Generating Descriptions of Entity Relationships

Nikos Voskarides[1], Edgar Meij[2], and Maarten de Rijke[1]

[1] University of Amsterdam, Amsterdam, The Netherlands
{n.voskarides,derijke}@uva.nl
[2] Bloomberg L.P., London, United Kingdom
edgar.meij@acm.org

Abstract. Large-scale knowledge graphs (KGs) store relationships between entities that are increasingly being used to improve the user experience in search applications. The structured nature of the data in KGs is typically not suitable to show to an end user and applications that utilize KGs therefore benefit from human-readable textual descriptions of KG relationships. We present a method that automatically generates textual descriptions of entity relationships by combining textual and KG information. Our method creates sentence templates for a particular relationship and then generates a textual description of a relationship instance by selecting the best template and filling it with appropriate entities. Experimental results show that a supervised variation of our method outperforms other variations as it best captures the semantic similarity between a relationship instance and a template, whilst providing more contextual information.

1 Introduction

Results displayed on a modern search engine result page (SERP) are sourced from multiple, heterogeneous sources. For so-called organic results it has been known for a long time that result snippets, i.e., brief descriptions explaining the result item and its relation to the query, positively influence the user experience [20]. In this paper, we focus on generating descriptions for results sourced from another important ingredient of modern SERPs: knowledge graphs. Knowledge graphs (KGs) contain information about entities and their relationships. A large and diverse set of search applications utilize KGs to improve the user experience. For instance, web search engines try to identify KG entities in queries and augment their result pages with knowledge graph panels that provide contextual entity information [3, 12]. Such panels usually focus on a single entity and may include attributes of the entity and other, related entities.

Entities can be connected with more than one relationship in a KG, however. For example, two actors might have appeared in the same film, be born in the same country and also be partners. Recent work has focused on finding relationships between a pair of entities and ranking the relationships by a predefined relevance criterion [5]. When using relationships in real-world search applications, with SERPs being the prime example, a crucial problem is that they are typically represented in a formal manner that is not suitable to present to an end user. Instead, human-readable descriptions that verbalize and provide context about entity relationships are more natural to use [7]. They can be used, e.g., for entity recommendations [2] or for KG-based timeline generation [1].

Descriptions of KG relationships themselves are usually not included in large-scale knowledge graphs and previous work on automatically generating such descriptions has either relied on hand-crafted templates [1] or on external text corpora [22]. The main limitations of the former are that manually creating these templates is expensive, not generalizable, and thus it does not scale well. The latter approach is limited as the underlying text corpus may not contain descriptions for all certain relationship instances; it will not produce meaningful results for instances that do not appear in the text corpus.

We propose a method that overcomes these limitations by automatically generating descriptions of KG entity relationships. Since there exist textual descriptions of a certain relationship for some relationship instances, we aim to use these descriptions to learn how the relationship is generally expressed in text and use this information to generate descriptions for other instances of the same relationship. Existing relationship descriptions are usually complex and tailored to the entities they discuss. Also, it is likely that the KG does not contain all the information included in a description. For example, the KG might not contain any information about the second part of the following sentence: "*Catherine Zeta-Jones starred in the romantic comedy The Rebound, in which she played a 40-year-old mother of two ...*". Nevertheless, descriptions of the same relationship share patterns that are specific to that relationship. Therefore, we first create sentence templates for a certain relationship and then, for a new relationship instance, we select appropriate templates, which we formulate as a ranking problem, and fill them with the appropriate entities to generate a description.

We propose a method that generates descriptions of entity relationships for a relationship instance given a knowledge graph and a set of relationship instances coupled with their descriptions; we evaluate this method using an automatic and manual evaluation method, and release the datasets used to the community.[1] We show that we generate contextually rich relationship descriptions that are meant to be valid under the KG closed-world assumption. Moreover, our template-based method is naturally robust against KG incompleteness, since in the case of lack of contextual information about the relationship instance, it can still generate a basic description.

2 Related work

Web search engine result pages (SERPs) can be augmented with information about the query and the documents from KGs in order to improve the user experience [12]. Also, SERPs can be augmented with textual descriptions and/or summaries with a prominent example being snippet generation for web search [20, 21]. Closest to our setting, relationship descriptions have been studied in the context of providing evidence for entity recommendation for web search [22] and timeline generation for knowledge base entities [1]. Our task, generating a description of a relationship instance given a KG, is similar to event headline generation, where the task is to generate a short sentence that summarizes a specific event. Similar to our templates, the headline patterns constructed in [17] consist of words and entity slots. Our method differs however, since relationships are more general than events and we thus have to deal with ambiguity at generation time when selecting which template matches a relationship instance.

