Using Patterns in Knowledge Graphs for Targeted Information Extraction

Mengfei Zhou and Vivi Nastase
Institute for Computational Linguistics
University of Heidelberg
Heidelberg, Germany
{zhou, nastase}@cl.uni-heidelberg.de

ABSTRACT

Knowledge graphs are useful structures for capturing information in machine readable format, that can be used for higher level tasks such as question answering. Knowledge graphs are incomplete. We present our analysis of a method for targeted information extraction for completing the knowledge graphs, that relies on patterns in the knowledge graphs to form more specific queries for completing $< source, relation, ? >$ triples. The results show that the baseline method – using a query consisting of "*source relation*" – performs best on a minority (13 – 39%) of the investigated relations, indicating that there is much to gain by using patterns from the knowledge graphs for query expansion.

CCS CONCEPTS

• **Information systems → Query reformulation**; *Web searching and information discovery*; *Data extraction and integration*;

KEYWORDS

targeted information extraction, query expansion, PRA, knowledge graph completion

ACM Reference Format:
Mengfei Zhou and Vivi Nastase. 2017. Using Patterns in Knowledge Graphs for Targeted Information Extraction. In *Proceedings of First Workshop on Knowledge Base Construction, Reasoning and Mining (KBCOM 2018).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Knowlegde graphs (KGs) are useful data structures that capture information in a relational model – most commonly in the form of $< source, relation, target >$ triples – that makes them easily accessible and usable for a variety of tasks such as question answering [10], evaluating trustworthiness of web content [5] and web search [4].

Knowledge of various types – lexical, common sense, world – has been the target for modeling in the graph format, but however much effort has been put into this, there is always more to add. Knowledge graphs are incomplete. A variety of approaches have been designed to help enrich and complete knowledge graphs. West et al. [17] distinguish two types of graph completions approaches: (i) "push" where facts extracted from a text collection and organized into or added to a knowledge graph; and (ii) "pull" where specific information is "pulled" from textual sources and added to the graph. In addition to these, link prediction approaches add new edges to a graph based on intrinsic regularities, such as node homophily, which can be captured by induced embeddings of the graph in continuous vector spaces [14]. The work presented here fits into the "pull" model, i.e. targeted information extraction.

KBCOM 2018, Feb 9th, Los Angeles, California, USA
2017. ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

West et al. [17] use query logs to extract information that completes specific relation triples – e.g. finding the *nationality* or the *place-of-birth* or *parent* of a given named entity. West et al. [17]'s hypothesis is that by making a query more specific by adding information that is connected to the desired answer, a search engine will be more likely to return a text snippet that contains the answer. For example, finding the *mother* of *Frank Zappa* is more difficult when forming a query based only on the relation name and the source entity – i.e. "Frank Zappa mother" – because Frank Zappa was part of a band called *The Mothers of Invention.* Adding information such as the place of birth makes the query more specific, and the answers returned by the search engine better. To find associated information for a desired query, West et al. [17] analyze Google's Web-search query logs.

In this work we focus on the same targeted information extraction task. To expand queries for completing a triple $< source, relation, ? >$ we use patterns discovered in knowledge graphs, in the form of relation paths associated with $relation$. The queries we form are used with the Google search engine. We use the paths produced by Das et al. [3], built with Gardner and Mitchell's[6] variation of the Path Ranking Algorithm (PRA) [8, 9]. We experiment with using different parts of the path information to generate different types of queries and understand the impact of this added knowledge for targeted information extraction. The results show that with respect to the evaluation criteria (precision, recall, comparable precision) the average results are higher overall for the baseline set-up – the query consists of "*source relation*" –, but this setting performs best on a minority of the studied relations (13% – 39%). This indicates that there is much to gain by using patterns from the knowledge graphs.

