Weakly-supervised Contextualization of Knowledge Graph Facts

Nikos Voskarides[*]
University of Amsterdam
Amsterdam, The Netherlands
n.voskarides@uva.nl

Edgar Meij
Bloomberg
London, U.K.
edgar.meij@acm.org

Ridho Reinanda
Bloomberg
London, U.K.
rreinanda@bloomberg.net

Abhinav Khaitan
Bloomberg
New York, U.S.
akhaitan10@bloomberg.net

Miles Osborne
Bloomberg
London, U.K.
mosborne29@bloomberg.net

Giorgio Stefanoni
Bloomberg
London, U.K.
gstefanoni1@bloomberg.net

Prabhanjan Kambadur
Bloomberg
New York, U.S.
pkambadur@bloomberg.net

Maarten de Rijke
University of Amsterdam
Amsterdam, The Netherlands
derijke@uva.nl

ABSTRACT

Knowledge graphs (KGs) model facts about the world; they consist of nodes (entities such as companies and people) that are connected by edges (relations such as *founderOf*). Facts encoded in KGs are frequently used by search applications to augment result pages. When presenting a KG fact to the user, providing other facts that are pertinent to that main fact can enrich the user experience and support exploratory information needs. *KG fact contextualization* is the task of augmenting a given KG fact with additional and useful KG facts. The task is challenging because of the large size of KGs; discovering other relevant facts even in a small neighborhood of the given fact results in an enormous amount of candidates.

We introduce a neural fact contextualization method (*NFCM*) to address the KG fact contextualization task. NFCM first generates a set of candidate facts in the neighborhood of a given fact and then ranks the candidate facts using a supervised learning to rank model. The ranking model combines features that we automatically learn from data and that represent the query-candidate facts with a set of hand-crafted features we devised or adjusted for this task. In order to obtain the annotations required to train the learning to rank model at scale, we generate training data automatically using distant supervision on a large entity-tagged text corpus. We show that ranking functions learned on this data are effective at contextualizing KG facts. Evaluation using human assessors shows that it significantly outperforms several competitive baselines.

CCS CONCEPTS

• **Information systems** → *Presentation of retrieval results*;

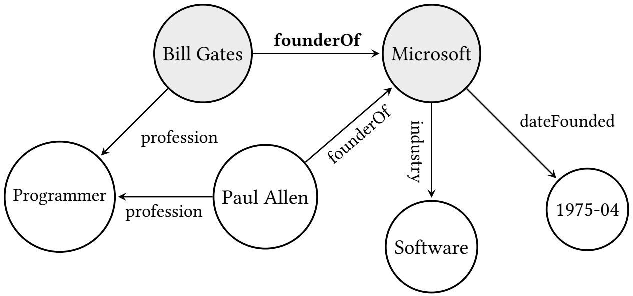

Figure 1: A Freebase subgraph that consists of relevant facts to the query fact *founderOf* (Bill Gates, Microsoft).

KEYWORDS

Knowledge graphs, Fact contextualization, Distant supervision

ACM Reference format:
Nikos Voskarides, Edgar Meij, Ridho Reinanda, Abhinav Khaitan, Miles Osborne, Giorgio Stefanoni, Prabhanjan Kambadur, and Maarten de Rijke. 2018. Weakly-supervised Contextualization of Knowledge Graph Facts. In *Proceedings of The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, Ann Arbor, MI, USA, July 8–12, 2018 (SIGIR '18),* 10 pages.
https://doi.org/10.1145/3209978.3210031

1 INTRODUCTION

Knowledge graphs (KGs) have become essential for applications such as search, query understanding, recommendation and question answering because they provide a unified view of real-world entities and the facts (i.e., relationships) that hold between them [6, 7, 22, 34]. For example, KGs are increasingly being used to provide direct answers to user queries [34], or to construct so-called *entity cards* that provide useful information about the entity identified in the query. Recent work [10, 17] suggests that search engine users find entity cards useful and engage with them when they contain information that is relevant to their search task, for instance in the form of a set of recommended entities and facts that are related to the query [6]. Previous work has focused on augmenting entity cards with facts that are centered around, i.e., one-hop away from, the main entity of the query [17].

[*] This work was done while Nikos was an intern at Bloomberg.

SIGIR '18, July 8–12, 2018, Ann Arbor, MI, USA

ACM ISBN 978-1-4503-5657-2/18/07…$15.00
https://doi.org/10.1145/3209978.3210031

However, oftentimes a user is interested in KG facts that by definition involve more than one entity (e.g., "Who founded Microsoft?" ⟶ "Bill Gates"). In such cases, we can exploit the richness of the KG by providing query-specific additional facts that increase the user's understanding of the fact as a whole, and that are not necessarily centered around only one of the entities. Additional relevant facts for the running example would include Bill Gates' profession, Microsoft's founding date, its main industry and its co-founder Paul Allen (see Figure 1). In this case, Bill Gates' personal life is less relevant to the fact that he founded Microsoft.

Query-specific relevant facts can also be used in other applications to enrich the user experience. For instance, they can be used to increase the utility of KG question answering (QA) systems that currently only return a single fact as an answer to a natural language question [5, 34]. Beyond QA, systems that focus on automatically generating natural language from KG facts [15, 20] would also benefit from query-specific relevant facts, which can make the generated text more natural and human-like. This becomes even more important for KG facts that involve tail entities, for which natural language text might not exist for training [32].

In this paper, we address the task of KG fact *contextualization*, that is, given a KG fact that consists of two entities and a relation that connects them, retrieve additional facts from the KG that are relevant to that fact. This task is analogous to ad-hoc retrieval: (i) the "query" is a KG fact, (ii) the "documents" are other facts in the KG that are in the neighborhood of the "query". We propose a *neural fact contextualization method* (NFCM), a method that first generates a set of candidate facts that are part of {1,2}-hop paths from the entities of the main fact. NFCM then ranks the candidate facts by how relevant they are for contextualizing the main fact. We estimate our learning to rank model using supervised data. The ranking model combines (i) features we automatically learn from data and (ii) those that represent the query-candidate facts with a set of hand-crafted features we devised or adjusted for this task. Due to the size and heterogeneous nature of KGs, i.e., the large number of entities and relationship types, we turn to distant supervision to gather training data. Using another, human-verified test collection we gauge the performance of our proposed method and compare it with several baselines. We sum up our contributions as follows.

- We introduce the task of KG fact contextualization where the goal is to, given a fact that consists of two entities and a relationship that connects them, rank other facts from a KG that are relevant to that fact.
- We propose NFCM, a method to solve KG fact contextualization using distant supervision and learning to rank. Our results show that: (i) distant supervision is an effective means for gathering training data for this task and (ii) a neural learning to rank model that is trained end-to-end outperforms several baselines on a human-curated evaluation set.
- We provide a detailed result analysis and insights into the nature of our task.

The remainder of the paper is organized as follows. We first provide a definition of our task in Section 2 and then introduce our method in Section 3. We describe our experimental setup and detail our results and analyses in Sections 4 and 5, respectively. We conclude with an overview of related work and an outlook on future directions.

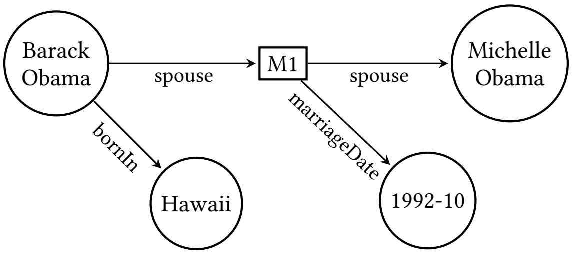

Figure 2: KG subgraph that consists of three facts: *bornIn*⟨**Barack Obama**, **Hawaii**⟩, *spouseOf*⟨**Barack Obama**, **Michelle Obama**⟩ **and** *marriageDate*⟨**M1**, **1992-10**⟩**. M1 is a CVT entity. Note that the third fact is an attribute of the second fact.**

2 PROBLEM STATEMENT

In this section we provide background definitions and formally define the task of KG fact contextualization.

