Towards Annotating Relational Data on the Web with Language Models

Matteo Cannaviccio
Roma Tre University
Rome, Italy
cannaviccio@uniroma3.it

Denilson Barbosa
University of Alberta
Edmonton, AB, Canada
denilson@ualberta.ca

Paolo Merialdo
Roma Tre University
Rome, Italy
paolo.merialdo@uniroma3.it

ABSTRACT

Tables and structured lists on Web pages are a potential source of valuable information, and several methods have been proposed to annotate them with semantics that can be leveraged for search, question answering and information extraction. This paper is concerned with the specific problem of finding and ranking relations from a given Knowledge Graph (KG) that hold over pairs of entities juxtaposed in a table or structured list. The state-of-the-art for this task is to attempt to link the entities mentioned in the table cells to objects in the KG and rank the relations that hold for those linked objects. As a result, these methods are hampered by the incompleteness and uneven coverage in even the best knowledge graphs available today. The alternative described here does not require entity linking, relying instead on ranking relations using generative language models derived from Web-scale corpora. As such, it can produce quality results even when the entities in the table are missing in the KG. The experimental validation, designed to expose the challenges posed by KG incompleteness, shows that our approach is robust and effective in practice.

ACM Reference Format:
Matteo Cannaviccio, Denilson Barbosa, and Paolo Merialdo. 2018. Towards Annotating Relational Data on the Web with Language Models. In *WWW 2018: The 2018 Web Conference, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3178876.3186029

1 INTRODUCTION

The Web is a vast source of intrinsically relational knowledge expressed in hundreds of millions of tables and many more structured lists within billions of documents. Web-scale table corpora have found many applications, including search and question answering [5, 15, 28], knowledge graph construction [8, 19, 26, 27], schema understanding and auto-complete [5, 31], to name a few. However, unlike with documents in which information is encoded in text amenable to natural language understanding tools, the facts and relationships encoded in tables are *implicit*, and therefore hard to extract automatically. The various approaches for *understanding* Web tables amount to two main tasks: (1) identifying the *type* of each column in a table, and (2) identifying the *relationship* between pairs of columns in the table. As a motivating example, Fig. 1 shows a snippet of a prototypical table (from Wikipedia in this case) that can be easily extracted and parsed with tools like Google tables or an automatic wrapper induced from a set DOM-trees. It is clear to a human looking at the table that the second column has *actors* who played in the *movies* in the first column. With little effort (e.g., after reading the article with that table) a human can infer that the third column has *countries* that were *filming locations* of the *movies* in the first column. Yet, extracting those types and relationships is out of reach of text-based Information Extraction tools that rely on linguistic patterns used to encode knowledge [17].

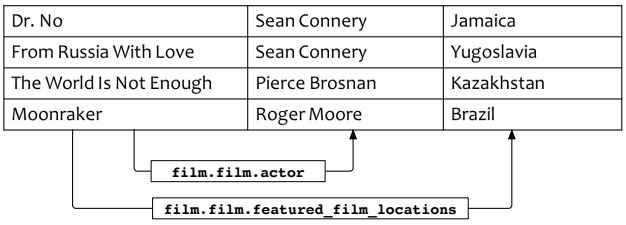


Figure 1: Web table with actors and filming locations of James Bond movies.

The first methods for Web table understanding [28] were strictly lexical, using frequently occurring keywords and phrases as annotations, and are primarily useful for keyword-based table search. The prevalent approach, however, is to leverage existing Web-scale Knowledge Graphs (KGs) for *semantic* Web table understanding, whereby one annotates columns with *classes* of entities and pairs of columns with *relationships* from the KG ontology [12, 15, 24, 25, 37]. To do so, these methods attempt to disambiguate the entities in the table by *linking* them to objects in the KG. If this can be done, one can immediately annotate each table column with the ontology type covering the entities in the column. Next, the relationship between a pair of columns can be inferred *ranking* KG relations based on their *coverage* of (entity pairs in the rows) of the table.

Although highly intuitive, the approach outlined above is hampered by the fact that even the best existing KGs are notoriously incomplete [18, 29, 32], missing many entities (not only obscure tail entities) as well as many relations among the entities. More precisely, KG incompleteness introduces two problems. First, if the Web table contains mostly entities that are not in the KG or cannot be easily linked, no table annotations are possible. It is worth mentioning that state-of-the-art entity linking methods rely on textual

WWW 2018, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5639-8/18/04.
https://doi.org/10.1145/3178876.3186029

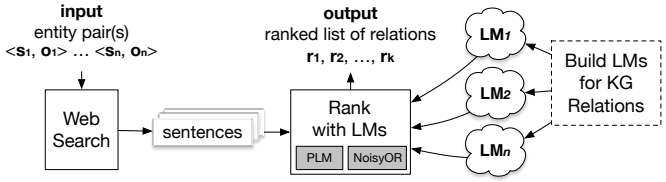


Figure 2: Overview of our approach.

phrase	freq.
was released in	0.23
topped the charts in	0.17
is available only in	0.14
...	...
shot in	0.04

(a) `music/release/region`

phrase	freq.
was filmed in	0.44
set in	0.26
shot in	0.14
...	...
was released in	0.03

(b) `.../featured_film_locations`

Figure 3: Generative language models for two relations between works of art and countries.

sentences	freq.
Dr. No filmed entirely in **Jamaica**	6
Dr. No location Ocho Rios **Jamaica**	3
Dr. No based and shot in **Jamaica**	2
Dr. No filmed in Kingston, **Jamaica**	1

Figure 4: Web search results for ⟨`"Dr. No","Jamaica"`⟩.

features (e.g., keyphrases) which are hardly available in the context of Web table understanding. Second, the KG coverage for specific relations is often biased, which can lead to unexpected results even if the entities can be correctly linked as illustrated next.

The example of Fig. 1 was chosen systematically to illustrate the problems caused by KG incompleteness. We picked a table with a column whose cells would: (1) be difficult to link to Freebase objects, and (2) participate in a partially populated relation of interest. In our example, the names of the movies are hard to distinguish from their respective *soundtrack* albums, even with sophisticated string matching methods (e.g., [30]). Moreover, although we can easily disambiguate the countries in the table, and although Freebase has a dedicated relation for filming locations of movies, the coverage of that relation is heavily biased towards recent movies, lacking the filming location of most Bond movies. Freebase is not as incomplete, however, in the music domain. In fact, it contains all countries were the soundtracks of the Bond movies were released. As a result, one would be biased towards annotating the relationship between the first and third columns of Fig. 1 with the predicate for the region where an album is released, which would be incorrect in this case. (In passing, at the time of writing, both YAGO and DBpedia also lack the filming locations of most Bond movies.)

1.1 Problem Statement

For convenience, we adopt the Freebase notation and terminology.