Our task is also similar to concept-to-text generation, where the task is to generate a textual description given a set of database records [18]. In this context, our task is most closely related to [10, 19]. Saldanha et al. [19] use a template-based approach for

[1] `https://github.com/nickvosk/ecir2017-gder-dataset/`

Table 1: Glossary.

Symbol	Description
$\mathcal{K}$	knowledge graph
$\mathcal{E}$	set of entities
$\mathcal{P}$	set of predicates
$\langle s, p, o\rangle$	knowledge graph triple with $s, o \in \mathcal{E}$ and $p \in \mathcal{P}$
v	word in vocabulary $\mathcal{V}$
a	sentence
r_i	relationship instance of relationship r
T_r	set of templates $t \in T_r$ for relationship r
R_t	set of relationship instances that support the template t
X	set of pairs $\langle r_{i'}, y'\rangle$, where y' is a textual description (a single sentence)
C	mapping from an entity to an entity cluster
K	entity dependency graph of a sentence
G	compression graph
P	set of paths in G

generating company descriptions from Freebase. They construct sentence templates by replacing the entities in existing sentences by the Freebase relation of the entity to the company (e.g., $\langle company\rangle$ was founded by $\langle founder\rangle$). They add a preprocessing step where they remove phrases from the sentence that contain entities that are not connected to the company directly. At generation time, the authors replace the entity slots with the appropriate entities. Lebret et al. [10] propose a neural model to generate the first sentence of a person's biography in Wikipedia conditioned on Wikipedia infoboxes. Our setting is different from these papers since our generated descriptions are neither restricted to having entities that are directly connected to the subject entity in a KG nor need they be contained in a Wikipedia infobox.

3 Problem definition

In this section we formally define the task of generating descriptions of entity relationships. Table 1 lists the main notation we use in the paper.

3.1 Prelimilaries Let $\mathcal{E}$ be a set of entities and $\mathcal{P}$ a set of predicates. A *knowledge graph* $\mathcal{K}$ is a set of triples $\langle s, p, o\rangle$, where $s, o \in \mathcal{E}$ and $p \in \mathcal{P}$. We follow the closed-world assumption for $\mathcal{K}$ and use Freebase as our knowledge graph [4, 15]. A *sentence* a is a sequence of words $[v_1, \ldots, v_n]$, where each $v_i \in a$ is also in $\mathcal{V}$. Non-overlapping sub-sequences of a might refer to a single entity $e \in \mathcal{E}$.

A *relationship* r is a logical form in λ-calculus that consists of two lambda variables (x and y), at least one predicate, and zero or one existential variables [24]. Lambda variables can be substituted with Freebase entities, excluding compound value type (CVT) entities.[2] Existential variables, on the other hand, can be substituted with Freebase entities, including CVT entities. For example, the logical form of the relationship $starsInFilm$ is $\lambda x.\lambda y.\exists z.actor\_film(x, z) \wedge film\_starring(z, y)$. Fig. 1 shows the equivalent graphical representation of this relationship.

A pair $r_i = r\langle s, o\rangle$ is a *relationship instance* of r for entities $s, o \in \mathcal{E}$ if by substituting $x = s$ and $y = o$ in r and by executing the resulting logical form in the knowledge

[2] CVT entities are special entities in Freebase that are used to model attributes of relationships (e.g., date of marriage).

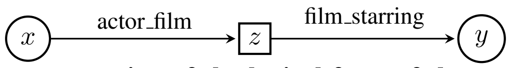

Fig. 1: Graphical representation of the logical form of the $starsInFilm$ relationship. Lambda variables are shown in circles and existential variables in rectangles.

graph $\mathcal{K}$ we get at least one result. For example, $starsInFilm(BradPitt, Troy)$ is a relationship instance of the $starsInFilm$ relationship.

3.2 Task definition We assume that a relationship instance r_i can be expressed with a human-readable description (such as a single sentence) that contains mentions of both s and o and possibly other entities which may provide contextual information for the relationship r or the entities s and o. The task we address in this paper is to generate such a textual description y of the relationship instance r_i given the KG. For this we leverage a set of pairs X, where each $x \in X$ is a pair of $r_{i'}$ and y', and y' is the description of $r_{i'}$. We describe how we obtain this set in Section 5.

