Image2Type: Entity Type Prediction via Image Analysis

Master Thesis

by

Patrick Eisele

Degree Course: Information Systems M.Sc.

Matriculation Number: [redacted]

FIZ Karlsruhe & AIFB

KIT Department of Economics and Management

Primary Examiner:	Prof. Dr. Harald Sack
Secondary Examiner:	Prof. J. Marius Zöllner
Supervisors:	Dr. Mehwish Alam
	Russa Biswas
Submitted:	January 14, 2022

Abstract

Entity type prediction is the task to assign an entity in a Knowledge Graph (KG) its semantic type. However, the type information in most open KGs such as DBpedia or Wikidata are incomplete and noisy. Thereby the type of an entity is a fundamental information. Currently, most state-of-the-art entity type prediction models use only structured data to perform entity type prediction. This reduces prediction accuracy, creates incomplete and noisy type information. At the same time 51.1% of entities in the data set DBpedia630k have at least one associated image. Hence it is obvious to extract the additional information incorporated in these images to improve accuracy of entity type prediction to complete type information. This thesis presents an approach which creates a combined entity representation consisting of image features and structural information. On top of it a fully connected neural network is deployed for classification. The model performance is measured on two newly created real-world benchmark datasets. The results suggest that the approach is suitable to perform entity type prediction, however optimization is needed to improve performance.

Contents

List of Abbreviations

AC accuracy.

AMRI Adjuested Mean Rank Index.

CC Combined Classification.

CC-P Combined Classification pretrained TransE embeddings.

KG Knowledge Graph.

KGE Knowledge Graph Embedding.

Ma-F_1 Macro-averaged F_1.

Mi-F_1 Micro-averaged F_1.

MRR Mean Reciprocal Rank.

SC Structural Classification.

SC-P Structural Classification pretrained TransE embeddings.

URI Uniform Resource Identifier.

List of Figures

List of Tables

1 Introduction

Over the past few years, a large number of Knowledge Graphs (KGs) have been published which are either or domain-specific cross-domain such as DBpedia [4] or Wikidata [30]. KGs represent information in a structured form, i.e., in the form of entities and relations. However, the type information in most of the open KGs are incomplete and noisy.

Thereby the type of an entity is a fundamental information and plays an important role in various knowledge-driven applications, for instance, entity linking [14], relation extraction [35] and question answering [21]. For example, an user could be interested in the question whether an airline flies to the city of Yakutsk (Russia). This question can be answered with the help of a KG by querying for entities of the type *Airline* which are associated with the entity *Yakutsk* or the entity *Yakutsk Airport*.

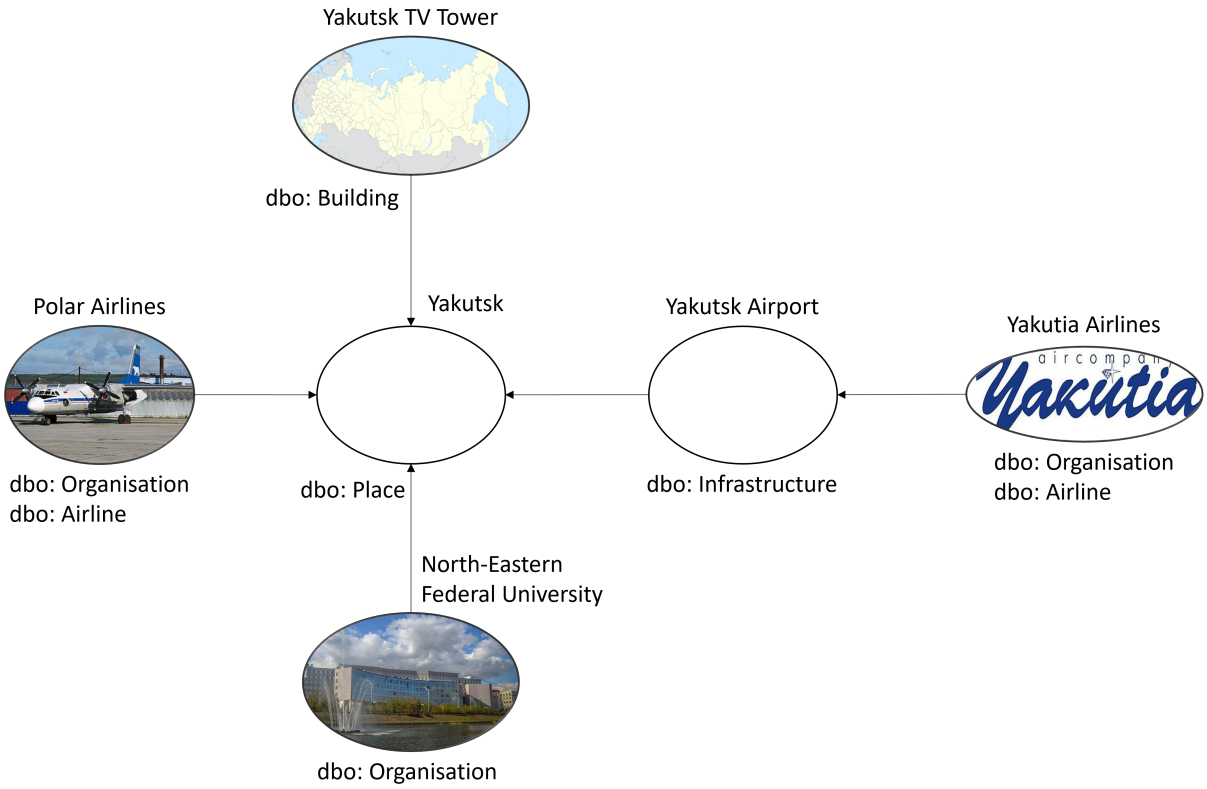


Figure 1: Exempted subgraph from DBpedia

Currently, most state-of-the-art entity type prediction models use only structured data, e.g., textual description, category or property of an entity to predict its type. This reduces prediction accuracy, creates incomplete and noisy type information [26, 8] as the main source for DBpedia [4] is Wikipedia where the information is extracted from info boxes. These texts are human-created and faulty in some cases.

At the same time 51.1% of entities in the data set DBpedia630k [19] (a subset of DBpedia) have at least one associated image. Hence it is obvious to extract the additional information incorporated in these images to improve accuracy of entity type prediction to complete type information in DBpedia.

To answer the question in the example previously introduced with the help of DBpedia: there are two Airlines in *Yakutsk*, *Yakutia Airlines* and *Polar Airlines*, both have an associated image. This image can be used in combination with structural information contained in the KG to increase the accuracy of entity type prediction.

However, the included images are also created by humans and can therefore be ambiguous which would cause an image classifier to learn wrong predictions. An illustration of this problem is the entity *Polar Airlines* whose image displays a plane that an image classifier would classify as the type *MeanOfTransportation (Aircraft)* instead of *Organisation (Airline)*. Therefore, given the information the main goals of this thesis are:

- **RQ1**: How to filter out misleading images from which an image classifier would learn wrong entity type predictions?
- **RQ2**: How to learn distributed representations of entities using their structural information?
- **RQ3**: How to perform classifications of the entities into their types using structural as well as image information?

The remainder of this thesis is structured as follows: We will first look at necessary fundamentals regarding KGs, families of embeddings models along with metrics needed for evaluation in Chapter 2 before we present representation learning and entity type prediction in Chapter 3. The following Chapter 4 presents the approaches developed for entity type prediction. Following Chapter 5 where we discuss the results and compare them with other approaches. The thesis is concluded with a conclusion in Chapter 6 along with future work in Chapter 7.

2 Fundamentals

In this Section we introduce the necessary fundamentals. First, we define the term KG, then we discuss how to create KGEs, discuss the downstream application of entity type prediction and finally introduce the metrics used in thesis for evaluation.

2.1 Knowledge Graphs

The term KG is used ubiquitously and there is no single commonly accepted definition [11]. Therefore, in the context of this thesis the following definition shall be used: A KG is represented as $G = (E, R, T)$, where E is the set of entities, R is the set of relations, and T is the set of triples with $T = E \times R \times E$ [37].

2.2 Knowledge Graph Embeddings

The task of creating KGEs aims to learn a low-dimensional vector space representation for each entity and relation in the KG while preserving their semantic meaning [17]. It is a critical research issue as it enables many downstream applications, like entity linking [14], relation extraction [35], and question answering [21] which would otherwise not be possible due to data sparsity and growing computational inefficiency in big KGs [22].

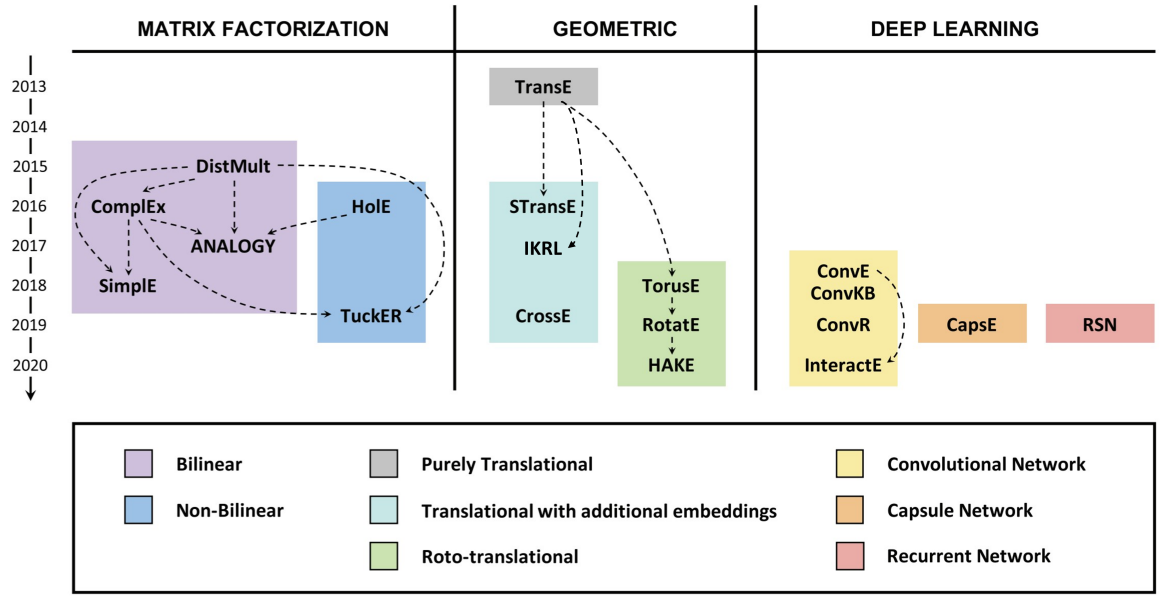


Figure 2: Different families of KGEs models. Dotted arrows indicate that the target method builds on the source method, based on [28]