As customary, we model a **knowledge graph** as a labeled, directed multi-graph $KG = (N, E, L)$ where N, E, and L are sets of nodes, edges, and labels. Nodes can be entities, represented using unique identifiers called "m-ids" (e.g., `m/03_gd`), text literals in quotes (e.g., `"James Cameron"`), or types denoted as paths in an ontology (e.g., `film/director`). Labels in L define relation names (e.g., `film/director/film`). Edges in $E \subseteq N \times L \times N$ are called statements or triples and can be used to assign types to entities (e.g., ⟨`m/03_gd`, `type/object/type`, `film/director`⟩); to describe entities (e.g., ⟨`m/03_gd`, `type/object/name`, `"James Cameron"`⟩) or to relate pairs of entities (e.g., ⟨`m/03_gd`, `film/director/film`, `m/0dr_4`⟩). We assume the KG ontology specifies types for the domain and range of every relation.

This paper seeks to describe and evaluate a principled and effective way of predicting KG relations that hold over columns of Web tables. (Note that doing so allows one to annotate the columns themselves with the domain and range types from the KG ontology for those relations.) Predicting which relation(s) hold for pairs of entities amounts to *ranking* all KG relations for those entities followed by either thresholding or taking the top-k relations in the ranking. Thus, in this paper we focus on, and evaluate relation rankings instead of predictions. Without loss of generality, we assume the input to be a set of entity pairs, as one can always convert a multi-column table or a nested list into one or more sets of pairs. Also, we assume that all pairs in the set are similarly related; or, in other words, the input is not random. More precisely:

Definition 1. *Given a set $I = \{\langle s_1, o_1\rangle, \ldots, \langle s_n, o_n\rangle\}$ of subject-object entity pairs, and a list L of relation names from a KG, produce a ranked list $r_1, \ldots, r_k$ of k relations from L that hold over pairs in I, sorted by decreasing relevance.*

1.2 Overview of Our Approach

Fig. 2 illustrates our method, which is heavily inspired by the established Language Models (LMs) for IR approach [36], in which the goal is to model each document separately and score documents w.r.t. queries based on how likely the corresponding models are to generate the query. Keeping with this analogy, the KG relations in our setting play the role of the "documents" and the pairs of entities play the role of the "queries".

Models of KG Relations. We model a KG relation distributionally using the *phrases* that are used to express it. We learn such models from the approximately 500M texts in English of the ClueWeb09 corpus, leveraging Google's FACC1 annotation corpus assigning m-ids of Freebase entities to the *mentions* in those texts where these entities appear. Following the state-of-the-art in open relation extraction [20], we gather phrases appearing *between* m-ids in the corpus and filter out phrases that describe ontological relations (e.g., "is a" and variants) and phrases that do not conform to known patterns defined at the level of parts-of-speech [7] tags. The details of the construction of LMs are given in Sec. 3.

Relation Ranking. To rank KG relations for entity pair $\langle s_i, o_i\rangle$, we perform a Web search with those entities and extract *relational phrases* connecting the entities in the result of the Web search. Then, we score the KG relations based on how likely their corresponding models are to generate the set of phrases extracted from the Web search. Continuing with our running example, Fig. 3a

shows some of the phrases of the LM associated with the relation associating albums to the countries they have been released, while Fig. 3b shows phrases for the relation for filming locations of movies. Fig. 4 shows some sentences from the Web search with entities ⟨"Dr. No","Jamaica"⟩. In this case, the relation about movie locations will rank higher than the relation about album releases.

Two models for KG relation scoring are considered here (see Sec. 4.2): (1) PLM, the "standard" *conjunctive* approach of maximizing the likelihood of all terms in the query, and (2) Noisy OR, a *disjunctive* approach where the ranking may be biased towards a small number of highly relevant phrases. Finally, note that when the input consists of multiple entity pairs, we need a way of finding aggregate scores for the relations relative to all such pairs. Again, two ways of doing so are discussed and evaluated here (Sec. 4.3): (*a*) a *global* strategy that combines all sentences of all pairs into a single *global* query used to rank the LMs (once), and (*b*) a *local* strategy where we produce a ranking for each pair, merging them to arrive at a final prediction.

Evaluation. To the best of our knowledge, there are no benchmarks concerned with KG incompleteness and how they affect table understanding. Therefore, we designed two experiments to illustrate how our method can overcome this problem. In the first we use a synthetic benchmark with facts that are both true and known to be missing from Freebase, DBpedia, and YAGO (Sec. 5), following our previous work [6]. In the second we experiment with tables from Wikipedia for which a state-of-the-art web table annotation tool [24] fails to produce any output (Sec. 5.7).

Contribution. We describe an effective method for ranking KG relations that applies to pairs of named entities that is less susceptible to KG incompleteness than the current state-of-the-art. Departing from previous work, our method does *not* require that the entities are linked to, or even exist in the KG. Instead, our method works whenever a Web search returns phrases describing how the entities are related. Our method is general: it is not specific to any KG, corpus or natural language, and the Web search can be replaced by a search on any large corpus. Finally, perhaps the biggest advantage of our method is that it predicts relations based on *corpus* statistics which are *independent* of, and not biased by the KG coverage. Our experimental evaluation, which has been conducted on publicly available datasets, indicates that our method can correctly predict relationships for in-KG and for out-of-KG entities, where state-of-the-art approaches fail, and thus can significantly contribute to improve previous work in this area.

2 RELATED WORK

Language Models have found many uses in Information Retrieval beyond document ranking [14, 22, 36]. A state-of-the-art entity search method [2] is based on "entity LMs" that harness entity categories (i.e., semantic types) for ranking and filtering answers based on a desired *type* (e.g., movies, albums, etc.). Other applications include searching and ranking over RDF-structured Linked Data and knowledge graphs with queries that combine keywords and entity examples [4] or interpret so-called telegraphic text queries [13] on the underlying structured data [23]. LMs have also been found useful in ranking the results of exact, relaxed and keyword-augmented graph-pattern queries over RDF graphs [10, 34], which has applications in translating natural language questions into SPARQL queries over KGs [35], among others. We use LMs for relation prediction.

Our work borrows from relation prediction methods that exploit the duality between KG relations and phrases occurring in text (e.g., [1, 3, 9, 17, 33]), except that we employ strict filters to remove non-relational and ontological patterns [7]. While we use LMs for prediction and achieve good results, other ranking models from Information Retrieval and/or other relation prediction models from the field of Information Extraction could be used for the same purpose. We leave as future work investigating other scoring models and how they fare in our setting.

We are motivated by the problem of understanding Web tables, widely recognized as a valuable knowledge source on the Web [5]. The first solution [28] is meant for search, annotating columns with keywords from an "is-a" database and relationships between columns with *keyphrases* frequently occurring with the entities in the table. On the other hand, by annotating pairs of columns with KG relations that hold over the respective entities, our method annotates the tables with semantic information. With our method, the columns can be annotated with the *expected* (semantic) types for the relations as per the KG schema.

