# Transformer based Semantic Relation Typing for Knowledge Graph Integration

Sven Hertling[0000−0003−0333−5888] and Heiko Paulheim[0000−0003−4386−8195]

Data and Web Science Group, University of Mannheim, Germany
{sven,heiko}@informatik.uni-mannheim.de

**Abstract.** More and more knowledge graphs (KGs) are generated in various domains. Applications using more than one KG require an integrated view of those KGs, which, in the first place, requires a common schema or ontology. Merging schemas requires not only equivalence mappings between classes but also other semantic relations, like subclass, superclass, etc. In this paper, we introduce *TaSeR*, a **Tra**nsformer based model for **Se**mantic **R**elation Typing, which is able to decide which type of relation holds between two given classes. The approach can differentiate between equivalent class, sub-/superclass, part of/has part, cohyponym, and no relation at all. With the latter outcome, it is not only possible to refine given class alignments, but also filter incorrect correspondences. The models are trained based on examples from general knowledge graphs as well as fine-tuned on the test case at hand. The former models can be directly used to predict a relation without further training. We show that those models are able to outperform other approaches which solve a similar task. For the evaluation, a new measure is introduced which credits for proximal matches.

**Keywords:** Relation Typing · Ontology Matching · Knowledge Graph Integration · Transformers

## 1 Introduction

Data integration comprises different tasks, such as schema matching, entity matching, and data fusion. In most approaches, schema or ontology matching is carried out as a first step, trying to find correspondences between schema elements of different knowledge graphs (KGs).

Most existing ontology matching tools only identify equivalent classes in two schemas. This restriction, however, is very limiting when it comes to existing integration problems. For example, if one schema defines a class `Person` (without any subclasses), and another one defines a class `Artist` (without any superclasses), such tools can either not find any correspondence between the two classes, or erroneously identify them as equivalent.

Since both solutions – not finding a correspondence at all, or erroneously identifying equivalence – are suboptimal, we argue that schema matching tools should output a wider range of relations between classes in schemas.

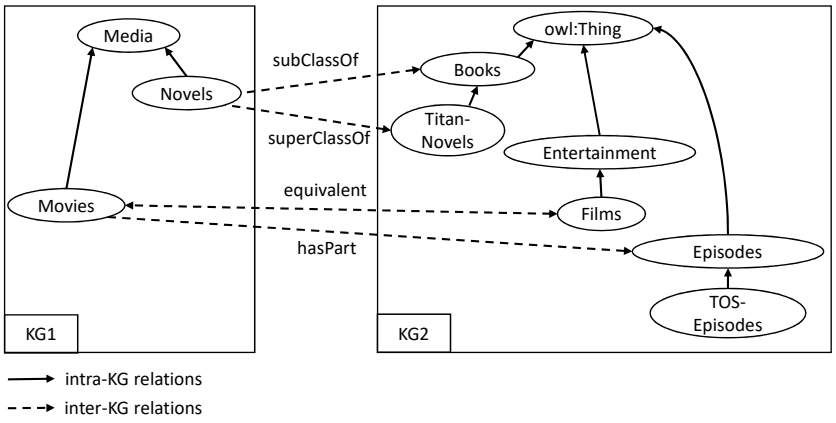


**Fig. 1.** Example of a matching task with semantic relations. All intra-KG relations represent rdfs:subClassOf.

In this work, we present *TaSeR*, a **Tra**nsformer based model for **Se**mantic **R**elation Typing. The input is a correspondence between two classes where the relation type is not yet determined. The task is to find out which type of relation actually holds between those inputs. We show that transformer based models can outperform state of the art approaches by fine-tuning them on general knowledge graphs such as DBpedia or Wikidata. The input KGs are further used to create a test case-specific model which often increases the performance.

Such models can be used to integrate ontologies of multiple KGs such as in the case of DBkWik [12]. Thousands of KGs are generated from Wikis by applying the DBpedia extraction framework. The result is a set of isolated KGs. When integrating their schemas, different relations can hold between their classes, not only equivalence.

Our contributions are: (1) a transformer based model for semantic relation prediction, (2) an existing dataset transformed to a new track at the OAEI, and (3) an improved evaluation measure for this task.

The paper is structured as follows: In the next section, we first define the task and present the relations our approach is able to predict. Afterwards related work is discussed. The following approach section is divided into training data generation, model tuning, and tuning on test cases. We evaluate the models on a gold standard in section 5 and conclude with an outlook on future work.

## 2 Task Definition

Figure 1 shows an example of the task. Given two input KGs KG1 and KG2, the task is to find correspondences between them that are enriched with semantic relations. Thus only inter-KG relations are of interest in this work. Nevertheless, it is possible to use intra-KG relation as an additional training signal which is later discussed and evaluated.

**Table 1.** Mapping of linguistic relations to correspondence types.

| Relation | DL | RDF representation | Correspondence | example |
|---|---|---|---|---|
| Synonym | $A \equiv B$ | owl:equivalentClass | equivalence | home <->domicile |
| Hyponym | $A \sqsubseteq B$ | rdfs:subClassOf | subsumed | apple ->fruit |
| Hypernym | $A \sqsupseteq B$ | rdfs:subClassOf$^{-1}$ | subsume | fruit->apple |
| Meronym | - | dcterms:isPartOf | part of | knee ->leg |
| Holonym | - | dcterms:hasPart | has a / has part | leg ->knee |
| Cohyponym | - | skos:related | cohyponym | dalmatian <->poodle |

The input is an alignment $A$ consisting of correspondences defined as a 4-tuple $< x, y, r, c >$ where $x$ and $y$ are entities of KGs one and two, $r$ represents the relation which holds between the entities (e.g., $=$ or $\sqsubseteq$), and finally a confidence value $c \in [0, 1]$. The output is an alignment $A'$ with the same pairs $(x, y)$, but potentially different relations and confidence scores.

