Modelling and Factorizing Large-Scale Knowledge Graph (DBPedia) for Fine-Grained Entity Type Inference

A. B. M. Moniruzzaman(✉)

University of Technology Sydney, Sydney, Australia
abm.moniruzzaman@student.uts.edu.au

Abstract. Recent years have witnessed a rapid growth of knowledge graphs (KGs) such as Freebase, DBpedia, or YAGO. These KGs store billions of facts about real-world entities (e.g. people, places, and things) in the form of triples. KGs are playing an increasingly important role in enhancing the intelligence of Web and enterprise search and in supporting information integration and retrieval. Linked Open Data (LOD) cloud interlinks KGs and other data sources using the W3C Resource Description Framework (RDF) and makes accessible on web querying. DBpedia, a large-scale KG extracted from Wikipedia has become one of the central interlinking hubs in the LOD cloud. Despite these impressive advances, there are still major limitations regarding coverage with missing information, such as type, properties, and relations. Defining fine-grained types of entities in KG allows Web search queries with a well-defined result sets. Our aim is to automatically identify entities to be semantically interpretable by having fine-grained types in DBpedia. This paper embeddings entire DBpedia, and applies a new approach based on a tensor model for fine-grained entity type inference. We demonstrate the benefits of our task in the context of fine-grained entity type inference applying on DBpedia, and by producing a large number of resources in different fine-grained entity types for connecting them to DBpedia type classes.

Keywords: Knowledge graph · DBPedia embedding · Type inference · Tensor factorization · Semantic web search

1 Introduction

Knowledge graphs (KGs), i.e., graph-based knowledge-bases, store information about real-world objects (e.g. people, places, and things) in the form of RDF triples (i.e. (subject, predicate, object)). Recent years have witnessed a rapid growth of KGs driven by academic and commercial efforts, such as Yago [26, 49], Freebase [13], DBpedia [10,36], NELL [15], Google's Knowledge Graph, Microsoft's Satori, Probase [3], and Google Knowledge Vault [25]. These KGs have reached an impressive size, for instance, DBPedia a large-scale KG extracted

M. Qiao et al. (Eds.): ADC 2021, LNCS 12610, pp. 204–219, 2021.
https://doi.org/10.1007/978-3-030-69377-0_17

from Wikipedia contains many millions of entities, organized in hundreds to hundred thousands of semantic classes, and billions of relational facts (triples) involving a large variety of predicates (relation types) between entities.

KGs are playing an increasingly important role in enhancing the intelligence of Web and enterprise search and in supporting information integration and retrieval. For Example, Freebase KG powered Google Knowledge Graph that supports Google's web search, or Microsoft's Satori that supports Bing by providing richer data for Entity Pane, Carousel, and Facts Across Segments in the search panel. Additionally, KGs are becoming important resources for different Artificial Intelligence (AI) and Natural Language Processing (NLP) applications, such as Question-Answering [11,22], Query Understanding through Knowledge-Based Conceptualization [12], and Short Sentence Texts Understanding [51,53] and Conceptualization using a probabilistic Knowledge bases. Despite these impressive advances, there are still major limitations regarding coverage and freshness, these KGs are incomplete with missing information, such as type, properties, and relations [18,42,45,48,52,57].

Types in KG are used to express the concept of classes. According to KG idiomatic usage, a KG object "has X, Y, Z types" is equivalent to an object "is a member of the X, Y, Z classes". In the case of Tom Hanks[1] , the KG object for Hanks would have the types *person* and *Actor* to indicate that the object is a member of the Persons and Actors. However, an entity is usually not associated to a limited set of generic types (Person, Location, and Organization) in KGs but rather to a set of more specific (fine-grained) types. Evidence suggests that performance of Web search queries (in case of exploring lists and collections) can be dramatically improved by defining large numbers of these fine-grained entity types in KG. Untyped entities and entities with incomplete set of types are a common problem in Semantic Web KGs [42,45]. For example, one can find by Web search queries the fact that Tom Hanks is a person, an actor, and a person from California, USA. All these types are correct but some may be too general to be interesting (e.g., person, actor), while other set of more specific (fine-grained) types may be interesting but may be identified by web searching (such as, list of films in any specific film genre of Tom Hanks film).

The Semantic Web's Linked Open Data (LOD) cloud interlinks KGs and other data sources using the W3C Resource Description Framework (RDF) [4] and makes accessible on web querying through SPARQL. This LOD cloud is growing rapidly. At the time of this writing, the LOD cloud contains 1,234 datasets with 16,136 links[2]. Several hundred data sets on the Web publish RDF links pointing to DBpedia themselves and thus make DBpedia one of the central interlinking hubs in the Linked Open Data (LOD) cloud [ref]. DBpedia ontology forms a subsumption hierarchy consisting of a standard limited (760) set of classes (types), and recent version of DBpedia has been incorporated a large number (570,276) of YAGO types (mostly file-grained types) by linking YAGO types taxonomy.

[1] http://dbpedia.org/resource/Tom_Hanks.
[2] https://lod-cloud.net/.

Although couple of approaches, such as SDType [47] a heuristic approach and, SNCN [40] a hierarchical classification approach have been applied on DBpedia for type inference, these approaches are successful in extracting commonly used coarse types, such as *'Person', 'Artist', 'Movie'*, or '*Actor*'. In DBpedia, a vast amount of entities missing of fine-grained types (depth of four to six in type hierarchy). For instance, (at the time of this writing) 18805 number of entities listed as American Film class (fine-grained type) within 94996 number of entities from movie class (type) in DBpedia [footnote reference]. However, according to current DBpedia online, only 83 entities (from actor class type) as identified as American Film Actor which evident that 98% of entities missing of this fine-grained entity type.