We aim to generate descriptions that are valid (expressing a relationship that can be found in the knowledge graph under the closed-world assumption), natural (grammatically correct), and informative, i.e., not just replicating the formal relationship but providing additional contextual information where possible.

We conclude our task definition with an example. Assume that we are given the relationship instance $starsInFilm(BradPitt, Troy)$. A possible description of this relationship instance is the following: "Brad Pitt appeared in the American epic adventure film Troy." This description not only contains mentions of the entities of the relationship instance and a verbalization of the relationship ("appeared in"), but also mentions of other entities that provide additional context. In particular, it contains mentions of `Troy`'s type (`Film`), its genres (`Epic, Adventure`), and its country of origin.

4 Generating textual descriptions

In this section we detail our method which consists of three main steps. First, we enrich the description y' for each pair $\langle r_{i'}, y' \rangle \in X$ with additional entities from the KG (Section 4.1). Second, we use $\mathcal{K}$ and the set X to create a set of sentence templates T_r for the relationship r (Section 4.2). Third, given a new relationship instance, we use T_r and $\mathcal{K}$ to generate a description (Section 4.3).

4.1 Enriching the textual descriptions In this step we perform entity linking to enrich the description y' for each pair $\langle r_{i'}, y' \rangle \in X$ with additional entities from the KG. This is done in order to facilitate the template creation step (Section 4.2). Each y' is a sentence that is about an entity $e \in \mathcal{E}$ and in the context of this paper we obtain these sentences from Wikipedia as our KG provides explicit links to Wikipedia articles. Although Wikipedia articles already contain explicit links to other articles and thus entities, these links are quite sparse. Therefore, we apply an algorithm for entity linking similar to [22].

Since y' originates from a Wikipedia article that is about a specific entity, we restrict the *candidate entities* (i.e., the entities that we consider adding to enrich y') to e itself, the in-links and out-links of the article of e in the Wikipedia structure, and the one-hop and two-hop neighbors of e in the KG. We infer the *surface forms* of each entity using the Wikipedia link structure, as is common in entity linking [14], and we also use

Table 2: Additional surface forms per entity type.

Entity type	Surface form
Person	“he” or “she”, person’s surname
Film	“the film”
Music album	“the album”
Music composition	“the song”, “the track”

Algorithm 1 Template creation

```
Input: A set $X$, the knowledge graph $\mathcal{K}$
Output: A set of templates $T_r$
 1: $X' \leftarrow []$
 2: for $\langle r_{i'}, y' \rangle \in X$ do
 3:     $K \leftarrow$ BuildEntityDependencyGraph($y', \mathcal{K}$)
 4:     $X'.append(\langle r_{i'}, y', K \rangle)$
 5: $C \leftarrow$ ClusterEntities($X'$)
 6: $G \leftarrow$ BuildCompressionGraph($X', C$)
 7: $P \leftarrow$ FindValidPaths($G$)
 8: $T_r \leftarrow \{\}$
 9: for $p \in P$ do
10:     $t \leftarrow$ ConstructTemplate($p, G, X'$)
11:     if $t \neq NULL$ then
12:         $T_r.add(t)$
```

the aliases of each entity provided by the KG.[3] In order to increase coverage for e, we enhance the set of surface forms of entity e using the rules in Table 2.

We iterate over the n-grams of the sentence that are not yet linked to an entity in decreasing order of length; if the n-gram matches a surface form of a candidate entity, we *link* the n-gram to the entity. If multiple entity candidates exist for a surface form, we rank the candidate entities by the number of entity neighbors they have in the sentence and select the top-ranked entity. Because of the very restricted set of candidate entities, the linking is usually unambiguous (with only one entity candidate per surface form).[4]

4.2 Creating sentence templates In this step, we create a set of templates T_r for a relationship r using the KG and the set of $\langle r_{i'}, y' \rangle$ pairs. The templates in T_r will be used in the next step to generate a novel description for the relationship instance r_i.

A *sentence template* t is a tuple (k, l, R_t), where (i) $k = [u_1 u_2 \dots u_n]$ is a sequence, such that $\forall u_i \in l : u_i \in \mathcal{V} \cup \mathcal{E}_t$, (ii) l is a logical form in λ-calculus that consists of all the lambda variables in $\mathcal{E}_t$, at least one predicate and zero or more existential variables, and (iii) R_t is a set of relationship instances that support t.