In the example knowledge graph on the right, there is no equivalent class to novels. But many inter-KG correspondences can still be created e.g. novel is a subclass of book. Thus it is important to not only use equivalence relations between classes for integrating knowledge graphs.

The following semantic relations can be used between KGs: equivalence, sub-/super class of, part of/has part, cohyponym, and no relation at all (to filter correspondences). We now discuss how the relations should be used, their semantics, and how they relate to each other. Given the fact that class A is a subclass of class B means that all instances of A are also instances of class B (by RDFS semantics). Thus A is the more specific class whereas B is the more general class. The inverse relation is called superclass of. Whenever A is a subclass of B then B is always a superclass of A. The part of relation which represents a composition of concepts is often miss-used with the subclass relation [1]. As an example, a leg is a part of the body and not a subclass because each instance of a leg is not an instance of type body. The inverse relation is called has part or has a. Based on the definition of [1], the cohyponym relation consists of a pair of concepts that are both direct subclasses of a common superclass (e.g. A and C are related iff A is a subclass of B and C is a subclass of B).

## 3 Related Work

The first area of related work is *knowledge graph completion* [23]. The main task is to find links between entities that are true, but not explicitly stated in a KG. Technically, the systems are required to retrieve a ranked list of entities that are most likely for a given source entity and a relation - essentially object entity prediction. In a similar way, the subject entity should be predicted for a given target entity and relation. Some approaches also use the task of predicting the relation given both source and target entities as an additional training task.

[1] https://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/

The approaches can be further divided into embedding-based approaches such as RESCAL [22], TransE [3], and ComplEx [29] as well as rule-based systems such as AnyBURL [18] and AMIE [7]. The main difference is that those approaches do only predict relations that are seen during training. This means that any relation which appears in the given KGs might be a candidate for the prediction. But in the presented use case the target relation is restricted to class relations.

The next field of related is the *natural language processing (NLP)* community. Given resources such as Wordnet [20] the relations between synsets can also be seen as semantic relations between classes in KGs. Table 1 shows the connection between linguistic relations used in Wordnet and relation types used in correspondences. [8] shows for example an approach to detect semantic relation between two given words/phrases using word embeddings [19]. The focus lies on the symmetric and asymmetric properties of the relations. Most of the datasets used in this work are derived from Wordnet.

[26] introduces a shared task about semantic relations. Based on this dataset, KEML, a meta learning framework for predicting lexical relations, is introduced [32].

Finally, the work in this paper falls into the area of *Knowledge Graph matching and fusion/integration* [13]. The closest related work is STROMA [1] which follows an enrichment strategy by refining given correspondences with a more meaningful relation than e.g. equivalence. The approach consists of five techniques consisting of linguistic approaches as well as the use of background knowledge (such as Wordnet).

The Tifi [5] approach combines multiple taxonomies together and uses lexical and graph features to determine if a given subsumption relation holds or not. The difference to our work is that the focus lies on the cleaning of those hierarchies and thus only intra-KG relations are analyzed. Furthermore, the prediction is binary and indicates only if two given concepts are subsumed. BERTSubs [4] focuses also on this relation. The idea is to embed concepts with BERT [6] by transforming classes and other OWL constructs such as restrictions into a textual representation. They evaluate their approach based on rankings. Given a concept A, they rank all other entities based on how likely it is that A is a subclass of this concept.

Except for STROMA [1], there is no system that can directly be applied to a given dataset to create correspondences between classes which includes semantic relations. Most of the approaches which use transformer based models, show specific techniques on how to train such models but do not create a usable model based on existing data. With TaSeR, we close this gap and present a model which is able to predict the relation type and can optionally be further fine-tuned on the test case at hand.

## 4 Approach

The overall approach is shown in Figure 2. Given two knowledge graphs KG1 and KG2, the task is to find an alignment A consisting of correspondences with

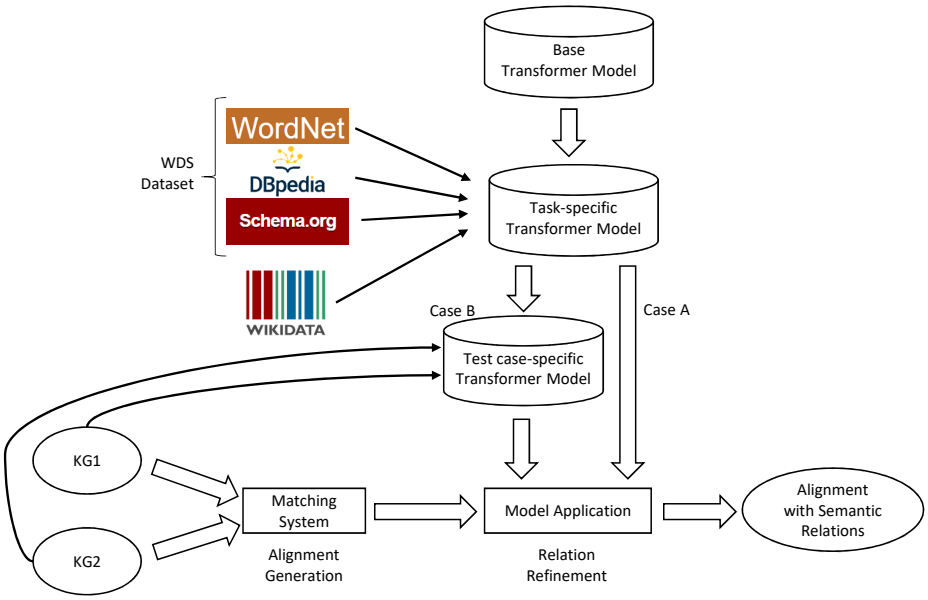


**Fig. 2.** Overall approach of TaSeR.

semantic relations (e.g. subsumption, equivalence, part of, etc). TaSeR is a two-step approach. The first step is to generate candidate correspondences between classes where the relation is not yet specified (equivalence by default). This can be achieved by running various matching systems which are able to produce class correspondences. In a second step, the relation is further refined with the help of a transformer model. Two cases are analyzed: Case A) a task-specific model which is directly applied to the candidate set of correspondences. Such a model is trained on external datasets like DBpedia, Wordnet, and Schema.org. Case B) on top of the task-specific model, a further fine-tuned model is trained on test case data such as intra-KG subclass relations. In the further sections, each step is explained in more detail.