In recent years, representation learning in form of latent variable methods [14,23,27,31,32,37–39,41,43,53] have increasingly been gained attention for the statistical modeling of KGs, learning latent embeddings for entities and relation-types from the data that can then be used as representations of their semantics. These models have successfully been applied on FB15K [14] dataset, is a subset of Freebase KG which has been commonly used to evaluate various KG completion task, and showing promising results in tasks related to link predictions. DBpedia data are represented in the form of RDF [4] triples <*subject, predicate, object*>, where the subject and object are entities and the predicate is the relation type. The representation learning from DBpedia (large-scale relational data) has therefore become emerged especially for fine-grained entity type inference.

Modeling and fatorizing entire DBpedia is not a trivial task, as DBpedia is very large scale (with millions of entities with billions of facts), and contains heterogeneous information where mappings are created via a world-wide crowd-sourcing effort to extract contents from the information created in various Wikimedia projects. Such information includes infobox templates, categorisation information, images, geo-coordinates, links to external web pages, disambiguation pages, redirects between pages, and links across different language editions of Wikipedia. Besides, A large number of fine-grained types (sub-class) from YAGO type taxonomy are not systematically consistent in the DBpedia ontology. Furthermore, (in the depth of five and six) are not coherently defined in context to sub-class types hierarchy. In addition, A good number of types redundant in DBpedia, such as 5 (five) different types exprese as Actor type class (in Table 3). Although many of these types mapped to DBpedia types using owl:equivalentClass, this leads inconsistency and miss proper fine-grained typing of entities in DBPedia. In this paper, we focus on the extraction of entity fine-grained types, i.e., assigning fine-grained types to – or typing – entities in DBpedia. The major two folds contributions of this paper are as follows:

1. This paper models entire DBpedia with a approach based on a tensor model that learns latent embeddings for entities, relation-types and properties to automatically identify entities to be semantically interpretable by having fine-grained types for connecting them to DBpedia classes. The key idea behind of modelling and applying factorization method is that it uses three-dimensional arrays (tensor) to represent DBpedia and obtain probabilistic likelihoods of

type-relations existing between entities (objects) by applying tensor factorization (TF) techniques on DBpedia.
2. This paper proposes TypePathSample algorithm an efficient way to reduce the computer complexity for the large-size of the dataset, yet operate on a representative subset there of is to use KG partition. This will capture (observing) rich interactions of all the entities of fine-grained types in populating tensor according to fine-grained type entity constraint. This will transform as unobserve from observing interactions of all the entities of coarse-grained types according to fine-grained type entity constraint. Applying this algorithm to DBpedia, we generate multiple samples of the coupled data with domain and type, we fit a Coupled Matrix and Tensor Factorization (CMTF) model to each sample and propose to simultaneous factorization by parallelization.
3. We demonstrate the benefits of this task in the context of fine-grained entity type inference with experiments on a large-scale KG by producing $1.3 \times 10^5$of resources in different fine-grained entity types for person entities from one sample in DBpedia.

This paper is structured as follows. The next section contains related work. In Sect. 2 explain BDpedia Knowledge Graph; modeling and factorizing DBpedia with tensor factorization model. In Sect. 3 In Sect. 3.1 we introduce our approach. In Sect. 3 we describe our experiments. We conclude in Sect. 5.

2 DBpedia Knowledge Graph

DBpedia [1,10,36], a large-scale KG extracted from Wikipedia currently describes 6.6M entities, and 5.5M resources are classified in a consistent ontology, such as 1.5M persons, 840K places, 496K works. Altogether the DBpedia 2016-10 release (see footnote 2) consists of 13 billion pieces of information (RDF triples). Each resource in the DBpedia data set is denoted by a de-referenceable Internationalized Resource Identifier (IRI)- or the Uniform Resource Identifier (URI)-based reference of the form http://dbpedia.org/resource/Name. URI uniquely identifying each entity in Semantic Web KGs. For instance, en entity *Tom Hanks* can be found in DBpedia[3], and in Wikipedia[4]. Every DBpedia entity name resolves to a description-oriented Web document (or Web resource).

DBpedia is served on the web in three forms: First, it is provided in the form of downloadable data sets where each data set contains the results of one of the extractors; second, DBpedia is served via a public SPARQL endpoint and, third, it provides dereferencable URIs according to the Linked Data principles. DBpedia datasets in N3/TURTLE serialisation, and each triple is represented as the form <*head entity, relation, tail entity*>.

The DBpedia data set can be accessed online via a SPARQL query endpoint[5] and as Linked Data[6]. All list of types (coarse-grained or fine-grained) of an

[3] http://dbpedia.org/resource/Name.
[4] http://en.wikipedia.org/wiki/Name.
[5] https://dbpedia.org/sparql.
[6] http://mappings.dbpedia.org/server/ontology/classes/.

