DAIKIRI
Diagnostische KI für industrielle Daten

DAIKIRI
Erklärbare Diagnostische KI für industrielle Daten

Project Number: 01IS19085B **Start Date of Project:** 01/01/2020 **Duration:** 24 months

Deliverable 3.1
Clustering of High-Dimensional Data

Dissemination Level	Public
Due Date of Deliverable	Month 12, 31/12/2020
Actual Submission Date	Month 12, 31/12/2020
Work Package	WP3 — Semantification
Tasks	T3.1 & T3.2
Type	Report
Approval Status	Final
Version	1.0
Number of Pages	18

Abstract: This deliverable presents an approach for clustering of high-dimensional data developed within the DAIKIRI project. Experiments were performed with different embedding algorithms and clustering algorithms on different datasets.

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

This project has received funding from the Federal Ministry of Education and Research (BMBF) within the project DAIKIRI under the grant no 01IS19085B.

History

Version	Date	Reason	Revised by
1.0	31/12/2020	Final version created	Hamada Zahera

Author List

Organization	Name	Contact Information
UPB	Hamada Zahera	hamada.zahera@uni-paderborn.de
UPB	Stefan Heindorf	heindorf@uni-paderborn.de
UPB	Axel-Cyrille Ngonga Ngomo	axel.ngonga@uni-paderborn.de
AI4BD	Semih Ymusak	semih.yumusak@ai4bd.com
AI4BD	Mehmet Buğrahan Duran	bugrahan.duran@ai4bd.com
AI4BD	Martin Voigt	martin.voigt@ai4bd.com

Executive Summary

When constructing knowledge graphs, e.g., for web search or industrial applications, an important task is to create the type hierarchy of entities. In this paper, we propose a fully *unsupervised* approach without requiring any training data. Our approach clusters entities by means of their embeddings in order to derive their types. One particular challenge is the high-dimensionality of embedding spaces: as the number of dimensions increases, many clustering algorithms suffer from the *curse of dimensionality* causing a degradation of clustering performance and rendering distance measures meaningless. To this end, we experiment with different embedding and clustering approaches and evaluate them both in terms of predictive performance and runtime performance, comparing them to unsupervised and supervised baselines. Our results show that unsupervised embedding-based approaches can reasonably predict the types with F1-measures up to 0.634 and 0.751 on the FB15k-237 and WN18RR benchmarking datasets respectively. As expected, unsupervised approaches could not reach the performance of fully-supervised approaches, raising the need to explore semi-supervised approaches requiring only small amounts of training data.

Contents

1 Introduction

In recent years, it has become a common practice to compute embeddings for various tasks—word embeddings for natural language processing, image embeddings for object/face recognition, and entity embeddings for knowledge graphs (KGs) completion and node classification. In the absence of labeled training data, clustering approaches allow, e.g., to identify similar nodes in knowledge graphs. However, to the best of our knowledge, knowledge graph embeddings have not been used for prediction yet.

Typical embedding dimensions for knowledge graphs go up to 512 Ruffinelli et al. [2020] and real-world datasets such as Wikidata and YAGO, contain over 100 million entities. Such large embedding dimensions and dataset sizes pose significant scalability challenges for traditional clustering algorithms such as centroid-based, hierarchical, or density-based clustering. Potential solutions that have been proposed for large-scale high-dimensional data are HDBSCAN Campello et al. [2013] and NG-DBSCAN Lulli et al. [2016] as well ask-means++Arthur and Vassilvitskii [2007] and k-means||Bahmani et al. [2012]. However, in high-dimensional data, traditional similarity measures (e.g. Euclidean distance), as used in conventional clustering algorithms, are usually not meaningful Kriegel et al. [2009]. This problem is known to be *the curse of dimensionality* and related phenomena require adaptations of clustering approaches to the nature of high-dimensional data. Moreover, clustering approaches have been hardly evaluated on entity embeddings and for the task of type prediction. The main challenge is the absence of information about entity types—an optimal clustering approach should allocate entities with the same types together in an unsupervised learning manner. Hence, entity representations (embeddings) play a major role in an efficient clustering process. Recently, embedding-based representation has been employed effectively in various tasks where entities with similar proprieties are *embed*ded together into a semantic embedding space.

In this paper, we leverage clustering approaches on high-dimensional knowledge graph embeddings to predict entity types. Our research goals are (1) to analyze the impact of different embedding representations on the performance of type prediction, (2) to compare different clustering approaches for this task, and (3) to evaluate their scalability. To this end, we performed several experiments: Based on standard benchmarking datasets for knowledge graph completion, e.g., FB15k-237 and WN18RR, we experimented with the embedding approaches RotatE, TransE, and DistMult and the clustering approaches HDBSCAN, agglomerative clustering, and k-means. As a baseline, we employ both random clusterings and supervised approaches, and we evaluate our approach in terms of accuracy, cluster purity as well as weighted macro-averaged recall, precision, and F1-measure. Our results show that the best embedding and clustering approach depends on the dataset—with k-means, HDBSCAN on top of TransE and DistMult, often producing good results while agglomerative clustering and RotatE seem to be less suitable for the task.

The remainder of this paper is organizing as follows: In Section 2, we discuss state-of-the-art clustering methods addressing the challenges of clustering high-dimensional data. Section 3 describes the full pipeline of our approach to benchmark clustering approaches in entity type prediction. In Section 4, we describe our experiments on two different datasets and discuss the performance results of baselines. Section6 concludes the paper and shows the current challenges in high-dimensional data for future work.