Recent work on Web table understanding links entities in *table cells* to KG objects and pairs of columns with KG relations that hold over them [12, 15, 25, 37]. An earlier work in this area [15] learns a probabilistic graphical model that collectively annotates cells with entity identifiers, columns with KG types and pairs of columns with KG relations, maximizing the joint probability of the assignment. Another idea is to model each row of the table as a set of (possibly multi-valued) attributes describing a single entity in the KG [25]; each row of the table is then matched with entities in DBpedia, taking into account the table headers and how well they match classes in the DBpedia ontology. Unlike these works, our method does not require linking entities to KG objects and, thus, should be less susceptible to KG incompleteness. We validate this hypothesis experimentally in two ways. First, we use a synthetic benchmark with facts that are both true and known to be missing from Freebase, DBpedia, and YAGO (Sec. 5), obtained from [6]. Second, we experiment with tables from Wikipedia for which a state-of-the-art web table annotation tool [24] fails to produce any output (Sec. 5.7). Our evaluation confirms our hypothesis and suggests our method can be *used in conjunction* with previous work to lead to better Web table understanding tools.

3 BUILDING LANGUAGE MODELS

Achieving accurate results in our setting requires LMs derived from a large corpus of phrases that are relational, grammatical, and frequent (so that they are likely to match evidence gathered at prediction time). Thus, we use the English subset of ClueWeb09 and the 5 billion annotations provided by Google's FACC1 corpus,[1] indicating which text spans contain mentions of entities known to Freebase, identified through their m-ids.

Replacing actual mentions to named entities in the text by their corresponding m-ids, we arrive at content such as the following:

`/m/06mr6` famously starred as `/m/06k5xq` besides `/m/0clpml`

[1] http://lemurproject.org/clueweb09

Since in Freebase the entities /m/06mr6 (actor Sir Sean Connery) and /m/06k5xq (fictional character Robin Hood) are related through relation film/actor/.../character, we add the phrase "famously starred as" to the language model of that relation (and also to the models of all other relations between these entities).

In a nutshell, building LMs boils down to: for every pair of m-ids that belong to a relation, extracting all phrases connecting those m-ids from the corpus, filtering uninformative phrases and aggregating the counts accordingly. Next, we explain the filtering steps we perform to increase the quality of our language models.

3.1 Filtering Phrases

Our goal is to predict relations between *pairs of entities* such as family and romantic relationships between people, employment relationships between people and organizations, and business relationships among organizations. In order to keep our LMs highly focused we discard generic and uninformative phrases with the help of lightweight natural language processing tools.

Filtering Uninformative Phrases. Not all phrases connecting entities are useful for relation prediction. For example, in the sentence above, the phrase "famously starred as" describes an actual relation between the surrounding entities while the phrase "besides" between /m/06k5xq (Robin Hood) and /m/0clpml (James Bond) does not. To filter out such noise, we parse the sentences containing pairs of m-ids and check if the phrases connecting the entities conform to known grammatical patterns that describe binary relations [7], discarding those that do not. This step eliminates the vast majority of the phrases but ensures our language models are grammatical and predictive.

Placeholder Generalization. We often can (and should) generalize the phrases that reveal the same relation but differ in some detail. For example, phrases "starred in the 3rd movie of" and "starred in the first movie of" express the same relation, and are generalized into "starred in the ORD movie of", where "ORD" stands for any ordinal number. We apply similar generalizations to instances of other common generic types such dates, distances and numbers. Fig. 5 summarizes the placeholders used with the relative frequency (i.e. number of phrases).

Further Filtering. We are not interested in ontological relations that describe class membership of entities, such as that Sir Sean Connery is an actor and that James Bond is a fictional character. Other prominent ontological relations concern the ethnicity of people, the business segment of organizations, etc. Thus, we discard phrases that are variants of the "is a" pattern, often used in these cases (e.g., "is a British actor" or "was an American activist").

3.2 Phrase Statistics

We were able to find 19M distinct pairs of m-ids that are connected by a (filtered) relational phrase in the ClueWeb09 corpus. Only 1.4M of these pairs (8%) belong to one of the approximately 5K Freebase relations. In total, these 1.4M pairs are related through 2.36M distinct phrases in the corpus. Although we found some fairly long phrases, the majority of them are relatively short (4.3 tokens per phrase on average). As expected, we observed that the distribution of phrases by frequency in the corpus follows a power-law.

Global Statistics	
annotated pairs of m-ids (related in KG)	19 M (1.4 M)
annotated relations ($\geqslant$ 1 phrase)	2739
distinct *relational* phrases	2.36 M
Placeholders	
AGE (3.2K), DATE (1K), LENGTH (3.4K)	
MONEY (5.5K), ORD (15K), SCORE (3.3K)	
TIME (2K), UNIT (1K), WEIGHT (100), YEAR (845)	
Filtered Models Statistics	
relations ($\geqslant$ 200 phrases)	500
LMs derived	1934
filtered phrases	27.7 K

Figure 5: Statistics after the filtering process.

3.3 Specializing LMs Based on Type

The Freebase ontology specifies the *expected* types of the entities that can participate in any given relation. For example, relation /film/film/subject, that describes the subject of a movie, has domain /film/film and range /film/film_subject. Although somewhat informative, these types are fairly generic. For example, the subjects of biographical movies are people (and thus instances of /people/person), while the subjects of documentaries can be organizations or locations. Note however, that the LMs for these different kinds of movies are likely to be very different: the relational phrase "is the biography of" is appropriate for movies whose subject are people, while "portrays the founding of" is suitable for movies about organizations. In order to account for such nuances, we partition the phrases associated with each relation based on generic entity types that can be inferred automatically by typical NER systems[2], and are available in Freebase as /people/person, /organization/organization, and /location/location. A catch-all "misc" type is used for all the other entities. This results in each FB relation having up to 16 different LMs, one for each possible combination of types.

3.4 Model Statistics

In the end, of the 4819 relations in Freebase, we are able to build models for 2739. For the purposes of the evaluations reported here, we experimented only with those relations for which we could find at least 200 distinct phrases. This corresponds to 500 relations and 1934 different models (on average 3.78 LMs per relation, based on the combination of NER types). Fig. 5 summarizes statistics about the number of language models we derived from ClueWeb09, and those we use in our experimentation.

4 RELATION RANKING

This section gives details the steps involved in relation ranking.

[2]Person (PER), Location (LOC), Organization (ORG), and Miscellaneous (MISC) as defined by Stanford NER [11].