The procedure we follow is outlined in Algorithm 1. First, we augment each $\langle r_{i'}, y' \rangle$ pair with an entity dependency graph K in order to capture dependencies between entities in a sentence (lines 1–4). Next, we build a mapping C that maps each entity in each sentence to a single cluster id (line 5). This is done in order to facilitate the detection of useful patterns in the sentences since each sentence describes a relationship for a

[3] We tag the sentences with POS tags and ignore unigram surface forms that are verbs.

[4] A manual evaluation of this algorithm on a held-out, random sample of 100 sentences in our dataset revealed an average of 93% precision and 85% recall per sentence.

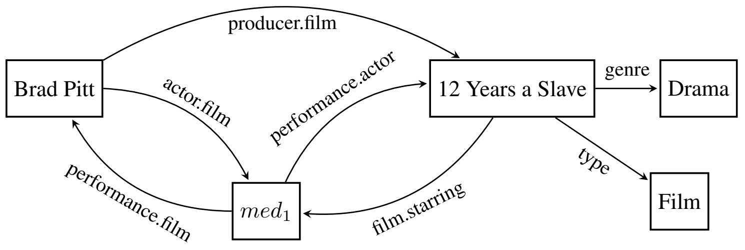

Fig. 2: Entity dependency graph for the sentence "Brad Pitt appeared in the drama film 12 Years a Slave". Nodes represent entities and edge labels represent predicates (med_1 is a CVT entity).

particular entity pair. Then, we build a compression graph G (line 6) and use it to find valid paths P (line 7). Finally, for each path $p \in P$, we construct a template t and add it to the set of templates (lines 8–12). We now describe each procedure in Algorithm 1.

BUILDENTITYDEPENDENCYGRAPH(.) In order to build the graph K for a sentence y', we retrieve all paths between each pair of entities mentioned in y' from the KG and add them to K. We only consider 1-hop paths and 2-hop paths that pass through a CVT entity. Fig. 2 shows the entity dependency graph for an example sentence.

CLUSTERENTITIES(.) In order to obtain C, we consider all $x' = \langle r_{i'}, y', K \rangle \in X'$ and map two entities in the same cluster if they share at least one incoming or outgoing edge label in their corresponding entity dependency graph K. For example, in the *starsInFilm* relationship, this procedure will create separate clusters for persons, films, dates and CVT entities.

BUILDCOMPRESSIONGRAPH(.) In this step, we build a compression graph $G = (V, E)$ using the sentence y' of each $\langle r_{i'}, y', K \rangle \in X'$. V is a set of nodes and E is a set of edges. We follow a similar procedure to [6], in which each node holds a list of $\langle sid, pid \rangle$ pairs, where sid is a sentence id and pid is the index of the word/entity in the sentence. In our case a node can be a word or an entity cluster. We map two words onto the same node if they have the same lowercase form and the same POS tag. We map two entities on the same node if they have the same cluster id.

FINDVALIDPATHS(.) In order to find valid paths in the graph G, we set all the entity cluster nodes as valid start/end nodes and traverse G to find a set of paths P from a start to an end node. In order to build templates that are natural we enforce the following constraints for the paths in P: (i) the path must contain a verb and (ii) the path must have been seen as a complete sentence at least once in the input sentences. For example, given the following sentences (the corresponding cluster id per entity are listed in brackets):

- y'_1: "Bruce_Willis[c_1] appeared in Moonrise_Kingdom[c_2]"
- y'_2: "Liam_Neeson[c_1] appeared in the action[c_3] film[c_4] Taken[c_2]"
- y'_3: "Brad_Pitt[c_1] appeared in the drama[c_3] film[c_4] 12_Years_a_Slave[c_2]"

we obtain the following valid paths by traversing the graph:

- p_1: "c_1 appeared in c_2"
- p_2: "c_1 appeared in the c_3 c_4 c_2"