2.1 Preliminaries

Let $E = E_n \cup E_c$ be a set of entities, where E_n and E_c are disjoint sets of non-CVT and CVT entities, respectively.[1] Furthermore, let P be a set of predicates. A *knowledge graph* K is a set of triples $\langle s, p, o \rangle$, where $s, o \in E$ and $p \in P$. By viewing each triple in K as a labelled directed edge, we can interpret K as a labelled directed graph. We use Freebase as our knowledge graph [8, 24].

A path in K is a non-empty sequence $\langle s_0, p_0, t_0 \rangle, \ldots, \langle s_m, p_m, t_m \rangle$ of triples from K such that $t_i = s_{i+1}$ for each $i \in 0, m-1$.

We define a *fact* as a path in K that either: (i) consists of 1 triple, $s_0 \in E$ and $t_0 \in E_n$ (i.e., s_0 may be a CVT entity), or (ii) consists of 2 triples, $s_0, t_1 \in E_n$ and $t_0 = s_1 \in E_c$ (i.e., $t_0 = s_1$ must be a CVT entity). A fact of type (i) can be an attribute of a fact of type (ii), iff they have a common CVT entity (see Figure 2 for an example).

Let R be a set of relationships where a *relationship* $r \in R$ is a label for a set of facts that share the same predicates but differ in at least one entity. For example, *spouseOf* is the label of the fact depicted in the top part of Figure 2 and consists of two triples. Our definition of a relationship corresponds to direct relationships between entities, i.e., one-hop paths or two-hop paths through a CVT entity. For the remainder of this paper, we refer to a specific fact f as $r\langle s, t \rangle$, where $r \in R$ and $s, t \in E$.

2.2 Task definition

Given a query fact f_q and a KG K, we aim to find a set of other, relevant facts from K. Specifically, we want to enumerate and rank a set of candidate facts $F = \{f_c : f_c \subseteq K, f_c \neq f_q\}$ based on their relevance to f_q. A candidate fact f_c is *relevant* to the query fact f_q if it provides useful and contextual information. Figure 1 shows an example part of our KG that is relevant to the query fact *founderOf*⟨Bill Gates, Microsoft⟩. Note that a candidate fact does not have to be directly connected to both entities of the query fact to be relevant, e.g., *profession*⟨Paul Allen, Programmer⟩. Similarly, a fact can be related to one or more entities in the relationship instance, e.g., *parentOf*⟨Bill Gates, Jennifer Katharine Gates⟩, but not provide any context, thus being considered irrelevant.

[1] Compound Value Type (CVT) entities are special entities frequently used in KGs such as Freebase and Wikidata to model fact attributes. See Figure 2 for an example.

Algorithm 1 Fact enumeration for a given query fact f_q.

```
Input: A query fact $f_q = r\langle s, t\rangle$
Output: A set of candidate facts $F$
$F \leftarrow \{\}$
for $e \in \{s, t\}$ do
    for $n \in$ GetOutNeighbors($e$) + GetInNeighbors($e$) do
        $F.addAll$(GetFacts($e, n$))
        if IsClassOrType($n$) then
            continue
        for $n_2 \in$ GetOutNeighbors($n$) do
            $F.addAll$(GetFacts($n, n_2$))
        for $n_2 \in$ GetInNeighbors($n$) do
            $F.addAll$(GetFacts($n_2, n$))
return $F$
```

3 METHOD

In this section we describe our proposed neural fact contextualization method (NFCM) which works in two steps. First, given a query fact f_q, we enumerate a set of candidate facts $F = \{f_c : f_c \subseteq K\}$ (see Section 3.1). Second, we rank the facts in F by relevance to f_q to obtain a final ranked list F' using a supervised learning to rank model (see Section 3.2). We describe how we use distant supervision to automatically gather the required annotations to train the supervised learning to rank model in Section 4.3.

3.1 Enumerating KG facts

In this section we describe how we obtain the set of candidate facts F from K given a query fact $f_q = r\langle s, t\rangle$. Because of the large size of real-world KGs—which can easily contain upwards of 50 million entities and 3 billion facts [25]— it is computationally infeasible to add all possible facts of K in F. Therefore, we limit F to the set of facts that are in the broader neighborhood of the two entities s and t. Intuitively, facts that are further away from the two entities of the query fact are less likely to be relevant.

The procedure we follow is outlined in Algorithm 1. This algorithm enumerates the candidate facts for $f_q = r\langle s, t\rangle$ that are at most 2 hops away from either s or t. Three exceptions are made to this rule: (i) CVT entities are not counted as hops, (ii) we do not include f_q in F as it is trivial, and (iii) to reduce the search space, we do not expand intermediate neighbors that represent an entity class or a type (e.g., “actor”) as these can have millions of neighbors. Figure 3 shows an example graph with a subset of the facts that we enumerate for the query fact *spouseOf*⟨Bill Gates, Melinda Gates⟩ using Algorithm 1.

3.2 Fact ranking

Next, we describe how we rank the set of enumerated candidate facts F with respect to their relevance to the query fact $f_q = r\langle s, t\rangle$. The overall methodology is as follows. For each candidate fact $f_c \in F$, we create a pair (f_q, f_c)—an analog to a query-document pair—and score it using a function $u : (f_q, f_c) \to [0, 1] \in R$ (higher values indicate higher relevance). We then obtain a ranked list of facts F' by sorting the facts in F based on their score.

We begin by describing the training procedure we follow and continue with the network architecture we use for learning our scoring function u.

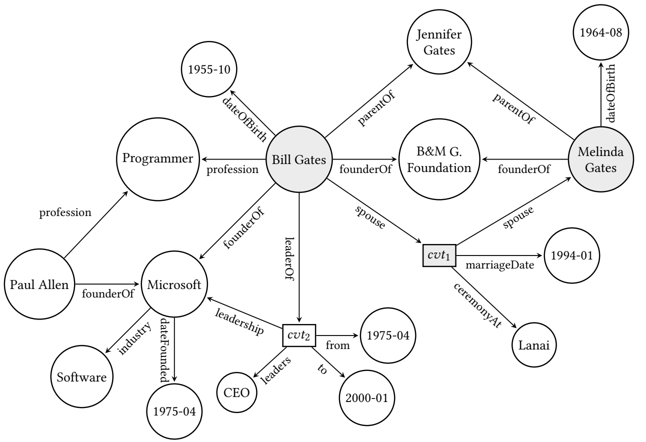

Figure 3: Graph with a subset of the facts that are enumerated for the query fact *spouseOf*⟨Bill Gates, Melinda Gates⟩. The entities of the query fact are shaded.