4.1 Gathering Evidence from the Web

Given an entity pair $\langle s_1, o_1 \rangle$, we perform a Web search looking for sentences mentioning both entities in the given order. For the purposes of this paper, we collected sentences from *snippets* returned by Google, saving time and bandwidth. The actual scoring of relations is done on relational phrases, extracted from the snippets, that match those used to build the language models. That is, we process the snippets in the same way described in Sec. 3.1. If multiple relational phrases are found in the same snippet our system uses all of them. Also, we attempt to minimize the effects of geo-localization and personalization in Google searches by periodically obtaining a new IP address via a Private Virtual Network service. Of course, our system is not restricted to Google. In fact, a local index of a large Web crawl (e.g., ClueWeb or the Web commons crawl) could be used instead of Google for this step.

We compare two strategies for matching phrases: exact matching (equality) and shallow approximate matching, which works as follows. Given a span of text we find candidate phrases using n-grams at character level (3-grams) and then we score them using a Fuzzy Jaccard Similarity [30] that takes into account fuzzy matchings between the individual words[3]. The approximate match comes with a clear precision and recall trade-off: it introduces noise but results in more phrases being matched. For example, the text "filmed entirely in" in a Web search snippet could match phrases "was filmed in" and "were filmed by", belonging to different LMs. We investigate this trade-off and confirm in the experimental evaluation that approximate matching generally improves the quality of the rankings.

4.2 Ranking for Individual Entity Pairs

As mentioned in Sec. 1.2, we take an IR approach to rank KG relations based on their relevance for an entity pair. To recap the notation and avoid confusion, each "document" D corresponds to a KG relation and a "query" Q corresponds to relational phrases connecting entities obtained through a Web search. Given an entity pair $\langle s_i, o_i \rangle$, we denote by $a(\langle s_i, o_i \rangle)$ the ranking of all documents for the query resulting from that pair, computed according to a $score(\cdot, \cdot)$ function (explained below).

Query Likelihood Scoring. The *query likelihood* retrieval model assumes that the query terms are samples drawn from a LM derived from a document. Formally, given query Q and document D, from which a model θ_D is derived, we rank the documents by decreasing score, defined as:

$$score(Q, D) = P(Q|\theta_D).$$

Many approaches have been used for estimating $P(Q|\theta_D)$. We start with the robust multinomial language model, which assumes that terms are generated independently, and avoid overfitting with interpolation (i.e., Jelinek-Mercer smoothing) [36]. More precisely, let C be a document containing all the phrases and S the set of phrases. Then:

$$score(Q, D) = \prod_{p \in S} P(p|\theta_D)^{c(p,Q)} \tag{1}$$

where:

$$P(p|\theta_D) = \lambda\, P(p|D) + (1 - \lambda)\, P(p|C) \tag{2}$$

$$P(p|D) = \frac{c(p, D)}{|D|} \text{ and } P(p|C) = \frac{c(p, C)}{|C|} \tag{3}$$

Above, $c(p, \cdot)$ denotes the frequency of phrase p in the query Q, document D or corpus C. We call this ranking approach PLM, for *Phrase Language Model* in the experimental evaluation.

We set $\lambda = 0.9$ experimentally.

Disjunctive Gate Scoring. The query likelihood approach uses a *conjunctive gate* to combine evidence from multiple phrases while predicting the likelihood of a relation: a model that cannot generate most phrases in the query is unlikely to rank high. In our setting, this is often an overkill. For example, one can be reasonably certain that the relation `/film/film/featured_film_locations` holds for a pair of entities connected by the phrase "was filmed in". An implicit assumption here is that the frequency of the phrases used to build the LMs and the queries is a good proxy for how reliable they are. While this seems reasonable at Web scale, one could take the trustworthiness of the sources [8] into account, e.g., via re-ranking or by a priori filtering.

To allow for more permissive predictions we calculate the score of each relation interpolating its prior and its posterior probability conditioned by every single phrase in the query. We aggregate the posterior of each phrase using a "noisy-OR" gate [21]:

$$score(Q, D) = \beta\, P(D|p_1, ..., p_Q) + (1 - \beta)\, P(D) \tag{4}$$

where:

$$P(D|p_1, ..., p_Q) = 1 - \prod_{p \in Q} (1 - P(D|p)) \tag{5}$$

$$P(D|p) = \frac{c(p, D)}{\sum_{l \in L} c(p, D_l)} \tag{6}$$

A "noisy-OR" gate combines evidence differently from the standard PLM: a relation scores high if the query contains *any* of the high frequency phrases associated with that relation. We call this ranking approach Noisy OR in the experimental evaluation.

As λ for the standard model, β is a coefficient to control the amount of smoothing. We set it experimentally to $\beta = 0.8$.

4.3 Ranking for Multiple Pairs

We now move on to the general form of the problem which applies to a set of entity pairs $\langle s_1, o_1 \rangle, \ldots, \langle s_n, o_n \rangle$, e.g., coming from different rows of the same table, and in which we need to rank KG relations in some aggregate form. We consider two ways of producing an aggregate ranking. The first is a *global* approach where we merge the results of the Web searches for each individual pair into a single query, used to rank all KG relations, while the second is a *local* aggregation approach, where we score KG relations by merging the individual rankings obtained for each pair separately.

Global Aggregation. In this aggregation approach we build a query Q' containing all phrases in each Q_i derived from entity pair $\langle s_i, o_i \rangle$, with the appropriate phrase counts, and obtain a single ranking using Eq. 1 or Eq. 4 to produce the final answer, depending on the scoring model used.

Local Aggregation. In this approach we first rank all relations for each of the entity pairs, and combine these rankings to score the relations. Let $a_i = a(\langle s_i, o_i \rangle)$ be the ranking obtained for entity

[3] We use Jaro-Winkler similarity with a threshold of 0.9.

pair $\langle s_i, o_i \rangle$, computed according to Eq. 1 or Eq. 4 depending on the scoring model:

$$a_1 = a(\langle s_1, o_1 \rangle) = r_1^{a_1}, r_2^{a_1}, \ldots, r_k^{a_1}$$
$$\vdots$$
$$a_n = a(\langle s_n, o_n \rangle) = r_1^{a_n}, r_2^{a_n}, \ldots, r_j^{a_n}$$

The *local* aggregated score of a relation is its mean (inverse) rank across all individual rankings:

$$agg_{score}(r) = \frac{1}{n} \sum_{a_i} \frac{1}{rank(r, a_i)} \tag{7}$$

In this way, a relation that ranks high for a large number of pairs will have a higher score and rank high for the set of entity pairs.