2 Related Work

2.1 Clustering of High-Dimensional Data

Kriegel et al. [2009] and Parsons et al. [2004] survey clustering approaches for high-dimensional data. In high dimensions, similarity measures such as Euclidean distance often become meaningless since the relative distance between the closest and farthest points converges to 0. Moreover, with increasing dimensions, there are often many irrelevant features representing distracting "noise". A common approach to overcome the issues of high-dimensional data is to perform feature selection or dimensionality reduction, e.g., with PCA. However, these operations are performed *globally,* whereas different features and different information are often required for different clusters. To overcome this problem, approaches for subspace clustering, pattern-based clustering, and correlation clustering have been proposed—each with their own assumptions on the underlying data. In our case of clustering neural embeddings, we cannot assume axis-parallel subspaces or pattern-based subspaces but must assume arbitrarily-oriented clusters. Hence, correlation-based clustering approaches appear particularly promising. While most of them rely on PCA with runtime cubic in the number of dimensions, ORCLUSAggarwal and Yu [2000] is a reasonably fast k-means-like algorithm and 4CBöhm et al. [2004] is a reasonably fast density-based algorithm that does not require the number of clusters in advance.

Esmin et al. [2015] review particle swarm optimization (PSO) algorithms for clustering high-dimensional data. They discuss the shortcomings of conventional algorithms such as their slowness of convergence, sensitivity to initialization values (e.g., K-means), and manual effort required, e.g., for hyper-parameter optimization. The authors survey particle swarm optimization algorithms to overcome some of the challenges. PSO explores the search space by means of interacting particles moving towards good solutions. While PSO requires few parameters to adjust, its convergence speed is slow near the global optimum when the search space is large and complex. Hence, PSO is a promising direction of future research, but it is not guaranteed to outperform conventional algorithms yet.

2.2 Clustering of Large Datasets

Besides a large number of dimensions, another challenge often encountered in practice are large datasets with millions if not billions of data points. Simple clustering approaches often taught in textbooks such as K-means or DBSCAN hardly scale to these sizes and a number of approaches have been proposed to improve their scalability. Centroid-based clustering approaches like k-means start by picking k points as cluster centroids and iteratively assigning the closest point to their respective clusters and updating the cluster centroids. Their scalability can be significantly improved by the initialization procedure of the k initial points, such as is done by k-mean++Arthur and Vassilvitskii [2007] and k-means||Bahmani et al. [2012].

Density-based approaches like DBSCAN group instances are located close to each other but far away from other clusters. By operating solely based on density, these algorithms can even cluster non-convex shapes. Scalable variants hierarchically decompose the space (HDBSCAN Campello et al. [2013]), efficiently approximate the densities (NG-DBSCAN Lulli et al. [2016]), employ reverse nearest neighbor counts (RNN-DBSCANBryant and Cios [2018]), or randomly partition the data (RP-DBSCAN Song and Lee [2018]).

Spectral clustering approaches use the Eigenvalues of the similarity/adjacency matrices. An efficient approximation approach was, for example, proposed by Yan et al. [2009]. Besides algorithmic advances, technical solutions increasingly employ the map-reduce paradigm to improve scalabil-

ity Cordeiro et al. [2011], Pandove et al. [2018]. Challenges in this direction include splitting data across machines, communication costs across machines, and merging solutions while maintaining a competitive clustering performance.

2.3 Clustering in Knowledge Graph

2.3.1 Clustering of Knowledge Graph Entities

Discovering Types in RDF Datasets: Kellou-Menouer and Kedad [2015] presented a clustering-based approach to discover entity types in linked data (e.g., RDF). The proposed approach employed a density-based clustering to group entities with similar types together into one cluster. Jaccard metric is used to measure the similarity between two entities. To *speed-up* the clustering process, they run the clustering algorithm once and assign types to new data as the nearest neighbors. Christodoulou et al. [2015] proposed a similar approach for structural inference in Linked data using clustering. They introduced an automatic approach to derive structural summary linked data as RDF triples. Then, they employed a hierarchical-based clustering algorithm to organize the data into clusters and then infer a structural summary. The experimental results of structural summaries are expressed in the form of classes, properties and relationships, and good performance in different types of linked data sources.

2.3.2 Entity Embedding for Type Prediction

Demir and Ngonga Ngomo [2019] proposed a scalable approach for knowledge graph embedding (dubbed PYKE) that combines a physical model based on Hooke's law Rychlewski [1984] and its inverse with ideas from simulated annealing to compute the embeddings for knowledge graphs efficiently. The experimental results demonstrated PYKE achieves a linear space complexity— PYKE is more than 22 times faster than existing embedding solutions in the best case and can *scale-up* to a knowledge graph containing millions of triples.

2.3.3 Hierarchical Type Prediction (without embeddings/clustering).

Paulheim and Bizer [2013] addressed the problem of type prediction in noisy and incomplete knowledge bases. They proposed a heuristic link-based approach for entity type completion (dubbed SDType). The basic idea is to use each link from and to an instance as an indicator for the resource's type. For each link, they use the statistical distribution (hence the name SDType) of types in the subject and object position of the property for predicting the instance's types.

Similar work was also carried out by Melo et al. [2017] where they formulate the problem of type prediction as a hierarchical multi-classification task. The authors exploited different features (including entity embeddings) to learn node representation and predict their types.