Learning procedure. We train a network that learns the scoring function $u(f_q, f_c)$ end-to-end in mini-batches using stochastic gradient descent (we define the network architecture below). We optimize the model parameters using Adam [19]. During training we minimize a pairwise loss to learn the function u, while during inference we use the learned function u to score a query-candidate fact pair (f_q, f_c). This paradigm has been shown to outperform pointwise learning methods in ranking tasks, while keeping inference efficient [13]. Each batch B consists of query-candidate fact pairs (f_q, f_c) of a single query fact f_q. For constructing B for a query fact f_q, we use all pairs (f_q, f_c) that are labeled as relevant and sample k pairs (f_q, f_c) that are labeled as irrelevant. During inference, we minimize the mean pairwise squared error between all pairs of (f_q, f_c) in $B \times B$:

$$L(B, \theta) = \frac{1}{|B|} \sum_{\langle x_1, x_2\rangle \in B \times B} ([l(x_1) - l(x_2)] - [u(x_1) - u(x_2)])^2, \quad (1)$$

where $x_1 = (f_q, f_{c_1})$ and $x_2 = (f_q, f_{c_2})$ are query-candidate fact pairs in the set $B \times B$, $l(x) \in \{0, 1\}$ is the relevance label of a query-candidate fact pair x, $|B|$ is the batch size, and θ are the parameters of the model which we define below.

Network architecture. Figure 4 shows the network architecture we designed for learning the scoring function $u(f_q, f_c)$. We encode the query fact f_q in a vector $\boldsymbol{v_q}$ using an RNN (see Section 3.2.1). As we will explain further in that section, we do not model the entities in the facts independently due to the large number of entities; instead, we model each entity as an aggregation of its types. Therefore, instead of modeling the candidate fact f_c in isolation and losing per-entity information, we first enumerate all the paths up to two hops away from both the entities of the query fact f_q (s and t) to all the entities of the candidate fact f_c (s' and t'). Let A_s denote the set of paths from s to all the entities of f_c. Let A_t denote the set of paths from t to all the entities of f_c. For each $A \in \{A_s, A_t\}$, we first encode all the paths in A using an RNN (Section 3.2.1), and then combine the resulting encoded paths using the procedure described in Section 3.2.2. We denote the vectors obtained from the above procedure for A_s and A_t as $\boldsymbol{v}_{as}$ and $\boldsymbol{v}_{at}$, respectively. Then we obtain a vector $\boldsymbol{v}_a = [\boldsymbol{v}_{as}, \boldsymbol{v}_{at}]$, where $[\cdot, \cdot]$ denotes the concatenation operation (middle part of Figure 4). Note that we use

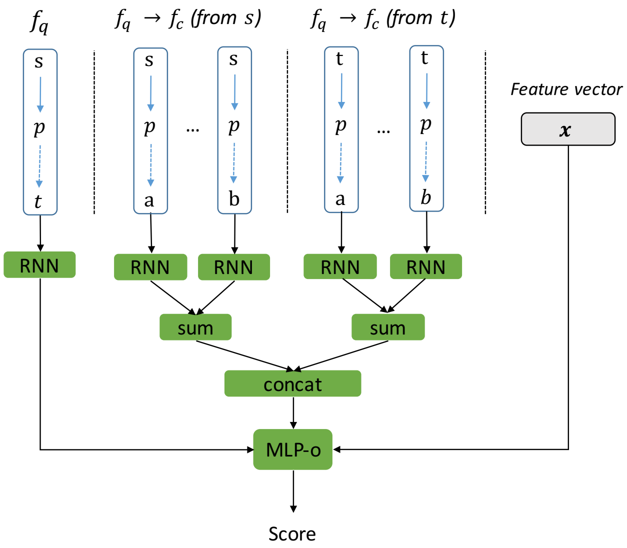
f_q
$f_q \rightarrow f_c$ (from s)
$f_q \rightarrow f_c$ (from t)
s
p
t
a
b
...
Feature vector
x
RNN
sum
concat
MLP-o
Score

Figure 4: Network architecture that learns a scoring function $u(f_q, f_c)$. Given a query fact $f_q = r\langle \mathrm{s}, \mathrm{t}\rangle$ and a candidate fact $f_c = r'\langle \mathrm{a}, \mathrm{b}\rangle$ it outputs a score $u(f_q, f_c)$. "$f_q \rightarrow f_c$ (from e)" is a label for the paths that start from an entity e of the query fact (either s or t) and end at an entity e' of the candidate fact f_c. Note that p is a variable in this figure, i.e., it might refer to different predicates.

the same RNN parameters for all the above operations. To further inform the scoring function, we design a set of hand-crafted features $\boldsymbol{x}$ (right-most part of Figure 4). We detail the hand-crafted features in Section 3.2.3.

Finally, MLP-o($[\boldsymbol{v}_q, \boldsymbol{v}_a, \boldsymbol{x}]$) is a multi-layer perceptron with α hidden layers of dimension β and one output layer that outputs $u(f_q, f_c)$. We use a ReLU activation function in the hidden layers and a sigmoid activation function in the output layer. We vary the number of layers to capture non-linear interactions between the features in $\boldsymbol{v}_q$, $\boldsymbol{v}_a$, and $\boldsymbol{x}$.

The remainder of this section is structured as follows. Section 3.2.1 describes how we encode a single fact, Section 3.2.2 describes how we combine the representations of a set of facts and finally Section 3.2.3 details the hand-crafted features.

3.2.1 Encoding a single fact. Recall from Section 2.1 that a fact f is a path in the KG. In order to model paths we turn to neural representation learning. More specifically, since paths are sequential by nature we employ recurrent neural networks (RNNs) to encode them in a single vector [12, 16]. This type of modeling has proven successful in predicting missing links in KGs [12]. One restriction that we have in modeling such paths is the very large number of entities (~ 1.5 million entities in our dataset) and, since learning an embedding for such large numbers of entities requires prohibitively large amounts of memory and data, we represent each entity using an aggregation of its types [12]. Formally, let $\boldsymbol{W}_z$ denote a $|Z| \times d_z$ matrix, where each row is an embedding of an entity type z, $|Z|$ is the number of entity types in our dataset and d_z is the entity type embedding dimension. Let $\boldsymbol{W}_p$ denote a $|P| \times d_p$ matrix, where each row is an embedding of a predicate p, $|P|$ is the number of predicates in our dataset, and d_p is the predicate embedding dimension. In order to model *inverse* predicates in paths (e.g., Microsoft → $founderOf^{-1}$ → Paul Allen), we also define a $|P| \times d_p$ matrix $\boldsymbol{W}_{p_i}$, which corresponds to embeddings of the inverse of each predicate [16].

The procedure we follow for modeling a fact f is as follows. For simplicity in the notation, in this Section we denote a path as a sequence of alternate entities and predicates $[s_0, p_0, \ldots t_m]$, instead of a sequence of triples as defined in Section 2.1. For each entity $e \in f$, we first retrieve the types of e in K. From these, we only keep the 7 most frequent types in K, which we denote as Z_e [12]. We then project each $z \in Z_e$ to its corresponding type embedding $\boldsymbol{w}_z \in \boldsymbol{W}_z$ and perform element-wise sum on these embeddings to obtain an embedding w_e for entity e. We project each predicate $p \in f$ to its corresponding embedding $\boldsymbol{w}_p$ ($\boldsymbol{w}_p \in \boldsymbol{W}_{p_i}$ if p is inverse, $\boldsymbol{w}_p \in \boldsymbol{W}_p$ otherwise).

The resulting projected sequence $X_f = [\boldsymbol{w}_{s_0}, \boldsymbol{w}_{p_0}, \ldots, \boldsymbol{w}_{t_m}]$ is passed to a uni-directional recurrent neural network (RNN). The RNN has a sequence of hidden states $[\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_n]$, where $\boldsymbol{h}_i = tanh(\boldsymbol{W_{hh}} \boldsymbol{h}_{i-1} + \boldsymbol{W_{xh}} \boldsymbol{x}_i)$, and $\boldsymbol{W_{hh}}$ and $\boldsymbol{W_{xh}}$ are the parameters of the RNN. The RNN is initialized with zero state values. We use the last state of the RNN $\boldsymbol{h}_n$ as the representation of the fact f.