Table 1. Different RDF triple relations in DBpedia

a. Example of Type relation in a RDF triple	
Subject/head entity	<dbpedia.org/resource/Tom_Hanks>
Predicate/relation	<www.w3.org/1999/02/22-rdf-syntax-ns#type>
Object/tail entity	<dbpedia.org/ontology/Actor>

b. Example of SPARQL query for finding types of an entity
select distinct ?Subject where ?Subject ?Predicate ?Object filter (?Object = <dbpedia.org/ontology/Actor> && ?Predicate = <www.w3.org/1999/02/22-rdf-syntax-ns#type>)

entity, or all list of entities for any types can be obtained by SPARQL search on DBpedia (see in Table 1(b));

2.1 Modeling DBpedia

From modelling perspective, tensor representations are appealing to KG because they provide an elegant way to represent multiple RDF triples. The interpretation of DBpedia can be interpreted as a tensor, where first mode of a tensor therefore models the occurrences of all entities as a subject, the second mode models the occurrences of all entities as an object, and the third mode models the different relations, as illustrated in Fig. 2. Entities in DBpedia can be subjects or objects in multiple relations (RDF triples) depending on relation types. For instance, in a relation *<Tom Hanks, starring, Inferno>*, and in another relation *<Inferno, rdf-type, Movie>*, where entity *Inferno* is a subject in one relation and an object in another relation.

The DBpedia ontology consists of 760 classes (such as Thing, Person or Movie) which form a subsumption hierarchy. Figure 1 depicts a part of the DBpedia ontology, indicating the relations from the top class of the DBpedia ontology, i.e. the classes with the highest number of instances. The complete DBpedia ontology can be browsed online (see footnote 3). The file Instance Types (see footnote 4) contains triples of the form <object> <rdf:type> <class> from the mapping based extraction. We can therefore easily model a class as an object in a rdf triple and populate to a tensor. However, a large number of fine-grained types (sub-class) from YAGO type taxonomy are not systematically consistent in the DBpedia ontology. Furthermore, (in the depth of five and six) are not coherently defined in context to sub-class types hierarchy. To address this issue we extended DBpedia type class hierarchy with YAGO type classes for each of DBpedia ontology class using SPARQL [9] query with <http://www.w3.org/2000/01/rdf-schema#subClassOf>. For instance, all 2502 [footnote reference] sub classes of DBpedia ontology class for Actor can be derived by this SPARQL query.

Since DBpedia data are high-dimensional but very sparse, we approach the problem of learning positive examples only from DBpedia, by assuming that missing triples are very likely not true. We use the *weighted-tensor* interpretation scheme to effectively model DBPedia for constructing the tensor. The weighting tensor with different values for KG and text data is a particularly important component of our model. Without applying weight in tensor construction, the objective function would place equal importance on fitting both observed and unobserved values. Since DBPedia KB is very large scale, and constructed tensors are therefore often very sparse, this will result in fitting a large number of unobserved data and uncertainty in the observations. The weighting tensor prevents this from happening by emphasizing only the observed (or certain) entries. we approach the problem of learning positive examples only from DBpedia, by assuming that missing triples are very likely not true. DBpedia consisting of e entities and r different relations can then be represented in form of a tensor $X \in R^{e \times e \times r}$ with entities

$$X_{i,j,k} = \begin{cases} 1, & \text{if the relationship } r_k(e_i; e_j) \text{ exists in DBpedia.} \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

The values ('2' or '1') and ('0') of X_{ijk} come from tensor model are regarded as observed and un-observed data respectively, the representations of DBpedia are therefore becomes possible for tensor factorization.

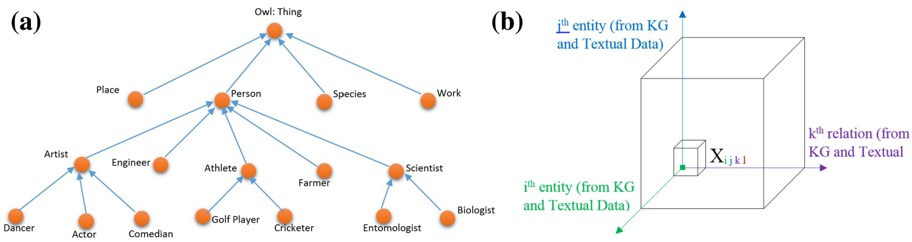

Fig. 1. (a) A part of DBPedia type hierarchy. (b) Representation of a third-order tensor with RDF triples.

The schema information for RDF, which provides the concepts rdfs:domain and rdfs:range for a semantic description of the entities contained in the KG. These concepts are used to represent type-constraints on relation-types by defining the classes or types of entities which they should relate, where the domain covers the subject entity classes and the range the object entity classes in a RDF-Triple. DBPedia domain information can be found in the property of a relation by SPARQL [5] query with http://www.w3.org/1999/02/22-rdf-syntax-ns#Property. For example, a set of actor typed entities can be a set of author typed where these entities are involved with different domains. Ignoring these information (inter-domains collaboration activities of entities) may effect on latent features learning by factorization.

Leveraging Domain Knowledge. Most KGs (such as DBPedia, Freebase, or Yago) store facts about real-world objects covering only numbers of specific domains (e.g. "*Movie*", "*Book*", or "*Place*"). For instance, types in KG such as *actor, film, director,* or *producer* and fine-grained types such as *filmActor, TVActor, regularActor, guestActor, executiveDirector, AssistantProducer* are in "*film*" domain. Given the importance the fine-grained inference task in KG, typed entities (objects) for given fine-grained types in one domain (such as "*film*" domain) are less likely to be entities in other domains (such as "*book*", or "*place*"). For instance, inferring entities for fine-grained types (such as regularActor, guestActor) would be a typed in Actor, those entities generally are in same domain in KG.