5 EXPERIMENTS

We now report on an experimental evaluation of the LM-based relation ranking approach to show it does not suffer from the KG incompleteness problem and, thus, can be a viable alternative to Entity Linking (EL) approaches. To do that, we experiment first on two corpora of facts involving pairs of in-KG entities, comprising 9 relations from the person domain (Tab. 1 shows the relations). The first corpus, called LectorFacts, consists of facts *known to be missing* from DBpedia and Freebase. The second corpus, called KGFacts, comprises the same relations as LectorFacts, but with facts that are present in both DBpedia and Freebase. With these two corpora, we can simulate the scenarios where EL methods should work (KGFacts) and the scenario where they would not (LectorFacts). We thoroughly evaluate our system both with individual pairs of entities (Sec. 5.2) and also with sets of pairs (Sec. 5.3) as input, under a variety of scenarios. Then, to further illustrate the KG incompleteness issue, we evaluate our approach on pairs of columns from Wikipedia tables mixing in-KG and out-of-KG entities but for which a state-of-the-art EL method, T2K Match [24–26], fails to identify the correct relations (Sec. 5.7).

For the experiments reported here, we trained our method was to predict 500 different Freebase relations (recall Sec. 3.4) and used Google for the Web search step.

Statistics about LectorFacts and KGFacts. To the best of our knowledge, no previous benchmark for Web table understanding considers the case where KG incompleteness prevents an EL approach to predict a good relation even when all entities are in the KG. Therefore, we rely on the only corpus of facts *known to be missing* from mainstream and publicly available KGs [6][4], and call it LectorFacts here. We restrict our evaluation to the 9 non-ontological relations in the original corpus[5], and use 50 facts (i.e., entity-pairs) from each relation. For the sake of comparison, we created a similar benchmark, KGFacts, by randomly picking 50 entity pairs from each of the relations in LectorFacts, among those pairs that appear in both DBpedia and Freebase. In a sense, these two benchmarks complement each other: facts from KGFacts concern prominent pairs of entities and generate more hits on a Web search. As shown in Tab. 1, on average, we are able to obtain twice as many hits (the "sent." column in the table) on that corpus compared to LectorFacts. The two columns under "phrase" in the table show the (average) number of relational phrases (i.e., the ones used to build the LMs) that *match* the sentences returned by the Web search. These phrases are used (as queries) to predict the relations in our approach. We experiment with exact and approximate matching of sentences and relational phrases. For clarity, all results reported use exact matching, except those in Sec. 5.4.

Relation	LectorFacts			KGFacts		
	sent.	phrase		sent.	phrase	
		ex.	ap.		ex.	ap.
people/person/parents	39.5	2.8	4.9	93.5	8.6	16.6
people/person/education	22.3	1.3	2.2	113.3	8.6	16.8
sports/pro_athlete/teams	61.4	3.5	6.3	131.4	17.3	29.9
people/person/place_of_death	55.1	1.5	3.3	126.3	10.9	21.5
government/politician/party	37.7	1.6	2.5	102.8	11.0	21.9
people/person/place_of_birth	54.2	2.5	4.4	119.9	9.7	18.5
award/award_winner/awards_won	58.1	2.7	6.1	94.2	8.2	15.0
people/person/spouse	49.0	2.7	5.5	84.2	8.5	15.4
people/person/children	43.2	2.0	4.1	82.0	7.1	14.2
Mean	46.7	2.3	4.4	105.3	10.0	18.9

Table 1: Mean number of sentences retrieved from the Web search and corresponding number of matching phrases obtained with the exact and the approximate method. Each relation consists of 50 entity pairs.

5.1 Metrics

Given a set $I = \{\langle s_1, o_1 \rangle, \ldots, \langle s_n, o_n \rangle\}$ of subject-object entity pairs we have only one correct answer (the pairs are labeled with a single relation). In order to evaluate the ranking produced by the system we use the reciprocal rank [16] that corresponds to the multiplicative inverse of the rank of the correct relation. More precisely, let $a(I) = r_1, \ldots, r_k$ be the response of a prediction for the input pairs I in which relations are given in decreasing order of relevance, and let $truth(I)$ be the ground truth relation for I, the $reciprocal_{rank}(a, I)$ is:

$$reciprocal_{rank}(a, I) = \sum_{i=1}^{k} \frac{\mathbb{1}(truth(I) = a[i])}{i} \tag{8}$$

where $\mathbb{1}(\cdot)$ is the indicator function (returns 1 if the condition appearing as its argument holds, 0 otherwise) and $a[i]$ is the relation in position i in the ranking. Note that the metric implicitly takes recall into account. Indeed if the system does not predict any ranking or the correct relation is not present the reciprocal rank is 0. Finally, we use the Mean Reciprocal Ranking (MRR) [16] to evaluate the results of multiple input sets $\mathcal{I} = I_1, \ldots, I_n$:

$$MRR(\mathcal{I}) = \frac{1}{|\mathcal{I}|} \sum_{I_j \in \mathcal{I}} reciprocal_{rank}(a, I_j) \tag{9}$$

5.2 MRR on Individual Entity Pairs

Fig. 6 shows the $MRR(\mathcal{I})$, per relation, obtained on predicting the relation for each of the 50 pairs individually. As expected, relation prediction on KGFacts is easier than on LectorFacts regardless

[4] Available at: http://downloads.dbpedia.org/2016-04/ext/lector_facts/

[5] `people/person/nationality`, `people/person/religion` and `people/person/ethnicity` are ontological relations and thus ignored.

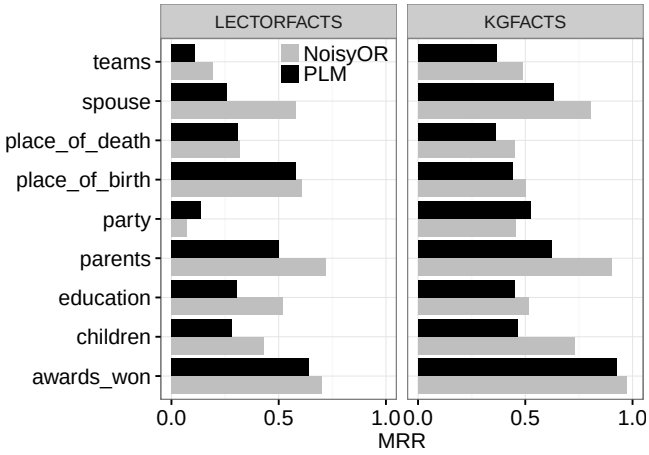
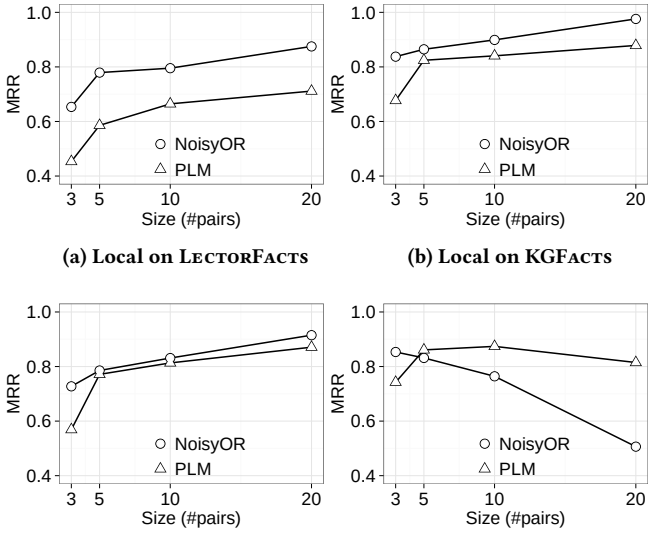


Figure 6: MRR on individual entity pairs.