3.2.2 Combining a set of facts. We obtain the representation of the set of encoded facts using element-wise summation of the encoded facts (vectors). We leave more elaborate methods for combining facts such as attention mechanisms [4, 12] for future work.

3.2.3 Hand-crafted features. Here, we detail the hand-crafted features $\boldsymbol{x}$ we designed or adjusted for this task. Table 1 lists the notation we use. We generate features based on feature templates that are divided into three groups: (i) those that give us a sense of *importance* of a fact, (ii) those that give us a sense of *relevance* of (f_q, f_c), and (iii) a set of miscellaneous features. Note that we use log-computations to avoid underflows.

(i) Fact importance. This group of feature templates give us a sense on how important a fact f is when taking statistics of the knowledge graph K into account at a global level. Note that we calculate these features for both facts f_q and f_c. The first of these feature templates measures *normalized predicate frequency* of each predicate p that participates in fact f (we also include the minimum, maximum and average value for each fact as metafeatures [9]). This is defined as the ratio of the size of the set of triples that have predicate p in the KG to the total number of triples:

$$PredFreq(p) = \frac{|TriplesPred(p)|}{NumTriples}. \tag{2}$$

The second feature template is the *normalized entity frequency* for each entity e that participates in fact f (we also include the minimum, maximum and average value for each fact as metafeatures). This is defined as the ratio of the number of triples in which e occurs in the KG over the number of triples in the KG:

$$EntFreq(e) = \frac{|TriplesEnt(e)|}{NumTriples}. \tag{3}$$

The final feature template in this feature group is *path informativeness*, proposed by Pirrò [26], which we apply for both f_q and f_c (recall from Section 2.1 that a fact f is a path in the KG). This feature is analog to TF.IDF and aims to estimate the importance of predicates for an entity. The informativeness of a path π is defined

Table 1: Notation

Name	Description	Definition
NumTriples	Number of triples in K	$\lvert\{\langle s,p,t\rangle : \langle s,p,t\rangle \in K\}\rvert$
TriplesPred(p)	Set of triples that have predicate p	$\{\langle s,p',t\rangle : \langle s,p',t\rangle \in K, p' = p\}$
TriplesEnt(e)	Set of triples that have entity e	$\{\langle s,p,t\rangle : \langle s,p,t\rangle \in K, s = e \lor t = e\}$
TriplesSubj(e)	Set of triples that have entity e as subject	$\{\langle s,p,t\rangle : \langle s,p,t\rangle \in K, s = e\}$
TriplesObj(e)	Set of triples that have entity e as object	$\{\langle s,p,t\rangle : \langle s,p,t\rangle \in K, t = e\}$
UniqEnt(T)	The unique set of entities in a set of triples T	$\bigcup\{\{s,t\} : \langle s,p,t\rangle \in T\}$
Types(e)	The set of types of entity e	$\{z : \langle e, type, z\rangle \in K\}$
Entities(f)	The set of entities of fact f	$\bigcup\{\{s,t\} : \forall\langle s,p,t\rangle \in f\}$
Preds(f)	The set of predicates of fact f	$\{p : \langle s,p,t\rangle \in f\}$

as follows [26]:

$$I(\pi) = \frac{1}{2|\pi|} \sum_{\langle s,p,t\rangle \in \pi} \mathit{PFITF}_{out}(p,s,K) + \mathit{PFITF}_{in}(p,t,K), \quad (4)$$

where:

$$\mathit{PFITF}_x(p,e,K) = \mathit{PF}_x(p,e) * \mathit{ITF}(p), x \in \{in, out\},$$

where $\mathit{ITF}(p)$ is the inverse triple frequency of predicate p:

$$\mathit{ITF}(p) = \log \frac{\mathit{NumTriples}}{|\mathit{TriplesPred}(p)|},$$

$\mathit{PF}_{out}(p,e)$ is the outgoing predicate frequency of e when p is the predicate:

$$\mathit{PF}_{out}(p,e) = \frac{|\mathit{TriplesSubj}(e) \cap \mathit{TriplesPred}(p)|}{|\mathit{TriplesSubj}(e)|},$$

and $\mathit{PF}_{in}(p,e)$ is the incoming predicate frequency of e when p is the predicate:

$$\mathit{PF}_{in}(p,e) = \frac{|\mathit{TriplesObj}(e) \cap \mathit{TriplesPred}(p)|}{|\mathit{TriplesObj}(e)|}.$$

(ii) Relevance. This group of feature templates gives us signal on the relevance of a candidate fact f_c w.r.t. the query fact f_q. The first of these feature templates measures *entity similarity* for each pair $(e_1, e_2) \in \mathit{Entities}(f_q) \times \mathit{Entities}(f_c)$ (we also include the minimum, maximum and average entity similarity as metafeatures). We measure entity similarity using type-based Jaccard similarity:

$$\mathit{EntTypeSim}(e_1, e_2) = \mathit{JaccardSim}(\mathit{Types}(e_1), \mathit{Types}(e_2)). \quad (5)$$

The next feature template in the *relevance* category is *entity distance*, which allows us to reason about the distance of two entities $(e_1, e_2) \in \mathit{Entities}(f_q) \times \mathit{Entities}(f_c)$ (we also include the minimum, maximum and average entity distance as metafeatures). This feature is defined as the length of the shortest path between e_1 and e_2 in K. The intuition is that we can get a signal for the relevance of f_c by measuring how "close" the entities in f_c are to the entities of f_q in the KG.

The next set of features measure *predicate similarity* between every pair of predicates $(p_1, p_2) \in \mathit{Preds}(f_q) \times \mathit{Preds}(f_c)$ (we also include the minimum, maximum and average predicate similarity as metafeatures). The intuition is that if f_c has predicates that are highly similar to the predicates in f_q, then f_c might be relevant to f_q. We measure predicate similarity in two ways. First, by measuring the co-occurrence of entities that participate in the predicates p_1 and p_2:

$$\mathit{PredCooccSim}(p_1, p_2) = \quad (6)$$
$$\mathit{JaccardSim}(\mathit{UniqEnt}(\mathit{TriplesPred}(p_1)), \mathit{UniqEnt}(\mathit{TriplesPred}(p_2))).$$

For instance, $\mathit{PredCooccSim}(p_1, p_2)$ would be high for $p1 = \mathit{starredIn}$ and $p2 = \mathit{directedBy}$. Second, by measuring the jaccard similarity of the set of predicates in f_q with the set of predicates in f_c [26]:

$$\mathit{SetPredicatesJaccardSim}(f_q, f_c) = \quad (7)$$
$$\mathit{JaccardSim}(\mathit{Preds}(f_q), \mathit{Preds}(f_c)).$$

Finally, we add a binary feature that captures whether f_q and f_c have the same CVT entity, i.e., f_c is an attribute of f_q.

(iii) Miscellaneous. This set of features includes whether f_q has a CVT entity (same for f_c). We also include whether an entity is a date (for all entities of f_q and f_c). Finally, we include the concatenation of the predicates of f_q as a feature using one-hot encoding.

4 EXPERIMENTAL SETUP

In this section we describe the setup of our experiments that aim to answer the following research questions:

RQ1 How does NFCM perform compared to a set of heuristic baselines on a crowdsourced dataset?

RQ2 How does NFCM perform compared to a scoring function that scores candidate facts w.r.t. a query fact using the relevance labels gathered from distant supervision on a crowdsourced dataset?

RQ3 Does NFCM benefit from both the handcrafted features and the automatically learned features?

RQ4 What is the per-relationship performance of NFCM? How does the number of instances per relationship affect the ranking performance?