2.3.4 Ontology Learning

Learning ontology from relational databases: Li et al. [2005] reviewed different techniques to evaluate ontology learning. Depending on the ontology types and the purpose of evaluation, the authors summarize the most common approaches that fall into one of the categories such as 1) comparing the ontology to a "golden standard", 2) Usage of ontology in applications and evaluate the results. 3)

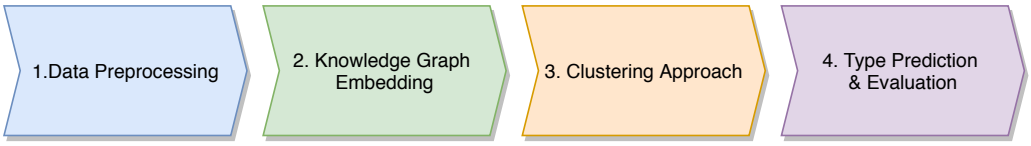


Figure 1: The Pipeline of our proposed approach.

human-based evaluation with a set of predefined criteria. A survey of ontology evaluation techniques: Brank et al. [2005] proposed an approach that can automatically learn OWL ontology from a relational database by a set of learning rules. The approach can obtain OWL ontologies like classes, properties, properties, characteristics, cardinality, and instances.

3 Methodology

The problem of type prediction in KGs requires *scalable* approaches that can handle high-dimensional representations of entities and their relations. In this section, we describe our proposed approach based on four main-folds as depicted in Figure 1. In the following subsections, we first define the task of entity prediction, then we describe the details of each module in our approach.

3.1 Task Definition:

Let $\mathcal{V} = (\mathcal{E}, \mathcal{R})$ be a knowledge graph of linked data where $\mathcal{E} = (e_1, e_2, \dots e_n)$ are entities and $\mathcal{R}$ represents their relations. $\mathcal{E}$ are commonly organized using a set of types $\mathcal{T} = (t_1, t_2, \dots, t_k)$ (e.g., persons, countries, vehicles). Entity typing is the task of assigning a type t to an entity e and is a fundamental task in KG completion. In this paper, we aim to predict $\mathcal{T}$, given unlabeled data (i.e., an unsupervised learning scenario). We assume an input data is in the form of RDF triples, a single instance is formulated as (e_s, r_t, e_o), where e_s and e_o denote the subject and object entities. Our approach aims to predict e_o— i.e., the type of subject entity e_s, given type relation r_t(e.g., `rdf:type`). For example, the triple `<dbr:Albert Einstein, rdf:type, dbo:Scientist>` states that Albert Einstein (e_s) is of type type (r_t) Scientist (e_o).

3.2 Data Preprocessing:

The first step in our approach is to load the input data and generate its knowledge graph representation in RDF format. This is an essential step to formulate the data for the next step, where we train the embedding model to learn the representation of entities and their relationships. Tabular data might be converted with `Vectograph`, [1] an open-source library to convert tabular data into RDF.

3.3 Knowledge Graph Embedding

This step aims to learn the representation of entities and their relations in a knowledge graph. To capture entities' latent semantics, we employed a KG embedding where entities and relations are represented as low-dimensional vectors. Moreover, entities with similar properties are then *embed*ded close to each other in the embedding space. Recently, several KGs embedding methods have been

[1] `https://github.com/dice-group/Vectograph`

proposed to generate vector representations for entities and their relationships. We used the `GraphVite` library to train KGs models (TransE, RotatE, and DistMult) on the Freebase and WordNet datasets for benchmarking our experiments. The model's hyper-parameters are tuned to achieve the best performance. In particular, we used a batch size of $2,000$ triples and train the models for $1,000$ epochs. We use the Adam optimizer with learning rate $2 \cdot 10^{-6}$. Finally, entities' vectors are generated to be clustered in the next step. Each vector represents the entity's features in 256 dimensions.

3.4 Entity Clustering

In this step, we aim to cluster entities, a good clustering method will produce high-quality clusters with high *intra-class* (i.e, the *distance between clusters*) similarity and low inter-class (i.e, the *distance between entities in the same cluster*). In our approach, we employ a density-based clustering approach (HDBSCAN) to cluster entity vectors into groups. HDBSCAN requires two inputs parameters: ϵ is a radius within identity how many neighbor points and `minPoint` is the minimum number of neighbor points to consider a core point. We used *cosine similarity* as a distance metric to compute nearest entities. Each entity is assigned to a cluster based on its nearest core points. HDBSCAN outputs k clusters of entities, including anomalies entities (which do not belong to any cluster).

3.5 Entity Typing

Finally, we inference types of entities based on the previously computed clusters. We *propagate* the major type $\mathbf{t}^{\star}$ to all entities in a cluster C. To this end, our approach clusters similar entities w.r.t their types, where each cluster will have a single type of entities.

4 Experimental Evaluation

We conducted several experiments to verify the effectiveness of our approach on two benchmark datasets for entity typing. Our goal is to answer the following research questions:

Q.1. Which embedding approach yields the best clusters of entity types?

Q.2. What clustering approach yields the best clusters of entity types?

Q.3. How scalable are the clustering algorithms both in terms of number of embedding dimensions and number of samples?

4.1 Evaluation Setup

4.1.1 Datasets

We employ two labeled datasets:

4.1.1.1 FB15k-237

Toutanova et al. [2015] compiled the dataset to benchmark the embedding performance of type and link prediction tasks. It includes a subset of the Freebase Knowledge Graph with 14,951 entities

Table 1: Dataset statistics.