Since we utilize only structural information to create KGEs in this thesis, we focus on comparison on models that learn soly from KG structure. A comparison of further models which combine different literals, for instance text, images or numerical values see [13]. Rossi et. al propose to differentiate between three main families of KGEs models [28]:

Tensor decomposition models. This group is called matrix factorization models in Figure 2. They use a three-dimensional adjacency matrix that can be decomposed in several low-dimensional vectors for entity and relation embedding. The adjacency matrix is only partially existent due to incompleteness and noise. **Bilinear models** take only single facts into consideration when computing the embedded representation. They define the relational embedding as a bidimensional matrix $\boldsymbol{R} \in \mathbb{R}^{d \times d}$ for given head embedding $\boldsymbol{h} \in \mathbb{R}^d$, tail embedding $\boldsymbol{t} \in \mathbb{R}^d$. Their scoring function is the linear product
$\phi(\boldsymbol{h}, \boldsymbol{R}, \boldsymbol{t}) = \boldsymbol{h}^t \boldsymbol{R} \boldsymbol{t}$

Geometric models. This family of models describe the relation as a geometric transformation between head and tail in the latent space. To compute the embedding of the tail entity, a spatial transformation τ is applied. As scoring function they use the distance between the tail vector and the resulting vector $\phi(\boldsymbol{h}, \boldsymbol{R}, \boldsymbol{t}) = \delta(\tau(\boldsymbol{h}, \boldsymbol{r}), \boldsymbol{t})$. Similar to Tensor decomposition models, these family of models computes backpropagation usually direct on entities and does not share parameters. [28] identify three groups of models within this family: **Pure translational models** rely on the assumption that the relational embedding added to the head embedding is expected to be spatially close to the tail embedding. **Translational models with additional embeddings** learn several embeddings for each element in the KG. **Roto-translational models** employ in addition to or instead of translational operations a rotation-like transformation.

Deep learning models. This model family uses different types of deep neural networks to learn patterns from the KG's input data. Each layer of a **Convolutional Neural Networks** applies a low-dimensional filter to its input data, like embedding KG elements in a fact. **Capsulate Neural Networks** are composed of convolutional layers as well. They solve a weakness of convolutional neural networks by being able to recognize features without loss of spatial information what leads to a more stable representation. **Recurrent Neural Networks** use recurrent layers to analyze complete paths in a KG instead of just individual facts.

It should be noted that the term KG representation learning is often used synonymously with KGE in the literature.

2.3 Entity Type Prediction

In this thesis we consider entity typing as a classification task with the entity types as classes. For this purpose a prediction function $P(t|e)$ that infers the type based on an entity representation $v(e)$ is learned [34]. Based on the entity representation, different types of neural networks are deployed for the classification. In this Section entity typing models are categorized into three parts depending on how they construct the entity representation $v(e)$:

Feature-based methods exploit various kinds of information, such as textual description, property and category, to create the feature representation of an entity.
KYLIN [32] is one of the first papers to perform type prediction for Wikipedia infoboxes. It searches for classes of Wikipedia pages with similar infoboxes and determines their common attributes in order to learn a CRF extractor.
MuLR [34] learns multilevel representations of entities by combining the complementary information of character, word, and entity embeddings followed by a hierarchical multi-label classification for fine-grained entity typing.

Structure-based methods are based on annotated corpus to learn low-dimensional entity representations:
SDType [26] is a statistical heuristic link-based type prediction mechanism to take care of noisy and incorrect data. It exploits links between instances to infer their types using weighted voting. The model assumes that certain relations occur only with particular types. Therefore, an instance has heuristically certain types if it is connected to other instances with some particular relations.

Methods that combine both information:
Jin et. al [19] propose the GCN-based model HMGCN to predict the entity types based on textual entity descriptions, its relations and Wikipedia categories.
Yogatama et al. [36] developed an embedding-based model taking features of structural information and labels from textual descriptions into account. Their model uses a ranking loss to be more robust against noisy labels due to fine-grained type predictions. Then a deep neural network is used to first learn entity representation and secondly predict the type hierarchy.

APE [18] constructs a partially-labeled attributed entity network that consists of structural, attribute and type information followed by deep neural network for representation learning. This enables them to combine three types of information without the need for large annotated corpuses.

Further, a distinction can be made between entity typing and fine-grained entity typing. Entity typing focuses on small sets of types like *Person*, *Organization* or *Animal*, while fine-grained entity typing assigns an entity more specific types, between which there is a hierarchical relationship in the form of a tree or directed acyclic graph [18, 19]. In particular, methods for fine-grained type prediction combine (several) feature representations with low-dimensional structural representations to utilize various kinds of information in order to improve accuracy.

Our model, on the other hand, combines structural information from TransE embeddings with images features for entity type prediction.

2.4 Evaluation

2.4.1 Hits @ K

$$Hits@K = \frac{1}{|\mathcal{I}|} \sum_{r \in \mathcal{I}} \| [r \leq k] \tag{2.1}$$

Hits @ K describes the portion of true entities which appear in an ordered rank of the first k entities. The value range is $[0, 1]$ where the closer to 1 the better [5]. Hits @ 1 represents a special case and corresponds to precision [37]. This metric does not differentiate between ranks larger than k, what means that all $r > k$ have the same influence on the final score Hits @ K. Consequently, other metrics are more suitable to compare different models.

2.4.2 Adjusted Mean Rank Index

The Adjuested Mean Rank Index (AMRI) was introduced by [5] and is defined as following:

$$MR = \frac{1}{|\mathcal{I}|} \sum_{r \in \mathcal{I}} r \tag{2.2}$$

$$AMRI = 1 - \frac{MR - 1}{\mathbb{E}\left[MR\right]} = \frac{2 \sum_{i=1}^{n} r_i}{\sum_{i=1}^{n} \left(|S_i| + 1\right)} \tag{2.3}$$

It's value range is $[-1, 1]$, whereby 1 is equal to the optimal performance and a value of 0 is equal to assigning random values.

2.4.3 Mean Reciprocal Rank

[5] The reciprocal rank is the multiplicative inverse rank of the first true type prediction. Thus the Mean Reciprocal Rank (MRR) is the average of the reciprocal ranks for a series of type predictions. It has a value range of $(0, 1]$ with the closer to 1 the better and is defined as follows:

$$MRR = \frac{1}{|\mathcal{I}|} \sum_{r \in \mathcal{I}} \frac{1}{r} \tag{2.4}$$

At the moment there is a controversy whether to use the MRR and if it has some theoretical flaws [12, 29]

2.4.4 Early Stopping

During supervised training of a neural network, validation data can be used to detect when overfitting starts to occur, in this case training is stopped early before convergence to avoid overfitting [27]. It used to speed up the training time or to improve generalization of a neural network.

Despite the above mentioned controversy the MRR is often used during the creation of KG embeddings for early stopping due to its ability to be more influenced by changes in low rank values in comparison to high rank values, without ignoring them completely like Hits @ K when the rank is larger than k [2].

3 Related Work

After discussing some general concepts, we will investigate representation learning and entity type prediction in more detail. In Section 3.1 we discuss one pure translational model along with a translational based model that additional embeds image literals. Next, we will discuss entity type prediction based on embeddings in Section 4.2.

3.1 Representation Learning

Representation learning models encode KGs into vector spaces to create embeddings. With the help of the learned embeddings relation semantics can be evaluated. In this subsection two transnational distance-based models shall be presented.

3.1.1 Translational distance models

[9] TransE models entities and relations into the same low-dimensional vector space. It is based on the assumption that for a given triple (h, r, t) the embedding of the tail entity t should be similar to the head entity's embedding h plus some relation r which leads to the following energy function:

$$E\left(h,r,t\right) = ||\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t}|| \tag{3.1}$$

$h + r \approx t$ lasts if both entities h, t are near neighbors, under other conditions $h + r$ is very different from t. The assumption can be visualized in vector representation as a vector sum of h, t and measures the distance to t (see Figure 3).

TransE utilizes a margin-based ranking loss to learn embeddings and consists out of two parts. The first part prefers lower values for given training triples while the second part of corrupted triples is generated by swapping the head or tail entity with another random entity. Since these are artificial entities, a high value is expected for this part of the loss.

$$L = \sum_{(h,r,t)\in S} \sum_{\left(h',r,t'\right)\in S'_{h,r,t}} \left[\gamma + d\left(\boldsymbol{h} + \boldsymbol{r}, \boldsymbol{t}\right) - d\left(\boldsymbol{h}' + \boldsymbol{r}, \boldsymbol{t}'\right)\right]_+ \tag{3.2}$$

TransE is an effective, efficient and at the same time simple approach to model entities and relations between them. But its simplicity is also its weakness as TransE only considers one-to-one relations, which can create conflicts when complicated one-to-many, many-to-

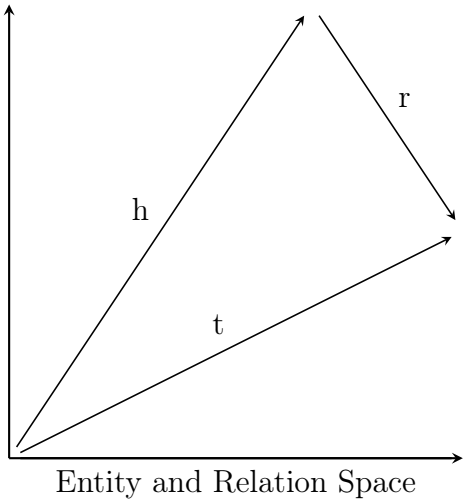


Figure 3: Illustration of TransE embedding method

many relations are modelled. Therefore, several extensions based on the same assumption have been created: TransH [31] models relations in a hyperplane with a translation-specific operation. TransR [23] uses separate vector spaces for entities and relations. TransD [16] extends TransR by considering the diversity of relations and entities.