2 RELATED WORK

The main inspiration for this work is [17]. West et al. [17] frame the problem of knowledge base completion as question answering – finding the answer to the question "*source relation _?_*" to complete the triple $< source, relation, ?target >$. For this targeted information extraction task they use Google's Web search query logs to solve two subtasks: (i) finding alternative lexicalizations for the $relation$ in the incomplete triple, and (ii) query augmentation, which aims to discover associated information for given relations in the query logs, and gather them in the form of augmentation templates associated with particular relations. For example, for the *mother* relation, the augmentation template would be PLACE OF BIRTH, which for the incomplete triple $< Frank\ Zappa, mother, ? >$ would be instantiated to *Baltimore*. The lexicalizations and query augmentation templates are used to form multiple queries for an incomplete triple, and the results for all queries are aggregated to produce an overall ranking for the search results obtained with each query.

West et al. [17]'s use of query logs for finding information / attributes associated with a given relation cannot be replicated or used outside the search engine's organization. There are however alternative sources of information that can provide such information. We explore here knowledge graphs. FreeBase [1], YAGO [15], NELL [2], among others, capture knowledge in graph format, where nodes are entities, and directed links represent different types of relations among these entities (e.g. *nationality, profession, playsForTeam, birthPlace*). Lao et al. [9] propose a method for finding paths in a knowledge graph that are predictive of direct links. This work was further explored by [3, 6, 16], who improve on the original PageRank-based algorithm for traversing the graph and finding relevant paths. The motivation for finding such paths in the cited work was link prediction – when a path associated with a direct link is found, the direct link is proposed for completion of the graph.

We develop a method for targeted information extraction using path-relation associations produced by Das et al. [3], using a variation of the Path Ranking Algorithm implemented by Gardner and Mitchell [6]. We use the highest scoring paths p associated with a relation r to produce queries that contain different types of information, and study which type of associated information from a path – relations in the path, entities in the path, or both – has the most impact when used to form a query to complete a given incomplete triple using a search engine.

Lao et al. [9] and Yahya et al. [18] also perform a form of targeted information extraction using knowledge represented in graphs, but with a different purpose and in a different form. Lao et al. [9] use bibliometric information about scientific publications to form queries for addressing: (i) reference recommendation – finding relevant citations for a new paper; (ii) expert finding – finding a domain expert for a particular topic; (iii) gene recommendation – predicting the genes an author will write about in the next year. Queries consist of paths built based on "training data" consisting of the bibliometric network of scientific publications within a certain time slice, which are matched with metadata that describes each article in terms of authors, citations, title. Yahya et al. [18] move towards using open text for completing a knowledge graph, by processing a given collection and producing subject-predicate-object (SPO) triples, and implementing an extension of SPARQL triple patterns to cover structured search not only over the knowledge graph but also over the collected SPO-structured data.

Kotnis et al. [7] add bridging entities for the "on-demand augmentation" of a knowledge base using subject-verb-object (SVO) triples extracted from an external corpus. A source and a target entity which are not connected in the knowledge base will be linked through two SVO triples that share one of the arguments. This noun phrase will be a bridging entity, and will be added to the graph. To find the relevant SVO triples and the bridging entities, the "SVO" graph of the corpus is traversed through limited depth DFS (depth first search) starting from the source entity in the KB and ending at the target entity.

Similarly to these approaches we use structured data in form of a knowledge graph, and use this knowledge graph to provide "augmentation templates" in the form of patterns, which are paths associated with direct relations. We will use these patterns to augment queries for searching the web using a search engine. [1]

3 TARGETED INFORMATION EXTRACTION FOR KNOWLEDGE GRAPH COMPLETION

Adding information to knowledge graphs through targeted information extraction implies finding a specific piece of information. West et al. [17] call this a "pull" model – pulling specific information from available textual data – and contrast it with the "push" model, where all facts extracted from a text collection are organized into/added to a knowledge graph. In contrast with link prediction methods, the pull model aims to add new nodes to the graph by filling in incomplete $< source, relation, ? >$ triples.

As in [17], we assume the missing information would correspond to a new node in the knowledge graph, that is the missing part of a given relation $< source, relation, ? >$. We form queries for a search engine to help find this information. To enrich the information in the query, and thus make it more focused, we use patterns discovered in knowledge graphs, that associate paths to direct links, as explained in Section 3.1. In Section 3.2 we show how this information is used to form augmented queries.