Algorithm 2 CONSTRUCTTEMPLATE(.)

```
Input: A path $p$, the compression graph $G$, a set $X'$, parameters $\alpha, \beta$
Output: A template $t$
 1: $D_g \leftarrow []$                                  ▷ entity dependency graphs
 2: $R_t \leftarrow []$                ▷ relationship instances that support the template
 3: for $\langle r_{i'}, y', K \rangle \in X'$ do
 4:     if ISSUBSEQUENCE($p, y', G$) then
 5:         $h \leftarrow$ GETSUBSEQUENCE($p, y', G$)          ▷ get the actual subsequence
 6:         $\langle s, o \rangle \leftarrow r_{i'}$              ▷ subject/object of the relationship instance
 7:         if CONTAINSLINK($h, s$) and CONTAINSLINK($h, o$) then
 8:             $D_g.append(K)$
 9:             $R_t.append(r_{i'})$
10: if $|R_t| < \alpha$ then                          ▷ too few relationship instances
11:     return $NULL$
12: $l \leftarrow$ BUILDLOGICALFORM($D_g, \beta$)        ▷ aggregate the entity dependency graphs
13: $k \leftarrow$ REPLACECLUSTERIDSWITHVARIABLES($p$)
14: $t = (k, l, R_t)$
```

CONSTRUCTTEMPLATE(.) Algorithm 2 outlines the procedure for constructing a template t from a path p. First, for each $\langle r_{i'}, y', K \rangle \in X'$, we check whether y' is a (possibly non-continuous) subsequence h of path p by using the positional information of each node in p from G.[5] If it is, we check whether h contains links to both the subject and the object of the relationship instance $r_{i'}$. If it does, we store the entity dependency graph and the relationship instance. Next, if the number of instances is less than a parameter α, we consider the template to be invalid. Subsequently, we build the logical form l by aggregating the entity dependency graphs D_g. Entity nodes that were part of the path p become lambda variables (nodes constructed from subject and object entities have special identifiers). Entity nodes that were not part of the path p (CVT entities) become existential variables. We ignore edges appearing in less than $|D_g| \cdot \beta$ entity dependency graphs. Lastly, we replace the cluster ids in p with the corresponding lambda variables to obtain a sequence k.

Fig. 3 shows the logical form of a template constructed using the example sentences y'_1, y'_2 and y'_3 and their corresponding instances in graphical form ($\beta = 0.5$). Note that the edge "producer.film" has been eliminated since it only appears in one out of the three instances.

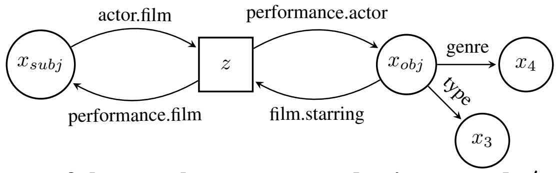

Fig. 3: Logical form of the template constructed using p_2 and y'_1, y'_2, y'_3 (with their corresponding relationship instances). k ="x_{subj} appeared in the x_3 x_4 x_{obj}". Lambda variables are shown in circles and existential variables in rectangles.

[5] For example, the path p_1 is a subsequence of y'_2.

4.3 Generating the description In this step we generate a novel description for a relationship instance r_i using the set of templates T_r and the knowledge graph $\mathcal{K}$. This comes down to selecting the template from T_r that best describes the relationship instance r_i and filling it with the appropriate entities.

The procedure is as follows. First, we rank the templates in T_r for the relationship instance using a scoring function $f(r_i, t)$. Subsequently, for each template $t = (k, l, R_t)$ we replace the subject and object lambda variables in l to obtain $l' = l[x_{subj} = s, x_{obj} = o]$. We then query the knowledge graph $\mathcal{K}$ using l' and if at least one instantiation of l' exists, we randomly pick one and replace all the entity variables in k with the entity names to generate the description y, otherwise we proceed to the next template. As an example, assume we are given the instance $r_i = \mathit{starsInFilm}(\mathit{Ryan\_Reynolds}, \mathit{Deadpool})$ and we consider the template shown in Fig. 3. A possible instantiation of the template for this relationship instance will result in the description "Ryan Reynolds appeared in the comedy film Deadpool".[6]

The template scoring function $f(r_i, t)$ returns a score for a relationship instance r_i and template t. As we want to generate descriptions that are valid under the closed-world assumption of the KG, we promote templates that are semantically closest to the relationship instance. For a new relationship instance r_i we extract binary features for each entity in the r_i. Recall that r_i has two or more entities (subject s, object o and possibly a CVT entity z). For each entity e of r_i, we extract all triples $\langle e, p, e' \rangle$ from the KG $\mathcal{K}$. We restrict the feature space by discriminating between entity attributes and entity relations depending on the predicate p as in [13]. If the predicate p is an attribute (e.g., "gender"), we use the complete triple as a feature (e.g. $\langle s, \mathit{gender}, \mathit{female} \rangle$). If the predicate p is a relation (e.g., "date_of_death"), we only keep the subject and the predicate of the triple as a feature (e.g., $\langle e, \mathit{person.date\_of\_death} \rangle$). We also add a count feature for the relation predicates (e.g., $\langle s, \mathit{person.children}, 2 \rangle$, i.e., a person has two children). We denote the resulting binary vector for r_i as $\mathit{vec}(r_i)$. We obtain a vector $\mathit{vec}(t)$ for template t by summing the vectors of all the instances R_t of t. We also compute a vector $\mathit{vec\_tfidf}(t)$ that is a TF.IDF weighted vector of $\mathit{vec}(t)$, where IDF is calculated at the template level. Based on these ingredients, we define two scoring functions:

- **Cosine** Calculates the cosine similarity between vectors $vec(r_i)$ and $vec\_tfidf(t)$.
- **Supervised** Learns a scoring function using a supervised learning to rank algorithm. We treat r_i as a "query" and t as a "document."

We create training data for the supervised algorithm as follows. Recall that each r_i is coupled with a description y'. For each r_i, we assign a relevance label of 3 for templates that best match y (measured by the number of entities) and a relevance label of 2 for the rest of the templates that match y. In order to create "negative" training data, we sample templates that are dissimilar to the ones that match y in the following way. First, we calculate the average vector of all the templates that match y and build a distribution of templates based on the cosine distance from the average vector to each of the templates in T_r (excluding the ones that match y). Lastly, we sample at most the number of

[6] Note that there might be multiple instantiations (e.g., Deadpool is also a science fiction film) and selecting the optimal one depends on the application—we leave this for future work.

matching templates from the resulting distribution and assign them a relevance label of 1 (we ignore templates that have a cosine similarity to the average vector greater than 0.9). For the supervised model we use the following features: each element/value pair in $vec(r_i)$, the cosine similarity between vectors $vec(r_i)$ and $vec\_tfidf(t)$, the words in t, the number of entities in t and the size of R_t. We use LambdaMART [23] as the learning algorithm and optimize for NDCG@1.[7]

5 Experimental setup

In this section we describe our experimental setup.

5.1 Datasets We use an English Wikipedia dump dated 5 February 2015 as our document corpus. We perform sentence splitting and POS tagging using the Stanford CoreNLP toolkit. We use a subset of the last version of Freebase as our KG [4]: all the triples in the people, film and music domains, as these are well-represented in Freebase.

In order to create an evaluation dataset for our task, we first need a set of KG relationships. We rank the predicates in each domain by the number of instances and keep the 10 top-ranked predicates. We exclude trivial predicates such as "dateOfDeath". We then use the predicates to manually construct the logical forms of the relationships (see Figure 1 for an example). Second, we need a set of $\langle r_{i'}, y' \rangle$ pairs for each relationship r, where $r_{i'} = r\langle s', o' \rangle$ is an instance of relationship r, s' and o' are entities and y' is a description of $r_{i'}$. To this end, for each relationship r, we randomly sample 12 000 relationship instances from the KG. For each relationship instance $r_{i'}$, we pick the first sentence in the Wikipedia article of the subject entity s' that contains links to both s' and o'. If such a sentence does not exist, we proceed to the next instance. We manually inspected a subset of the sentences selected with this heuristic and the quality of the selected sentences was relatively good. Our final dataset contains 10 relationships and 90 058 $\langle r_{i'}, y' \rangle$ instances in total and 8 187 instances on average per relationship. We randomly select 80% of each relationship sub-dataset for training and 20% for testing.

5.2 Evaluation metrics We perform two types of evaluation: automatic and manual. For automatic evaluation we use METEOR [9], ROUGE-L [11] and BLEU-4 [16] as metrics. METEOR was originally proposed in the context of machine translation but has also been used in a task similar to ours [19]. ROUGE is a standard metric in summarization and BLEU is widely used in machine translation and generation. As is common in text generation [8], we also employ manual evaluation. We ask human annotators to annotate each output sentence on three dimensions: validity under the KG closed-world assumption (0 or 1), informativeness (1–5) and grammaticality (1–5). One human annotator (not one of the authors) annotated 11 generated sentences per relationship per system (440 sentences in total).

5.3 Compared approaches We compare 4 variations of our method. The variations differ in the way they rank templates for a given relationship instance. The first variation (*Random*) ranks the templates randomly. The second (*Most-freq*) ranks templates by the number of relationship instances that support the template. The third (*Cosine*) ranks templates based on the cosine similarity between the vectors of the relationship instance and the template (Section 4.3). The fourth (*Supervised*) ranks templates using a learning

[7] For this method we use 20% of the training data as validation data. The same test data is used for all methods.