However, all these models only utilize structural information from KGs and do not take additional information like images into account, although 51.1% of entities have at least one associated (see Section 4.1).

3.1.2 Models with image information

[33] IKRL extends structural embedding models like TransE [9] by jointly combining structural information and image information into the same vector space. Therefore they propose two representations for every entity: Structure-based representations $\boldsymbol{h}_S, \boldsymbol{t}_S$ that are conventionally learned in representation learning (see Section 3.1.1) and image-based representations $\boldsymbol{h}_I, \boldsymbol{h}_I$ constructed from constructed the head and tail entity‘s associated images.

The overall energy function follows the approach of other translation approaches.

$$E\left(h, r, t\right) = E_{SS} + E_{SI} + E_{IS} + E_{II} \tag{3.3}$$

TransE's energy function is extended by combining all structure-based representations and image-based representations into the same vector space for every triplet. E_{SS} only uses structure-based representations $\boldsymbol{h}_S, \boldsymbol{t}_S$ and has the same energy function as proposed

in TransE. E.g., the energy function $E_{SI} = ||\boldsymbol{h}_S + \boldsymbol{r} - \boldsymbol{t}_I||$ depends on the structure-based representation from the head entity and image-based representation from the tail entity.

To encode images they utilizes the neural network AlexNet [20] to create the image representation. Then the extracted features are projected from image space to entity space to extract discriminative features. For every image i, the image-based representation $\boldsymbol{p}_i$ is computed as following:

$$\boldsymbol{p_i} = \boldsymbol{M} \cdot f\left(img_i\right) \tag{3.4}$$

with the projection matrix $\boldsymbol{M} \in \mathbb{R}^{d_i \times d_s}$ of dimension d_i which stands for the dimension of image features and d_s representing the dimension of entities.

IKRL combines structural information and image information for representation learning of KGs. However, IKRL assumes that every image is associated with exactly one entity. Indeed, an image could inherit information of two entities, but its information is in this case only utilized for the representation of one entity [13]. This could lead to a decrease of performance in representation learning as 48.9% have no associated image (see Section 4.1). Moreover, IKRL requires due to Formula 3.3 the head entity as well as the tail entity of a given triplet to have an associated image [13].

Furthermore, the authors propose an attention mechanism to aggregate all image representations of an entity into an aggregated image-based representation. Since we assume that every entity has at most image representation, we skip its explanation at this point. Those who are interested can read about it at [33].

3.2 Entity Type Prediction

All the papers presented in this Section use different types of embeddings but the classification part should now be discussed. Therefore, the embeddings are only briefly discussed.

[7] Cat2Type creates Wikipedia category embeddings based on textual information in the category names with the help of widely used language models like Word2Vec, Glove or Wikipedia2Vec in combination with structural features of Wikipedia categories by training Node2Vec on a category-category network. The final entity representation derived from category name features is formally given by

$$E_i^{Model_{LM}} = \frac{1}{m} \sum_{j=1}^{m} C_j^{LM} \tag{3.5}$$

where the entity representation $E_i^{Model_{LM}}$ is the average of category representations C_j^{LM} generated by m language models. The same applies accordingly to the node embedding models. On top of the entity representation a fully connected neural network with two hidden layers is deployed for classification. The authors show state of the art results in entity type prediction for 14 non-overlapping types in comparison with other approaches. However, the question arises whether the created entity representation can be used for fine-grained typing as well. Especially averaging all category representations in Equation 3.5 instead of using all available features for classification could introduce noise.

[34] Schütze, Yaghoobzadeh on the other hand create a joint entity representation by combining three complementary entity representations for fine-grained entity typing. They create the vector representation by concatenating character, word, and entity level representations as input for a multilayer perceptron with one hidden layer. Since fine-grained entity prediction is performed, an entity can be assigned to multiple types. Therefore, to set up a binary classification problem a binary cross-entropy loss is used in the last layer, formally given by

$$CE_{loss} = \sum_i -\Big(t_i \log p_i + (1 - t_i) \log (1 - p_i)\Big) \tag{3.6}$$

where t_i is the ground truth and p_i the prediction that an entity is assigned.

This approach achieves consistently good results especially when combining all three levels of embeddings, but needs a large, annotated corpus to be trained on.

A solution to this problem is proposed by Jin et. al [18] to reduce the size of the annotated corpus needed to learn a joint representation consisting of low-dimensional structural, entity attribute and entity type representations.

Based on a concatenated feature vector consisting of structural and attributed representation, the entity type is predicted here with the help of a multilayer perceptron but only for the share of entities with type information. For an unlabeled entity its combined structural and attribute information is used for neighbor prediction.

This approach allows them to use the information of all entities contained in the network to improve the accuracy of the classification and combines the advantages of semi-supervised learning as well as attributed embeddings.

[19] A completely different approach is taken by Jin et. al. They construct three undirected entity graphs, namely co-occurrence graph, category-based graph and property-

based graph to capture different semantic correlations between entities. Every entity graph is fed to its corresponding GCN model, which consists of several stacked GCN layers, formally given by

$$\boldsymbol{Z}^{(i)} = \sigma\left(\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{Z}^{(i-1)}\boldsymbol{W}^{(i)}\right) \tag{3.7}$$

$\widetilde{\boldsymbol{A}} = \boldsymbol{A}+\boldsymbol{I_n}$ is the adjacency matrix with selfloops, $\boldsymbol{I_n}$ the identity matrix. $\widetilde{\boldsymbol{D}}_{ii} = \sum_j \widetilde{\boldsymbol{A}}_{ij}$, for all types j and entity i. Due to the definition of a GCN layer in Equation 3.7, the feature vectors of all directly connected neighbor entities are linearly combined per layer. So two layers recursively add the features of two distant neighbors using $\boldsymbol{Z}$. With more than three hidden layers, more noise than useful information is added to an entity representation even when using an attention function [37].

These three GCN models share parameters such as model weight $\boldsymbol{W}^{(i)}$ and unsupervised consistency regularization what enables them to decide together based on three different semantic perspectives. To optimize the jointly entity type prediction the difference between two GCN predictions $\hat{\boldsymbol{Z}}^{cat}$, $\hat{\boldsymbol{Z}}^{co}$ for all entities N is minimized, formally given by:

$$L_{c1} = \sum_{i=1}^{N} \parallel \hat{\boldsymbol{Z}}^{cat}_{i,:} - \hat{\boldsymbol{Z}}^{co}_{i,:} \parallel^2 \tag{3.8}$$

In detail, the regularization differs, but the basic idea is the same for all threee GCN models. The overall loss is the sum of the supervised loss L_s (from predicting the type of labeled entities) and all regularizations.

$$Loss = L_s + \lambda\left(t\right)\left(L_{c1} + L_{c2}\right) + \lambda^{'} L_{hie} \tag{3.9}$$

HMGCN can outperform all other in this Section discussed approaches. However, its performance is highly dependent on the optimal weighting by $\lambda^{'}$ and the chosen function $\lambda\left(t\right)$ of the different regularization terms in the combined loss function for the data set.

4 Methodology

This Chapter describes the necessary data pre-processing in Section 4.1 to be able to create TransE embeddings based on the dataset DBpedia630k$_{\text{image}}$ and be able to carry out entity type prediction. Therefore, we present two approaches in Section 4.2, one baseline and our newly developed approach.

4.1 Data Preprocessing

Along with structural information, each entity in DBpedia [4] also contains textual information in the form of an abstract as well as an associated image. 51.1% of entities in the data set DBpedia630k [19] (a subset of DBpedia) have at least one image.

In order to extract the additional information incorporated in these images to develop a classifier for the task of entity type prediction to complete type information in DBpedia the data set must be pre-processed.

The original data set DBpedia630k is from 2019 and as most KGs are incomplete and lack of type information, we expect outdated entities. So, the first step is to update as well as to complete type information, names and DBpedia Uniform Resource Identifiers (URIs) where possible. Furthermore, outdated entities which do not exist anymore, without type information, or changed type information that is not considered in this subset of DBpedia are excluded. That includes 13.5% of data. For 1.8% of the entities the type information was updated during this process, of which 0.8% remain in the dataset since their updated type information belongs to the 14 coarse grained types considered. Most of the changes could be determined for the fine-grained type of Plant to Insect or Animal.

In the next step we extract the main Wikipedia image for every of the remaining entities, if possible, with the help of the web scraper *Beautiful Soup*[1] to create the subset DBpedia630k$_{\text{image}}$. Images whose width or height is not greater than 200 are not considered, as they are standard images that can be found in a large part of the Wikipedia articles, without relevance for a specific entity. We extract the main Wikipedia image under the assumption that it is the most accurate entity representation and therefore contains the most information. DBpedia encodes its entity URIs in UTF-8 what includes chars that are not supported by most operating systems as file names. To avoid this restriction, we developed a function that replaces all special characters in image names

[1] `https://www.crummy.com/software/BeautifulSoup/bs4/doc/`

with ASCII conform symbols and maps them to given entity URI.

Wikipedia supports images in different formats[2] in any size. However ResNet-50 [15], the image analyzer used for classification expects images to be in the format JPEG with the shape 224 x 224 along with exactly three colour channels of RGB. The same requirements apply to Vision Transformer [10] with the difference that images of shape 224 x 224 and 384 x 284 are supported. All images are transformed so that they meet the stated requirements for classification.

This new dataset is called DBpedia630k$_{\text{image}}$ as it contains all valid entities with one associated image.

As already introduced in Chapter 1 there are two types of complications associated with the newly created dataset are expected: An entity is labelled wrong if its label does not match the content of the associated image. Further an image is considered as ambiguous if it displays the correct entity but from a context which would cause an image classifier to learn wrong predictions. An illustration of this problem is the entity *Polar Airlines* in Figure 1 whose image displays a plane that an image classifier would classify as the type *Aircraft (MeanOfTransportation)* instead of *Airline (Company)*. There is still a certain semantic connection between the predicted *Aircraft (MeanOfTransportation)* and its actual type *Airline (Company)*. However, the difference between the actual type and the type predicted with the help of the image can assign a completely wrong type to the entity. For example, the main Wikipedia image of the entity *Yakutsk TV Tower* shows a map of Russia since the city of Yakutsk is located in Russia, although the correct type of *Yakutsk TV Tower* is Building.