4.1 Knowledge graph

We use the latest edition of Freebase as our knowledge graph [8]. We include Freebase relations from the following set of domains: *People, Film, Music, Award, Government, Business, Organization, Education.* Following previous work [23], we exclude triples that have an equivalent reversed triple.

4.2 Dataset

Our dataset consists of query facts, candidate facts, and a relevance label for each query-candidate fact pair. In order to construct our evaluation dataset we need to start with a set of relationships. Given that most of our domains are people-centric, we obtain this set by extracting all relationships from Freebase that have an entity of type

Table 2: Examples of relationships used in this work.

Domain	Relationship
People	*spouseOf* (*person*, *person*) *parentOf* (*person*, *person*) *educatedAt*(*person*, *organization*)
Business	*founderOf* (*person*, *organization*) *boardMemberOf* (*person*, *organization*) *leaderOf* (*person*, *organization*)
Film	*starredIn*(*person*, *film*) *directorOf* (*person*, *film*) *producerOf* (*person*, *film*)

Person as one of the entities. In the end, we are left with 65 unique relationships in total (see Table 2 for example relationships). We then proceed to gather our set of query facts. For each relationship, we sample at most 2,000 query facts, provided that they have at least one relevant fact after applying the procedure described in Section 4.3. In total, the dataset contains 62,044 query facts (954.52 on average per relationship). After gathering query facts for each relation, we enumerate candidate facts for each query fact using the procedure described in Section 3.1. Finally, we randomly split the dataset per relationship (70% of the query facts for training, 10% for validation, 20% for testing). Table 3 shows statistics of the resulting dataset.

Table 3: Statistics of the dataset gathered using distant supervision (see Section 4.3).

Part	# query facts	# candidate facts			
		average	median	max.	min.
Training	44,632	1,420	741	9,937	2
Validation	4,983	1,424	749	9,796	3
Test	12,429	1,427	771	9,924	3

Note that we train and tune the fact ranking models with the training and validation sets in Table 3 respectively, using the automatically gathered relevance labels (see Section 4.3). The test set was only used for preliminary experiments (not reported) and for constructing our manually curated evaluation dataset (see Section 4.4). We describe how we automatically gather noisy relevance labels for our dataset in the next section.

4.3 Gathering noisy relevance labels

Gathering relevance labels for our task is challenging due to the size and heterogeneous nature of KGs, i.e., having a large number of facts and relationship types. Therefore, we turn to distant supervision [23] to gather relevance labels at scale. We choose to get a supervision signal from Wikipedia for the following reasons: (i) it has a high overlap of entities with the KG we use, and (ii) facts that are in KGs are usually expressed in Wikipedia articles alongside other, related facts. We filter Wikipedia to select articles whose main entity is in Freebase, and the entity type corresponds to one of the domains listed in Section 4.1. This results in a set of 1,743,191 Wikipedia articles.

The procedure we follow for gathering relevance labels given a query fact f_q and its set of candidate facts F is as follows. For a query fact $f_q = r\langle s, t\rangle$, we focus on the Wikipedia article of entity s. First, as Wikipedia style guidelines dictate that only the first mention of another entity should be linked, we augment the articles with additional entity links using an entity linking method proposed in [32]. Next, we retain only segments of the Wikipedia article that contain references to t. Here, a segment refers to the sentence that has a reference to t and also one sentence before and one after the sentence. For each such extracted segment, we assume that it expresses the fact f_q, which is a common assumption in gathering noisy training data for relation extraction [23]. From the segments, we then collect a set of other entities, O, that occur in the same sentence that mentions t: for computational efficiency, we enforce $|O| \leq 20$. Then, we extract facts for all possible pairs of entities $\langle e_1, e_2\rangle \in \{O \cup \{s, t\}\} \times \{O \cup \{s, t\}\}$. If there is a single fact f_c in K that connects e_1 and e_2, we deem f_c relevant for f_q. However, if there are multiple facts connecting e_1 and e_2 in K, the mention of the fact in the specific segment is ambiguous and thus we do not deem any of these facts as relevant [30]. The rest of the facts in F are deemed irrelevant for f_q.

The distribution of relevant/non-relevant labels in the distantly supervised data is heavily skewed: out of 87,998,956 facts in total, only 225,032 are deemed to be relevant (0.26%). This is expected since the candidate fact enumeration step can generate thousands of facts for a certain query fact (see Section 3.1).

As a sanity check, we evaluate the performance of our approach to collect distant supervision data by sampling 5 query facts for each relation in our dataset. For these query facts, we perform manual annotations on the extracted candidate facts that were deemed as relevant by the distant supervision procedure. We obtain an overall precision of 76% when comparing the relevance labels of the distant supervision against our manual annotations. This demonstrates the potential of our distant supervision strategy for creating training data.

4.4 Manually curated evaluation dataset

In order to evaluate the performance of NFCM on the KG fact contextualization task, we perform crowdsourcing to collect a human-curated evaluation dataset. The procedure we use to construct this evaluation dataset is as follows. First, for each of the 65 relationships we consider, we sample five query facts of the relationship from the test set (see Section 4.2). Since fact enumeration for a query fact can yield hundreds or thousands of facts (Section 3.1), it is infeasible to consider all the candidate facts for manual annotation. Therefore, we only include a candidate fact in the set of facts to be annotated if: (i) the candidate fact was deemed relevant by the automatic data gathering procedure (Section 4.3), or (ii) the candidate fact matches a fact pattern that is built using relevant facts that appear in at least 10% of the query facts of a certain relationship. An example fact pattern is *parentOf*⟨?, ?⟩, which would match the fact *parentOf*⟨Bill Gates, Jennifer Gates⟩.

We use the CrowdFlower platform, and ask the annotators to judge a candidate fact w.r.t. its relevance to a query fact. We provide the annotators with the following scenario (details omitted for brevity):

> *We are given a specific real-world fact, e.g., "Bill Gates is the founder of Microsoft", which we call the query fact. We are interested in writing a description of the query fact (a sentence or a small paragraph). The purpose of this assessment task is to identify other facts that could be included in a description of the query fact. Note that*

Table 4: Relevance label distribution of the crowdsourced evaluation dataset.

Relevance	Non-attribute facts (%)	Attribute facts (%)
Irrelevant	60.86	34.34
Somewhat relevant	34.49	57.81
Very relevant	4.63	7.84

> *even though all facts presented for assessment will be accurate, not all will be relevant or equally important to the description of the main fact.*

We ask the annotators to assess the relevance of a candidate fact in a 3-graded scale:

- *very relevant*: I would include the candidate fact in the description of the query fact; the candidate fact provides additional context to the query fact.
- *somewhat relevant*: I would include the candidate fact in the description of the query fact, but only if there is space.
- *irrelevant*: I would not include the candidate fact in the description of the query fact.

Alongside each query-candidate fact pair, we provide a set of extra facts that could possibly be used to decide on the relevance of a candidate fact. These include facts that connect the entities in the query fact with the entities in the candidate fact. For example, if we present the annotators with the query fact *spouseOf*⟨Bill Gates, Melinda Gates⟩ and the candidate fact *parentOf*⟨Melinda Gates, Jennifer Gates⟩ we also show the fact *parentOf*⟨Bill Gates, Jennifer Gates⟩.

Each query-candidate fact pair is annotated by three annotators. We use majority voting to obtain the gold labels, breaking ties arbitrarily. The annotators get a payment of 0.03 dollars per query-candidate fact pair.