### 4.1 Creating Candidates / Blocking

Our approach TaSeR needs input class correspondences to further refine the relation. An advantage is that TaSeR is also able to filter correspondences in case they are not related at all. This allows using candidate generators that return many correspondences which may also contain incorrect correspondences (aligned classes that should not be connected at all, e.g. car and actor).

In general, any matching system or any combination of them can be used to generate candidates. A low threshold should be used to increase recall and allow the system to include also near matches. Due to the fact that this is not the focus of this work, LogMap [15] is chosen as it is a state of the art ontology matching system.

**Table 2.** Analysis of the training datasets which are used to create the task-specific model.

| dataset | equivalence | sub-/superclass of | part of/has part | cohyponym | negatives |
|---|---|---|---|---|---|
| Wordnet | 215,672 | 84,501 | 9,092 | 44,329 | 410,960 |
| DBpedia | 58 | 246 | 0 | 198 | 0 |
| Schema.org | 0 | 1,421 | 0 | 826 | 0 |
| Wikidata | 927 | 127,659 | 230,897 | 0 | 0 |

### 4.2 Training Data for Task-specific Model

In this section, the creation of the task-specific model is described. This is one difference in comparison to other approaches like [4] which present techniques on how to train models for subsumption prediction but do not provide a concrete model which can be applied to a dataset at hand.

The overall task is to predict relations between different knowledge graphs (inter-KG relations) but for training inter- and intra-KG relations can be used. The reason is that no features are derived which depends on the fact that the training originates from two different sources. Creating such training sets requires KGs which have a lot of relations useful for the models such as `rdfs:subClassOf` or `skos:hasPart` etc. In the following, we describe how each KG is used to create the training dataset. Wherever possible, for all kinds of relations examples are generated. Table 2 shows an analysis of all datasets and their counts for each type of relation.

**Wordnet** [20]: Other approaches like STROMA [1] use Wordnet as their main background knowledge because of the quality and coverage of concepts. Therefore in this work, Wordnet is also included.

Wordnet is structured by so-called synsets. They contain different lemmas which describe the same concept (synonyms). Each synset can have multiple relations to other synsets such as hypernym, hyponym, and holonym (see also Table 1). Generating examples for the equivalence relation is achieved by using all possible combinations between the words in a synset. This also includes the tuple where the word is equivalent to itself. In each synset the words "are ordered by estimated frequency of use"[2] e.g. "dog" is the most used word but "Canis familiaris" is also a synonym for it. The sub-/superclass examples are produced by following the hyponym/hypernym relations between the synsets. As a label, the first word is chosen as a representation of the synset. For the part of relations, the same approach is applied but following the meronym/holonym relations. The second last relation is called cohyponym. Examples are generated by listing all hypernyms of a given synset and creating combinations between them. Usually, there are many hypernyms of a given synset and thus only a maximum of five examples are extracted per synset to not overemphasize this kind of relation. The last relation is an exception because its semantics is that no relation holds between the given concepts. The training data for this kind of negative example

[2] As stated in https://wordnet.princeton.edu/documentation/wn1wn

is generated by randomly sampling two synsets. It was ensured that no false negative is included by checking that the two synsets do not relate with either hyponym/hypernym and meronym/holonym relations.

**DBpedia** [2]: Beyond Wordnet, other KGs like DBpedia are used for training. The class hierarchy in DBpedia is manually created and curated in the mappings wiki[3]. The reason why such a mappings wiki exists is that the class information for each wiki page originates from the MediaWiki templates containing the text "infobox" which is usually rendered at the top right of the page to highlight some key facts of the concept. But many templates need to be mapped and processed to create a reasonable KG out of it e.g. the template infobox_aircraft_type[4] is mapped to the ontological class aircraft. Such a manual-created taxonomy is helpful in training subsumption relationships.

The training examples for equivalence relation from DBpedia are extracted by querying the official SPARQL endpoint[5] with the query in Listing 1.1 (the from statement was necessary to retrieve all results without duplicates). All English labels attached to the classes are queried in case they exist (optional statement). In cases where the label does not exist, the URI fragment[6] is used instead. In those fragments, no whitespaces are allowed and thus the text is split by camel case[7]. Hyphens and underscores are replaced by whitespace. Most of those equivalence mappings map two different datasets e.g. DBpedia to Schema.org and thus no label information is available for the external one. The YAGO mappings are skipped because the mapping only changes the URI but no label or textual representation. Still, in the resulting equivalence dataset of DBpedia, most examples map the exact same label to each other which is intentional. For sub-/superclass relationships, the same query as before is used but the property is replaced with `rdfs:subClassOf`. Due to the fact that there is no RDF property to express a superclass relation, the source and target entity of the subclass relation are switched. In the same way as for Wordnet, the examples with the cohyponym relation are created (at a maximum of five for each superclass). In the DBpedia ontology no further part of or has part relations are defined.