At this point it must be mentioned that a fine grained hierarchically type prediction was originally planned, but due to problems with the data set, ultimately only a flat type prediction is pursued, see Section 5.3. Hence the following methods are implemented for fine grained DBpedia types. Moreover, at the beginning of the thesis we had problems with the application for computing capacity at the BWUniCluster 2.0[3]. To avoid this problem, we did not use ResNet-50 [15] in this step, as in the further course of this thesis but rather Vision Transformer for image analysis. There is a very high-performance implementation for Vision Transformer [10] that runs in Google Colab[4].

To filter out misleading entities and ambiguous images, Vision Transformer [10] pre-

[2] `https://www.mediawiki.org/wiki/Help:Images`
[3] `https://wiki.bwhpc.de/e/Category:BwUniCluster_2.0`
[4] `https://research.google.com/colaboratory/`

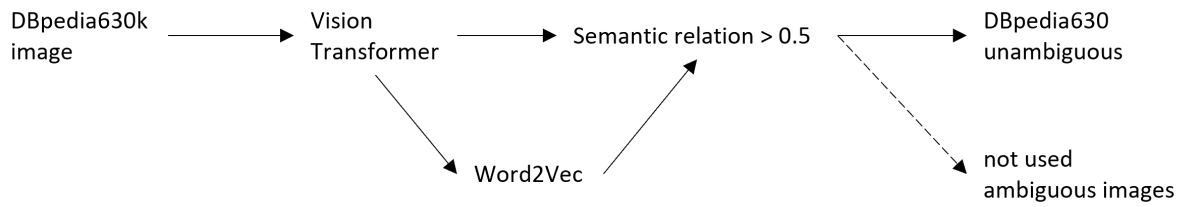


Figure 4: Pipeline to create DBpedia630k$_{\text{unambiguous}}$ from DBpedia630k$_{\text{image}}$

trained on ImageNet-21k[5] is used in combination with Word2Vec [25] to calculate the strength of the semantic relation between the actual type of an entity by Vision Transformer extracted image information. In detail we proceed as follows: First Vision Transformer predicts the ten most likely labels for the associated image of an entity. Next it is checked whether the one the labels equal one type in its type-path, whereby a type-path corresponds to all types which are contained in the type hierarchy of the DBpedia, subset DBpedia630k from coarse-grained to fine-grained. For example, *Polar Airlines* type path is

$$Organization \rightarrow Company \rightarrow PublicTransitSystem \rightarrow Airline$$

If one of the ten most likely contents of the image (predicted by Vision Transformer) corresponds to an element of the entity's type- path, we assume the highest semantic relation of one. If not, Word2Vec is used to calculate the semantic relation of the predicted ten types and all elements in the entity's type-path since there does not exist a mapping between the classes from ImageNet-21k and types from DBpedia. With the help of this procedure, we create several variants of DBpedia630k$_{\text{unambiguous}}$ with a base score of at least 0.5 to measure the influence of images on entity typing performance in Section 5.4.2

We call the second new dataset DBpedia630k$_{\text{unambiguous}}$ due to the fact that it is a subset of DBpedia630k$_{\text{image}}$ and contains only entities with an associated unambiguous image. The following sankey diagram in Figure 5 gives an overview about the number of entities in each created dataset.

KGs are dynamic structures whose entity URI along with relations change over time. This can lead to version conflicts between entity URIs in different models depending on the DBpedia version the model is trained on. To compare entity typing performance we use pretrained TransE embeddings which are based on entities of the DBpedia 2016-10 dataset, but DBpedia630k was created 2019. It is therefore necessary to create a mapping

[5]https://image-net.org/index.php

between revisions of an entities URI to be able to compare the results.

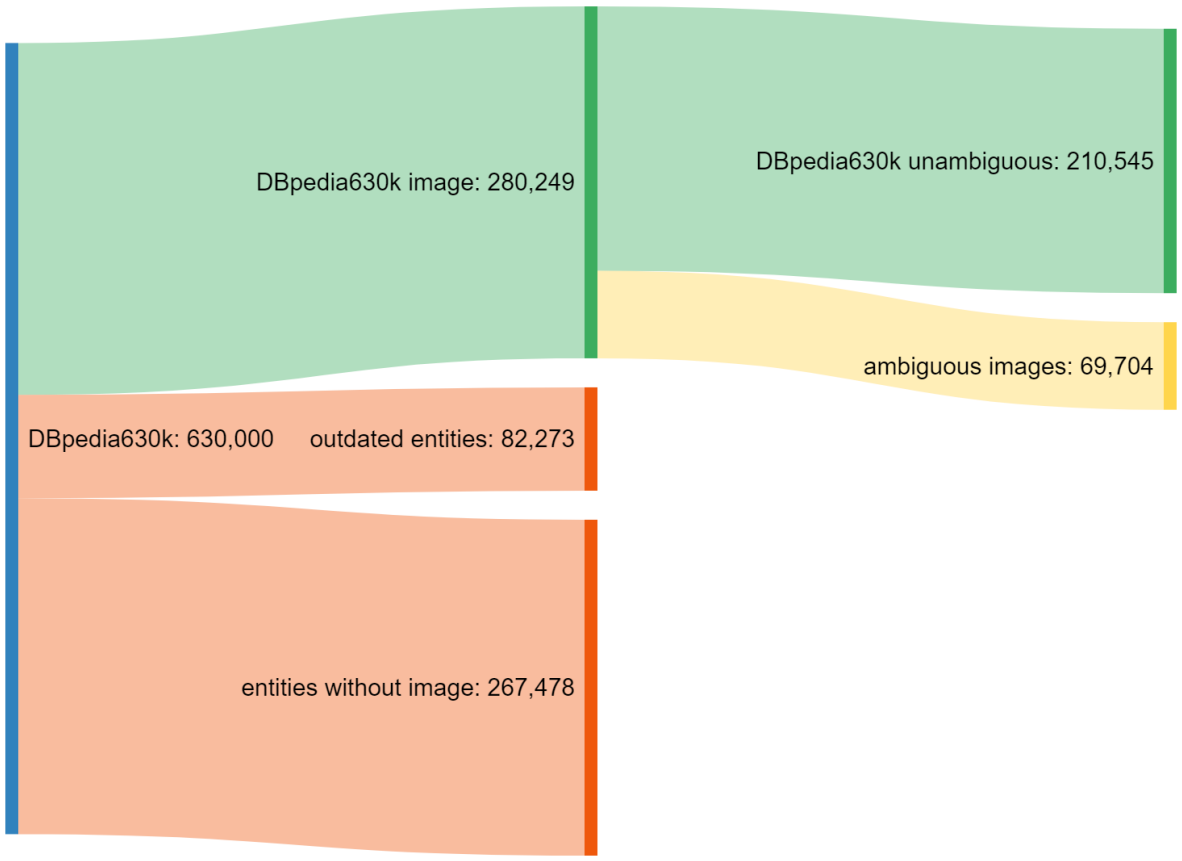


Figure 5: DBpedia630k after preprocessing and remaining derived entities used to create TransE embeddings and used for classification

4.2 Entity Type Prediction

As already mentioned in 2.3 we consider entity typing as a classification task in which entity types represent the classes. All definitions introduced in the following Section 4.2.1 also apply to Section 4.2.2 without further mentioning unless otherwise specified.

4.2.1 Structural Classification

Entity representation. The in Section 3.1.1 discussed translational model TransE generates a low-dimensional entity and relation representation called embedding based on the geometrical structure of the KG. Since we only want to determine the entity type, we discard the relation vectors and use only the entity representation for classification.

Classifier. Following [7, 6] we deployed a fully connected neural connected network consisting of two linear layers with ReLU as activation function on top of the entity representation for classification. Figure 6 shows a visualization of the network in which the numbers represent the shape of the tensor. The shape of the TransE vectors is either 175 or 200, depending which TransE vectors are used in the classification (see Section 5.3). The output layer's size 7 is the $|type|$ in which each element t of this layer outputs the probability for type t. At the last layer a SoftMax function in combination with cross entropy loss is used to calculate the probability of an entity belonging to one class. This step is formally given by:

$$S\left(x_i\right) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{4.1}$$

$$CE_{loss} = -\sum_i t_i log\left(S\left(x_i\right)\right) \tag{4.2}$$

The SoftMax function, shown in Equation 4.1 normalizes the outputs (predicted entity types) from the fully connected neural network to a probability distribution $\in [0, 1]$ over the predicted output labels, where x_i is the label of class i whose probability shall be calculated. x_j are the inferred labels for each the j classes.
With the help of Equation 4.2 the difference between the probability distribution of the true entity types and the predicted ones is calculated to propagate the loss back to optimize the weights of the fully connected neural network, where t_i is the ground truth and $S\left(x_i\right)$ probability for all i given classes.