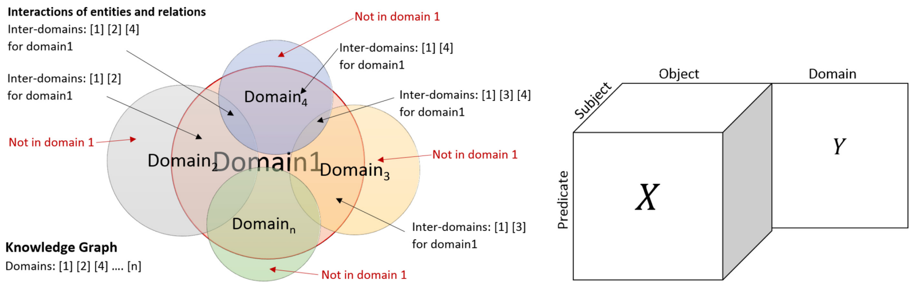

Fig. 2. (left) Representation of a KG with different domains. (right) Modelling domain knowledge in a tensor.

The collaborative activities between the entities in "*film*" domain are therefore higher importance for fine-grained type inference in this domain.

Partitioning via Type Path Hierarchy. Introduce *Fine-grained type entity constraint* for Knowledge Graph: This fine-grained type entity constraint will distinguish and separate types in KG into two sets of types – (a) Coarse-grained types and (b) Fine-grained types. Developing an effective algorithm for type-class path partitioning is an efficient way to reduce the computer complexity for the large-size of the dataset, yet operate on a representative subset thereof is to use KG partition. This will capture (observing) rich interactions of all the entities of fine-grained types in populating tensor according to 'fine-grained type entity constraint'. This will transform as unobserved from observing interactions of all the entities of coarse-grained types according to 'fine-grained type entity constraint'.

Proposed Algorithm:

Input: *Knowledge Graph* $G = (T, E, R)$*; where set* T *is a set of 5th level Types in KG,* $T = \{t_1, ..., t_{|t|}\}$*; set* E *is a set of entities (objects),* $E = \{e_1, ..., e_{|e|}\}$*, and* E_s *as subject set of entities which occur as subject in relation links, where* $E_s \in E$*; and and* E_o *as object set of entities which occur as object in relation links where* $\{E_o \in E\}$ *and, the set* R *is a set of relations (Predicates) between entities,* $R = \{r_1, ..., r_{|r|}\}$*.*

Output: set ∇T_c is a list of triples $<< (E_s)_i, (R)_i, (E_o)_i >>$ for each $d_i \in T$ (Types)

Start

```
function TypePathSample (T, E, R)
    d ← type
    for each r_i ∈ R do
        for each r_i ∈ d do
            source (e_s) : R → E_s     ▷ return source(e)
            target (e_o) : R → E_o     ▷ return target(e)
            e_d ← (e_s ∪ e_o)
            while e_i ∈ e_d OR r_i ∈ d do
                T_d ← SELECT-TRIPLES {(e_s)_i, (r)_i, (e_o)_i}
            end while
            ∇T_c ← T_c
        end for
    end for
end function
```

Applying this algorithm to DBpedia, we generate multiple partitions of data samples of the coupled data with domain and type, we fit a CMTF model to each sample and propose to simultaneous factorization by parallelization.

2.2 Factorizing DBpedia

Classical *Tensor Factorization Models (TFM)* such as *Singular Value Decomposition (SVD)* [20,28], *CANDECOMP/PARAFAC Decomposition (CPD)* [16,30] can be regarded as *Latent Factor Models (LFM)* for multi-relational data [34,43]. Since DBpedia data is multi-relational, the tensor entries from them can be therefore factorized in order to directly comparable by transforming subject entities, relations, object entities and domains to the same latent factor space. The global dependencies are captured during learning the latent representations of each of these dimensions of tensor. The latent ternary correlation subject, object, predicate and domain can be inferred after factorizing the tensor model. We use CP based *Coupled Matrix and Tensor Factorization (CMTF)* [8,9] for deriving the latent relationships between dimensions of the tensor model. After latent factors generation via tensor factorization, we therefore follow *tensor reconstruction* process to reveal new entries that are inferred from the latent factors.

As illustrated Fig. 2, the *CMTF* model, CP factorizes tensor $\mathbf{X} \in \mathbb{R}^{S \times O \times P}$, and a matrix $\mathbf{Y} \in \mathbb{R}^{P \times D}$, can be formulated as

$$f(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{V}) = \|\mathcal{X} - \|\mathbf{A}, \mathbf{B}, \mathbf{C}\|\|^{\mathbf{2}} + \left\|\mathbf{Y} - \mathbf{A}\mathbf{V}^{\mathbf{T}}\right\| \tag{2}$$

where $\mathcal{X}$ is factorized using a CP model on each mode-n matricization and results in four latent factors matrices, $\mathbf{A} \in \mathbb{R}^{S \times R}$, $\mathbf{B} \in \mathbb{R}^{O \times R}$ and $\mathbf{C} \in \mathbb{R}^{P \times R}$ corresponding to the each dimensions of tensor $\mathcal{X}$, $\mathbf{V} \in \mathbb{R}^{D \times R}$ are the factor matrices extracted from matrix Y through matrix factorization.

Probabilistic Inference. Since DBPedia is multi-relational data, the similarity of entities is therefore determined by the similarity of their relationships, following the intuition that "if two objects are in the same relation to the same object, this is evidence that they may be the same object". The collaborative activities of entities as subjects $\mathbf{A_s} \in \mathbb{R}^{S \times P}$ and objects $\mathbf{A_o} \in \mathbb{R}^{O \times P}$ in relations in a domain can be modelled by the entity matrix $\hat{\mathbf{A}}$, where $\hat{\mathbf{A}}$, is QR matrix factorization [17,29] of $\sum(A_s + A_o)$. For each domain the latent space $\hat{\mathbf{A}}$ therefore reflects the similarity of entities in the relational domain. The type or fine-grained type classes set $C_e = \{t_1, t_2, t_3, ...t_n\}$ where C_e is a set of Types in one KG. A list of type or fine-grained type classes that are considered for given fine-grained type. For each fine-grained type in C_e the candidate entities set, $\hat{E}_t = \{\hat{e}_1, \hat{e}_2, \hat{e}_3, ...\hat{e}_n\}$ where E_t is a set of typed entities in one KG.