**Schema.org** [9]: Other public common knowledge graphs such as the first version of YAGO [28] also use Wordnet as their top-level ontology which is then used as a type system for all Wikipedia pages. Starting from YAGO 4 [24] they changed the top-level ontology to Schema.org. It is a taxonomy primarily used for encoding knowledge in websites with RDFa or Microdata. Consequently, this top-level ontology is also included as one source of training data.

For Schema.org the corresponding CSV (comma separated values) file[8] is downloaded and all subclass relations are extracted. Again, superclass and cohy-

[3] http://mappings.dbpedia.org
[4] http://mappings.dbpedia.org/index.php/Mapping_en:Infobox_aircraft_type
[5] https://dbpedia.org/sparql
[6] The URI fragment is extracted by using the text after the last slash or hashtag.
[7] https://en.wikipedia.org/wiki/Camel_case
[8] https://schema.org/version/latest/schemaorg-current-http-types.csv

**Listing 1.1.** DBpedia query to retrieve all equivalence relations.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
SELECT *
FROM <http://dbpedia.org>
WHERE {
    ?left owl:equivalentClass ?right.
    OPTIONAL{
        ?left rdfs:label ?leftlabel.
        ?right rdfs:label ?rightlabel.
        FILTER (LANG(?leftlabel) = "en" &&
                LANG(?rightlabel) = "en" )
    }
}
```

ponym examples are generated as in DBpedia. No part of or equivalence relations are available.

**WDS Dataset**: There are not many relations extracted from DBpedia and Schema.org. Thus a combination of Wordnet, DBpedia, and Schema.org is created. This WDS dataset will later be used to train the corresponding models with a rather limited set of data. In the next section, a larger dataset is presented which is extracted from Wikidata.

**Wikidata** [31]: Usually, more training data helps to improve the classification. Thus Wikidata is used as an additional dataset because it is one of the largest KGs available. Initially, the official endpoint[9] of Wikidata is used to retrieve all subclass and part of relations. Due to a large number of classes and relations, the endpoint runs into timeouts. Therefore the endpoint of Virtuoso[10] is used. The corresponding query is shown in Listing 1.2 which uses the P279 predicate to retrieve all subclass relations together with all labels. Due to the availability of labels and the size of Wikidata, the labels are marked as mandatory in the query. The high number of results forced us to execute all queries with limits and offsets to iterate over pages of results. In a similar way, the part of/has part relation examples are collected (by replacing the property with wdt:P361). For the equivalence relation wdt:P1709 is used (equivalent property to owl:equivalenceClass). Similarly to DBpedia, one concept (either source or target) does not have literals included in Wikidata. Thus the URI fragment is used again.

### 4.3 Training of Task-specific Model

Given the general training dataset constructed in the previous section, the question remains how a transformer based model is trained. Many of the pre-trained transformer models such as BERT [6], RoBERTa [17], or Albert [16] are trained on large amounts of text. The training objective is called *masked language modeling* where the model should predict a masked token given that it can attend to

[9] https://query.wikidata.org
[10] https://wikidata.demo.openlinksw.com/sparql

**Listing 1.2.** Wikidata query to retrieve all subclass relations.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
SELECT *
FROM <http://dbpedia.org>
WHERE {
    ?left wdt:P279 ?right.
    ?left rdfs:label ?leftlabel.
    ?right rdfs:label ?rightlabel.
    FILTER (LANG(?leftlabel) = "en" &&
            LANG(?rightlabel) = "en" )
    LIMIT 10000
    OFFSET 0
}
```

the surrounding tokens bidirectionally. In this work, those models are used but with a different objective. The overall task is to classify a given example into one of seven classes (which represent the different relation types in Table 1 plus the negative class). Therefore each model gets an additional dropout and linear layer on top (this is also called the head of the model) which transforms the output of the model into seven neurons (each representing a relation type). For each example during training exactly one of them should result in the value of one and all others to zero. The binary cross entropy loss is used to reduce this error. Each model gets this classification head and is trained with the given examples. In addition, it is possible to only finetune the head and freeze all layers of the underlying model but in this work, we adjust all weights.

In the following, the input representation of the training concepts is described. For each class, a textual representation is generated. It is usually the `rdfs:label` of the class. In case there is more than one label, each label is transformed into a training example (cross-product in case both concepts have multiple labels). If no label is available, then the URI fragment is extracted and post-processed (camel case, hyphens, and underscores as in section 4.2). The extraction of textual representations can be further customized.

To separate the textual representation of the source and target class, the pre-trained special token `[SEP]` is used as a separator. Thus a training example for the relation has part can look like "body[SEP]leg". This is similar to the BERTSubs [4] approach called isolated class (IC). They showed that adding path context (PC) or breadth-first context (BC) does not improve the result much (e.g. 0.002 for inter-ontology named subsumption prediction on the HeLiS-FoodOn test case for hits at one) or even make it worse (e.g. 0.016 less in hits at five for the same task).

In comparison to BERTSubs we do not only rely on BERT as one prominent representative of transformer based language models, but also use and evaluate more common models such as albert-base-v2, bert-base-uncased, distilbert-base-uncased, and roberta-base.

For the actual training, the transformers library of huggingface [33] is used. In more detail, the trainer class is executed with the default parameters except

for the batch size to create a first model. The value for the batch size is increased as much as possible to A) decrease the training time and B) update the weights based on more examples with different target relations. Thus the weights are not changed drastically by a batch that e.g. contains only examples with the equivalence relation. Each model has different memory requirements and thus an approach is developed to automatically select the highest possible batch size given the dataset, model, and GPU. It works as follows: The data set is first sorted by the overall length of the two input texts such that the longest texts appear at the beginning. Afterwards different batch sizes $b$ are tried out by starting with four and multiplying by two after each trial (iterating over the powers of two). The dataset is cut to the top $b$ examples and only trained for one step in order to quickly test if the memory of the GPU is large enough to store everything. If the test works fine, the batch size is increased. If instead an out-of-memory error is detected, the batch size is divided by two to get the maximum working batch size. This approach allows for higher batches sizes than trying out the theoretical maximum which is a batch where each example exhausts the maximum number of tokens. In case the dataset consists only of rather short texts, this is an improvement because the batch size could be increased.