Dataset	Entities	Relations	Triples	Dim.	Types
FB15k-237	14,541	237	310,116	512	6
WN18RR	40,559	11	93,003	256	5

and 237 relations. Each triple $\mathbf{t} = (e_s, r, e_o)$ describes a relation r between subject entity e_s and object entity e_o, for example, "Mira Nair" (`/m/0kvsb`) "has profession" (`/people/person/profession`) "Professor" (`/m/01d30f`). We obtain the type of subjects via the property hierarchy, e.g., "Mira Nair" has type "people". If an entity has more than one type, we employ the most frequent one, in the same way as Toutanova and Chen [2015]. Similarly, we consider the types "education", "film", "location", "music", and "soccer". Our filtered and labeled version of the dataset has 7,891 entities and 6 ontology types (i.e, entity class).

4.1.1.2 WN18RR

This dataset includes a subset of WordNet Dettmers et al. [2017], which contains lexical information of the English language. The dataset contains 93,003 triples with 40,943 entities and 11 relations. Each triple represents the semantic relation between words (e.g, hyponym, hypernym, or antonymy). For example, a triple `dog.n.01 hypernym canine.n.02` models the hypernym relation between the two entities "dog" and "canine". For our experiments, we obtain the types transitively via the hypernym relation and consider the top-level types "animal", "communication", "food", "person", and "plant". For example, both "dog" and "canine" would be of type "animal". Our filtered and labeled version of the dataset has 15,583 entities and 5 ontology types.

4.1.2 Baselines

We investigated different baselines (both *unsupervised* and *supervised*) to evaluate the performance of entity typing. Regarding the former, we considered centroid-based, hierarchical, and density-based clustering algorithms (k-means, agglomerative clustering, and HDBSCAN). For comparison, to get an idea of the maximal achievable performance, we also explored supervised baselines such as logistic regression, K-nearest neighbors, and random forest.

4.1.3 Evaluation Metrics

We assess the clustering quality based on accuracy and purity criteria, i.e, good clusters should have one single type of entities. Moreover, we employ precision, recall, and F1-score to reflect the imbalanced nature of our datasets. Once the clusters have been computed, an oracle assigns the correct label, i.e., a type to a cluster. While in practice, the oracle, might be a human annotator, we employ a majority vote for our experiments.

4.1.3.1 Accuracy & Cluster Purity

Accuracy measures the fractions of entities assigned to the correct cluster (majority type within the cluster). Cluster purity evaluates the cluster homogeneity—i.e, the degree to which a cluster contains members from a single class (a fraction of samples belong to the cluster's majority)—and each cluster

is weighted according to its samples. In our setting of employing clustering for type prediction (with a majority vote oracle), cluster purity (before majority vote) and accuracy (after majority vote) are equal.

4.1.3.2 Precision, Recall, and F1-Score

Moreover, we report weighted macro-averaged precision, recall, and F1-scores: The scores are computed for each class (i.e., type in our case) and weighted according to the number of samples in this class.

4.1.4 Reproducibility

We provide an open-source implementation of our experiments along with a description of our input data and required libraries[2]. For training the embeddings models rotateE, transE, and distMult on the datasets FB15k-237 and WN18RR, we used the GraphVite[3]. Our experiments were run on a server with 16 processors (`Intel(R) Xeon(R) CPU E5-2695 v4 @ 2.10GHz`) and 126 GB of memory.

4.2 Embedding Algorithms

Targetting the question what embedding algorithm works best for entity type prediction ($Q.1$), we experiment with three state-of-the-art KGs embedding *TransE*, *DistMult*, and *RotatE*. Tables (2,3) report the performance results on each embedding approach in terms of accuracy (Acc.), precision (Pre.), recall (Rec.), F1-score (F1) and cluster purity (Purity). The table shows that the embedding approach has a substantial impact on the clustering performance. For example, in Table 2, HDBSCAN achieves an F1-score 62% with *TransE* compared to 11% with *RotatE*. We hypothesize that this performance margin is related to the way each model allocates similar entities together—i.e., how each model computes the similarity between entities. Further details are discussed in Section 5.0.0.1.

4.3 Clustering Algorithms

Asking ourselves which clustering algorithm would be most suitable ($Q.2$), we experiment with K-means, agglomerative clustering, and HDBSCAN. Table 2 shows that K-means achieves the best F1-measure (0.751) with *TransE* embeddings on the FB15-237 dataset. On the WN18RR dataset, HDBSCAN achieves the best F1-measure (0.634) with *DistMult* embeddings. Furthermore, we observed that K-means and Agglomerative approaches achieved very close performance in entity typing—with K-means slightly outperforming agglomerative clustering (+0.01 and +0.05 on FB15-237 and WN1855 respectively). While both K-means and HDBSCAN achieved good performance for unsupervised entity typing in our experiments, we observed HDBSCAN be a more robust and stable approach compared to K-means. K-means's performance varies with its initialization and requires the number of clusters as a hyper-parameter. In particular, we measure the stability of algorithms across different runs. HDBSCAN's has stable F1-scores with no variance, while for example, K-means showed standard deviation in it's F1-scores by 0.13% on FB15-237 dataset.

To visualize the clustering behavior, we projected embeddings including the clustering results using T-SNE projection (Figures **??**, 4c, 4d). The figures show that the embedding approach has a big impact on the distribution of samples in the embedding space. For example, the distance between

[2] `https://github.com/dice-group/DAIKIRI-Clustering`
[3] `https://graphvite.io/`

Table 2: Performance Results of Clustering Algorithms on Datasets: Freebase 15k-237. Best results in bold. ♠ refers to random clustering with equal probability. and ♣ for random clustering with custom probability based on type distribution in the dataset.