3.1 Patterns in knowledge graphs

Knowledge graphs are highly interconnected structures. In such a structure there may be paths that are strongly associated with direct links. Lao and Cohen [8] explored the structure of the knowledge graphs using random walks to discover such associations. An example is one of the paths associated with the relation */people/person/nationality*[2] which connects a person and their nationality in the Freebase dataset, which we present in Figure 1.

We can view the path as an explanation, or a justification for the existence of the direct link: person X has the same nationality as person Y if they were born in the same city.

Das et al. [3], Gardner and Mitchell [6], Neelakantan et al. [13] improve upon this method by incorporating selectional preferences on intermediary nodes [3], or focusing on the subgraph structure surrounding a node [6]. We use the paths produced and made available by Das et al. [3]. The dataset is presented in Section 4.1.

3.2 Forming queries

We adopt West et al. [17]'s assumption that information that is strongly associated with a given relation can be used to produce more specific (and thus better) queries for retrieving information that is missing in a knowledge graph, from textual sources. The

[1]Because we focus on the "pull" model, and want to evaluate how well we can extract targeted information from external textual sources, we do not compare with "graph internal" link predictions methods, an overview of which can be found in [14].
[2]The name of Freebase relations consists of such three-part strings (as delimited by the "/"). When the last part of the string is unambiguous, we will use only that to denote the relation.

Relation type: nationality (Person X has nationality Y)

Path type:
$S \xrightarrow{\text{place of birth}} X \xrightarrow{\text{place of birth}^{-1}} Y \xrightarrow{\text{nationality}} T$

Relation instance: *Andris Berzins* $\xrightarrow{\text{nationality}}$ Latvia

Path instance:
Andris Berzins
$\xrightarrow{\text{place of birth}}$ *Riga*
$\xrightarrow{\text{place of birth}^{-1}}$ *Karlis Vilertes*
$\xrightarrow{\text{nationality}}$ *Latvia*

Figure 1: Relation type with an associated path type and an example instance.

information we use consists of path patterns discovered through random walks that are strongly associated with specific relations.

When aiming to fill in information that would complete a triple $< source, relation, ? >$, the simplest approach would be to form a query Q_{s+r} = "*source relation*", and look for the answer (i.e. the missing argument of $relation$) in the documents returned by a search engine. If there are paths strongly associated with relation r, e.g. $p_r = \{r_1, r_2, ..., r_n\}$, we can form alternative queries that contain different types of additional information. We summarize them in the list below, with examples for an incomplete triple $< Andris\ Berzins, nationality, ? >$, with the associated path type

$$S \xrightarrow{\text{place of birth}} X \xrightarrow{\text{place of birth}^{-1}} Y \xrightarrow{\text{nationality}} T$$

and the path instance presented in Section 3.1:

Q$_{s+r}$ the query includes the source and relation information:
Andris Berzins nationality

Q$_{s+r+p}$ the query will include the source and relation information, plus the relation types on the path p_r:
Andris Berzins nationality place of birth

Q$_{s+r+n}$ the query will include the source and relation information, plus the intermediate nodes on the path p_r:
Andris Berzins nationality Riga Karlis Vilertes

Q$_{s+r+p+n}$ the query will include all available information from the path:
Andris Berzins nationality place of birth Riga Karlis Vilerts

In an extreme test scenario, the most realistic option for query expansion would be Q_{s+r+p} where we add the relation types on a potential path to the source and target. In such a scenario we don't assume that we can superimpose the path on the graph and find (at least some of the) intermediate entities, but use only the generalized path that was found associated with a specific relation. It is quite possible however that some intermediate nodes do already exist in the knowledge graph, so we include them to study their potential impact on the targeted information extraction task.

