Learning Fine-Grained Semantics for Multi-Relational Data

Nitisha Jain and Ralf Krestel

Hasso Plattner Institute
University of Potsdam, 14482 Potsdam, Germany
<firstname.lastname@hpi.de>

Abstract. The semantics of relations play a central role in the understanding and analysis of multi-relational data. Real-world relational datasets represented by knowledge graphs often contain polysemous relations between different types of entities, that represent multiple semantics. In this work, we present a data-driven method that can automatically discover the distinct semantics associated with high-level relations and derive an optimal number of sub-relations having fine-grained meaning. To this end, we perform clustering over vector representations of entities and relations obtained from knowledge graph embedding models.[1]

Keywords: relation disambiguation · knowledge graph embeddings.

1 Introduction

Relations between different words or phrases are important for the semantic understanding of text. For real-world data, the relations are oftentimes polysemous by nature, i.e., they exhibit distinct meanings in different contexts. Similar to the task of word-sense disambiguation, relation disambiguation is needed to interpret the specific contextual semantics of relations in such cases. Relation semantics are particularly important in the context of knowledge graphs (KGs) that are widely used as multi-relational databases and constructed from natural language texts, where relation polysemy occurs frequently [8]. Since the ontologies for most large scale KGs have been curated based on texts through manual or semi-automated efforts, relations between the entities are often abstracted for simplification and avoidance of redundancies. This may result in cases where a single high-level relation serves as a generic notion between various different types of KG entities and has more than one semantic meaning associated with it. Due to the diversity of the kinds of associations between entities, abstract relations may not be sufficiently representative of the underlying semantics that they are supposed to capture. E.g., in Yago3 [7] the majority of the relations have multiple entity types (concepts) associated with them, with generic relations

[1] https://hpi.de/naumann/projects/web-science/ontology-engineering.html

Table 1: Examples of multiple semantics of relations

created class types	**owns** class types
(artist, medium)	(country, club)
(artist, movie)	(company, club)
(player, movie)	(company, company)
(officeholder, movie)	(sovereign, building)
(writer, movie)	(company, airport)
(writer, television)	(organization, airport)
(writer, fictional_character)	
(artist, computer_game)	
(company, computer_game)	

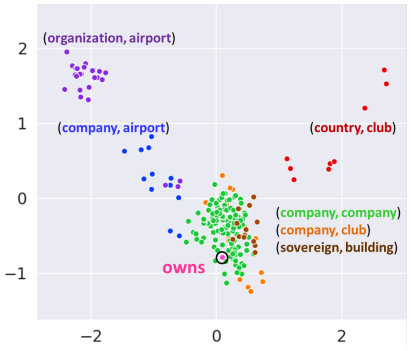

Fig. 1: Visualization of $\boldsymbol{\Delta_r}$ vectors for *owns* relation having different types

such as *owns* and *created* exhibiting very high type plurality. Some examples of the entity types that are associated with these relations are shown in Table 1 in the form of (head, tail) pairs. It can be seen that *created* is associated with (*company*, *computer game*), as well as (*painter*, *artwork*), despite the different contexts and dissimilar entity types. If such relations are split into fine-grained sub-relations, the precise contextual semantics as per the associated entity types can be clearly represented. Defining fine-grained semantics between the entities of a KG is important for facilitating several related applications such as question answering [2] and knowledge base completion [5].

However, due to the broad spectrum of semantic distances between the different entity types, it is a non-trivial task to determine an optimal way to split a relation into sub-relations, both in terms of the number of sub-relations, as well as the entity types that the sub-relations should represent. E.g., while some entity types are semantically similar to one another such as *television* and *movie* for *created* relation, other types are quite different, for instance, *company* and *writer* for *created*, or *airport* and *club* in the case of *owns* relation. We propose a data-driven, scalable method called *SemSplit* that can automatically determine an optimal configuration on a per-relation basis for a given dataset. Some previous works have discussed the identification of multiple relation semantics [6, 10]. However, they either perform manual clustering or propose rigid techniques with predefined number of clusters for all relations across a dataset, and thus, unlike our method, such approaches are not scalable or suitable for large and/or dynamic datasets.