Algorithms	**FB15k Dataset with Different Embeddings**														
	FB15k-TransE					**FB15k-DistMult**					**FB15k-RotatE**				
	Acc.	Pre.	Rec.	F1	Purity	Acc.	Pre.	Rec.	F1	Purity	Acc.	Pre.	Rec.	F1	Purity
RandomClustering[♠]	0.281	0.154	0.281	0.194	0.281	0.287	0.158	0.287	0.203	0.287	0.282	0.155	0.282	0.176	0.282
RandomClustering[♣]	0.285	0.157	0.285	0.199	0.285	0.280	0.154	0.280	0.198	0.280	0.280	0.154	0.280	0.198	0.280
Logistic Regression	0.900	0.908	0.900	0.900	0.900	0.905	0.907	0.905	0.905	0.905	0.266	0.217	0.266	0.208	0.283
Knn	0.874	0.874	0.874	0.873	0.874	0.802	0.835	0.802	0.798	0.802	0.232	0.182	0.232	0.172	0.282
RandomForest	**0.901**	**0.915**	**0.901**	**0.901**	**0.901**	**0.908**	**0.919**	**0.908**	**0.908**	**0.908**	**0.314**	**0.238**	**0.314**	**0.253**	**0.315**
K-Means	**0.784**	**0.753**	**0.784**	**0.751**	**0.784**	0.771	0.750	0.771	0.741	0.771	0.282	**0.200**	0.282	0.200	0.282
Agglomerative	0.779	0.749	0.779	0.746	0.779	0.781	0.752	**0.781**	**0.749**	**0.781**	**0.284**	0.156	**0.284**	**0.201**	**0.284**
HDBSCAN	0.678	0.617	0.678	0.624	0.678	0.475	0.321	0.475	0.362	0.475	0.276	0.076	0.276	0.119	0.276

Table 3: Performance Results of Clustering Algorithms on WN18RR Dataset. Best results in bold. ♠ refers to random clustering with equal probability. and ♣ for random clustering with custom probability based on type distribution in the dataset.

Algorithms	**WN18RR Dataset with Different Embeddings**														
	WN18RR-TransE					**WN18RR-DistMult**					**WN18RR-RotatE**				
	Acc.	Pre.	Rec.	F1	Purity	Acc.	Pre.	Rec.	F1	Purity	Acc.	Pre.	Rec.	F1	Purity
RandomClustering[♠]	0.274	0.075	0.274	0.117	0.274	0.274	0.075	0.274	0.117	0.274	0.274	0.075	0.274	0.117	0.274
RandomClustering[♣]	0.274	0.146	0.274	0.137	0.274	0.274	0.075	0.274	0.117	0.274	0.274	0.144	0.274	0.172	0.276
Logistic Regression	0.766	0.772	0.766	0.758	0.766	0.818	0.814	0.818	0.816	0.818	0.853	0.854	0.853	0.851	0.853
Knn	**0.936**	**0.936**	**0.936**	**0.936**	**0.936**	**0.943**	**0.943**	**0.943**	**0.943**	**0.943**	**0.938**	**0.938**	**0.938**	**0.938**	**0.938**
RandomForest	0.824	0.833	0.824	0.811	0.824	0.878	0.889	0.878	0.874	0.878	0.927	0.930	0.927	0.924	0.927
K-Means	0.546	0.497	0.546	0.474	0.546	0.568	0.575	0.568	0.513	0.568	0.594	0.592	0.594	0.547	0.594
Agglomerative	0.510	**0.690**	0.510	0.428	0.510	0.526	0.581	0.526	0.451	0.526	0.569	0.555	0.569	0.514	0.569
HDBSCAN	**0.592**	0.564	**0.592**	**0.539**	**0.592**	**0.609**	**0.719**	**0.609**	**0.634**	**0.609**	**0.617**	**0.719**	**0.617**	**0.620**	**0.617**

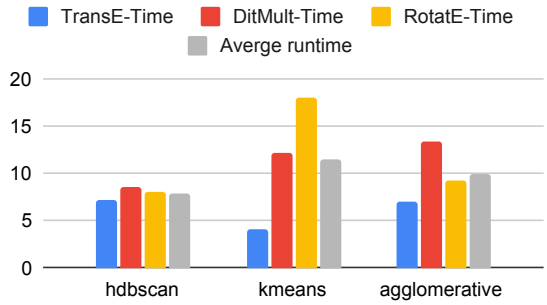
clusters is much large for *TransE* embeddings compared to *RotatE* embeddings on the FB15k-237 dataset.

4.4 Scalability

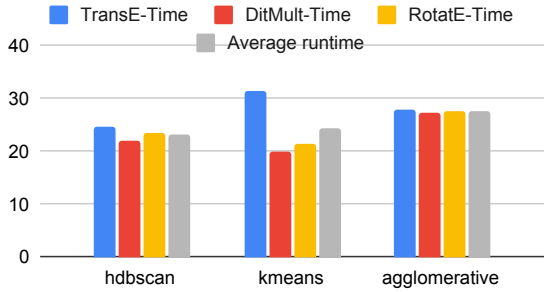
Analyzing the runtime of entity typing approaches ($Q.3$), we measure the runtimes to cluster the entities based on precomputed embeddings. It is apparent from Figure 2a that HDBSCAN has the most stable runtimes with different embeddings, while the runtime of other clustering approaches heavily depends on the kind of embeddings. On average, HDBSCAN achieves the best runtime performance on the FB15K-237 dataset.

Table 4: Performance Results of Clustering Algorithms on AI4BD Smart Logistics Dataset. Best results in bold.