# relation types	27,791
# query relation types	46
# relation instances	3.22M
avg. path length	4.7
max path length	7

Table 1: Statistics of the dataset [3]

	train	dev	test
# of relations	46	46	46
avg. # instances / relation	6621	897	3516
avg. # paths / relation	324649	51419	180752

Table 2: Relation and path statistics for the data used

4 EXPERIMENTS

We describe here the experimental set-up for evaluating the hypothesis that using patterns discovered in knowledge graphs can aid the targeted expansion of the knowledge graph. Subsection 4.1 describes the data – the knowledge graph and the path patterns – used for query expansion as presented in section 3.2. The queries formed were used with the Google search engine, and the results returned were analyzed and scored as explained in section 4.3.

4.1 Data

We use the dataset released by [3][3]. This data is a subset of Freebase enriched with ClueWeb information: Freebase relations with identifiers such as */people/person/nationality* are enhanced with lexical expressions from ClueWeb such as *born in*. The dataset consists of a knowledge graph expressed as relation triples
$< source, relation, target >$. For a subset of 46 relation types (called "query relations"), the Path Rank Algorithm (PRA) finds associated paths in the knowledge graph. For each instance of the query relations, the applicable paths are expanded with intermediate entities, gathered through a depth first traversal starting from the first entity of the entity pair in the relation instance and following the relation types on the path until reaching the last entity of the pair. The dataset comes split into training, development and test subsets. We use the training portion to select paths for the query relations to serve as expansion information, a subset of the development set for the development of the approach, and a subset of the test set for the final experiments.

There are numerous paths associated with each relation. On the other hand, there are paths that apply to very few instances of a relation. To select representative and productive paths for each relation type, we weigh them using two scores similar to those commonly used to score words in documents, computed based on relation and path information in the training partition of the data:

[3]http://iesl.cs.umass.edu/downloads/akbc16/

Relations from R$_{all}$ (sample)					
relations	# path types	# paths in dev	# filtered paths in dev	# paths in test	# filtered paths in test
/aviation/airport/serves	2	51	51	60	59
/book/book/characters	2	60	60	60	60
/book/written_work/original_language	2	60	60	60	58
/broadcast/content/artist	2	60	58	60	60
...					
total	92	2487	2384	2649	2548
The R$_3$ relations					
/people/person/nationality	5	277	258	463	430
/people/person/place_of_birth	5	88	88	175	169
/people/person/profession	5	528	523	500	497
total	15	894	869	1138	1096

Table 3: Working data statistics: the relations used, the number of path types for each relation, and the number of instances from development and test datasets

Score$_1$ is a form of df (document frequency), as we want paths that are relevant (i.e. have many instances) for a specific relation, but not for others. For a path type p, we compute its score with respect to an associated relation r and all relations r_i in the dataset as:

$$Score_1(p,r) = \frac{|\{< s,r,t > \mid < s,p,t >\}|}{\sum_{i=1}^{n} |\{< s,r_i,t > \mid < s,p,t >\}|}$$

where $\{< s,r,t > \mid < s,p,t >\}$ is the set of instances of relation r that can be connected through path p.

Score$_2$ is a frequency score that measures the proportion of instances of relation r that have associated an instantiation of a path p:

$$Score_2(p,r) = \frac{|\{< s,r,t > \mid < s,p,t >\}|}{|\{< s,r,t >\}|}$$

Because we want to use patterns (paths) that are both productive and also precise with respect to a given relation, we used the two scores in a two-step filtering process – from the most productive paths (high *Score*$_2$) we select those that appear with few other relations (high *Score*$_1$). We form two sets for evaluation, whose statistics are presented in Table 3:

R_{all} contains two of the highest scoring paths for each of the 46 query relations, and for each path we extract randomly approximately 30 relation instances from the development and test data (if there aren't 30 instances, we extract as many as there are).

R_3 contains 5 path types for 3 of the relations: */people/person/nationality, /people/person/place_of_birth, /people/person/ profession.* For each path type we randomly extract approximately 100 instances from each of the development and test data (if less than 100 instances exist, we extract as many as there are). We chose these relations for additional analysis, as they overlap with the relations investigated by West et al. [17].