We use the Bayes' theorem [24,55] for predicting the class candidate entity E_t that have the highest posterior probability given C_e, $p(C_e|E_t)$. The posterior probability is utilized to calculate the preference probability of an entity e to be fine-grained typed t in C_e type classes by observing current type classes of entity e, and latent similarity of entity e to fine-grained typed entity. The conditional probability can be formulated as:

$$p(C_e|E_t) = \frac{p(E_t|C_e)p(C_e)}{p(E_t)} \tag{3}$$

where *prior probability* $p(C_e)$ is the prior distributions of parameter set C_e in a single domain before E_t is observed, that is relative frequency with which observations from that class occur in a population. Generally, prior probability for fine-grained type classes are lower compared to top level type classes in KG. $p(C_e|E_t)$ is the joint probability of observing type class preference set C_e given E_t, and entity similarity preference given fine-grained type t. Using the assumption of multinomial event model distribution for the Naive Bayes classifier, the posterior probability p_{e_n,t_r} for fine-grained type t_e with fine-grained type class C_e for candidate entity e_n, an instance of E_t, is obtained by multiplying the *prior probability* of t_e, $P(C_e = t_e)$, with the probability of preference candidate entity e_n, an instance of E_t, given t_e, $P(e_n|C_e = t_e)$:

$$p_{e_n,t_e} = p(t_e|C_e) = \sum_{t=1}^{|C_e|} P(e_n|C_e = t_e) \prod_{\hat{e}=1}^{|\hat{E}_e|} P(\hat{E}_e|C_e) \tag{4}$$

where, $P(\hat{E}_e|C_e)$ is probability of likelihood for t_e in C_e, is derived from the entities set, $\hat{E}_t = \{\hat{e}_1, \hat{e}_2, \hat{e}_3, ...\hat{e}_n\}$ where values from reconstructed tensor $\hat{X}$, and entity similarity values from $\hat{A}$ are used.

3 Experiments

3.1 Evaluation and Apply on DBpedia and Freebase

Datasets. We apply and demonstrate the benefits of our task in the context of fine-grained entity type inference applying on on DBpedia 2016–10 release dataset. The DBpedia 2016-10 release dataset[7] published on the year 2017, at the time of this writing this release is latest full version of DBpedia KG. To make ready DBpedia dataset to apply tensor based model, we first A simple java program is used to transfer textual based triples into readable format for applying tensor based model. Prior to apply on DBpedia, we evaluate our approach on Freebase FB15K dataset[8]; FB15K-237 Knowledge Base Completion Dataset[9] and DBpedia 2016-10 release dataset (see Table 2). The FB15K (Bordes et al. 2013), is a subset of Freebase which has been commonly used to evaluate various KG completion models [14,31,32,37,38,53,54]. In the FB15K-237 Knowledge Base Completion Dataset, the triples (entity- textual-entity) are derived from 200 million sentences from the ClueWeb12 corpus coupled with Freebase entity. There are around 3.9 million text descriptions corresponding to the relation types in Freebase. The FB15K-237 dataset has been used in [50,51,56] for embedding representations for textual relations with Freebase entity mention annotations.

Table 2. Datesets used in the experiments.

DBpedia	
Dataset	DBpedia 2016-10 release
# Entities	5.72 million
# Relations as object properties	1,105
# Relations as datatype properties	1,622
# Relations as specialised datatype properties	132
# Entity class types	760
# YAGO class types	570,276
# RDF triples from DBpedia 2016-10 release	494 million
# RDF triples from online DBpedia by SPARQL	1.2 million

Freebase			
Datasets	**# Entities**	**# Relations**	**# Triples**
FB15K	14,951	1,345	486,641
FB15K-327	14,951	2,766,477	3,977,677

Implementation for Experiment. For implementation, we use *tensor-toolbox* [6] and *poblano-toolbox* [2] in Matlab. We construct a 3th order tensor where the tensor size (5.72M × 5.72M × 27K) in 52 different domain with 494M entries from DBpedia. First and second orders of this tensor are defined as *Entity* and third order as *Relation*. We fit tensor factorization based model [41] to this tensor where domain is coupled with relation in tensor; and apply TypePathSample to make samples of the model. Each sample model is density reduced tensor with same size. For instance, in first sample all other samples tensor entries are transformed to unobserved. For evaluation we apply *weighted tensor scheme* in constructing tensor from Freebase where the tensor size (14951 × 14951 × 2,767,822) with 486541 entries from KG (FB15K); and 4460819 entries from textual dataset (FB15K-237). We then apply *domain-relevance weighted tensor (DrWT)* to construct 4th order tensor with domain entries, where the tensor size

[7] https://wiki.dbpedia.org/develop/datasets/dbpedia-version-2016-10.

[8] https://developers.google.com/freebase/.

[9] FB15K-237 Knowledge Base Completion Dataset https://www.microsoft.com/en-us/download/details.aspx?id=5231.