In addition to the default hyperparameter (HP) of the trainer, an HP tuning is executed. The training of transformer based models requires a lot of time and thus population based training [14] (PBT) is selected as the HP search algorithm. The underlying idea is based on evolutionary algorithms. In the beginning, there is a set of models with random HPs based on an initial distribution. After training and evaluating the models for some number of batches, the hyperparameters of good performing models are used as a replacement for models which perform worse. With such an approach the HP can change over time during training. Thus it is necessary to use the trained model directly because there is no fixed set of HP such that one can tune a new model based on it.[11]

### 4.4 Training of Test Case Specific Model

In the previous section, a task-specific model is created. It is trained on general knowledge graphs and can thus be used directly to predict a semantic relation for a given pair of classes. But given the input KGs there is more training data available. The overall task is to predict inter-KG relationships between classes and thus intra-KG relations can be used to further fine-tune the model to a given test case (consisting of two KGs). This will incorporate some knowledge about the concepts which are later used in the prediction.

[11] The number of trained models is fixed to ten and the following hyperparameters are tuned: learning rate (loguniform between 1e-6 and 1e-4), train epochs (between 1 to 10), seed (uniform distribution from 1 to 40), batch size (choice of 4, 8, 16, 32, 64, 128 until the maximum possible batch size). The mutations of HPs are defined by: weight decay (uniform between 0.0 and 0.3), learning rate (uniform between 1e-5 and 5e-5), batch size (choice of 4, 8, 16, 32, 64, 128 until the maximum possible batch size).

**Table 3.** Analysis of the evaluation datasets.

| dataset | equivalence | subclass of | superclass of | part of | has part | cohyponym |
|---|---|---|---|---|---|---|
| g1-web | 275 | 29 | 26 | 2 | 3 | 4 |
| g2-diseases | 316 | 27 | 11 | 0 | 1 | 0 |
| g3-text | 70 | 425 | 267 | 0 | 0 | 0 |
| g4-furniture | 13 | 107 | 4 | 0 | 11 | 1 |
| g5-groceries | 29 | 14 | 113 | 0 | 2 | 11 |
| g6-clothing | 10 | 0 | 124 | 0 | 8 | 0 |
| g7-literature | 12 | 18 | 52 | 1 | 0 | 0 |

To select training examples for the subsumption, we use all triples in each of the input KG where rdfs:subClassOf is used as a relation. In addition, the source and target of this relation are switched to also generate examples for the inverse relation. Similarly, for the equivalence relation the owl:equivalentClass predicate is used. For all other relations, we use the vocabulary terms defined in Table 1.

In most of the input KGs, only a taxonomy is defined. This results in training examples consisting only of subsumption relations. Fine-tuning the task-specific model with such a training set causes the model to adapt to these relations. Thus another fine-tuning dataset is created. In addition to the examples extracted from the input KGs, the same number of equivalence examples are sampled from the corresponding task-specific training dataset. This fine-tuning approach is later called in the evaluation "test case +".

## 5 Evaluation

In this section, the approach TaSeR is evaluated. Due to the fact that it is a two-step approach, we first evaluate only the prediction of semantic relations and later define a new measure for evaluating the whole system. In the next section, the corresponding evaluation datasets are selected.

### 5.1 Datasets

Most of the existing datasets are not suitable to evaluate our approach because they do only evaluate equivalence relationships such as the Ontology Alignment Evaluation Initiative (OAEI) [25]. Only the complex track allows the matching system to create correspondence which are composed of multiple concepts. But this is not the same task as presented in this paper where the relation between two concepts should be evaluated. Thus the dataset of STROMA [1] is used for evaluation. It contains seven pairs of ontologies from different domains. Each dataset was in a different format such as tabular separated files and various OWL formats. Furthermore, the identifier for concepts is not defined by URIs but by the path of labels from the given concept to the top concept. Due to the possibility of multiple inheritance, this is not unique.

Therefore, all datasets are transformed into the OAEI standard format which consists of a source and target KG in RDF/XML. The reference alignment contains correspondences where each one consists of a source and target URI which needs to be present in the input KGs. This was explicitly checked and errors (in the KGs as well as in the reference alignment) were corrected.

All characteristics of the dataset are included in Table 3. It shows for each test case and semantic relation the number of correspondences included in the reference alignment. All correspondences are directed and thus the number of sub-/superclass and part of/has part are not symmetric.

### 5.2 Results

All trained models are evaluated on the before mentioned dataset. The experiments are executed on NVIDIA Tesla V100 graphic cards and Intel Xeon Gold 6230 processors (2.1GHz). Due to the fact that the models are either trained on external datasets or on the input KGs, no further train test split needs to be created. The provided gold standard is solely used for testing. In this section, we evaluate the predictions of semantic relations given the class correspondences. Thus not using any candidate generation step. In essence, this boils down to a multi-class classification task.

Table 5 shows the overall results for each trained model. The runtime of each model is always below one minute (exact runtimes are given in the supplementary material). We differentiate between the base model, the dataset, and the fine-tuning method. The WDS dataset consists of Wordnet, DBpedia, and Schema.org whereas WDS + Wikidata represents the same dataset but adds all training examples from Wikidata (this increases the number of examples drastically). To compare all models, the micro averaged $F_1$ is computed across all classes which in this case represents the kind of semantic relation. Micro averaged precision and recall are the same as $F_1$ because *all* false instances are counted. Thus the following equation holds true (c represents the classes): $\sum_c FP_c = \sum_c FN_c$.