4.2 Query expansion

Some node ids from the selected $< source, relation, target >$ instances could not be mapped to an entity name, so they are filtered out. For the remaining ($< source, relation, target >, path$) instances in the working data, we form the four types of queries described in Section 3.2. We consider the *target* unknown, and try to retrieve it by querying a search engine (Google) with the query variations built. We pass the query to Google through a command line interface, and we retain the first page of the results as our candidates for finding the missing *target* entity in the input relation triple. We save the results of the search as an html file which we process with the BeautifulSoup python package.

Not all queries return at least 10 answers. Table 4 shows statistics about the number of queries that returned at least k answers ($k \in \{1, 3, 10\}$), and the number of relations represented in the final answer set.

k	Q_{s+r} # queries	Q_{s+r+n} # queries	$Q_{s+r+p+n}$ # queries	Q_{s+r+p} # queries
R_{all}				
1	2548	2546	2543	2548
3	2548	2530	2537	2547
10	1202	1273	1847	1947
R_3				
1	1096	1095	1093	1096
3	1096	1093	1091	1096
10	620	524	793	862

Table 4: Number of queries for which the search engine returned at least k answers.

We consider that the returned text fragment (the text snippet on Google's results page for a query) is correct if it contains the *target* entity. Because an entity can be referred to through various phrases – e.g. "United Kingdom", "UK" all refer to the same named entity – we use an index of alternative names for entities extracted from Wikipedia's article titles, redirect pages, disambiguation pages and anchor texts [12][4]. Using such a reference is reasonable since Freebase contains much data harvested from Wikipedia itself, and contains mostly named entities. When a potential name for the *target* is found, we consider it as referring to the correct entity. The context we look at is the snippet returned by the search engine, in which the entities disambiguate each other. Because the entities we consider are closely connected in the knowledge graph, the assumption that the matched entity refers to the one we look for is reasonable, and it reflects the coherence assumptions made by word sense disambiguation/entity linking approaches: that words/entities disambiguate each other (e.g. [11]). A manual analysis on the development data has shown that variations in entity names are those most commonly used, which are covered by the resource.

We do not consider variations in the name of the *source* and intermediate entities and of the *relation* and the relations on the associated path (other than what we obtain from ClueWeb). Trying multiple combinations of these is impractical. We rely on the redundancy of information on the Web to compensate for this lack of lexicalization variation, and on Google's internal indexing mechanisms that will partly mitigate this problem.

4.3 Evaluation methodology

To evaluate the results of the query we rely on commonly used metrics in information retrieval – average precision@k, average recall and comparable precision. As mentioned in the previous section, we evaluate the text snippets in the first page of Google search results to a given query, that includes up to 10 answers. We consider an answer correct if the corresponding text snippet contains a mention of the *target* entity:

$$P@k = \frac{|\{d_i | d_i \text{ contains } target\}|}{k}$$

$$avgP@k = \frac{\sum_{q(Doc\#)\geq k} P@k}{n_{q(Doc\#)\geq k}}$$

$$avgRecall = \frac{\sum_{q(Doc\#)\geq k} P@k}{N}$$

$$comparable \quad precision = \frac{\sum_{q_{all}(Doc\#)\geq k} P@k}{n_{q_{all}(Doc\#)\geq k}}$$

In P@k, k corresponds to the number of relevant results on the first search page. We use $k \in \{1, 3, 10\}$, because not all queries return at least 10 results. To compute each score we combine the results for the queries that return at least k answers (see Table 4). In the formula for average recall, N denotes the total number of query data.

Each query expansion variation covers a different number of query data – they are those for which Google returns at least k snippets. Because of this we compute *comparable precision* as the average precision@k for queries which have at least k answers for all query expansion variations.

[4]WikiNet downloadable from https://drive.google.com/file/d/0B-PzM1Nn4gpMeUVGdnBvb3VXWVE/view?usp=sharing