By following the crowdsourcing procedure described above, we obtain 28,281 fact judgments for 2,275 query facts (65 relations, 5 query facts each). Table 4 details the distribution of the relevance labels. One interesting observation is that facts that are attributes of other facts (see Section 2.1) tend to have relatively more relevant judgments than the ones that are not. This is expected since some of them are attributes of the query fact (e.g., date of marriage for a *spouseOf* query fact). Finally, Fleiss' kappa is $\kappa = 0.4307$, which is considered moderate agreement. Note that all the results reported in Section 5 are on the manually curated dataset described here.

Evaluation metrics. We use the following standard retrieval evaluation metrics: MAP, NDCG@5, NDCG@10 and MRR. In the case of MAP and MRR, which expect binary labels, we consider "very relevant" and"somewhat relevant" as "relevant". We report on statistical significance with a paired two-tailed t-test.

4.5 Heuristic baselines

To the best of our knowledge, there is no previously published method that addresses the task introduced in this paper. Therefore, we devise a set of intuitive baselines that are used to showcase that our task is not trivial. We derive them by combining features we introduced in Section 3.2.3. We define these heuristic functions below:

- *Fact informativeness (FI).* Informativeness of the candidate fact f_c [26, Eq. 4]. This baseline is independent of f_q.
- *Average predicate similarity (APS).* Average predicate similarity of all pairs of predicates $(p_1, p_2) \in Preds(f_q) \times Preds(f_c)$ (Eq. 6). The intuition here is that f_c might be relevant to f_q if it contains predicates that are similar to the predicates of f_q.
- *Average entity similarity (AES).* Average entity similarity of all pairs of entities in $(e_1, e_2) \in Entities(f_q) \times Entities(f_c)$ (Eq. 5). The assumption here is that f_c might be relevant to f_q if it contains entities that are similar to the entities of f_q.

4.6 Implementation details

The models described in Section 3.2 are implemented in TensorFlow v.1.4.1 [1]. Table 5 lists the hyperparameters of NFCM. We tune the variable hyper-parameters of this table on the validation set and optimize for NDCG@5.

Table 5: Hyperparameters of NFCM, tuned on the validation set.

Description	Value(s)
# negative samples k during training	[1, 10, 100]
Learning rate	[0.01, 0.001, 0.0001]
d_z: entity type embedding dimension	[64, 128, 256]
d_p: Predicate embedding dimension	[64, 128, 256]
RNN cell size	[64, 128, 256]
RNN cell dropout	[0.0, 0.2]
α: # hidden layers of MLP-o	[0, 1, 2]
β: # dimension of MLP-o hidden layers	[50, 100]
L2 regularization factor for MLP-o kernel	[0.0, 0.1, 0.2]

5 RESULTS AND DISCUSSION

In this section we discuss and analyze the results of our evaluation, answering the research questions listed in Section 4.

In our first experiment, we compare NFCM to a set of heuristic baselines we derived to answer **RQ1**. Table 6 shows the results. We observe that NFCM significantly outperforms the heuristic baselines by a large margin. We have also experimented with linear combinations of the above heuristics but the performance does not improve over the individual ones and therefore we omit those results. We conclude that the task we define in this paper is not trivial to solve and simple heuristic functions are not sufficient.

In our second experiment we compare NFCM with distant supervision and aim to answer **RQ2**. That is, how does NFCM perform compared to DistSup, a scoring function that scores candidate facts w.r.t. a query fact using the relevance labels gathered from distant supervision. The aim of this experiment is to investigate whether it is beneficial to learn ranking functions based on the signal gathered from distant supervision, and to see if we can improve performance over the latter. Table 7 shows the results. We observe that NFCM significantly outperforms DistSup on MAP, NDCG@5, and NDCG@10 and conclude that learning ranking functions (and in particular NFCM) based on the signal gathered from distant supervision is beneficial for this task. We also observe that NFCM performs significantly worse than DistSup on MRR. One possible reason for this is that NFCM returns facts that are indeed relevant but were not selected for annotation and thus assumed not relevant, since the data annotation procedure is biased towards DistSup (see Section 4.4). We aim to validate this hypothesis by conducting an

Table 6: Comparison between NFCM and the heuristic baselines. Significance is tested between NFCM and AES, the best performing baseline. We depict a significant improvement of NFCM over AES for $p < 0.05$ as $^\blacktriangle$.

Method	MAP	NDCG@5	NDCG@10	MRR
FI	0.1222	0.0978	0.1149	0.1928
APS	0.2147	0.2175	0.2354	0.3760
AES	0.2950	0.3284	0.3391	0.5214
NFCM	**0.4874**$^\blacktriangle$	**0.5110**$^\blacktriangle$	**0.5289**$^\blacktriangle$	**0.7749**$^\blacktriangle$

Table 7: Comparison between NFCM and the distant supervision baseline. We depict a significant improvement of NFCM over DistSup as $^\blacktriangle$ and a significant decrease as $^\blacktriangledown$ ($p < 0.05$).

Method	MAP	NDCG@5	NDCG@10	MRR
DistSup	0.2831	0.4489	0.3983	**0.8256**
NFCM	**0.4874**$^\blacktriangle$	**0.5110**$^\blacktriangle$	**0.5289**$^\blacktriangle$	0.7749$^\blacktriangledown$

Table 8: Comparison between the full NFCM model and its variations. Significance is tested between NFCM and its best variation (LF). We depict a significant improvement of NFCM over LF for $p < 0.05$ as $^\blacktriangle$.

Method	MAP	NDCG@5	NDCG@10	MRR
HF	0.4620	0.4753	0.4989	0.7180
LF	0.4676	0.4993	0.5134	0.7647
NFCM	**0.4874**$^\blacktriangle$	**0.5110**	**0.5289**$^\blacktriangle$	**0.7749**

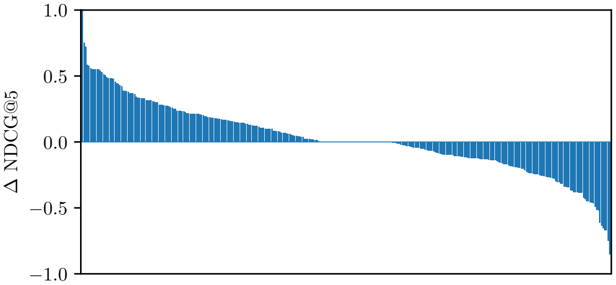

Figure 5: Per query fact differences in NDCG@5 between the variation of NFCM that only uses the learned features (LF) and the best-performing variation of NFCM that only uses the hand-crafted features (HF). A positive value indicates that LF performs better than HF on a query fact and vice versa.

additional user study in future work. Nevertheless, having an automatic method for KG fact contextualization trained with distant supervision becomes increasingly important for tail entities for which we might only have information in the KG itself and not in external text corpora or other sources.

In order to answer **RQ3**, that is, whether NFCM benefits from both the hand-crafted features and the learned features, we perform an ablation study. Specifically, we test the following variations of NFCM that only modify the final layer of the architecture (see Section 3.2):

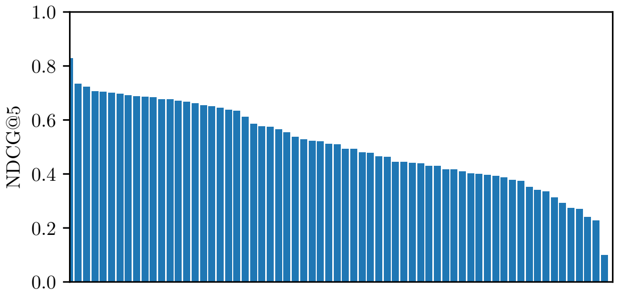

Figure 6: NDCG@5 for NFCM per relationship.