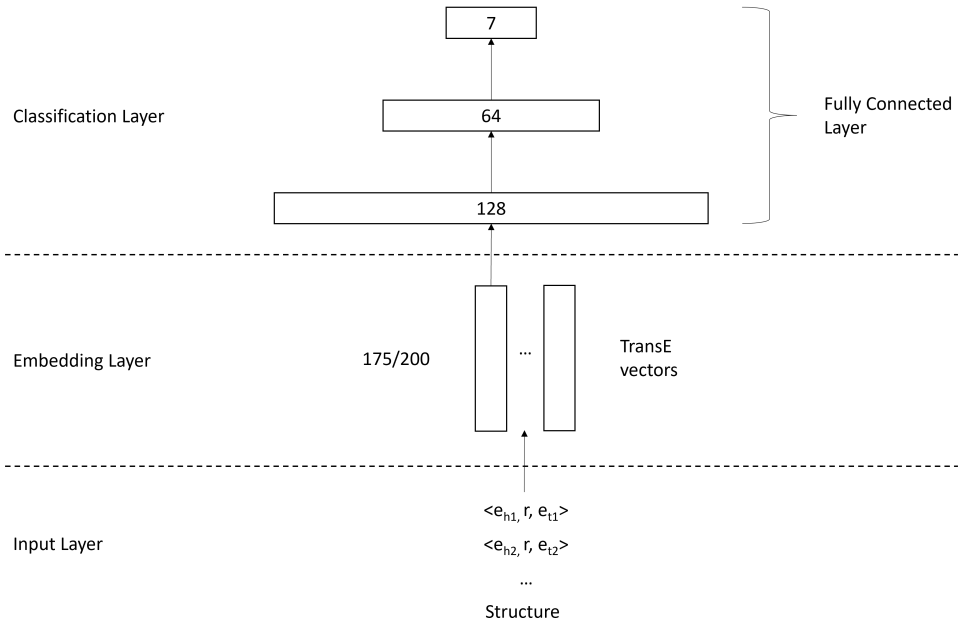


Figure 6: Visualization of structural classification

4.2.2 Combined Classification

Image feature extraction. In order to extract image features from the associated image of the entity, we utilize ResNet-50 [15] pretrained an ImageNet, an established state of the art residual neural network. It consists of two max pooling layers, 50 convolutional layers, one fully connected layer and a SoftMax layer. To deal with degrading gradients what is often a problem for very deep neural networks, the authors introduce identity mappings that are skip connections between one or more layers. We take the 1000-dimensional image feature as one of the two inputs for entity type prediction.

Combined feature vector. Since the structural TransE features and image features are from different vector spaces the extracted features are projected into a unified vector space. To ensure that the TransE vector as well as the image feature vector have the same influence on subsequent classification of the combined feature vector by the fully connected neural network, both vectors have the same dimension of 200 each before being concatenated.

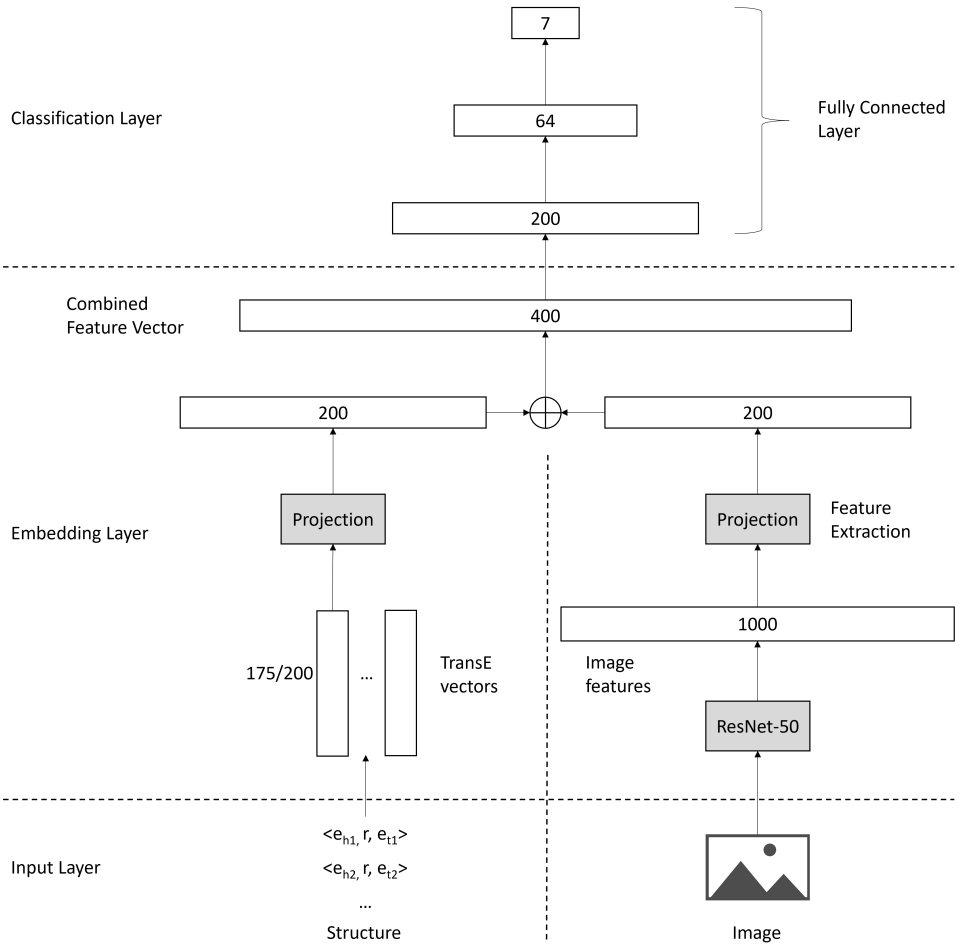


Figure 7: Visualization of combined classification

5 Evaluation

In Chapter 4 we discussed different approaches to predict the type of an entity. Now in this Section, we want to present the results and compare them with different approaches. First, we will introduce the experimental setup along with the data used in the experiments. Afterwards, the results of the structural model and the developed classifiers are shown in Section 5.3 and 5.4 as well as compared with different approaches.

5.1 Experimental setup

We implement all experiments using Python[6] in version 3.9 as programming language and Anaconda[7] as development environment. For creation of the KG embeddings, we use the KG embedding framework PyKEEN [3] which extends Pytorch[8]. Following we use Pytorch to implement the different classifications. All experiments are performed on BWUniCluster 2.0[9] with 30GB RAM with NVIDIA Tesla V100 GPU.

5.1.1 Structural Model

We search among the following values for hyper-parameters: The embedding dim in {100, 125, 150, 175, 200}, the learning rate between values in {0.001, 0.1}, the batch size in {64, 200} with a step size of 16. We created inverse triples, used early stopping and a rank-based evaluator. The selected setting is that embedding dim is 175, the learning rate is 0.08 and the batch size is 192. We run the embedding experiment with a time limit of 48 hours to find the presented configuration.

5.1.2 Classification

For the task of classification, we search among the following configurations: The learning rate in {0.01, 0.05, 0.1}, the number of epochs in {10, 20, 50, 100}. In the final setting we use learning rate = 0.05, number of epochs is 20 for the structural classification along with 10 for the combined classification and select a batch size of 64. We used this batch size because [24] found smaller batch sizes generalize better than larger batch sizes. Further

[6] `https://www.python.org/`
[7] `https://www.anaconda.com/`
[8] `https://pytorch.org/`
[9] `https://wiki.bwhpc.de/e/Category:BwUniCluster_2.0`

the activation function is ReLU, Adam the optimization algorithm and cross entropy loss the loss function.

5.2 Datasets

Following the results of [7, 6] the created subsets DBpedia630k$_{\text{image}}$ and DBpedia630k$_{\text{unambiguous}}$ of DBpedia630k are used to evaluate the results. DBpedia630k [38] consist of 14 non-overlapping classes and was originally constructed for text classification. It contains 630000 entities with type, 5191887 triples and 570 different relations (see Table 1). The constructed subsets DBpedia630k$_{\text{image}}$, DBpedia630k$_{\text{unambiguous}}$ are the resulting datasets with an image for every entity, for more information on how these splits were generated see Section 4.1. There are no entities shared between the train, test, and validation set in none of the created DBpedia630k variants.

5.2.1 Dataset for Structural Model

The split DBpedia630k$_{\text{image}}$ is used to create TransE embeddings. It contains 350000 entities from every of the 280249 entities have an associated image. The entities of DBpedia630k$_{\text{image}}$ are randomly splitted into a train set which consists of 50% of the entities, a test set with 30% and a validation set with 20% of the overall entities.

Split	purpose	Triples	Entities Train	Entities Test	Entities Validation	Entities with type	Different relations
DBpedia630k		5191887	315000	189000	126000	630000	570
DBpedia630k image	TransE embedding	1753724	140125	84074	56050	280249	372

Table 1: DBpedia splits used to create TransE embeddings

5.2.2 Datasets for Classification

To evaluate the performance of the different classifications two subsets of DBpedia630k, DBpedia630k$_{\text{image}}$ and DBpedia630k$_{\text{unambiguous}}$ each containing a balanced and an unbalanced variant (with additional variants for DBpedia630k$_{\text{unambiguous}}$) are created. DBpedia630k$_{\text{image}}$ is used to create TransE embeddings and consequently the structural classification is based

on these embeddings. In addition to that we measure the extent to which an image recognition algorithm can learn from the with entities associated images with the help of three variants each for the balanced and the unbalanced version of DBpedia630k$_{\text{unambiguous}}$. See Section 4.1 for how these different subsets based on DBpedia630k are created and Section 4.2.2 for the analysis. Unless otherwise mentioned, we always refer to the DBpedia630k$_{\text{unambiguous}}$ the base variant with a semantic relationship of 0.5 in the following sections.

Every split is divided randomly into 95% training data and 5% test data and consist of 7 non-overlapping coarse-grained classes. These are created by going up on level in DBpedia's type hierarchy in comparison with the 14 classes in DBpedia630k.


```
{
  "Person": {
    "occurrence": 42092,
    "Artist": 14914,
    "Athlete": 13192,
    "OfficeHolder": 13986
  },
  "Organisation": {
    "occurrence": 47493,
    "Company": 25073,
    "EducationalInstitute": 22420
  },
  "MeanOfTransportation": 20187,
  "Place": {
    "occurrence": 67650,
    "Building": 25476,
    "NaturalPlace": 21347,
    "Village": 20827
  },
  "Animal": 21980,
  "Plant": 24330,
  "Work": {
    "occurrence": 56584,
    "Album": 17415,
    "Film": 17447,
    "WrittenWork": 21722
  }
}
```


(a)


```
{
  "Person": {
    "occurrence": 4000,
    "Artist": 1333,
    "Athlete": 1333,
    "OfficeHolder": 1334
  },
  "Organisation": {
    "occurrence": 4000,
    "Company": 2000,
    "EducationalInstitute": 2000
  },
  "MeansofTransportation": 4000,
  "Place": {
    "occurrence": 4000,
    "Building": 1333,
    "NaturalPlace": 1333,
    "Village": 1334
  },
  "Animal": 4000,
  "Plant": 4000,
  "Work": {
    "occurrence": 4000,
    "Album": 1333,
    "Film": 1333,
    "WrittenWork": 1334
  },
}
```


(b)

Figure 8: Visualization of DBpedia630k$_{\text{image}}$ used for classification with respective subclasses of all entities (a) and balanced (b)

The unbalanced version of a split contains all available entities of a class whereas the balanced version contains 4000 of each of the 7 coarse-grained classes with an equal distribution of the corresponding subclasses. For example, in Figure 8, there are 14000 different entities of the type of *OfficeHolder* in the unbalanced version of the DBpedia$_{\text{image}}$

dataset, whereas the balanced version of the same dataset contains 1334 entities due to the required equal distribution of sub types.