AI4BD Smart Logistics-TransE					
Algorithms	Acc.	Pre.	Rec.	F1	Purity
RandomClustering	0.501	0.501	0.501	0.501	0.501
Logistic Regression	0.576	0.577	0.57	0.576	0.576
Knn	**0.749**	**0.749**	**0.749**	**0.749**	**0.749**
RandomForest	0.683	0.690	0.683	0.680	0.683
K-Means	0.576	0.577	0.576	0.577	0.576
Agglomerative	0.576	0.576	0.576	0.575	0.576
HDBSCAN	**0.611**	**0.582**	**0.611**	**0.597**	**0.577**



(a) FB15k-237 Dataset.



(b) WN18RR Dataset.

Figure 2: Runtime (seconds) Comparison of Entity Typing.

Similarly, Figure 2b compares the runtime performance on the WN18RR dataset. The overall results show that HDBSCAN approach has the best runtime performance compared to other approaches. Interestingly, we observed that agglomerative clustering took *consistent* runtime, however, HDBSCAN was faster by -10% than agglomerative clustering. Overall, these results suggest that HDBSCAN is a good choice for clustering high-dimensional data. It can compromise trade-off performance properly between accuracy and runtime metrics.

5 Discussion

5.0.0.1 Scoring function of embedding approaches

Score function in KGs has a significant impact on the quality of learned embeddings. They are used to train the KG models so that the entities connected by relations are close to each other. In our experiments, we trained the KGs models: *TransE, DistMult*, and *RotatE* to learn embedding representation for entities and relations. Each model uses a different score function which learns the embedding in a different way. We aim to answer $Q.1$ to find which embedding representation yields better in entity typing tasks. *TransE* model employs a distance-based scoring function which scores entities using distances in the embedding space, while *DistMult* uses a multiplicative approach.

We observed that transitional-based models (*TransE*) achieved outperforming scores when using

the same score function during embedding as in the clustering method. For example, HDBSCAN with cosine similarity as a distance function achieved F1 scores with *TransE* embedding with score function? compared to HDBSCAN with Euclidean distance. Hence, we demonstrate the impact of the score function on the learned embeddings w.r.t the task which will be applied. Few studies Zhang et al. [2019], Gao et al. [2019] highlighted this research gap with computed KGs embedding in a data-driven or task-driven way rather than generally learned embeddings.

5.0.0.2 Clustering approaches without embeddings

We will perform an ablation study to testify the influence of entity representation with/without KG embedding on clustering performance. In particular, we will conduct our experiments without entity embeddings, we will explore traditional features representation (e.g, one-hot encoding or entity's proprieties as bag-of-words) to benchmark the extent-to-which entity embeddings affect the performance of clustering and entity typing task.

5.0.0.3 Future work: semi-supervised type prediction

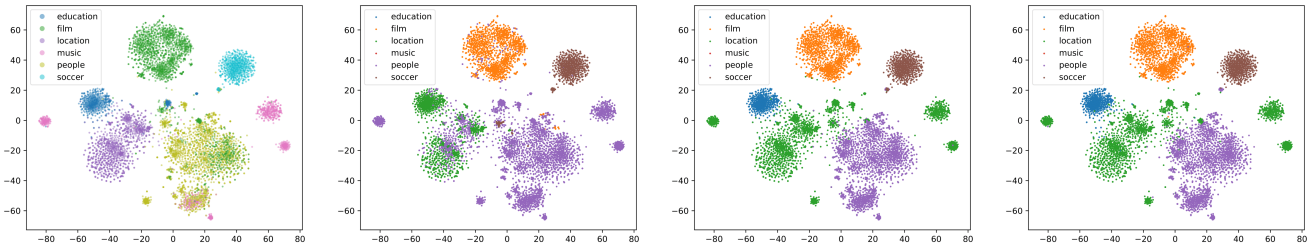
Unsupervised approaches showed promising results in predicting entity types without label data; the best performance achieved with 0.63 and 0.75 on FB15k-237 and WN18RR. On the other hand, supervised approaches showed very good performance (up to 0.90 and 0.94 on FB15k-237 and WN18RR), these approaches require a labeled dataset which is expensive either in labeling the training data by crowd-sourcing workers and time-consuming. In our further experiments, we will extend our approach to handle these challenges using semi-supervised approaches. At first, we will employ an unsupervised clustering method to group similar entities together into K clusters; then we will identify the top M entities as a cluster representation to be labeled by human experiments. The top-M entities can be chosen based on their distances to the cluster centroid. Finally, human experts will label (i.e, assign each entity an ontology type) the selected entities from each cluster, the final dataset contains K x M label entities. Finally, we will train a supervised classifier (e.g KNN) using our final dataset and predict the types of unlabelled entities.

6 Conclusion

Scalability and high dimensionality are two common problems associated with document clustering. In this paper, we present a clustering-based approach to predict entity types in KGS. Our approach comprises of four steps: 1) loading and generating KG; 2) KGs Embeddings; 3) Clustering; 4) Type Prediction. Our experimental results showed that leveraging density-based clustering with KGs embeddings representation improved significantly the performance of type predictions without training data. Moreover, our approach showed scalable and robust clustering performance on two benchmark datasets. In our future work, we will extend our experiments and benchmarks on industrial datasets as well. Moreover, we will investigate task adaptation for KGs embeddings in clustering problems.