(i) LF: Keeps the learned features ($\boldsymbol{v_q}$ and $\boldsymbol{v_a}$), and ignores the hand-crafted features $\boldsymbol{x}$.
(ii) HF: Keeps the hand-crafted features ($\boldsymbol{x}$) and ignores the learned features ($\boldsymbol{v_q}$ and $\boldsymbol{v_a}$).

We tune the parameters of LF and HF on the validation set. Table 8 shows the results. First, we observe that NFCM outperforms HF by a large margin. Also, NFCM outperforms LF on all metrics (significantly so for MAP and NDCG@10) which means that by combining HF and LF we are able to obtain more relevant results at lower positions of the ranking. We aim to explore more sophisticated ways of combining LF and HF in future work. In order to verify whether LF and HF have complementary signals, we plot the per-query differences in NDCG@5 for LF and HF in Figure 5. We observe that the performance of LF and HF varies across query facts, confirming the hypotheses that LF and HF yield complementary signals.

In order to answer **RQ4**, we conduct a performance analysis per relationship. Figure 6 shows the per-relationship NDCG@5 performance of NFCM – query fact scores are averaged per relationship. The relationship for which NFCM performs best is *profession*, which has a NDCG@5 score of 0.8275. The relationship for which NFCM performs worst at is *awardNominated*, which has a NDCG@5 score of 0.1. Further analysis showed that *awardNominated* has a very large number of candidate facts on average, which might explain the poor performance on that relationship.

Furthermore, we investigate how the number of queries we have in the training set for each relationship affects the ranking performance. Figure 7 shows the results. From this figure we conclude that there is no clear relationship and thus that NFCM is robust to the size of the training data for each relationship.

Next, we analyse the performance of NFCM with respect to the number of candidates per query fact; Figure 8 shows the results. We observe that the performance decreases when we have more candidate facts for a query, although not by a large margin, and that there does not seem to be a clear relationship between performance and the number of candidates to rank.

6 RELATED WORK

The specific task we introduce in this paper has not been addressed before, but there is related work in three main areas: entity relationship explanation, distant supervision, and fact ranking.

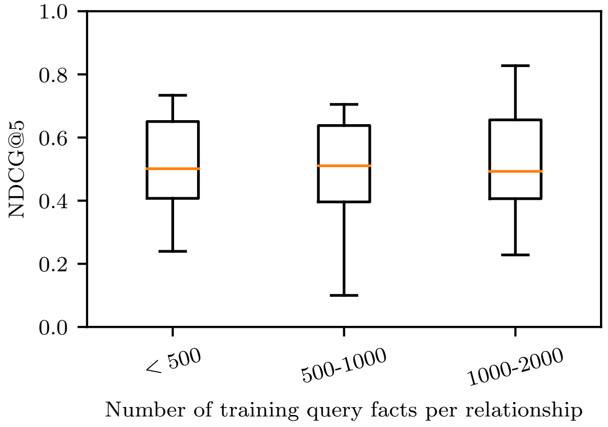

Figure 7: Box plot that shows NDCG@5 per number of training query facts of each relationship (binned). Each box shows the median score with an orange line and the upper and lower quartiles (maximum and lower values shown outside each box).

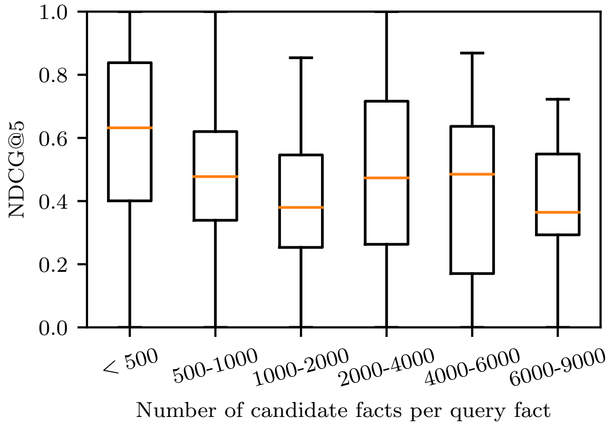

Figure 8: Box plot that shows NDCG@5 per number of candidate facts of each query fact (binned). Each box shows the median score with an orange line and the upper and lower quartiles (maximum and lower values shown outside each box).

6.1 Relationship Explanation

Explanations for relationships between pairs of entities can be provided in two ways: *structurally*, i.e., by providing paths or subgraphs in a KG containing the entities, or *textually*, by ranking or generating text snippets that explain the connection.

Fang et al. [14] focus on explaining connections between entities by mining relationship explanation patterns from the KG. Their approach consists of two main components: explanation enumeration and explanation ranking. The first phase generates all patterns in the form of paths connecting the two entities in the KG, which are then combined to form explanations. In the final stage, the candidate explanations are ranked using notions of interestingness. Seufert et al. [29] propose a similar approach for entity sets. Their method focuses on explaining the connections between entity sets based on the concept of relatedness cores, i.e., dense subgraphs that have strong relations with both query sets. Pirrò [26] also provide explanations of the relation between entities in terms of the top-k most informative paths between a query pair of entities; such paths are ranked and selected based on path informativeness and diversity, and pattern informativeness.

As to textual explanations for entity relationships, Voskarides et al. [33] focus on human-readable descriptions. They model the task as a learning to rank problem for sentences and employ a rich set of features. Huang et al. [18] build on the aforementioned work and propose a pairwise ranking model that leverages clickthrough data and uses a convolutional neural network architecture. While these approaches rank existing candidate explanations, Voskarides et al. [32] focus on generating explanations from scratch. They automatically identify the most common sentence templates for a particular relationship and, for each new relationship instance, these templates are ranked and instantiated using contextual information from the KG.

The work described above focuses on explaining entity relationships in KGs; no previous work has focused on ranking additional KG facts for an input entity relationship as we do in this paper.

6.2 Distant Supervision

When obtaining labeled data is expensive, training data can be generated automatically. Mintz et al. [23] introduce distant supervision for relation extraction; for a pair of entities that is connected by a KG relation, they treat all sentences that contain those entities in a text corpus as positive examples for that relation. Follow-up work on relation extraction address the issue of noise related to distant supervision. Alfonseca et al. [3], Riedel et al. [28], Surdeanu et al. [31] refine the model by relaxing the assumptions in the original method or by modeling noisy labels.

Beyond relation extraction, distant supervision has also been applied in other KG-related tasks. Ren et al. [27] introduce a joint approach entity recognition and classification based on distant supervision. Ling and Weld [21] used distant supervision to automatically label data for fine-grained entity recognition.

6.3 Fact Ranking

In fact ranking, the goal is to rank a set of attributes with respect to an entity. Hasibi et al. [17] consider fact ranking as a component for entity summarization for entity cards. They approach fact ranking as a learning to rank problem. They learn a ranking model based on importance, relevance, and other features relating a query and the facts. Aleman-Meza et al. [2] explore a similar task, but rank facts with respect to a pair of entities to discover paths that contain informative facts between the pair.

Graph matching involves matching two graphs and discovering the patterns of relationships between them to infer their similarity [11]. Although our task can be considered as comparing a small query subgraph (i.e., query triples) and a knowledge graph, the goal is different from graph matching which mainly concerns aligning two graphs rather than enhancing one query graph.

Our work differs from the work discussed above in the following major ways. First, we enrich a query fact between two entities by providing relevant additional facts in the context of the query fact, taking into account both the entities and the relation of the query fact. Second, we rank whole facts from the KG instead of just entities. Last, we provide a distant supervision framework for generating the training data so as to make our approach scalable.