For the test cases G1 and G2, the task-specific model outperforms the STROMA baseline by 0.038 and 0.053 $F_1$. This is achieved by directly applying the task-specific model to the dataset without any knowledge of the task. This shows the overall usability of such models. Starting from G3 one can see that the fine-tuning on the test cases is really helpful in deciding which relation holds true between the classes. Except for G6, it turns out that fine-tuning on the test case together with samples from the training dataset (the one where the corresponding task-specific model is trained on) is useful (higher $F_1$ for test case + than for test case). Only training on the subsumption relations of the input KGs usually gives too high weights to this relation such that the model overly often predicts those relations. The F1 gains are significant (95% confidence interval) for test cases G2, G3, and G7.

We finally select the best model which works without test case fine-tuning such that the research community can directly apply the model without any changes. For the model selection, all $F_1$ scores are added and the model with the

**Table 4.** Two aggregated confusion matrices for DistilBERT over all tasks.

| | | Prediction WDS | | | | | | Prediction WDS+Wikidata | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | equivalence | subclass of | superclass of | part of | has part | cohyponym | equivalence | subclass of | superclass of | part of | has part | cohyponym |
| Actual | equivalence | 654 | 24 | 15 | 8 | 15 | 9 | 298 | 93 | 55 | 118 | 156 | 5 |
| | subclass of | 173 | 428 | 9 | 0 | 0 | 10 | 23 | 475 | 15 | 93 | 8 | 6 |
| | superclass of | 202 | 20 | 358 | 0 | 2 | 15 | 27 | 24 | 386 | 30 | 120 | 10 |
| | part of | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 |
| | has part | 20 | 2 | 0 | 0 | 0 | 3 | 1 | 7 | 0 | 3 | 13 | 1 |
| | cohyponym | 8 | 2 | 5 | 0 | 0 | 1 | 1 | 2 | 3 | 5 | 5 | 0 |

highest score is selected. It turns out, that more training data or hyperparameter optimization is not always helpful because DistilBERT trained on the WDS dataset is achieving the overall best performance (not regarding test case fine tuning). The training time of the task-specific models can take up to 48 hours. The best model is shared on the huggingface model hub[12].

Table 4 shows two confusion matrices (values summed all over all tasks) of DistilBERT models. The left one is only trained on the WDS dataset (the chosen best model). Equivalence, subclass of, and superclass of can be detected quite well whereas part of and has part is not easy. The right confusion matrix is the result of the model trained on WDS and Wikidata (which contains more part of relations), therefore, the predictions for these types of relations are better.

### 5.3 Evaluation of the Complete System

To evaluate the complete system, the candidate generation step is added and all resulting relations of the correspondences are further refined with TaSeR. As already mentioned in [1], the system might find correspondences that are not in the reference alignment but still valid (especially for subclass relations) e.g. the system finds that movies are a subclass of entertainment (cf. Figure 1). Such statements would count as false positives which is incorrect.

Therefore we propose a new measure to circumvent this problem. The following closure approach is applied to each reference alignment as well as system alignment. If the alignment maps class A to class X with a subclass relation, then additionally all subclasses of A are also subclasses of X as well as all of all superclasses of X. Thus many more (implicit) relations between concepts are added. The equivalence relation is handled as two subclass relations in both directions. After the reference and system alignment are processed, precision, recall, and f-measure are computed as usual. We call those measures $P_{closure}$, $R_{closure}$, and $F_{1-closure}$. With such a definition, it is also possible to extend it to the case where the gold standard is not complete but partial.

[12] https://huggingface.co/dwsunimannheim/TaSeR

**Table 5.** Micro averaged $F_1$ results of all trained models differentiated by base model, datasets used for training, and kind of fine-tuning. STROMA serves as a baseline that is optimized for this gold standard. The top three values are printed in bold.