k	query exp.	avg. p@k	avg. recall	comp. precision
R_{all}				
1	Q_{s+r}	**0.35**	**0.35**	**0.35**
	Q_{s+r+n}	0.30	0.30	0.30
	$Q_{s+r+p+n}$	0.26	0.26	0.26
	Q_{s+r+p}	0.25	0.25	0.25
3	Q_{s+r}	**0.28**	**0.28**	**0.28**
	Q_{s+r+n}	0.26	0.25	0.26
	$Q_{s+r+p+n}$	0.22	0.22	0.22
	Q_{s+r+p}	0.21	0.21	0.20
10	Q_{s+r}	0.19	0.09	0.20
	Q_{s+r+n}	**0.22**	0.11	**0.23**
	$Q_{s+r+p+n}$	0.19	**0.13**	0.19
	Q_{s+r+p}	0.16	0.12	0.14
R_3				
1	Q_{s+r}	**0.55**	**0.55**	**0.55**
	Q_{s+r+n}	0.45	0.45	0.45
	$Q_{s+r+p+n}$	0.51	0.51	0.51
	Q_{s+r+p}	0.49	0.49	0.49
3	Q_{s+r}	**0.42**	**0.42**	**0.42**
	Q_{s+r+n}	0.38	0.38	0.38
	$Q_{s+r+p+n}$	0.41	0.41	0.41
	Q_{s+r+p}	0.38	0.38	0.38
10	Q_{s+r}	**0.34**	0.19	**0.34**
	Q_{s+r+n}	0.32	0.15	0.32
	$Q_{s+r+p+n}$	0.32	**0.23**	0.33
	Q_{s+r+p}	0.29	**0.23**	0.28

Table 5: Average p@k, recall and comparable precision of different query expansion settings on the test dataset

4.4 Results

We first look at average results over the test data, in terms of average precision, recall and comparable precision (Table 5) for the 4 different query expansion variations.

With a few exceptions, the results on the baseline – the query formed from only the source and relation – are higher. This however does not adequately show what the impact of query expansion on targeted information extraction is. A closer look shows that not always the same expansion method performs best. We analyzed the best performing query expansion for the relations in our dataset, and include the results in Table 6 and a summary of relative performance in Figure 2. The statistics show for how many relations (*nr*. and which proportion of the data this corresponds to – %) each query expansion performed best. Ties are assigned to all tied query types, and we include the evaluation in terms of the three metrics. For each set of relations that has higher performance with one of the query expansion methods, we include the score for the baseline (Q_{s+r}) setting in parentheses. The results show that the improvement when using a query expansion method compared to

k	Metrics	Q_{s+r}			Q_{s+r+n}			Q_{s+r}	$Q_{s+r+p+n}$			Q_{s+r}	Q_{s+r+p}			Q_{s+r}
		nr.	%	score	nr.	%	score	score	nr.	%	score	score	nr.	%	score	score
1	avg. p@1	22	(0.39)	0.4557	16	(0.29)	0.3178	(0.2459)	12	(0.22)	0.1785	(0.1496)	6	(0.11)	0.1802	(0.1377)
	avg. recall	22	(0.39)	0.4557	17	(0.30)	0.3079	(0.2373)	11	(0.21)	0.1783	(0.1496)	6	(0.11)	0.1802	(0.1377)
	comp. prec.	22	(0.39)	0.4565	17	(0.30)	0.3076	(0.2379)	11	(0.21)	0.1785	(0.1497)	6	(0.11)	0.1802	(0.1377)
3	avg. p@3	18	(0.33)	0.3762	18	(0.33)	0.2594	(0.1988)	13	(0.24)	0.1453	(0.1208)	6	(0.11)	0.2843	(0.2431)
	avg. recall	18	(0.33)	0.3762	18	(0.33)	0.2591	(0.1988)	13	(0.24)	0.1453	(0.1208)	6	(0.11)	0.2843	(0.2431)
	comp. prec.	18	(0.33)	0.3785	18	(0.33)	0.2596	(0.1979)	13	(0.24)	0.1453	(0.1209)	6	(0.11)	0.2843	(0.2431)
10	avg. p@10	17	(0.36)	0.2329	14	(0.30)	0.2942	(0.201)	9	(0.19)	0.1209	(0.0731)	7	(0.15)	0.1754	(0.1425)
	avg. recall	6	(0.13)	0.1328	11	(0.23)	0.2113	(0.146)	18	(0.38)	0.1339	(0.0524)	13	(0.27)	0.1279	(0.0688)
	comp. prec.	16	(0.30)	0.2377	19	(0.35)	0.2957	(0.2102)	13	(0.24)	0.1072	(0.0703)	6	(0.11)	0.0508	(0.0374)