Table 3. Fine-grained Entity Types Inference on DBpedia

Fine-grained types	# Entities present in DBpedia	# Entities new identified
http://dbpedia.org/class/yago/WikicatAmericanFilmActors	83	9787
http://dbpedia.org/class/yago/WikicatTelevisionActors	21	5225
http://dbpedia.org/class/yago/WikicatFilmDirectors	291	5310
http://dbpedia.org/class/yago/WikicatFilmsByAmericanDirectors	101	18805
http://dbpedia.org/class/yago/WikicatFilmProducers	2196	1695
http://dbpedia.org/class/yago/WikicatActionFilms	107	235
http://dbpedia.org/class/yago/WikicatAdventureFilms	110	582
http://dbpedia.org/class/yago/WikicatComedyFilms	124	3524
http://dbpedia.org/class/yago/WikicatHorrorFilms	116	1194
http://dbpedia.org/class/yago/WikicatDramaFilms	189	2360
http://dbpedia.org/class/yago/WikicatCrimeFilms	124	980
http://dbpedia.org/class/yago/WikicatMysteryFilms	118	511
http://dbpedia.org/class/yago/WikicatMusicalFilms	125	374
http://dbpedia.org/class/yago/WikicatFantasyFilms	116	718
http://dbpedia.org/class/yago/WikicatScienceFictionFilms	117	276
http://dbpedia.org/class/yago/WikicatRomanceFilms	109	492
http://dbpedia.org/class/yago/WikicatThrillerFilms	126	560
http://dbpedia.org/class/yago/WikicatAnimatedFilms	104	492
http://dbpedia.org/class/yago/WikicatArtFilmss	192	205
http://dbpedia.org/class/yago/WikicatRomanticComedyFilms	98	256
http://dbpedia.org/class/yago/WikicatShortFilms	204	245
http://dbpedia.org/class/yago/WikicatDocumentaryFilms	109	2105
http://dbpedia.org/class/yago/WikicatWarFilms	114	178
http://dbpedia.org/class/yago/WikicatPoliticalFilms	24	178
http://dbpedia.org/class/yago/WikicatTelevisionFilms	227	1652
http://dbpedia.org/class/yago/WikicatTelevisionActors	5200	248
http://dbpedia.org/class/yago/WikicatAmericanFilmActresses	7278	495
http://dbpedia.org/class/yago/WikicatVoiceActors	708	451
http://dbpedia.org/class/yago/WikicatChildActors	911	98
http://dbpedia.org/class/yago/WikicatMusicalTheatreActors	56	61
http://dbpedia.org/class/yago/WikicatVideoGameActors	17	98
http://dbpedia.org/class/yago/WikicatStageActors	209	2119
http://dbpedia.org/class/yago/WikicatAmericanActors	8823	964

(14951 × 14951 × 2,767,822 × 52). We use CP [16,30] based *4th-order Tensor Factorization* for the latent factor generation, and use CP-ALS algorithm [19,21,33] for computing tensor factorization. Since domain information is not depended in one other dimension of the tensor, we use 4th order tensor factorization instead of using *Coupled Matrix Tensor Factorization (CMTF)* [8,9]. We also apply *non-negativity constrain* [35] for effectively interpreting factor components from tensor factorization.

We demonstrate the benefits of our approach in the context of fine-grained entity type inference with experiments on a large-scale KG DBpedia by producing a large number of resources indifferent fine-grained entity types for connecting them to DBpedia type classes. Some new resources unidentified in Film domain in DBpedia are listed in Table 3. In Table 3, new identified entities for fine grained types http://dbpedia.org/class/yago/WikicatAmericanFilmActors http://dbpedia.org/class/yago/WikicatTelevisionActors and http://dbpedia.org/class/yago/WikicatFilmDirectors are 9787; 5225 and 5225 respectfully.

4 Related Work

RESCAL [43] is the state-of-the-art method for link prediction and type inference in KGs that has been used for type inference on YAGO [7] entire KG [44]. This approach defines statistical models for modeling tensor representation of binary relational data on KGs and explains triples via pairwise interactions of latent features Though, YAGO one of the large scale KB in LOD cloud is factorized with RESCAL and able to predict the likelihood of any of the 4.3 x $10^1 4$ possible triples in the YAGO 2 core ontology [44]; DBpedia is not yet modelled with such latent factor model. Paulheim, H. and Bizer, C. proposed SDType algorithm [46,47] a probabilistic method for predicting missing type of entities in DBpedia. Their approach which is based on conditional probabilities, such that predicts approximate types of entities by considering the observed types of subjects and objects in a relation. For each relation, this approach uses the statistical distribution of types in DBpedia based on the property of the subject and object for assuming the types of entities. This approach heuristically suggest that an entity should have certain types if it has certain relations connected to other entities. For example, a statement like <Tom Hanks, starring, Inferno>, this may give result that Tom Hanks is an actor [47].

Though, SDType algorithm has been applied to DBpedia and produced meaningful results in predicting entities for generic (coarse-grained type) classes, (such as actor, writer, or movie); in context to more specific (fine-grained types) classes, (such as American film actor, science-fiction writer, or thriller movie) this heuristic approach is not capable to produce meaningful results. This is because, SDType uses relations between entities as indicators for types, and relations between entities in DBpedia are coherently specific to generic entity types whereas too general to more specific types. Considering previous example, starring relations may be indicators for actor (generic type), however this is too general for all sub-class types of actor (such as film actor, voice actor or television actor) to indicate or distinguish. One recent state-of-the-art fine-grained type entity inference approach [41] which mainly focus on the fine-grained type entity inference task in the KGs via tensor factorization and probabilistic inference methods. First, it looks into the scope of utilizing embedded knowledge inside the KGs that will be efficiently captured to the fine-grained type entity inference task. Besides, it explores the advantages of using linked entity supplementary information to this task by effective incorporation of additional data to KGs. Furthermore, the use of similarity of entities in the KGs is also considered to the fine-grained type entity inference task. Experimental results show that this novel approach has achieved a significant improvement in the accuracy of fine-grained types entity inference in a KG. We models entire DBpedia following this tensor model based approach that learns latent embeddings for entities, relation-types and properties to automatically identify entities to be semantically interpretable by having fine-grained types for connecting them to DBpedia classes.