Table 3: Automatic evaluation results, averaged per relationship.

Method	**BLEU**	**METEOR**	**ROUGE**
Random	1.14	16.56	24.13
Most-freq	0.13	13.99	21.96
Cosine	$1.76^{\blacktriangle}$	17.37	$25.84^{\blacktriangle}$
Supervised	$\mathbf{2.14}^{\blacktriangle}$	$\mathbf{19.18}^{\blacktriangle}$	$\mathbf{26.54}^{\blacktriangle}$

Table 4: Manual evaluation results, averaged per relationship.

Method	**Validity**	**Informativeness**	**Grammaticality**
Random	0.4545	1.98	3.67
Most-freq	0.5000	1.60	3.62
Cosine	$0.5636^{\blacktriangle}$	2.05	**4.00**
Supervised	$\mathbf{0.5818}^{\blacktriangle}$	$\mathbf{2.18}^{\blacktriangle}$	3.90

to rank model (Section 4.3), for which we use LambdaMART with the default number of trees (1000). We set $\alpha = 20$ and $\beta = 0.5$ (Section 4.3). We depict a significant improvement in performance over *Random* with $^{\blacktriangle}$ (paired two-tailed t-test, $p < 0.05$).

6 Results

In this section we describe our experimental results. We compare all methods discussed previously, using the automatic and manual setups, respectively.

6.1 Automatic evaluation Table 3 shows the automatic evaluation results. We observe that *Supervised* and *Cosine* outperform *Random* and *Most-freq* on all metrics. This is expected since the former two try to capture the semantic similarity between a relationship instance and a template. Although *Supervised* consistently outperforms *Cosine*, the differences between *Cosine* and *Supervised* are not significant.

We also observe that the scores for the automatic measures are relatively low. This is because of two reasons: (i) we generally generate much shorter sentences than the reference sentence as not all information that appears in the reference sentence is represented in the KG, and (ii) since the reference sentences are extracted automatically, some of the reference sentences describe a minor aspect of the relationship or do not discuss the relationship at all.

6.2 Manual evaluation Table 4 shows the results for manual evaluation. The results follow a similar trend as in the automatic evaluation; *Supervised* and *Cosine* outperform *Random* and *Most-freq* on all metrics. *Supervised* significantly outperforms *Random* in terms of validity and informativeness. The differences between *Cosine* and *Supervised* are not significant.

6.3 Analysis We have also examined specific examples and identify cases where the best performing approach (*Supervised*) succeeds or fails. In terms of validity, it succeeds in matching attributes of the relationship instance and the template. E.g., in the context of the relationship $parentOf$, it correctly figures out what the genders of the entities are and the semantically valid expression of the relationship between them, often better than *Cosine*, as illustrated by the following example:

(*Supervised*) “Emperor Francis I (1708 - 1765) was the father of Emperor Leopold II” (VALID)

(*Cosine*) "Emperor Francis I was the son of Emperor Leopold II" (INVALID)

Supervised benefits from training a model that combines multiple features such as the template words with attributes of the relationship instance to describe whether the relationship is still ongoing or not. One of the main cases where *Supervised* fails is in ranking a relationship instance in a temporal dimension with regards to other relationship instances, as illustrated by the following example for the *childOf* relationship:

"Thomas Howard was the second son of Henry Howard and Frances de Vere."
(INVALID: Thomas Howard was the *first* son of Henry Howard)

The fact that our best performing approach (*Supervised*) has a relatively low validity score (0.5818) shows that there is room for improvement in capturing the semantic similarity between a relationship instance and a template.

In terms of informativeness, *Supervised* succeeds in offering contextual information about the relationship instance, such as dates, locations, occupations and film genres. The fact that informativeness scores are relatively low is because they are dependent on validity: when a generated sentence was assigned a validity of score 0, it was also assigned an informativeness score of just 1.

Grammaticality scores are high for all the systems with no significant differences. This is expected as the templates were generated using the same procedure for all the compared systems. Mainly, grammaticality is harmed when some entities in the generated sentence have the wrong surface form (e.g., 'Britain', 'British'), which is not surprising as we do simple surface realization (deciding which surface form of the entity best fits with the generated sentence) and only use the entity names as surface forms.