| Base model | Dataset | Fine-tuning | G1 | G2 | G3 | G4 | G5 | G6 | G7 |
|---|---|---|---|---|---|---|---|---|---|
| Albert | WDS | Task | 0.422 | 0.673 | 0.906 | 0.125 | 0.314 | 0.261 | 0.651 |
| | | Test case | 0.127 | 0.096 | 0.886 | 0.728 | 0.621 | 0.824 | 0.771 |
| | | Test case + | 0.434 | 0.746 | 0.912 | 0.721 | 0.663 | 0.761 | **0.940** |
| | WDS +HP Tuning | Task | 0.540 | 0.758 | 0.886 | 0.213 | 0.361 | 0.211 | 0.566 |
| | | Test case | 0.091 | 0.096 | 0.895 | 0.757 | 0.645 | 0.810 | 0.807 |
| | | Test case + | 0.575 | 0.797 | 0.878 | **0.779** | 0.710 | 0.817 | 0.687 |
| | WDS +Wikidata | Task | 0.121 | 0.273 | 0.941 | 0.375 | 0.485 | 0.451 | 0.627 |
| | | Test case | 0.118 | 0.099 | 0.932 | 0.699 | 0.680 | 0.838 | 0.783 |
| | | Test case + | 0.248 | 0.727 | 0.915 | 0.757 | 0.704 | 0.824 | **0.880** |
| Bert | WDS | Task | 0.534 | 0.752 | 0.919 | 0.228 | 0.385 | 0.261 | 0.639 |
| | | Test case | 0.109 | 0.093 | 0.932 | 0.699 | 0.663 | 0.831 | 0.759 |
| | | Test case + | 0.348 | 0.718 | 0.927 | **0.779** | 0.728 | 0.852 | 0.723 |
| | WDS +HP Tuning | Task | 0.560 | 0.741 | 0.896 | 0.250 | 0.385 | 0.268 | 0.651 |
| | | Test case | 0.103 | 0.087 | 0.913 | 0.699 | 0.663 | 0.817 | 0.735 |
| | | Test case + | 0.440 | 0.758 | 0.904 | 0.691 | 0.716 | 0.845 | 0.747 |
| | WDS +Wikidata | Task | 0.136 | 0.400 | **0.944** | 0.500 | 0.485 | 0.331 | 0.663 |
| | | Test case | 0.124 | 0.115 | **0.946** | 0.772 | 0.680 | 0.852 | 0.783 |
| | | Test case + | 0.428 | 0.775 | **0.944** | 0.691 | 0.710 | 0.845 | **0.831** |
| DistilBERT | WDS | Task | **0.767** | **0.828** | 0.929 | 0.154 | 0.420 | 0.268 | 0.590 |
| | | Test case | 0.100 | 0.085 | 0.883 | 0.699 | 0.698 | **0.852** | 0.783 |
| | | Test case + | 0.378 | 0.727 | 0.924 | 0.662 | **0.746** | 0.838 | 0.771 |
| | WDS +HP Tuning | Task | 0.720 | 0.786 | 0.925 | 0.191 | 0.373 | 0.197 | 0.566 |
| | | Test case | 0.094 | 0.087 | 0.928 | 0.721 | 0.645 | **0.852** | 0.783 |
| | | Test case + | 0.372 | 0.783 | 0.919 | 0.706 | **0.769** | 0.824 | 0.747 |
| | WDS +Wikidata | Task | 0.183 | 0.510 | **0.946** | 0.331 | 0.367 | 0.317 | 0.675 |
| | | Test case | 0.115 | 0.144 | 0.942 | 0.728 | 0.704 | **0.866** | 0.771 |
| | | Test case + | 0.419 | 0.735 | 0.934 | 0.743 | 0.734 | 0.838 | 0.771 |
| Roberta | WDS | Task | **0.799** | **0.820** | 0.916 | 0.272 | 0.373 | 0.204 | 0.554 |
| | | Test case | 0.118 | 0.099 | 0.865 | 0.750 | 0.550 | 0.831 | 0.675 |
| | | Test case + | 0.363 | 0.445 | 0.930 | 0.735 | 0.627 | 0.810 | 0.759 |
| | WDS +HP Tuning | Task | **0.799** | **0.845** | 0.841 | 0.250 | 0.296 | 0.155 | 0.494 |
| | | Test case | 0.124 | 0.099 | 0.883 | 0.728 | 0.669 | 0.803 | 0.663 |
| | | Test case + | 0.472 | 0.727 | 0.898 | 0.728 | 0.669 | 0.824 | 0.747 |
| | WDS +Wikidata | Task | 0.021 | 0.090 | 0.886 | 0.287 | 0.479 | 0.275 | 0.518 |
| | | Test case | 0.115 | 0.101 | 0.883 | 0.728 | 0.680 | 0.838 | 0.771 |
| | | Test case + | 0.268 | 0.811 | 0.882 | 0.743 | **0.740** | 0.831 | 0.771 |
| STROMA | - | - | 0.761 | 0.792 | 0.854 | **0.765** | 0.716 | **0.866** | 0.807 |

We evaluate the complete system with this measure and show the results in Table 6. The recall can be increased by using candidate generation models which return more correspondences and are thus more recall-oriented (due to the fact that TaSeR is also able to filter correspondences that are not related at all).

**Table 6.** Evaluation of complete system (micro averaged).

| Measure | G1 | G2 | G3 | G4 | G5 | G6 | G7 |
|---|---|---|---|---|---|---|---|
| $P_{closure}$ | 0.701 | 0.401 | 0.422 | 0.688 | 0.946 | 0.715 | 0.825 |
| $R_{closure}$ | 0.815 | 0.695 | 0.162 | 0.630 | 0.658 | 0.305 | 0.371 |
| $F_{1-closure}$ | 0.754 | 0.508 | 0.234 | 0.658 | 0.776 | 0.428 | 0.512 |

## 6 Conclusion and Outlook

In this paper, we presented TaSeR, a transformer based model for semantic relation prediction. Most state of the art matching systems do only output equivalence relations for classes which is too imprecise to use it for KG integration. TaSeR is based on two steps: (1) Generate candidate correspondences with any matching system, and (2) predict the most appropriate relation between them.

In addition, we transformed a given gold standard into the OAEI format and plan to submit a new track in 2023. We show that transformer models trained on general knowledge graphs such as Wordnet, DBpedia, and Schema.org can outperform STROMA which is one strong system developed especially for the gold standard used here. It could be shown that fine-tuning on the test case (by using intra-KG relations) can further improve the prediction.

In the future, we plan to extend our model to also deal with multilingual input. This can be achieved by using multilingual transformer models such as bert-base-multilingual-uncased together with additional training data. Those can be created from datasets like EuroWordNet [30], Wiktionary [27], and BabelNet [21]. Other datasets used in this approach like Wikidata also include labels for a concept in different languages which would also help. In such a case, one can also train the model to find semantic relations between classes of different languages e.g. car (English) is a subclass of vehículo (Spanish).

Another direction for future work is to increase the number of possible semantic relations. Probably the most interesting relation is the "is a" relation which is different from the subclass relationship because the former connects an instance to a class whereas the latter should only connect classes. In essence, this boils down to detecting if a concept is an instance or a class. Training data for this relation is not directly available in Wordnet, but in other knowledge graphs like DBpedia or Wikidata. It is still unclear how the training of such models should look like and how much data is actually necessary to achieve good results. Such a model could be used for WebIsALOD [11] to create a proper ontology.