Accordingly, every balanced subset based on DBpedia630k subset contains 26600 entities for training and 1400 to test. For more detailed information about the different splits, see Table 2.

Split	purpose	Triples	Entities Train	Entities Test	Different relations
DBpedia630k image	classification	1753724	266237	14012	372
DBpedia630k image	classification balanced	1753724	26600	1400	372
DBpedia630k unambiguous	classification	1262376	200017	10527	356
DBpedia630k unambiguous	classification balanced	1262376	26600	1400	356

Table 2: DBpedia splits used for classification

5.3 Analysis of Structural Model

Table 3 displays the link prediction performance of the created TransE embeddings for $\text{DBpedia630k}_{\text{image}}$. Link predictions means during this thesis the ability of the embeddings to predict the tail entity of a relation based on the head entity. The results are very poor, and each metric indicates the same fact, that the learned embedding cannot capture the relationships of any triple from $\text{DBpedia630k}_{\text{image}}$. This is surprising as TransE is an embedding model which is known to be easy to train because of the small number of parameters.

Split	Hits@1	Hits@3	Hits@5	Hits@10	AMRI	MRR
DBpedia630k image	0	1e-6	2.001e-6	8.005e-6	0.012	0.002

Table 3: Link prediction performance of TransE embeddings created on $\text{DBpedia630k}_{\text{image}}$

Especially the AMRI score of 0.012 indicates that the model is returning random tail predictions, and this is in fact true for the case of this evaluation. In total, $\text{DBpedia630k}_{\text{image}}$ contains 683260 different entities URIs in its triples file from which 321420 entities (47%)

appear only once and respectively 394722 entities (57.7%) are contained at most twice in the dataset.

The result is that 66 - 70% of the entities in the train split are not in the test or validation split. The difference is varying due to different random seeds from PyKEEN's triple factory[10] we use to split the overall dataset in the three subsequent parts. However, in order to perform a proper evaluation every entity must be included in the train, test and validation split at least once to be a predictable tail entity of a relation. If not TransE assigns a random entity as tail and consequently the evaluation results are of no significance, and this is especially the case here.

To be still able to interpret the results, we compare the classification performance of our own TransE embeddings based on $\text{DBpedia630}_{\text{image}}$ with the performance of TransE embeddings[11] which were pretrained on whole DBpedia. However, the inherited problem of $\text{DBpedia630}_{\text{image}}$ remains what makes it impossible to correctly learn the low-dimensional vector representation of entities, relations since it is impossible to compute the gradient of the loss function in backpropagation without correct test data. Further none of TransE's hyperparameters can be optimized due to the fact there is no correct validation data either. We write about the effects on the classification performance in Sections 4.2.1, 4.2.2.

For the original split DBpedia630k is the same problem existent with comparable differences between train, test, and validation splits: Of 1863230 different entities 818230 entities (43.39%) are included only once, respectively 1105652 entities (59.34%) are included at most twice in the dataset what leads to a difference of 66% between the three splits. Overall, we conclude that the data set is not suitable for creating embeddings and tasks in general which need several occurrences of the identical entity URI.

5.4 Analysis of Classification

In order to evaluate the performance of the structural and combined classification Macro-averaged F_1 (Ma-F_1), Micro-averaged F_1 (Mi-F_1) and accuracy (AC) are reported. Different variants of both classifiers are evaluated on several variants of DBpedia630k. It must be noted that all baseline models, except for Cat2Type 7 (types), predict 14 types on DBpedia630k and we only 7 types on two DBpedia630k subsets. All baseline results

[10] `https://pykeen.readthedocs.io/en/stable/reference/triples.html`
[11] `https://github.com/nheist/KBE-for-Data-Mining`

are taken from [7]. We use two subsets of DBpedia630k instead of the original dataset for evaluation because every entity must have one associated image for a combined classification based on structural information and image features. Since the data sets used for the evaluation are each a subset of DBpedia630k and the difference in performance between Cat2Type 14 and Cat2Type 7 is only 2.37%, the results are comparable.

Methods	DBpedia630k	
	Ma-F_1	Mi-F_1
MuLR [34]	0.752	0.777
APE [18]	0.760	0.784
HMGCN [19]	0.790	0.816
Cat2Type 14 [7]	0.948	0.947
Cat2Type 7 [7]	0.971	0.972

Table 4: Classification performance of baseline approaches on DBpedia630k, taken from [7]

Methods	DBpedia630k image			DBpedia630k unambiguous		
	Ma-F_1	Mi-F_1	AC	Ma-F_1	Mi-F_1	AC
SC	0.445	0.533	0.533	0.406	0.539	0.539
SC-P	0.985	0.987	0.987	0.994	0.994	0.994
CC	0.052	0.238	0.238	0.056	0.245	0.245
CC-P	0.050	0.232	0.232	0.054	0.234	0.234

Table 5: Entity type prediction performance of structural classification (SC), structural classification pretrained TransE embeddings (SC-P), combined classification (CC), combined classification pretrained TransE embeddings (CC-P) on DBpedia630k variants

5.4.1 Structural Classification

The SC-P variant of the proposed SC outperforms all baseline models with an average improvement of 14.3% on Ma-F_1, respectively 13% on Mi-F_1 as shown in Table 5. In the structural classification model, a fully connected neural network is deployed on top of TransE embeddings. This shows that the combination of TransE embeddings and a fully connected neural network is very capable of learning the underlying semantics and predict the entity type. Especially the performance improvement of 1.5% on Ma-F_1 and 1.6% on Mi-F_1 in comparison to Cat2Type 7, the only model in comparison that predicts seven types as well, confirms the performance results.

Impact of TransE embeddings. The SC-P variant outperforms the SC variant with a difference of 54.9% on Ma-F_1, respectively 46% on Mi-F_1. This expected big difference in performance validates the results from Section 5.3 and shows that TransE is unable to learn the semantic relationships in DBpedia630k$_{\text{image}}$ due to faulty test and validation data.

In comparison to SC-P, SC has poorer generalization properties. The 12% difference between Ma-F_1 and Mi-F_1 for SC compared to only 4% for SC-P shows that not all 7 types are equally affected by the insufficiently learned TransE embeddings. This difference results from the structure of DBpedia630k$_{\text{image}}$: For heterogeneous classes like *Person* with a large number of different subtypes in the type hierarchy, missing test and validation data have a significantly greater influence on learning the semantic structure than for homogeneous classes like *Plant* in which 94.2% of the entities correspond to the identical fine-grained type. Missing test and validation data can at least partially be compensated for by the number of entities the class contains.

Impact of semantic similarity of the images. Further there is no performance difference observable for all metrics on average between DBpedia630k$_{\text{image}}$ and DBpedia630k$_{\text{unambiguous}}$. This result is expected since the structural classification is based only on TransE embeddings.

Methods	DBpedia630k image			DBpedia630k unambiguous		
	Ma-F_1	Mi-F_1	AC	Ma-F_1	Mi-F_1	AC
SC	0.445	0.533	0.533	0.406	0.539	0.539
SC balanced	0.463	0.496	0.496	0.488	0.532	0.532
SC-P	0.985	0.987	0.987	0.993	0.994	0.994
SC-P balanced	0.978	0.987	0.987	0.994	0.994	0.994

Table 6: Comparison of structural classification performance for balanced and unbalanced splits of DBpedia630k$_{\text{image}}$, DBpedia630k$_{\text{unambiguous}}$

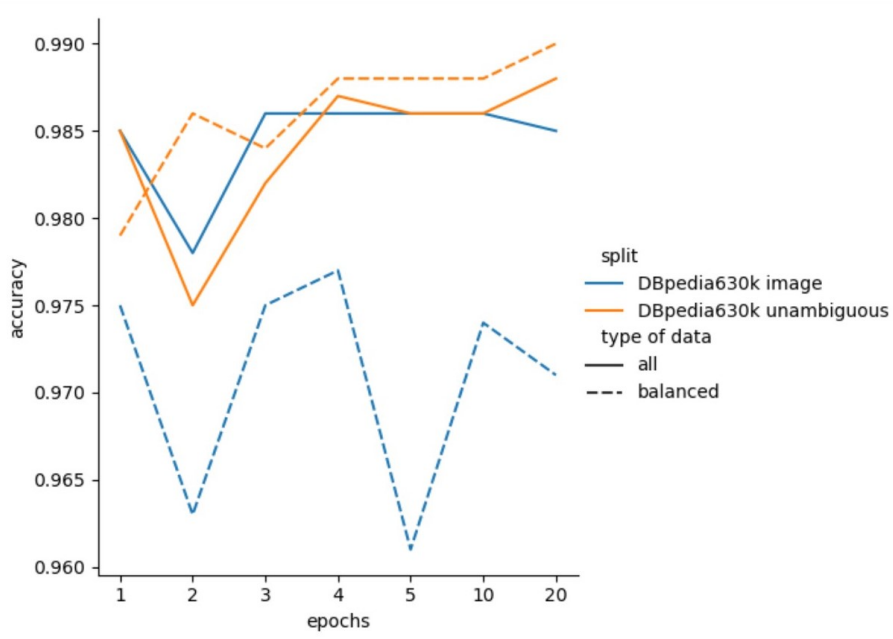
Impact of balanced dataset. The balanced variants of DBpedia630k$_{\text{image}}$ as well as DBpedia630k$_{\text{unambiguous}}$ contain 4000 entities per type with an equal distribution for every respective subtype, whereas the unbalanced variants contain all entities. This results in

a 6.9% diminution on AC for DBpedia630k$_{\text{image}}$ balanced on SC.