5 Conclusion

The performance of Web search queries (in case of exploring lists and collections) can be dramatically improved by defining large numbers of these fine-grained entity types in KG. This paper models entire DBpedia with a approach based on a tensor model that learns latent embeddings for entities, relation-types and properties to automatically identify entities to be semantically interpretable by having fine-grained types for connecting them to DBpedia classes. The key idea behind of modelling and applying factorization method is that it uses three-dimensional arrays (tensor) to represent DBpedia and obtain probabilistic likelihoods of type-relations existing between entities (objects) by applying tensor factorization (TF) techniques on DBpedia. This paper proposes a novel way to reduce the computer complexity for the large-size of the dataset, yet operate on a representative subset there of is to use KG partition. Applying this algorithm to DBpedia, we generate multiple samples of the coupled data with domain and type, we fit a Coupled Matrix and Tensor Factorization (CMTF) model to each sample and propose to simultaneous factorization by parallelization. We demonstrate the benefits of this task in the context of fine-grained entity type inference with experiments on a large-scale KG by producing 1.3×10^5 of resources in different fine-grained entity types for person entities from one sample in DBpedia.

References

1. “DBPedia” Public Semantic Knowledge Graph. http://wiki.dbpedia.org/. Accessed 08 Aug 2017
2. “Poblano Toolbox” Poblano - Sandia software - Sandia National Laboratories. https://software.sandia.gov/trac/poblano/. Accessed 29 Aug 2018
3. “Probase” Knowledge Base. https://www.microsoft.com/en-us/research/project/probase/. Accessed 29 Sept 2018
4. “RDF” Resource Description Framework. https://www.w3.org/RDF/. Accessed 29 Sept 2018
5. “SPRQL” Query Language for RDF. https://www.w3.org/TR/rdf-sparql-query/. Accessed 29 Sept 2018
6. “Tensor Toolbox” MATLAB Tensor Toolbox. https://www.sandia.gov/~tgkolda/TensorToolbox/index-2.6.html. Accessed 09 Aug 2018
7. “YAGO” semantic knowledge base. http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/. Accessed 09 Aug 2017
8. Acar, E., Kolda, T.G., Dunlavy, D.M.: All-at-once optimization for coupled matrix and tensor factorizations. arXiv preprint arXiv:1105.3422 (2011)
9. Acar, E., Rasmussen, M.A., Savorani, F., Næs, T., Bro, R.: Understanding data fusion within the framework of coupled matrix and tensor factorizations. Chemometr. Intell. Lab. Syst. **129**, 53–63 (2013)

10. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: a nucleus for a web of open data. In: Aberer, K., et al. (eds.) ASWC/ISWC -2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76298-0_52
11. Azmy, M., Shi, P., Lin, J., Ilyas, I.: Farewell freebase: migrating the simplequestions dataset to DBpedia. In: Proceedings of the 27th International Conference on Computational Linguistics, pp. 2093–2103 (2018)
12. Berant, J., Chou, A., Frostig, R., Liang, P.: Semantic parsing on freebase from question-answer pairs. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 1533–1544 (2013)
13. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 1247–1250. ACM (2008)
14. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in Neural Information Processing Systems, pp. 2787–2795 (2013)
15. Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr., E.R., Mitchell, T.M.: Toward an architecture for never-ending language learning. In: AAAI, vol. 5, p. 3 (2010)
16. Carroll, J.D., Chang, J.-J.: Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition. Psychometrika **35**(3), 283–319 (1970)
17. Chan, T.F.: Rank revealing QR factorizations. Linear Algebra Appl. **88**, 67–82 (1987)
18. Chang, L., Zhu, M., Gu, T., Bin, C., Qian, J., Zhang, J.: Knowledge graph embedding by dynamic translation. IEEE Access **5**, 20898–20907 (2017)
19. Cichocki, A., Zdunek, R., Phan, A.H., Amari, S.: Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation. Wiley, Hoboken (2009)
20. De Lathauwer, L., De Moor, B., Vandewalle, J.: A multilinear singular value decomposition. SIAM J. Matrix Anal. Appl. **21**(4), 1253–1278 (2000)
21. De Lathauwer, L., Nion, D.: Decompositions of a higher-order tensor in block terms-part III: alternating least squares algorithms. SIAM J. Matrix Anal. Appl. **30**(3), 1067–1083 (2008)
22. Diefenbach, D., Tanon, T., Singh, K., Maret, P.: Question answering benchmarks for Wikidata. In: ISWC 2017 (2017)
23. Ding, L., Pan, R., Finin, T., Joshi, A., Peng, Y., Kolari, P.: Finding and ranking knowledge on the semantic web. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 156–170. Springer, Heidelberg (2005). https://doi.org/10.1007/11574620_14
24. Domingos, P., Pazzani, M.: On the optimality of the simple bayesian classifier under zero-one loss. Mach. Learn. **29**(2–3), 103–130 (1997)
25. Dong, X., et al.: Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 601–610. ACM (2014)
26. Fabian, M.S., Gjergji, K., Gerhard, W., et al.: YAGO: a core of semantic knowledge unifying WordNet and Wikipedia. In: 16th International World Wide Web Conference, WWW, pp. 697–706 (2007)