In this work, we mainly used the dataset given by STROMA [1]. Other datasets do not contain many semantic relations but only one or two as in the case of BERTSubs [4]. Still, we would like to include more datasets in the future to get a higher variety of datasets even though some of them might only care about subsumption.

All supplementary materials can be found at figshare [10].

## References


1. Arnold, P., Rahm, E.: Enriching ontology mappings with semantic relations. Data Knowl. Eng. **93**, 1–18 (2014). https://doi.org/10.1016/j.datak.2014.07.001
2. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. In: The Semantic Web: 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007+ ASWC 2007, Busan, Korea, November 11-15, 2007. Proceedings. pp. 722–735. Springer (2007)
3. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. Advances in neural information processing systems **26** (2013)
4. Chen, J., He, Y., Geng, Y., Jimenez-Ruiz, E., Dong, H., Horrocks, I.: Contextual semantic embeddings for ontology subsumption prediction. arXiv preprint arXiv:2202.09791 (2022)
5. Chu, C.X., Razniewski, S., Weikum, G.: Tifi: Taxonomy induction for fictional domains. In: The World Wide Web Conference (WWW). pp. 2673–2679 (2019). https://doi.org/10.1145/3308558.3313519
6. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Burstein, J., Doran, C., Solorio, T. (eds.) NAACL-HLT. pp. 4171–4186. Association for Computational Linguistics (2019). https://doi.org/10.18653/v1/n19-1423
7. Galárraga, L., Teflioudi, C., Hose, K., Suchanek, F.M.: Fast rule mining in ontological knowledge bases with amie+. The VLDB Journal **24**(6), 707–730 (2015)
8. Glavaš, G., Ponzetto, S.P.: Dual tensor model for detecting asymmetric lexico-semantic relations. In: Conference on Empirical Methods in Natural Language Processing, EMNLP. Association for Computational Linguistics (2017)
9. Guha, R.V., Brickley, D., Macbeth, S.: Schema. org: evolution of structured data on the web. Communications of the ACM **59**(2), 44–51 (2016)
10. Hertling, S.: TaSeR (3 2023). https://doi.org/10.6084/m9.figshare.21750338.v1, https://figshare.com/articles/dataset/TaSeR/21750338
11. Hertling, S., Paulheim, H.: Webisalod: providing hypernymy relations extracted from the web as linked open data. In: ISWC. pp. 111–119 (2017)
12. Hertling, S., Paulheim, H.: Dbkwik: extracting and integrating knowledge from thousands of wikis. Knowledge and Information Systems **62**(6), 2169–2190 (2020)
13. Hertling, S., Paulheim, H.: The knowledge graph track at oaei: Gold standards, baselines, and the golden hammer bias. In: The Semantic Web: 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31–June 4, 2020, Proceedings 17. pp. 343–359. Springer (2020)
14. Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W.M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., et al.: Population based training of neural networks. arXiv preprint arXiv:1711.09846 (2017)
15. Jiménez-Ruiz, E., Grau, B.C., Zhou, Y., Horrocks, I.: Large-scale interactive ontology matching: Algorithms and implementation. In: ECAI. vol. 242, pp. 444–449 (2012)
16. Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R.: Albert: A lite bert for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942 (2019)
17. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019)

18. Meilicke, C., Chekol, M.W., Ruffinelli, D., Stuckenschmidt, H.: Anytime bottom-up rule learning for knowledge graph completion. In: International Joint Conference on Artificial Intelligence (IJCAI). pp. 3137–3143 (2019). https://doi.org/10.24963/ijcai.2019/435
19. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems **26** (2013)
20. Miller, G.A.: Wordnet: a lexical database for english. Communications of the ACM **38**(11), 39–41 (1995)
21. Navigli, R., Ponzetto, S.P.: Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. Artificial intelligence **193**, 217–250 (2012)
22. Nickel, M., Tresp, V., Kriegel, H.P.: A three-way model for collective learning on multi-relational data. In: ICML (2011)
23. Paulheim, H.: Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic web **8**(3), 489–508 (2017)
24. Pellissier Tanon, T., Weikum, G., Suchanek, F.: Yago 4: A reason-able knowledge base. In: European Semantic Web Conference. pp. 583–596. Springer (2020)
25. Pour, M., Algergawy, A., Amardeilh, F., Amini, R., Fallatah, O., Faria, D., Fundulaki, I., Harrow, I., Hertling, S., Hitzler, P., et al.: Results of the ontology alignment evaluation initiative 2021. In: CEUR Workshop Proceedings 2021. vol. 3063, pp. 62–108. CEUR (2021)
26. Santus, E., Gladkova, A., Evert, S., Lenci, A.: The cogalex-v shared task on the corpus-based identification of semantic relations. In: Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon (CogALex-V). pp. 69–79 (2016)
27. Sérasset, G.: Dbnary: Wiktionary as a lemon-based multilingual lexical resource in rdf. Semantic Web **6**(4), 355–361 (2015)
28. 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 (2007)
29. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: International conference on machine learning. pp. 2071–2080. PMLR (2016)
30. Vossen, P.: Eurowordnet: a multilingual database for information retrieval. In: Proceedings of the DELOS workshop on Cross-language Information Retrieval, March 5-7, 1997 Zurich. Vrije Universiteit (1997)
31. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Communications of the ACM **57**(10), 78–85 (2014)
32. Wang, C., Qiu, M., Huang, J., He, X.: Keml: A knowledge-enriched meta-learning framework for lexical relation classification. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 13924–13932 (2021)
33. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T.L., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M.: Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. pp. 38–45. Association for Computational Linguistics, Online (Oct 2020), https://www.aclweb.org/anthology/2020.emnlp-demos.6