(a) Local on LectorFacts **(b) Local on KGFacts**

(c) Global on LectorFacts **(d) Global on KGFacts**

Figure 7: MRR on sets of entity pairs.

of the ranking model. Quantitatively speaking, Noisy OR was 20% more effective than PLM on the KGFacts corpus (MRR of 0.64 and 0.53, respectively), and 31% more effective on LectorFacts (MRR of 0.46 and 0.35, respectively). This can be explained by the larger number of relational phrases that can be obtained with pairs of entities in KGFacts (recall Tab. 1). Looking at the MRR results across relations, one can see that some relations are harder to predict than others. This is explained by the ambiguity of the phrases in some language models. For example, "is the daughter of" is almost exclusively used in `people/person/parents`. Similarly, descriptive phrases like "won the" and "was awarded the" are strongly associated with `award/award_winner/awards_won`. Other relations are expressed through generic phrases that can only be interpreted in context, such as "leader of the", "led the" or "left the" which appear in the models of `.../person/employment`, `.../politician/party` or `.../pro_athlete/teams`.

It should be noted that the difficulties of dealing with ambiguity are more pronounced because the system evaluated here is configured to predict 500 different relations from all Freebase domains. Much better results are to be expected if one learns models for domain-specific relations.

5.3 MRR on Multiple Pairs

Fig. 7 shows the MRR for each combination of: corpus (LectorFacts and KGFacts), scoring model (Noisy OR and PLM), and method for combining evidence (local aggregation and global query), when considering multiple entity pairs. We vary the size of the input set from 3 to 20 pairs, and each plot is the average of 10 different samples. We cap the size to 20 pairs as previous work has reported that this is the average number of rows found in real Web Tables [26]. Several general observations are possible from the figure: (1) using multiple pairs for relation ranking has a significant positive impact on MRR; (2) aggregating pairwise rankings is more robust, regardless of scoring model; and (3) Noisy OR generally outperforms the standard language model PLM. Next, we discuss these findings in more detail.

More Entity Pairs is Better. Unsurprisingly, the more entity pairs we use, the higher the MRR. To quantify this, the highest MRR achieved with individual pairs was 0.64, obtained with the Noisy OR on KGFacts. Testing the same model on the same corpus with local aggregation, we achieve MRR of 0.84 using 3 pairs and 0.98 using 20 pairs, corresponding to improvements of 31% and 53%, respectively. Looking at LectorFacts with Noisy OR and local aggregation, even more pronounced gains are evident: 43% with 3 pairs and 89% with 20 pairs. The highest MRR are obtained with 20 entity pairs and the Noisy OR model: 0.98 on KGFacts with local aggregation, and 0.91 on LectorFacts with global queries. We conjecture on this discrepancy below. These results underscore the high potential of our method for Web table understanding, where predictions are made on multiple entity pairs, and not just one.

To see how multiple pairs help relation scoring, suppose the input is a table with soccer players and the teams they played for. If we are given just the pair ⟨`"Luis Enrique","FC Barcelona"`⟩, we will not be able to discern whether `.../pro_athlete/teams` or `soccer/.../manager` should rank higher as both relations apply to that pair. However, if the input also contains another pair like ⟨`"Neymar Jr.","Paris St Germain"`⟩, we will be much more likely to predict `.../pro_athlete/teams`. In other words, the ambiguity of the input reduces with more pairs, as expected.

Local Aggregation is More Robust. When it comes to the local aggregation of multiple pairwise predictions versus a single prediction using a global query formed by grouping all sentences from the Web searches, there seems to be a dependence on the actual number of phrases that are used: with a small number of phrases (e.g., as in LectorFacts), it is better to use the global approach while with many phrases it is much better to use the local aggregation. Using the AUC of the plots in Fig. 7 as a proxy, we can quantify this argument: for LectorFacts, the global approach is superior by 4% (Fig. 7(a) and (c)) while for KGFacts the local aggregation is better by 7% (Fig. 7(b) and (d)). As a matter of fact, the global aggregation method gets worse on KGFacts as more

entity pairs are used. This happens because with more pairs there are more (and more diverse) sentences, which leads to the global query to lose focus. For example, looking at some entity pairs in relation `people/person/education` we can find phrases such as "who became president of", "is the co founder of" or "played football at" that are associated (sometimes very strongly) with many relations. The problem is more evident with the Noisy OR model as it does not take phrase frequency into account and thus does not have any way of weighing importance of the phrases in the query. To verify this hypothesis, we pruned all but the 5 most frequent phrases in the queries used in Fig. 7(d), and scored these pruned queries with Noisy OR, resulting in a 5% improvement.

Noisy OR is Better. A comparison of the AUC of the plots in Fig. 7 reveals that scoring with Noisy OR is generally superior to that with PLM. In quantitative terms, the relative improvements can be as high as 24% (see Fig. 7(a)) and considering all the configurations Noisy OR obtains a relative gains of 20% over PLM. In conclusion, a Noisy OR scoring model is very robust if applied with a local aggregation approach independently from the popularity of the entities involved.

5.4 Matching Phrases Approximately

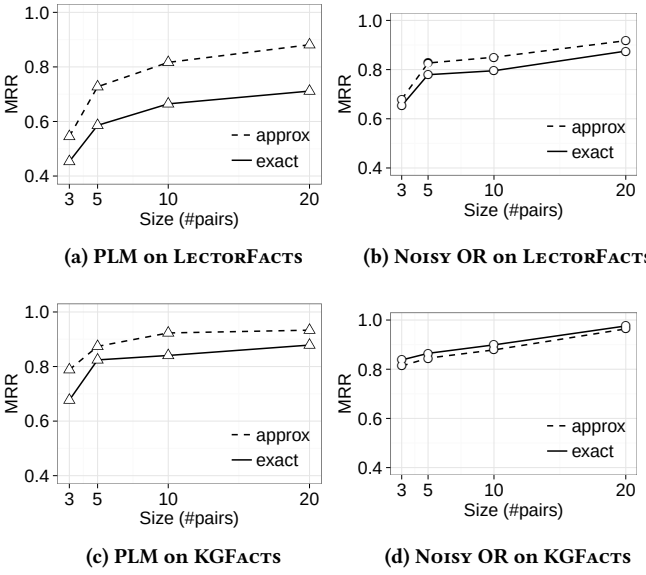
All results shown so far were obtained by *exactly* matching sentences from the Web search against relational phrases in the LMs (to form the queries used for relation ranking). Because exact matching leads to low hit counts (sometimes, even empty queries), we experimented with alternative ways of approximately matching sentences to relational phrases, settling for the scheme described in Sec. 4.1. Fig. 8 shows that approximate matching (dashed line) generally improves on exact matching (solid line). For clarity, we show the results with local aggregation only. In quantitative terms, comparing the AUC of the plots, we observe the biggest improvement (19%) for the PLM scoring. This improvement can be attributed to the fact that approximate matching effectively doubles the number of hits as shown in Tab. 1 (see the *phrase* columns).