27. Franz, T., Schultz, A., Sizov, S., Staab, S.: TripleRank: ranking semantic web data by tensor decomposition. In: Bernstein, A., et al. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 213–228. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04930-9_14
28. Golub, G.H., Reinsch, C.: Singular value decomposition and least squares solutions. Numerische Math. **14**(5), 403–420 (1970)
29. Gu, M., Eisenstat, S.C.: Efficient algorithms for computing a strong rank-revealing QR factorization. SIAM J. Sci. Comput. **17**(4), 848–869 (1996)
30. Harshman, R.A.: Foundations of the PARAFAC procedure: models and conditions for an "explanatory" multimodal factor analysis (1970)
31. He, S., Liu, K., Ji, G., Zhao, J.: Learning to represent knowledge graphs with Gaussian embedding. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, pp. 623–632. ACM (2015)
32. Ji, G., He, S., Xu, L., Liu, K., 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), vol. 1, pp. 687–696 (2015)
33. Kim, H., Park, H., Eldén, L.: Non-negative tensor factorization based on alternating large-scale non-negativity-constrained least squares. In: Proceedings of the 7th IEEE International Conference on Bioinformatics and Bioengineering, BIBE 2007, pp. 1147–1151. IEEE (2007)
34. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Rev. **51**(3), 455–500 (2009)
35. Lee, D.D., Seung, H.S.: Learning the parts of objects by non-negative matrix factorization. Nature **401**(6755), 788 (1999)
36. Lehmann, J., et al.: DBpedia-a large-scale, multilingual knowledge base extracted from Wikipedia. Semant. Web **6**(2), 167–195 (2015)
37. Lin, Y., Liu, Z., Luan, H., Sun, M., Rao, S., Liu, S.: Modeling relation paths for representation learning of knowledge bases. arXiv preprint arXiv:1506.00379 (2015)
38. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: AAAI, pp. 2181–2187 (2015)
39. Meilicke, C., Fink, M., Wang, Y., Ruffinelli, D., Gemulla, R., Stuckenschmidt, H.: Fine-grained evaluation of rule- and embedding-based systems for knowledge graph completion. In: Vrandečić, D., et al. (eds.) ISWC 2018. LNCS, vol. 11136, pp. 3–20. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00671-6_1
40. Melo, A., Völker, J., Paulheim, H.: Type prediction in noisy rdf knowledge bases using hierarchical multilabel classification with graph and latent features. Int. J. Artif. Intell. Tools **26**(02), 1760011 (2017)
41. Moniruzzaman, A.B.M., Nayak, R., Tang, M., Balasubramaniam, T.: Fine-grained type inference in knowledge graphs via probabilistic and tensor factorization methods. In: The World Wide Web Conference, pp. 3093–3100. ACM (2019)
42. Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs. Proc. IEEE **104**(1), 11–33 (2016)
43. Nickel, M., Tresp, V., Kriegel, H.-P.: A three-way model for collective learning on multi-relational data. In: Proceedings of the 28th International Conference on Machine Learning (ICML-11), pp. 809–816 (2011)
44. Nickel, M., Tresp, V., Kriegel, H.-P.: Factorizing YAGO: scalable machine learning for linked data. In: Proceedings of the 21st international conference on World Wide Web, pp. 271–280. ACM (2012)
45. Paulheim, H.: Knowledge graph refinement: a survey of approaches and evaluation methods. Semant. Web **8**(3), 489–508 (2017)

46. Paulheim, Heiko, Bizer, Christian: Type inference on noisy RDF data. In: Alani, H., et al. (eds.) ISWC 2013. LNCS, vol. 8218, pp. 510–525. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41335-3_32
47. Paulheim, H., Bizer, C.: Improving the quality of linked data using statistical distributions. Int. J. Semant. Web Inf. Syst. (IJSWIS) **10**(2), 63–86 (2014)
48. Porrini, R., Palmonari, M., Cruz, I.F.: Facet annotation using reference knowledge bases. In: Proceedings of the 2018 World Wide Web Conference on World Wide Web, pp. 1215–1224. International World Wide Web Conferences Steering Committee (2018)
49. Suchanek, F.M., Kasneci, G., Weikum, G.: YAGO: a core of semantic knowledge. In: Proceedings of the 16th International Conference on World Wide Web, pp. 697–706. ACM (2007)
50. Toutanova, K., Chen, D.: Observed versus latent features for knowledge base and text inference. In: Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality, pp. 57–66 (2015)
51. Toutanova, K., Chen, D., Pantel, P., Poon, H., Choudhury, P., Gamon, M.: Representing text for joint embedding of text and knowledge bases. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1499–1509 (2015)
52. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: a survey of approaches and applications. IEEE Trans. Knowl. Data Eng. **29**(12), 2724–2743 (2017)
53. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph and text jointly embedding. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1591–1601 (2014)
54. Xiao, H., Huang, M., Zhu, X.: TransG: a generative model for knowledge graph embedding. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), vol. 1, pp. 2316–2325 (2016)
55. Zhang, H.: The optimality of naive bayes. AA **1**(2), 3 (2004)
56. Zhang, J., Lu, C.-T., Cao, B., Chang, Y., Philip, S.Y.: Connecting emerging relationships from news via tensor factorization. In: 2017 IEEE International Conference on Big Data (Big Data), pp. 1223–1232. IEEE (2017)
57. Zupanc, K.: Davis, J.: Estimating rule quality for knowledge base completion with the relationship between coverage assumption. In: Proceedings of the Web Conference 2018, pp. 1–9 (2018)