7 CONCLUSION

In this paper, we introduced the knowledge graph fact contextualization task and proposed NFCM, a weakly-supervised method to address it. NFCM first generates a candidate set for a query fact by looking at 1 or 2-hop neighbors and then ranks the candidate facts using supervised machine learning. NFCM combines handcrafted features with features that are automatically identified using deep learning. We use distant supervision to boost the gathering of training data by using a large entity-tagged text corpus that has a high overlap with entities in the KG we use. Our experimental results show that (i) distant supervision is an effective means for gathering training data for this task, (ii) NFCM significantly outperforms several heuristic baselines for this task, and (iii) both the handcrafted and automatically-learned features contribute to the retrieval effectiveness of NFCM. For future work, we aim to explore more sophisticated ways of combining handcrafted with automatically learned features for ranking. Additionally, we want to explore other data sources for gathering training data, such as news articles and click logs. Finally, we want to explore methods for combining and presenting the ranked facts in search engine result pages in a diversified fashion.

Data

To facilitate reproducibility of our results, we share the data used to run our experiments at https://www.techatbloomberg.com/research-weakly-supervised-contextualization-knowledge-graph-facts/.

Acknowledgements

The authors would like to thank the anonymous reviewers (and especially reviewer #1) for their useful and constructive feedback. This research was supported by Ahold Delhaize, Amsterdam Data Science, the Bloomberg Research Grant program, the China Scholarship Council, the Criteo Faculty Research Award program, Elsevier, the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement nr 312827 (VOX-Pol), the Google Faculty Research Awards program, the Microsoft Research Ph.D. program, the Netherlands Institute for Sound and Vision, the Netherlands Organisation for Scientific Research (NWO) under project nrs CI-14-25, 652.002.001, 612.001.551, 652.001.003, and Yandex. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. TensorFlow: A System for Large-Scale Machine Learning.. In *OSDI*, Vol. 16. USENIX Association, 265–283.

[2] B. Aleman-Meza, C. Halaschek-Weiner, I. B. Arpinar, Cartic Ramakrishnan, and A. P. Sheth. 2005. Ranking complex relationships on the semantic Web. *IEEE Internet Computing* 9, 3 (May 2005), 37–44.

[3] Enrique Alfonseca, Katja Filippova, Jean-Yves Delort, and Guillermo Garrido. 2012. Pattern Learning for Relation Extraction with a Hierarchical Topic Model. In *ACL*. ACL, Stroudsburg, PA, USA, 54–59.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR* abs/1409.0473 (2014), 1–15.

[5] Hannah Bast, Buchhold Björn, and Elmar Haussmann. 2016. Semantic search on text and knowledge bases. *Found. Trends Inf. Retr.* 10, 2-3 (June 2016), 119–271.

[6] Roi Blanco, Berkant Barla Cambazoglu, Peter Mika, and Nicolas Torzec. 2013. Entity Recommendations in Web Search. In *ISWC*. Springer Berlin Heidelberg, Berlin, Heidelberg, 33–48.

[7] Roi Blanco, Giuseppe Ottaviano, and Edgar Meij. 2015. Fast and Space-Efficient Entity Linking for Queries. In *WSDM*. ACM, New York, NY, USA, 179–188.

[8] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *SIGMOD*. ACM, New York, NY, USA, 1247–1250.

[9] Alexey Borisov, Pavel Serdyukov, and Maarten de Rijke. 2016. Using metafeatures to increase the effectiveness of latent semantic models in web search. In *WWW*. ACM, New York, NY, USA, 1081–1091.

[10] Horatiu Bota, Ke Zhou, and Joemon M. Jose. 2016. Playing Your Cards Right: The Effect of Entity Cards on Search Behaviour and Workload. In *CHIIR*. ACM, New York, NY, USA, 131–140.

[11] M. Cho, K. Alahari, and J. Ponce. 2013. Learning Graphs to Match. In *ICCV*. IEEE, 25–32.

[12] Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2017. Chains of Reasoning over Entities, Relations, and Text using Recurrent Neural Networks. In *EACL*. ACL, Valencia, Spain, 132–141.

[13] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W Bruce Croft. 2017. Neural ranking models with weak supervision. In *SIGIR*. ACM, New York, NY, USA, 65–74.

[14] Lujun Fang, Anish Das Sarma, Cong Yu, and Philip Bohannon. 2011. Rex: explaining relationships between entity pairs. *VLDB* 5, 3 (2011), 241–252.

[15] Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating Training Corpora for NLG Micro-Planners. In *ACL*. ACL, 179–188.

[16] Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing Knowledge Graphs in Vector Space. In *EMNLP*. ACL, 318–327.

[17] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. 2017. Dynamic Factual Summaries for Entity Cards. In *SIGIR*. ACM, New York, NY, USA, 773–782.

[18] Jizhou Huang, Wei Zhang, Shiqi Zhao, Shiqiang Ding, and Haifeng Wang. 2017. Learning to Explain Entity Relationships by Pairwise Ranking with Convolutional Neural Networks. In *IJCAI*. IJCAI, 4018–4025.

[19] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014), 1–15. arXiv:1412.6980

[20] Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural Text Generation from Structured Data with Application to the Biography Domain. In *EMNLP*. ACL, 1203–1213.

[21] Xiao Ling and Daniel S. Weld. 2012. Fine-grained Entity Recognition. In *AAAI*. AAAI Press, 94–100.

[22] Iris Miliaraki, Roi Blanco, and Mounia Lalmas. 2015. From "Selena Gomez" to "Marlon Brando": Understanding Explorative Entity Search. In *WWW*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 765–775.

[23] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL/AFNLP*. ACL, 1003–1011.

[24] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016. 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 (2016), 11–33.

[25] Thomas Pellissier Tanon, Denny Vrandečić, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher. 2016. From Freebase to Wikidata: The Great Migration. In *WWW*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 1419–1428.

[26] Giuseppe Pirrò. 2015. Explaining and Suggesting Relatedness in Knowledge Graphs. In *ISWC*. Springer-Verlag New York, Inc., New York, NY, USA, 622–639.

[27] Xiang Ren, Ahmed El-Kishky, Chi Wang, Fangbo Tao, Clare R. Voss, and Jiawei Han. 2015. ClusType: Effective Entity Recognition and Typing by Relation Phrase-Based Clustering. In *KDD*. ACM, New York, NY, USA, 995–1004.

[28] Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling Relations and Their Mentions Without Labeled Text. In *ECML-PKDD*. Springer-Verlag, Berlin, Heidelberg, 148–163.

[29] Stephan Seufert, Klaus Berberich, Srikanta J. Bedathur, Sarath Kumar Kondreddi, Patrick Ernst, and Gerhard Weikum. 2016. ESPRESSO: Explaining Relationships Between Entity Sets. In *CIKM*. ACM, New York, NY, USA, 1311–1320.

[30] Daniil Sorokin and Iryna Gurevych. 2017. Context-Aware Representations for Knowledge Base Relation Extraction. In *EMNLP*. ACL, 1784–1789.

[31] Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance Multi-label Learning for Relation Extraction. In *EMNLP-CoNLL*. ACL, 455–465.

[32] Nikos Voskarides, Edgar Meij, and Maarten de Rijke. 2017. Generating Descriptions of Entity Relationships. In *ECIR*. Springer International Publishing, Cham, 317–330.

[33] Nikos Voskarides, Edgar Meij, Manos Tsagkias, Maarten de Rijke, and Wouter Weerkamp. 2015. Learning to Explain Entity Relationships in Knowledge Graphs. In *ACL-IJCNLP*. ACL, 564–574.

[34] Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base. In *ACL-IJCNLP*. ACL, 1321–1331.