It is worth noting that approximate matching closes the gap in MRR between KGFacts and LectorFacts considerably. Comparing the AUC between plots, the best setting for KGFacts (Noisy OR with local aggregation and exact phrase matching) is only 6% superior to the best setting for LectorFacts (Noisy OR with local aggregation and approximate phrase matching).

5.5 Pruning LMs

While our concern in this work is effectiveness, we report on a brief experiment on the natural trade-off between the LM sizes, inference time and accuracy. Tab. 2 shows the MRR and the inference time of the two models on samples of 20 pairs from LectorFacts for different values of the minimum *support* (F in the table) required for a phrase to be used in any language model[6]. Unsurprisingly, increasing the minimum support leads to fewer and smaller models, and, consequently: a drop in MRR, and a drop in inference time. For the Noisy OR scoring model, however, the trade-off is quite favorable: with F =100 there is no drop in MRR while the inference time is cut in half. This is easily explained by the power-law distribution

[6]For clarity, all other results reported here are on LMs with F =1.



(a) PLM on LectorFacts (b) Noisy OR on LectorFacts

(c) PLM on KGFacts (d) Noisy OR on KGFacts

Figure 8: Approximate (dashed lines) versus exact (solid lines) matching on multiple entity pairs with local aggregation.

F	PLM		Noisy OR	
	MRR	time (ms)	MRR	time (ms)
1	0.71	230.0 ±11.1	0.88	78.5 ±11.0
10	0.70	180.5 ± 3.5	0.87	78.1 ± 7.1
100	0.55	178.1 ± 4.6	0.88	44.1 ± 2.0
1000	0.14	167.4 ± 3.5	0.85	39.1 ± 1.3

Table 2: MRR and inference time versus minimum phrase *support* (F).

of phrases in the ClueWeb09 corpus (Sec. 3.2) and the fact Noisy OR relies on phrases that are *strongly* associated with a relation, which inevitably ignores phrases with low support.

5.6 Towards Practical Tools

The observations in the experiments above indicate that the LM-based relation ranking can perform well enough to be useful in practice. Moreover, they indicate that a configuration of Noisy OR with local aggregation and approximate sentence matching to be very robust. One possible tuning would be to use the global query approach if the number of sentences returned by the Web search is below an application specific threshold.

5.7 Towards Web Table Understanding

We now report on a preliminary experiment to show that relation prediction with LMs can be a viable alternative to Entity Linking (EL) for Web Table Understanding. While EL methods fail on tables with entity pairs from LectorFacts, even though they concern

EPG	EPL	ENG	ENL	APG	APL	ANG	ANL
35	38	43	49	32	41	28	52

Table 3: Relations correctly predicted (out of 80) on Web Table Understanding for different combinations of phrase matching (A-approximate, E-exact), model (N-Noisy OR, P-PLM) and aggregation approach (L-local, G-global).

in-KG entities, the issue is not artificially introduced by that benchmark. To show this, we collected actual Wikipedia tables containing in-KG and out-of-KG entities and facts from different domains and for which even the state-of-the-art EL method T2K Match [24–26] fails. Since an independent benchmark for this task is lacking, we manually extracted 80 pairs of columns from 65 different Wikipedia tables, for which T2K Match was unable to assign a relation. These pairs cover a range of topics and relations beyond LectorFacts, including geography, sports, actors and directors of movies and TV shows, among others.

The average number of entity pairs in this test was 31. The average number of sentences retrieved by the Web search is 39. We tested our method with these 80 pairs and manually evaluated the results to determine if the predicted relations were appropriate. [7]

We evaluate the results using precision@1 (1 if the first relation in the output ranking is correct, 0 otherwise) of each pair. Tab. 3 shows the number of entity pairs for which the top ranked relation was judged appropriate, for different configurations. The best results (52 relations out of 80) are obtained with approximate matching (A), using the Noisy OR score model (N) and locally aggregating the multiple pairs (L), which is also the best configuration on LectorFacts. Developing a proper benchmark for this task is non-trivial and important future work for this area.

6 CONCLUSION

This paper studied the problem of predicting relations from a Knowledge Graph that hold over relational data found on the Web, such as tables, structured lists, CSV/TSV files, among others. The methods described here fill a gap left by text-based Information Extraction tools and overcome the main obstacle of table understanding methods relying on linking entities to objects in the graph.

Our approach borrows from Information Retrieval and Information Extraction work and uses generative language models to represent the relations in a KG with relational phrases, extracted from a Web-scale corpus of independently annotated text using the state-of-the-art in Open Information Extraction to filter out noise. Thus, relation prediction is done by ranking the respective models based on how likely they are to generate the phrases on the Web that connect the entities given as input. The paper evaluates different approaches for model ranking and for aggregating evidence to predict relations for a set of entity pairs. Further evaluations with other standard ranking approaches are left for future work.

[7] All data used for this experiment, including the top-3 relations produced by each of our methods, can be found at https://ln.sync.com/dl/b7e5a9f40#rfyp47tn-irz5dvhn-bwmn58kr-cetrhxmj

Finally, the paper reports on an experimental evaluation designed to stress the issues caused by KG incompleteness, a limiting factor of the most related previous work. The encouraging results indicate the method overcome the KG incompleteness issue and that its accuracy is high enough to suggest it can be successfully applied on tasks such as KG construction and augmentation.

ACKNOWLEDGMENTS

This work was supported in part by grants from the Natural Sciences and Engineering Research Council of Canada. D. Barbosa was a visiting researcher at the Max Planck Institute for Informatics, Saarbrucken, Germany during the development of parts of this work. The authors thank Y. A. Sekhavat, who contributed to our preliminary work.

REFERENCES

[1] Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting Relations from Large Plain-text Collections. In *Proceedings of the Fifth ACM Conference on Digital Libraries (DL '00)*. ACM, 85–94.

[2] Krisztian Balog, Marc Bron, and Maarten De Rijke. 2011. Query Modeling for Entity Search Based on Terms, Categories, and Examples. *ACM Trans. Inf. Syst.* 29, 4 (2011), 22:1–22:31.