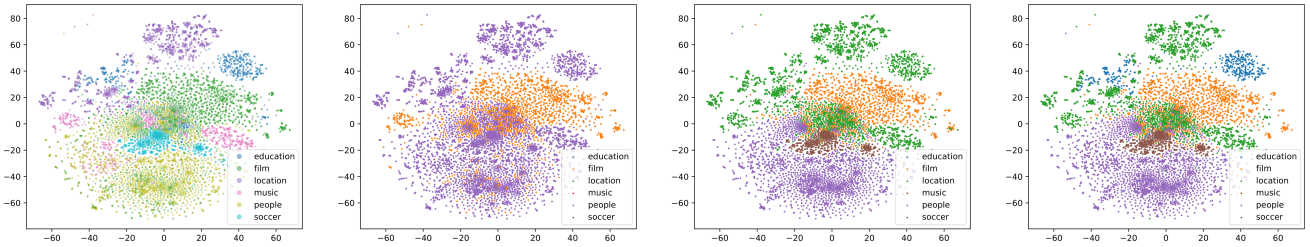
Acknowledgement

This work has been supported by the German Federal Ministry of Education and Research (BMBF) within the project DAIKIRI (under grant no 01IS19085B).



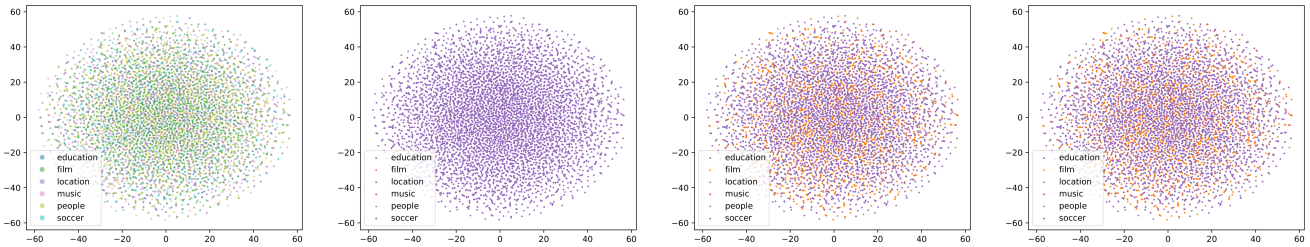
(a) (b) (c) (d)

Figure 3: T-SNE Visualization of FB15k TransE Embedding. Unsupervised Baselines: (a) HDBSCAN; (b) K-means; (c) Agglomerative.



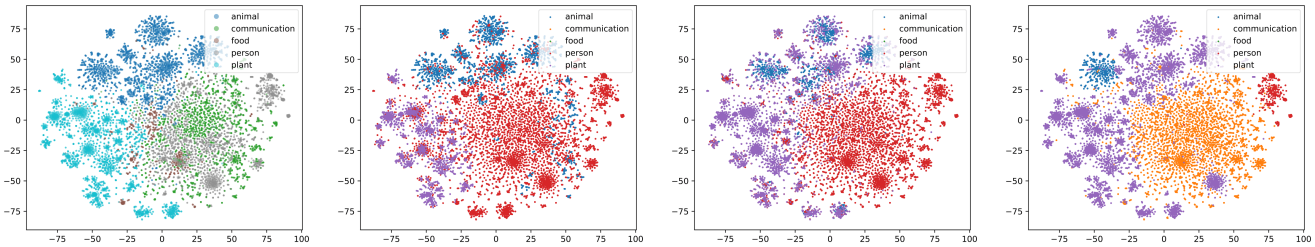
(a) (b) (c) (d)

Figure 4: T-SNE Visualization of FB15k DistMult Embedding. Unsupervised Baselines: (a) HDBSCAN; (b) Kmeans; (c) Agglomerative.



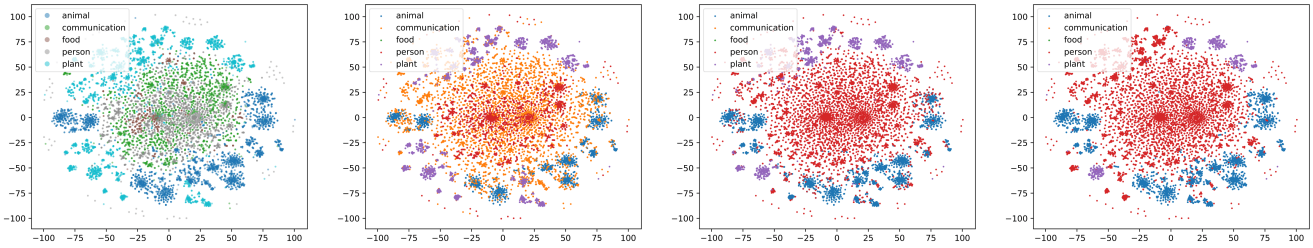
(a) (b) (c) (d)

Figure 5: T-SNE Visualization of FB15k RotatE Embedding. Unsupervised Baselines: (a) HDBSCAN; (b) Kmeans; (c) Agglomerative.



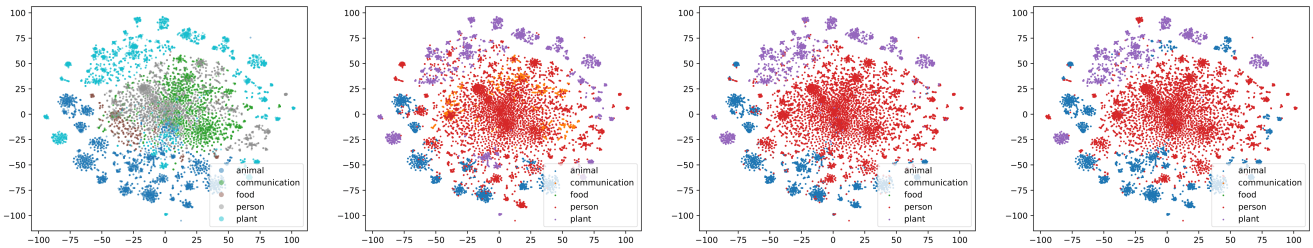
(a) (b) (c) (d)

Figure 6: T-SNE Visualization of WordNet TransE Embedding. Unsupervised Baselines: (a) HDBSCAN; (b) K-means; (c) Agglomerative.