Table 6: The number of relations (and percentage) with highest results for different query expansion settings for R$_{all}$(Test), and average baseline (Q_{s+r}) results for the relations that perform better with each query expansion type

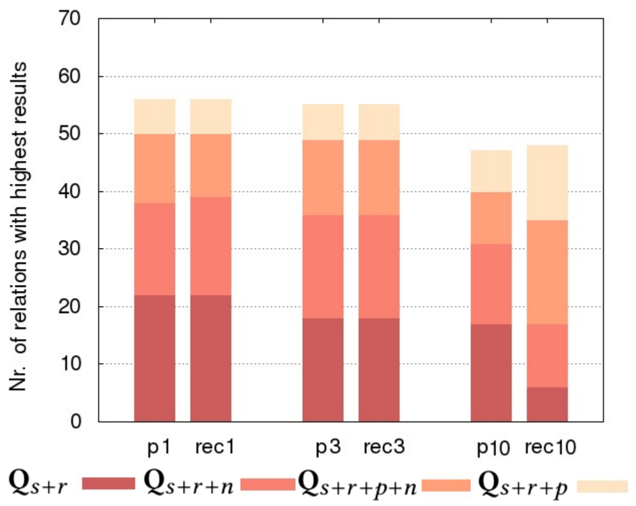

Figure 2: An overview of the comparative performance of the different query expansion methods in terms of precision and recall (at 1, 3 and 10).

the baseline is considerable. When a query expansion gives better results than the baseline, the baseline resuts are very low. This indicates that expansion helps with the more difficult cases.

The results show that the baseline performs best for less than a third of the relations in the test set. We also notice that different relations perform best with different query expansion types. We looked into the relations that perform best at k=1 when adding only relation types from the associated path – which would be the ideal situation for question answering. The problem is that only 6 relations performed well under these conditions, some of which are quite specific, e.g. */film/film/country, /music/artist/origin* but others are quite general, e.g. */location/location/contains*. The paths used to expand the queries for the more specific relations also contain quite specific relations – */film/film/country* is expanded with /film/film/language, and */music/artist/origin* is expanded with */music/group_membership/group*$^{-1}$, */music/group_membership/member, /people/person/place_of_birth.*

There is no obvious pattern for which expansion type would work best for each relation. Even for relations that have best results with the baseline, there are both specific relations, e.g. */education/ educational-institution/school_type, /film/film/sequel*, and more general (*/geography/river/cities, /people/person/nationality*). For the more specific relation, it is intuitive that additional information would not be necessary. For the more general relations we find that the cause for the baseline working best is that there are both good and bad paths that are associated with these relations. We exemplify this with the *nationality* relation, which is part of R_3 and was analyzed more deeply with a higher number of instances and paths.

An example good path for *nationality* for which one of the query expansion methods leads to better performance than the baseline has been shown in Section 3.1:

$$\textit{Andris Berzins} \xrightarrow{\text{place of birth}} \textit{Riga} \xrightarrow{\text{place of birth}^{-1}} \textit{Karlis Vilertes} \xrightarrow{\text{nationality}} \textit{Latvia}$$

This path associated with the *nationality* direct link can be interpreted as: people who have the same place of birth have the same nationality. For this example, Q_{s+r+n} and Q_{s+r+p} work better than the baseline Q_{s+r}.