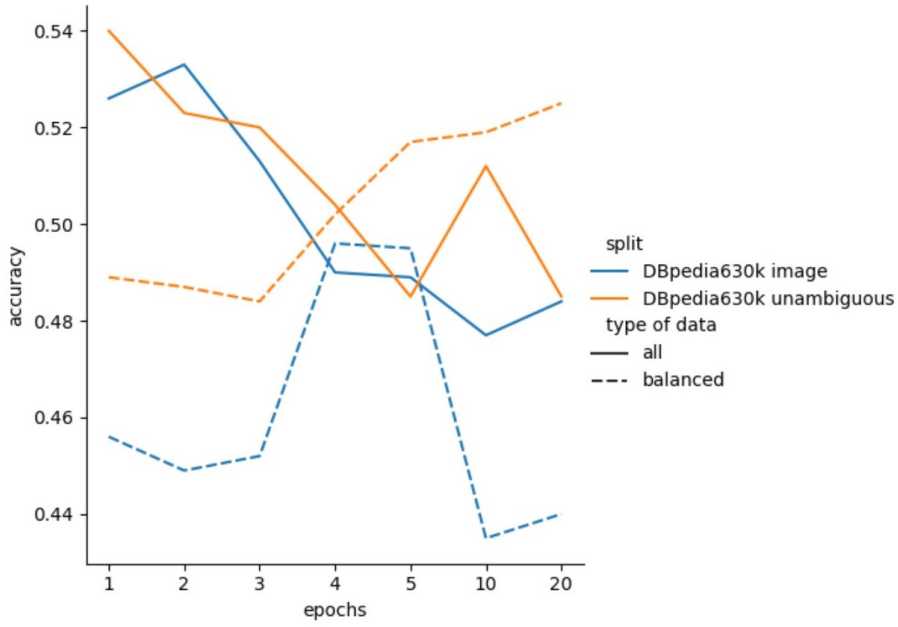
We assume that the difference in the prediction performance is caused by the combination of a different underlying distribution of entity types and an insufficient generalization ability of the classifier: The difference of SC between Ma-F_1 and Mi-F_1 for DBpedia630k$_{\text{image}}$ unbalanced is 16.6% in comparison to 6.7% for DBpedia630k$_{\text{image}}$ balanced. A macro-average will compute the F_1 metric independently for each class and then takes the average, hence all classes are treated equally. This means a good precision of a few classes contributes to a decent overall precision, what can be misleading due to imbalanced data. A micro-average will aggregate the contributes of all classes to compute the average F_1 metric, so classes are weighted with their relative frequency. Transferred to the data set here, this means some types are predicted by SC with a significantly higher degree of accuracy. For SC-P no significant change in performance can be observed.

Impact of number of epochs in training. Figure 9 visualizes the accuracy evolution over 20 epochs for SC-P in (a) and SC in (b). Except for training on DBpedia630k$_{\text{image}}$ balanced all trainings show a very high consistency on AC, whereby the variation for DBpedia630k$_{\text{image}}$ balanced is also very small with a maximum of 1.5%. From epoch 4 the results are constant, and the model will start too overfit.

Figure 9 (b), on the other hand, shows a very different picture for SC: The AC is very inconsistent depending on the epoch with a maximum reduction of 5.8% for DBpedia630k$_{\text{image}}$ balanced. We assume that the model underfits due to the insufficiently learned TransE embeddings.



(a)



(b)

Figure 9: Evolution of accuracy over epochs for SC-P (a) and SC (b)

5.4.2 Combined Classification

The developed Combined Classification (CC) along with the variation Combined Classification pretrained TransE embeddings (CC-P) in Table 5 perform significantly below average in comparison to all baselines models with average decrease of 82.5% on Ma-F_1 for both variants. The combined classification model, like the structural classification

model, uses TransE vectors. In this case, they are concatenated with image features. The classification performance of CC on Mi-F_1 for DBpedia630k$_{\text{image}}$ is only minimally better than a random prediction with uniform distribution and gives the impression that CC has not learned anything. However, as will become apparent in the remainder of this analysis, this assumption is only partial true.

Datasets	CC			CC-P		
	Ma-F_1	Mi-F_1	AC	Ma-F_1	Mi-F_1	AC
DBpedia630k image	0.052	0.238	0.238	0.050	0.232	0.232
balanced	0.039	0.157	0.157	0.038	0.156	0.156
DBpedia630k unambiguous 0.5	0.056	0.245	0.245	0.054	0.234	0.234
balanced	0.037	0.152	0.152	0.038	0.156	0.156
DBpedia630k unambiguous 0.6	0.073	0.342	0.342	0.069	0.324	0.324
balanced	0.043	0.177	0.177	0.044	0.184	0.184
DBpedia630k unambiguous 0.7	0.078	0.372	0.372	0.075	0.356	0.356
balanced	0.056	0.244	0.244	0.062	0.277	0.277

Table 7: Comparison of combined classification performance for balanced and unbalanced splits of DBpedia630k$_{\text{image}}$, DBpedia630k$_{\text{unambiguous}}$

Impact of TransE embeddings. The performance difference between SC and SC-P is 14.3% on Ma-F_1, respectively 13% on Mi-F_1. With the implementation of CC it could be reduced to less than 0.1% on average and can no longer be measured. Even though the overall classification performance is poor, this confirms the influence of image features on entity type prediction, as intended.

Surprisingly, with an average of 0,069 on Ma-F_1, respectively 0,319 on Mi-F_1 CC achieves better classification results than CC-P with 0,066 on Ma-F_1 and 0,305 on Mi-F_1 on each of the unbalanced datasets presented in Table 7. For CC-P it is the other way around, which performs better for the balanced version of all datasets. The differences are small overall but still consistent for all possibilities considered.

Influence of the combination of feature vectors Both SC, see Section 4.2.1, and classification of an entity using only its image achieve substantially higher performance

on Mi-F_1 and Ma-F_1 compared to CC.

To evaluate the theoretical performance of ResNet-50 in entity type prediction, a custom fully connected layer was added. In the next step, we trained the model 10 epochs on the respective data set from Table 8 to which the performance is to be compared. The gradient of all layers, except the newly added, is frozen in order not to destroy the already learned features [1]. Predicting the type of an entity only by its associated image performs only 15.5% worse in comparison to SC-P on Ma-F_1 and Mi-F_1. Moreover this simple approach outperforms three of the five baseline approaches listed in Table 5.

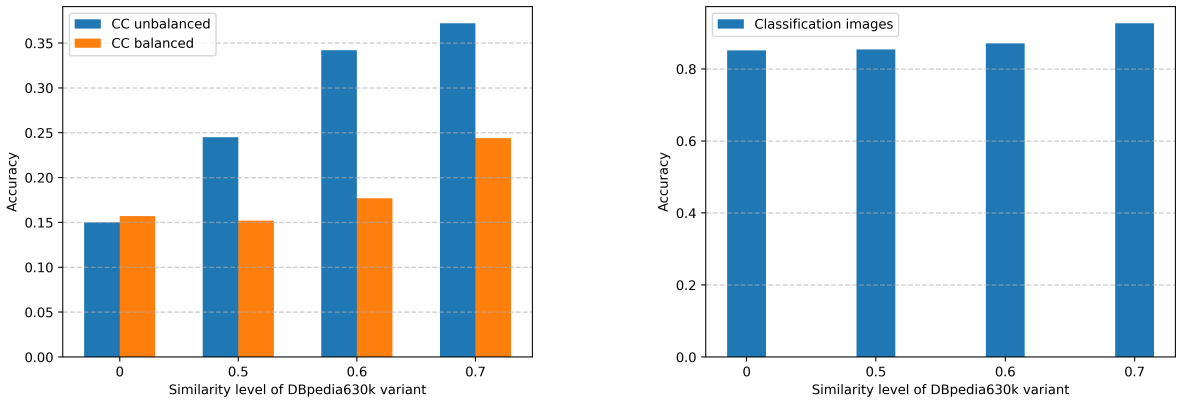
Split	similarity level	Ma-F_1	Mi-F_1	AC
DBpedia630k image		0.832	0.834	0.834
DBpedia630k unambiguous	0.5	0.855	0.854	0.854
DBpedia630k unambiguous	0.6	0.801	0.871	0.871
DBpedia630k unambiguous	0.7	0.829	0.927	0.927

Table 8: Comparison of image classification performance for balanced and unbalanced splits of DBpedia630k$_{\text{image}}$ and several similarity levels of DBpedia630k$_{\text{unambiguous}}$

Thus, SC-P as well as classification only using images achieve significantly better results than their combined classification. This leads to the conclusion that the inadequate classification performance is caused by the concatenation of the image features and structural features in connection with the on top deployed fully connected neural network. However, it is currently unclear which part is the cause of the problem. We assume that an ensemble learning method based on stacking that combines SC-P and classification based on images achieves significantly better results than CC and also outperforms the individual classifiers [39].

Impact of semantic similarity of the images. With higher semantic similarity, the classification accuracy increases by 58.4% when comparing DBpedia630k$_{\text{image}}$ with DBpedia630k$_{\text{unambiguous}}$ 0.7. For the balanced versions an improvement of 35.7% can be observed. For the already very good qualification performance solely based on the associated image, an improvement of 10.1% can be observed in the comparison of DBpedia630k$_{\text{image}}$

to DBpedia630k$_{\text{unambiguous}}$ 0.7 in Table 8. Meanwhile, a large number of entities have more than one associated image. Therefore, it should be considered to use the additional available information for entity type prediction as well, instead of simply assuming that the main Wikipedia image is the most accurate entity representation and therefore contains the most information. For this purpose, an attention mechanism could be integrated into the classification to select the most accurate image.



(a) (b)

Figure 10: Evolution of accuracy for SC over epochs for pretrained TransE embeddings (a) and own TransE embeddings (b)

Impact of balanced dataset. In contrast to SC, a difference of 78.4% on Ma-F$_1$ for all unbalanced data sets and 76.5% on Mi-F$_1$ for all balanced data sets can be measured on average for both CC and CC-P. Thus, the difference between Ma-F$_1$ and Mi-F$_1$ is almost constant and independent of the data set used. Overall, this means that the accuracy of the classification is strongly correlated with the predicted class, but is now independent of the datasets tested.