The proposed method, *SemSplit*, leverages knowledge graph embeddings that provide semantic representations of the KG entities and relations in a continuous vector space. Popular KG embedding models such as TransE [1] are trained such that the vectors $\boldsymbol{h}$, $\boldsymbol{r}$ and $\boldsymbol{t}$ associated with a triple $<h, r, t>$ satisfy $\boldsymbol{h} + \boldsymbol{r} = \boldsymbol{t}$ or $\boldsymbol{r} = \boldsymbol{t}$ - $\boldsymbol{h}$. Previous work has shown that these embeddings are able to capture relation similarity, i.e., relations having similar meanings stay close in the embedding space [4]. On the contrary, we found that relations having multiple semantics are not well represented in the vector space, i.e., for the

triples of a relation r, the $\boldsymbol{t}$ - $\boldsymbol{h}$ vectors denoted by $\boldsymbol{\Delta_r}$, are not necessarily in the neighborhood of the actual relation vector. This is illustrated in Figure 1 where the original relation vector $\boldsymbol{r}$ for *owns* (that was calculated from all its fact triples) does not serve as the center of a single cluster for all the $\boldsymbol{\Delta_r}$ vectors. This is because the different type pairs connected by the same relation are quite different from one another semantically, and thus, form their own clusters. In this work, we perform clustering on the $\boldsymbol{\Delta_r}$ vectors and identify an optimal number of clusters that represent diverse semantics exhibited by the entity type pairs connected by the relation. These clusters can be employed for splitting the relation into several fine-grained sub-relations.

2 SemSplit: Clustering Relations

For every unique relation r in a given KG, the first step is to obtain the corresponding set of fact triples $<h, r, t>$. For every such triple, the entity types of the head and tail entities, denoted as T_h and T_t, are extracted from the KG ontology. The triples are categorized by the unique type pairs (T_h, T_t) that will be used to derive the labels of the clusters during the clustering stage. The set of all unique type pairs for relation r is denoted by S_r. For each type pair in S_r, the vectors $\boldsymbol{\Delta_r} = \boldsymbol{t} - \boldsymbol{h}$ are calculated and saved for all the triples associated with it. Here, the vectors $\boldsymbol{t}$ and $\boldsymbol{h}$ for the entities are derived from pre-trained KG embeddings.

For the clustering, we leverage the $\boldsymbol{\Delta_r}$ vectors for finding suitable clusters that will represent the different semantic meanings for a relation and hence derive the sub-relations for a given relation. Since $\boldsymbol{\Delta_j}$ vectors capture the semantics of the type pairs, vectors that are close in the embedding space would convey similar semantic meaning. A maximal splitting of the relation, with one sub-relation for every different type pair, would be inefficient and lead to a large number of unnecessary sub-relations. For example, the *created* relation has type pairs *(artist, painting)* and *(artist, music)* that have the same head entity type and hence related meanings, while the type pair *(company, computer_game)* portrays a different context. The *SemSplit* clustering performs the challenging task of finding an optimal number and composition of clusters $\mathcal{C}_{opt}$ for the type pairs, that can convey the distinct semantics of the relations based on the KG triples, by combining similar type pairs while separating the dissimilar ones.

In order to automatically determine the optimal configuration, clustering is performed for several iterations with a varying number of clusters. It starts with $\mathcal{L} = |S_i|$ clusters, where every cluster corresponds to a distinct type pair for the relation, and the cluster labels are assigned accordingly. To narrow down the search space for the optimal clusters in further iterations, *SemSplit* leverages the semantic similarity of type pairs. For this, the cosine similarity scores between all combinations of type pairs are calculated with the help of ConVec embeddings [9] that represent the semantics of the types quite well. For each subsequent iteration, the most similar type pairs (with ties broken arbitrarily) are merged and assigned a combined cluster label that serves as ground truth,

Table 2: Examples of *SemSplit* Clusters

Relation (algorithm)	*SemSplit* optimal clusters $\mathcal{C}_{opt}$	Homegeneity Score (# clusters) $\mathcal{C}_{orig}$	$\mathcal{C}_{max}(\mathcal{L})$	$\mathcal{C}_{opt}(\mathcal{N})$
created (OP)	{(artist, computer_game)}, {(artist, medium) (officeholder, movie)}, {(writer, movie) (writer, television) (artist, movie) (player, movie)}, {(company, computer_game)}, {(writer, fictional_character)}	0.14 (1)	0.29 (9)	**0.49** (5)
owns (SP)	{(company, airport) (organization, airport)}, {(sovereign, building)}, {(company, club) (company, company) (country, club)}	0.03 (1)	0.32 (6)	**0.52** (3)
isAffiliatedTo (HA)	{(artist, club)}, {(cricketer, club)}, {(player, club) (hockey_player, club) (hockey_player, university) (hockey_player, team)}, {(officeholder, club)}	0.09 (1)	0.43 (7)	**0.61** (4)

along with the labels of other type pairs. This process of reducing the number of clusters and updating the labels by merging type pairs is followed until all the type pairs have been progressively combined back together in one single cluster. Finally, the optimal number of clusters $\mathcal{N} = |\mathcal{C}_{opt}|$ is determined based on a cluster quality score calculated after each iteration.