[3] Danushka Tarupathi Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. 2010. Relational Duality: Unsupervised Extraction of Semantic Relations Between Entities on the Web. In *Proceedings of the 19th International Conference on World Wide Web (WWW 2010)*. ACM, 151–160.

[4] Marc Bron, Krisztian Balog, and Maarten de Rijke. 2013. Example Based Entity Search in the Web of Data. In *Proceedings of the 35th European Conference on Advances in Information Retrieval (ECIR'13)*. Springer-Verlag, 392–403.

[5] Michael J. Cafarella, Alon Y. Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. WebTables: exploring the power of tables on the web. *PVLDB* 1, 1 (2008), 538–549.

[6] Matteo Cannaviccio, Denilson Barbosa, and Paolo Merialdo. 2016. Accurate Fact Harvesting from Natural Language Text in Wikipedia with Lector. In *Proceedings of the 19th International Workshop on Web and Databases (WebDB '16)*. ACM, 9:1–9:6.

[7] Filipe de Sá Mesquita, Jordan Schmidek, and Denilson Barbosa. 2013. Effectiveness and Efficiency of Open Relation Extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 447–457.

[8] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge Vault: A Web-scale Approach to Probabilistic Knowledge Fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 601–610.

[9] Doug Downey, Stefan Schoenmackers, and Oren Etzioni. 2007. Sparse Information Extraction: Unsupervised Language Models to the Rescue. In *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.

[10] Shady Elbassuoni, Maya Ramanath, Ralf Schenkel, Marcin Sydow, and Gerhard Weikum. 2009. Language-model-based Ranking for Queries on RDF-graphs. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM '09)*. ACM, 977–986.

[11] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL'05)*. Association for Computational Linguistics, 363–370.

[12] Yusra Ibrahim, Mirek Riedewald, and Gerhard Weikum. 2016. Making Sense of Entities and Quantities in Web Tables. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM '16)*. ACM, 1703–1712.

[13] Mandar Joshi, Uma Sawant, and Soumen Chakrabarti. 2014. Knowledge Graph and Corpus Driven Segmentation and Answer Inference for Telegraphic Entity-seeking Queries. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*. 1104–1114.

[14] John Lafferty and Chengxiang Zhai. 2017. Document Language Models, Query Models, and Risk Minimization for Information Retrieval. *SIGIR Forum* 51, 2, 251–259.

[15] Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. 2010. Annotating and Searching Web Tables Using Entities, Types and Relationships. *PVLDB* 3, 1 (2010), 1338–1347.

[16] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval*. Cambridge University Press.

[17] Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open Language Learning for Information Extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. 523–534.

[18] Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant Supervision for Relation Extraction with an Incomplete Knowledge Base. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*. 777–782.

[19] Emir Muñoz, Aidan Hogan, and Alessandra Mileo. 2014. Using linked data to mine RDF from wikipedia's tables. In *Proceedings of the 7th ACM international conference on Web search and data mining*. ACM, 533–542.

[20] Ndapandula Nakashole, Gerhard Weikum, and Fabian M. Suchanek. 2012. PATTY: A Taxonomy of Relational Patterns with Semantic Types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. 1135–1145.

[21] Judea Pearl. 1989. *Probabilistic reasoning in intelligent systems - networks of plausible inference*. Morgan Kaufmann.

[22] Jay M. Ponte and W. Bruce Croft. 1998. A Language Modeling Approach to Information Retrieval. In *SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 275–281.

[23] Jeffrey Pound, Ihab F. Ilyas, and Grant E. Weddell. 2010. QUICK: Expressive and Flexible Search over Knowledge Bases and Text Collections. *PVLDB* 3, 2 (2010), 1573–1576.

[24] Dominique Ritze and Christian Bizer. 2017. Matching Web Tables To DBpedia - A Feature Utility Study. In *Proceedings of the 20th International Conference on Extending Database Technology, EDBT 2017, Venice, Italy, March 21-24, 2017*. 210–221.

[25] Dominique Ritze, Oliver Lehmberg, and Christian Bizer. 2015. Matching HTML Tables to DBpedia. In *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics (WIMS '15)*. ACM, Article 10, 10:1–10:6 pages.

[26] Dominique Ritze, Oliver Lehmberg, Yaser Oulabi, and Christian Bizer. 2016. Profiling the Potential of Web Tables for Augmenting Cross-domain Knowledge Bases. In *Proceedings of the 25th International Conference on World Wide Web*. 251–261.

[27] Yoones A. Sekhavat, Francesco Di Paolo, Denilson Barbosa, and Paolo Merialdo. 2014. Knowledge Base Augmentation using Tabular Data. In *Proceedings of the Workshop on Linked Data on the Web co-located with the 23rd International World Wide Web Conference (WWW 2014), Seoul, Korea, April 8, 2014.*

[28] Petros Venetis, Alon Y. Halevy, Jayant Madhavan, Marius Pasca, Warren Shen, Fei Wu, Gengxin Miao, and Chung Wu. 2011. Recovering Semantics of Tables on the Web. *PVLDB* 4, 9 (2011), 528–538.

[29] Chi Wang, Kaushik Chakrabarti, Tao Cheng, and Surajit Chaudhuri. 2012. Targeted Disambiguation of Ad-hoc, Homogeneous Sets of Named Entities. In *Proceedings of the 21st International Conference on World Wide Web (WWW '12)*. ACM, 719–728.

[30] Jiannan Wang, Guoliang Li, and Jianhua Feng. 2014. Extending String Similarity Join to Tolerant Fuzzy Token Matching. *ACM Trans. Database Syst.* 39, 1, Article 7 (Jan. 2014), 7:1–7:45 pages.

[31] Yue Wang and Yeye He. 2017. Synthesizing mapping relationships using table corpus. In *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, 1117–1132.

[32] Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge Base Completion via Search-based Question Answering. In *Proceedings of the 23rd International Conference on World Wide Web (WWW '14)*. ACM, 515–526.

[33] Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting Language and Knowledge Bases with Embedding Models for Relation Extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 1366–1371.

[34] Mohamed Yahya, Denilson Barbosa, Klaus Berberich, Qiuyue Wang, and Gerhard Weikum. 2016. Relationship Queries on Extended Knowledge Graphs. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining (WSDM '16)*. ACM, 605–614.

[35] Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, Maya Ramanath, Volker Tresp, and Gerhard Weikum. 2012. Natural Language Questions for the Web of Data. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL '12)*. Association for Computational Linguistics, 379–390.

[36] ChengXiang Zhai. 2008. *Statistical Language Models for Information Retrieval*. Morgan & Claypool Publishers.

[37] Ziqi Zhang. 2014. Towards Efficient and Effective Semantic Table Interpretation. In *Proceedings of the 13th International Semantic Web Conference - Part I (ISWC '14)*. Springer-Verlag New York, Inc., 487–502.