6 Conclusion

While existing entity type prediction models achieve state-of-the-art results based on given curated datasets, most of them use only structured data like textual description or category to predict its type. Since the main source of this datasets from DBpedia is Wikipedia where the information is extracted from by human created info boxes, this reduces prediction accuracy, creates incompleteness and noisy type information in real world applications.

In this thesis, we aim to perform classifications of the entities into their types using structural as well as image information. To achieve this, we investigate different possibilities on how to create KGEs and perform entity typing based on literals along with structural information.

Since the incorporation of images for entity type prediction is a new task, we first create two new datasets to compare results. Both datasets are subsets of DBpedia630k and contain 7 non-overlapping classes. The first dataset DBpedia630k$_{\text{image}}$ contains all entities with one associated image. In order to create DBpedia630k$_{\text{unambiguous}}$, we developed an approach to exclude entities whose associated image displays a scene from which a classifier would learn false predictions.

The developed approach concatenates image features with structural information and deploys a fully connected neural network on top of the combined feature vector.

We compare the approach with the help of the above-mentioned datasets with approaches that utilize structural information, literals, or a combination of both. We find out that our created structural baseline approach generates the best performance in entity type prediction, outperforming the combined classification approach. Furthermore, we investigate the influence of the associated images on the classification performance and find an improvement in the classification performance on DBpedia630k$_{\text{unambiguous}}$. From this we conclude that the combined classification can only learn the patterns to a limited extent at the moment and that the inadequate classification performance is caused by the concatenation of the image features and structural features in connection with the on top deployed fully connected neural network. Besides that, we find inherited problems in the dataset DBpedia630k due to which it is not suitable for learning KGEs.

7 Future Work

For this thesis we assumed that the main Wikipedia image is the most accurate entity representation and therefore contains the most information. However, this statement only applies to a part of the entities as shown in the course of this work when comparing entity typing on $\text{DBpedia630k}_{\text{image}}$ with $\text{DBpedia630k}_{\text{unambiguous}}$ 0.7. At the same time, the majority of entities with an image associated have more than one image associated. It should therefore be considered how all images can be used for entity type prediction and what improvements in classification can be achieved with them. One possible approach is to include an attention mechanism in the classification.

We performed experiments only on subsets of DBpedia630k, so experiments must be validated on other datasets as well. The already existing dataset *WN9-IMG* could be utilized for this task [33].

In this work it could be shown that images in combination with other KG information (here TransE embeddings) can be used for entity type prediction. However, the achieved performance is not sufficient and must be improved. In this context, the concatenation of image features and TransE embeddings in combination with a fully connected neural network has proven to be inadequate.

Entity-level representations are often uninformative for rare entities, so that using only entity embeddings for fine-grained entity type prediction is likely to produce poor results[34]. Indeed, deep neural networks can distinguish between a wide variety of types with high accuracy for the task of image classification. For example, ResNet-50 achieves an AC of 79.26% on ImageNet[15]. This suggests to use images together with other literals for fine-grained entity type prediction.

A Appendix

Link to Repository: `https://git.scc.kit.edu/usdrz/master-thesis`

References

[1] Transfer learning and fine-tuning. `https://www.tensorflow.org/guide/keras/transfer_learning`, 2022. Accessed: 2022-01-06.

[2] Understanding the evaluation. `https://pykeen.readthedocs.io/en/latest/tutorial/understanding_evaluation.html`, 2022. Accessed: 2022-01-04.

[3] Ali, M., Berrendorf, M., Hoyt, C. T., Vermue, L., Sharifzadeh, S., Tresp, V., and Lehmann, J. PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings. *Journal of Machine Learning Research 22*, 82 (2021), 1–6.

[4] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. Dbpedia: A nucleus for a web of open data. *The Semantic Web Lecture Notes in Computer Science* (2007), 722–735.

[5] Berrendorf, M., Faerman, E., Vermue, L., and Tresp, V. On the ambiguity of rank-based evaluation of entity alignment or link prediction methods, 2021.

[6] Biswas, R., Sofronova, R., Alam, M., Heist, N., Paulheim, H., and Sack, H. *Do Judge an Entity by Its Name! Entity Typing Using Language Models*. 07 2021, pp. 65–70.

[7] Biswas, R., Sofronova, R., Sack, H., and Alam, M. Cat2type: Wikipedia category embeddings for entity typing in knowledge graphs. In *Proceedings of the 11th on Knowledge Capture Conference* (New York, NY, USA, 2021), K-CAP '21, Association for Computing Machinery, p. 81–88.

[8] Biswas, R., Turker, R., Bakhshandegan Moghaddam, F., Koutraki, M., and Sack, H. Wikipedia infobox type prediction using embeddings.

[9] Bordes, A., Usunier, N., García-Durán, A., Weston, J., and Yakhnenko, O. Translating embeddings for modeling multi-relational data. In *NIPS* (2013), C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger, Eds., pp. 2787–2795.

[10] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S.,

Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

[11] Fensel, D., Şimşek, U., Angele, K., Huaman, E., Kärle, E., Panasiuk, O., Toma, I., Umbrich, J., and Wahler, A. *Introduction: What Is a Knowledge Graph?* Springer International Publishing, Cham, 2020, pp. 1–10.

[12] Fuhr, N. Some common mistakes in ir evaluation, and how they can be avoided. *SIGIR Forum 51*, 3 (Feb. 2018), 32–41.

[13] Gesese, G. A., Biswas, R., Alam, M., and Sack, H. A survey on knowledge graph embeddings with literals: Which model links better literal-ly?, 2020.

[14] Gupta, N., Singh, S., and Roth, D. Entity linking via joint encoding of types, descriptions, and context. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (2017).

[15] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition, 2015.

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

[17] Ji, S., Pan, S., Cambria, E., Marttinen, P., and Yu, P. S. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems* (2021), 1–21.

[18] Jin, H., Hou, L., Li, J., and Dong, T. Attributed and predictive entity embedding for fine-grained entity typing in knowledge bases. In *Proceedings of the 27th International Conference on Computational Linguistics* (Santa Fe, New Mexico, USA, Aug. 2018), Association for Computational Linguistics, pp. 282–292.

[19] Jin, H., Hou, L., Li, J., and Dong, T. Fine-grained entity typing via hierarchical multi graph convolutional networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (Hong Kong, China, Nov. 2019), Association for Computational Linguistics, pp. 4969–4978.

[20] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1* (Red Hook, NY, USA, 2012), NIPS'12, Curran Associates Inc., p. 1097–1105.

[21] Lin, T., Mausam, and Etzioni, O. No noun phrase left behind: Detecting and typing unlinkable entities. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning* (Jeju Island, Korea, July 2012), Association for Computational Linguistics, pp. 893–903.

[22] Lin, Y., Han, X., Xie, R., Liu, Z., and Sun, M. Knowledge representation learning: A quantitative review, 2018.

[23] Lin, Y., Liu, Z., Sun, M., Liu, Y., and Zhu, X. Learning entity and relation embeddings for knowledge graph completion. In *AAAI* (2015).

[24] Masters, D., and Luschi, C. Revisiting small batch training for deep neural networks, 2018.

[25] Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space, 2013.

[26] Paulheim, H., and Bizer, C. Type inference on noisy rdf data. *Advanced Information Systems Engineering Lecture Notes in Computer Science* (2013), 510–525.

[27] Prechelt, L. *Early Stopping - But When?* Springer Berlin Heidelberg, Berlin, Heidelberg, 1998, pp. 55–69.

[28] Rossi, A., Barbosa, D., Firmani, D., Matinata, A., and Merialdo, P. Knowledge graph embedding for link prediction: A comparative analysis. *ACM Trans. Knowl. Discov. Data 15*, 2 (jan 2021).

[29] Sakai, T. On fuhr's guideline for ir evaluation. *SIGIR Forum 54*, 1 (Feb. 2021).

[30] Vrandečić, D., and Krötzsch, M. Wikidata. *Communications of the ACM 57*, 10 (2014), 78–85.

[31] Wang, Z., Zhang, J., Feng, J., and Chen, Z. Knowledge graph embedding by translating on hyperplanes. In *AAAI* (2014), C. E. Brodley and P. Stone, Eds., AAAI Press, pp. 1112–1119.

[32] Wu, F., and Weld, D. S. Autonomously semantifying wikipedia. *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management - CIKM 07* (2007).

[33] Xie, R., Liu, Z., Luan, H., and Sun, M. Image-embodied knowledge representation learning, 2017.

[34] Yaghoobzadeh, Y., and Schütze, H. Multi-level representations for fine-grained typing of knowledge base entities, 2017.

[35] Yao, L., Riedel, S., and McCallum, A. Collective cross-document relation extraction without labelled data. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing* (Cambridge, MA, Oct. 2010), Association for Computational Linguistics, pp. 1013–1023.

[36] Yogatama, D., Gillick, D., and Lazic, N. Embedding methods for fine grained entity type classification. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)* (2015).

[37] Zequn Sun, Chengming Wang, W. H. M. C. J. D. W. Z. Y. Q. Knowledge graph alignment network with gated multi-hop neighborhood aggregation. In *AAAI* (2020), pp. 222–229.

[38] Zhang, X., Zhao, J., and LeCun, Y. Character-level convolutional networks for text classification, 2016.

[39] Zhou, Z.-H. *Ensemble Learning*. Springer Singapore, Singapore, 2021, pp. 181–210.

Assertion

Ich versichere wahrheitsgemäß, die Arbeit selbstständig verfasst, alle benutzten Quellen und Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde, sowie die Satzung des KIT zur Sicherung guter wissenschaftlicher Praxis in der jeweils geltenden Fassung beachtet zu haben.

This work was performed on the computational resource bwUniCluster funded by the Ministry of Science, Research and the Arts Baden-Württemberg and the Universities of the State of Baden-Württemberg, Germany, within the framework program bwHPC.

Karlsruhe, January 14, 2022 Patrick Eisele