3 Evaluation

Dataset. For this work, we have prepared a dataset that is derived from the Yago3 knowledge base and ontology. The class types (concepts) of all entities were extracted and the 53 most frequent concepts (having at least 10,000 entities associated with them) were taken into consideration. Thereafter, we extracted the fact triples that were comprised of entities associated with the chosen concepts. This resulted in a dataset of 1,492,078 triples with 917,325 unique entities and 31 relations.

Cluster Quality. To analyze the performance of *SemSplit*, we employed several algorithms to obtain the clusters in high dimensional vector space : Spectral (SP), Optics (OP) and Hierarchical Agglomerative (HA) clustering. The entity and relation vectors were obtained from pre-trained ConvE KG embeddings [3]. The quality of the clusters, and thereby, the resultant sub-relations is measured in terms of homogeneity score [5] that favors a clustering scheme where every cluster represents a unique dominant type. Table 2 shows examples of the optimal cluster configurations obtained by *SemSplit* with different algorithms. The performance of $\mathcal{C}_{opt}$ is compared to two baseline configurations — original relation cluster ($\mathcal{C}_{orig}$) and maximal splitting clusters ($\mathcal{C}_{max}$), in terms of the cluster homogeneity scores and the number of clusters. The results indicate that while it is favorable to split the original relation cluster into multiple sub-relation clusters, a naive splitting leading to maximal sub-relation clusters is also not the ideal solution for representing the fine-grained semantics. It can be seen that the $\mathcal{C}_{opt}$ clusters obtained by *SemSplit* perform better than both the original relation as well as the naive maximal splitting of the relations in terms of homogeneity scores, thus indicating the efficacy of the method for finding optimal fine-grained sub-relations.

4 Conclusion

In this paper, we have studied the need for relation disambiguation for knowledge graphs due to the inherent relation polysemy in these datasets. We have proposed a scalable, data-driven method *SemSplit* that automatically determines an optimal configuration for deriving sub-relations with concrete semantics. First experiments have confirmed the importance of learning fine-grained relation semantics for real-world data and shown promising results for *SemSplit* performance. We plan to perform a systematic analysis of the utility and impact of our method on semantic tasks, such as relation extraction and question answering over KGs.

References

1. 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)
2. Cui, W., Xiao, Y., Wang, H., Song, Y., Hwang, S.w., Wang, W.: KBQA: Learning Question Answering over QA Corpora and Knowledge Bases. Proc. of the VLDB Endowment **10**(5), 565–576 (2017)
3. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2D knowledge graph embeddings. In: Proc. of the 32nd AAAI Conf. on Artificial Intelligence (2018)
4. Do, K., Tran, T., Venkatesh, S.: Knowledge Graph Embedding with Multiple Relation Projections. In: Proc. of the 24th International Conf. on Pattern Recognition (2018)
5. Jain, P., Kumar, P., Chakrabarti, S., et al.: Type-sensitive Knowledge Base Inference without Explicit Type Supervision. In: Proc. of the 56th Annual Meeting of the Association for Computational Linguistics. pp. 75–80 (2018)
6. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning Entity and Relation Embeddings for Knowledge Graph Completion. In: Proc. of the 29th AAAI Conference on Artificial Intelligence (2015)
7. Mahdisoltani, F., Biega, J., Suchanek, F.: YAGO3:A Knowledge Base from Multilingual Wikipedias. In: Proc. of the 7th Biennial Conference on Innovative Data Systems Research (2014)
8. Min, B., Shi, S., Grishman, R., Lin, C.Y.: Ensemble Semantics for Large-scale Unsupervised Relation Extraction. In: Proc. of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (2012)
9. Sherkat, E., Milios, E.E.: Vector Embedding of Wikipedia Concepts and Entities. In: Proc. of the International Conference on Applications of Natural Language to Information Systems. pp. 418–428. Springer (2017)
10. Zhang, Z., Zhuang, F., Qu, M., Lin, F., He, Q.: Knowledge Graph Embedding with Hierarchical Relation Structure. In: Proc. of the 2018 Conference on Empirical Methods in Natural Language Processing. pp. 3198–3207 (2018)