7 Conclusion

We have addressed the problem of generating descriptions of entity relationships from KGs. We have introduced a method that first creates sentence templates for a specific relationship, and then, for a new relationship instance, it generates a novel description by selecting the best template and filling the template slots with the appropriate entities from the KG. We have experimented with different scoring functions for ranking templates for a relationship instance and performed an automatic and a manual evaluation.

When using information about the relationship instance and the template taken from the KG, both automatic and manual evaluation outcomes are improved. A supervised method that uses both KG features and other template features (template words, number of entities) consistently outperforms an unsupervised method on all automatic evaluation metrics and also in terms of validity and informativeness.

As to future work, our error analysis showed that we need more sophisticated modeling for capturing the semantic similarity between a relationship instance and a template, especially for capturing temporal dimensions that also involve other relationship instances. We also want to explore more sophisticated methods for selecting the correct surface form for an entity to improve grammaticality. Finally, we aim to evaluate our method on generating descriptions for less popular KG relationships.

Acknowledgments. This research was supported by the Netherlands Institute for Sound and Vision and the Netherlands Organisation for Scientific Research (NWO) under project nr. CI-14-25. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

Bibliography

[1] T. Althoff, X. L. Dong, K. Murphy, S. Alai, V. Dang, and W. Zhang. Timemachine: Timeline generation for knowledge-base entities. In *KDD*, 2015.

[2] R. Blanco, B. B. Cambazoglu, P. Mika, and N. Torzec. Entity recommendations in web search. In *ISWC*, 2013.

[3] R. Blanco, G. Ottaviano, and E. Meij. Fast and space-efficient entity linking for queries. In *WSDM*, 2015.

[4] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 2008.

[5] L. Fang, A. D. Sarma, C. Yu, and P. Bohannon. Rex: explaining relationships between entity pairs. In *VLDB*, 2011.

[6] K. Ganesan, C. Zhai, and J. Han. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *COLING*, 2010.

[7] D. Gkatzia, O. Lemon, and V. Rieser. Natural language generation enhances human decision-making with uncertain information. In *ACL*, 2016.

[8] I. Konstas and M. Lapata. A global model for concept-to-text generation. *JAIR*, 48:305–346, 2013.

[9] A. Lavie and A. Agarwal. METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments. In *WMT*, 2007.

[10] R. Lebret, D. Grangier, and M. Auli. Neural text generation from structured data with application to the biography domain. In *EMNLP*, 2016.

[11] C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, 2004.

[12] T. Lin, P. Pantel, M. Gamon, A. Kannan, and A. Fuxman. Active objects: Actions for entity-centric search. In *WWW*, 2012.

[13] Y. Lin, Z. Liu, and M. Sun. Knowledge representation learning with entities, attributes and relations. In *IJCAI*, 2016.

[14] E. Meij, W. Weerkamp, and M. de Rijke. Adding semantics to microblog posts. In *WSDM 2012*, 2012.

[15] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich. A review of relational machine learning for knowledge graphs: From multi-relational link prediction to automated knowledge graph construction. *Proc. of the IEEE*, 104(1):11–33, 2016.

[16] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. BLEU: A method for automatic evaluation of machine translation. In *ACL*, 2002.

[17] D. Pighin, M. Cornolti, E. Alfonseca, and K. Filippova. Modelling events through memory-based, open-ie patterns for abstractive summarization. In *ACL*, 2014.

[18] E. Reiter, R. Dale, and Z. Feng. *Building Natural Language Generation Systems*. MIT Press, 2000.

[19] G. Saldanha, O. Biran, K. McKeown, and A. Gliozzo. An entity-focused approach to generating company descriptions. In *ACL*, 2016.

[20] A. Tombros and M. Sanderson. Advantages of query biased summaries in information retrieval. In *SIGIR*, 1998.

[21] A. Turpin, Y. Tsegay, D. Hawking, and H. E. Williams. Fast generation of result snippets in web search. In *SIGIR*, 2007.

[22] N. Voskarides, E. Meij, M. Tsagkias, M. de Rijke, and W. Weerkamp. Learning to explain entity relationships in knowledge graphs. In *ACL-IJCNLP*, 2015.

[23] Q. Wu, C. J. Burges, K. M. Svore, and J. Gao. Ranking, boosting, and model adaptation. Techn. report, Technical report, Microsoft Research, 2008.

[24] W.-t. Yih, M.-W. Chang, X. He, and J. Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL*, 2015.