(a) (b) (c) (d)

Figure 7: T-SNE Visualization of WordNet DistMult Embedding. Unsupervised Baselines: (a) HDBSCAN; (b) Kmeans; (c) Agglomerative.



(a) (b) (c) (d)

Figure 8: T-SNE Visualization of WordNet RotatE Embedding. Unsupervised Baselines: (a) HDBSCAN; (b) Kmeans; (c) Agglomerative.

References

Charu C. Aggarwal and Philip S. Yu. Finding generalized projected clusters in high dimensional spaces. In *SIGMOD Conference*, pages 70–81. ACM, 2000.

David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *SODA*, pages 1027–1035. SIAM, 2007.

Bahman Bahmani, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. Scalable k-means++. *Proc. VLDB Endow.*, 5(7):622–633, 2012.

Christian Böhm, Karin Kailing, Hans-Peter Kriegel, and Peer Kröger. Density connected clustering with local subspace preferences. In *ICDM*, pages 27–34. IEEE Computer Society, 2004.

Janez Brank, Marko Grobelnik, and Dunja Mladenić. A survey of ontology evaluation techniques. In *In In Proceedings of the Conference on Data Mining and Data Warehouses (SiKDD 2005)*, 2005.

Avory Bryant and Krzysztof J. Cios. RNN-DBSCAN: A density-based clustering algorithm using reverse nearest neighbor density estimates. *IEEE Trans. Knowl. Data Eng.*, 30(6):1109–1121, 2018.

Ricardo J. G. B. Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *PAKDD (2)*, volume 7819 of *Lecture Notes in Computer Science*, pages 160–172. Springer, 2013.

Klitos Christodoulou, Norman W. Paton, and Alvaro A. A. Fernandes. Structure inference for linked data sources using clustering. *Trans. Large Scale Data Knowl. Centered Syst.*, 19:1–25, 2015.

Robson Leonardo Ferreira Cordeiro, Caetano Traina Jr., Agma Juci Machado Traina, Julio César López-Hernández, U Kang, and Christos Faloutsos. Clustering very large multi-dimensional datasets with mapreduce. In *KDD*, pages 690–698. ACM, 2011.

Caglar Demir and Axel-Cyrille Ngonga Ngomo. A physical embedding model for knowledge graphs. In *JIST*, volume 12032 of *Lecture Notes in Computer Science*, pages 192–209. Springer, 2019.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. *arXiv preprint arXiv:1707.01476*, 2017.

Ahmed Ali Abdalla Esmin, Rodrigo A. Coelho, and Stan Matwin. A review on particle swarm optimization algorithm and its variants to clustering high-dimensional data. *Artif. Intell. Rev.*, 44(1): 23–45, 2015.

Huan Gao, Xianda Zheng, Weizhuo Li, Guilin Qi, and Meng Wang. Cosine-based embedding for completing schematic knowledge. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 249–261. Springer, 2019.

Kenza Kellou-Menouer and Zoubida Kedad. Discovering types in RDF datasets. In *ESWC (Satellite Events)*, volume 9341 of *Lecture Notes in Computer Science*, pages 77–81. Springer, 2015.

Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Trans. Knowl. Discov. Data*, 3(1):1:1–1:58, 2009.

Man Li, Xiao-Yong Du, and Shan Wang. Learning ontology from relational database. In *2005 International Conference on Machine Learning and Cybernetics*, volume 6, pages 3410–3415. IEEE, 2005.

Alessandro Lulli, Matteo Dell'Amico, Pietro Michiardi, and Laura Ricci. NG-DBSCAN: scalable density-based clustering for arbitrary data. *Proc. VLDB Endow.*, 10(3):157–168, 2016.

André Melo, Johanna Völker, and Heiko Paulheim. Type prediction in noisy RDF knowledge bases using hierarchical multilabel classification with graph and latent features. *Int. J. Artif. Intell. Tools*, 26(2):1760011:1–1760011:32, 2017.

Divya Pandove, Shivani Goel, and Rinkle Rani. Systematic review of clustering high-dimensional and large datasets. *ACM Trans. Knowl. Discov. Data*, 12(2):16:1–16:68, 2018.

Lance Parsons, Ehtesham Haque, and Huan Liu. Subspace clustering for high dimensional data: a review. *SIGKDD Explor.*, 6(1):90–105, 2004.

Heiko Paulheim and Christian Bizer. Type inference on noisy RDF data. In *International Semantic Web Conference (1)*, volume 8218 of *Lecture Notes in Computer Science*, pages 510–525. Springer, 2013.

Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. You CAN teach an old dog new tricks! on training knowledge graph embeddings. In *ICLR*. OpenReview.net, 2020.

Jan Rychlewski. On hooke's law. *Journal of Applied Mathematics and Mechanics*, 48(3):303–314, 1984.

Hwanjun Song and Jae-Gil Lee. Rp-dbscan: A superfast parallel dbscan algorithm based on random partitioning. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1173–1187, 2018.

Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66, 2015.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1499–1509, 2015.

Donghui Yan, Ling Huang, and Michael I. Jordan. Fast approximate spectral clustering. In *KDD*, pages 907–916. ACM, 2009.

Yongqi Zhang, Quanming Yao, Wenyuan Dai, and Lei Chen. Autosf: Searching scoring functions for knowledge graph embedding. *arXiv preprint arXiv:1904.11682*, 2019.