Another path associated with the *nationality* relation also leads to better results when used for query expansion:

$$\textit{Andre Malraux} \xrightarrow{\text{places lived}} \textit{Paris} \xrightarrow{\text{place of birth}^{-1}} \textit{Georges Bernanos} \xrightarrow{\text{nationality}} \textit{France}$$

which can be interpreted as: if a person lives in a place that is the birth place of someone else, the two persons could have the same

$S \xrightarrow{/people/person/place\_of\_birth} T$
$S \xrightarrow{/people/person/nationality} \ldots \xrightarrow{/people/person/nationality^{-1}} \ldots \xrightarrow{/people/person/places\_lived} \ldots \xrightarrow{/people/place\_lived/location} T$
$S \xrightarrow{/people/deceased\_person/place\_of\_death} \ldots \xrightarrow{/location/location/contains^{-1}} \ldots \xrightarrow{/people/person/nationality^{-1}} \ldots \xrightarrow{/people/person/place\_of\_birth} T$
$S \xrightarrow{/people/deceased\_person/place\_of\_death} \ldots \xrightarrow{/people/deceased\_person/place\_of\_death^{-1}} \ldots \xrightarrow{/people/person/place\_of\_birth} T$
$S \xrightarrow{/people/deceased\_person/place\_of\_death} \ldots \xrightarrow{/location/location/contains^{-1}} \ldots \xrightarrow{/location/location/contains} T$
$S \xrightarrow{/education/education/student^{-1}} \ldots \xrightarrow{/education/educational\_institution/students\_graduates^{-1}} \ldots \xrightarrow{/location/location/contains^{-1}} T$

Table 7: Highest ranking paths associated with the */people/person/place_of_birth* relation that we used for query expansion

nationality. For this path type the query expansions Q_{s+r+n} and Q_{s+r+p} lead to better search results than Q_{s+r}.

The path extraction algorithm also extracts incorrect paths, that lead to wrong results. An example of this is:

$$\textit{Sean Connery} \xrightarrow{\text{profession}^{-1}} \textit{Actor} \xrightarrow{\text{profession}} \textit{Harrison Ford} \xrightarrow{\text{nationality}} \textit{United States}$$

This path is too general, since people with the same profession do not always have same nationality. The fact that such a path has been found as highly correlated with the *nationality* relation may have been caused by skewness in the data, where American actors are a majority of the actors present. For such erroneous paths, the baseline Q_{s+r} leads to better search results.

We used three relations for more in-depth analysis, including */people/person/place_of_birth* and */people/person/nationality* which are two of the relations analyzed by West et al. [17]. We cannot directly compare the results, but we can compare the relations on the paths we use to expand the query with their augmentation templates, since they have a similar purpose and effect. West et al. [17] show the heatmap that captures the improvement in performance for the *place of birth* relation relative to the query expansion templates extracted from the query logs, which we reproduce here (Figure 3). West et al. [17] obtain the highest results either when not augmenting the query, or when using the *parents* augmentation template.

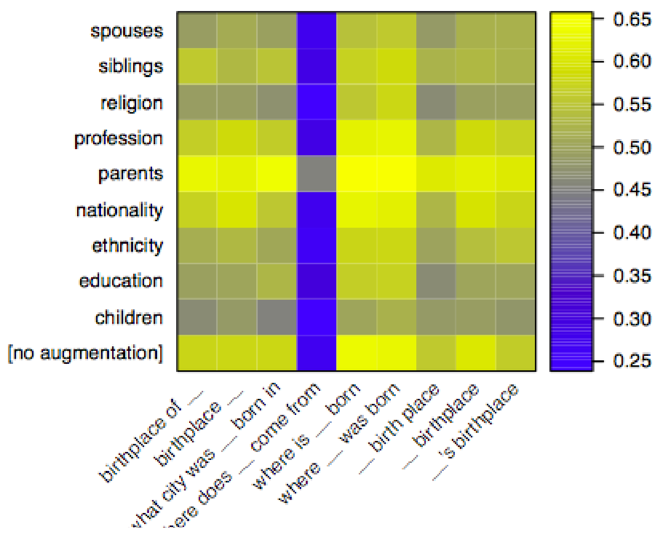

Figure 3: Heatmap for the *place of birth* relation with lexicalization templates (on the x axis) and expansion templates (on the y axis) from West et al. [17]

Table 7 shows the paths associated with the */people/person/place_of_birth* that we used for query expansion. In the k=1 setting, the best results (in terms of all metrics) are obtained with the Q_{s+r+p} setting (0.4 for Q_{s+r+p} vs 0.37 for Q_{s+r}). We note some overlap though with respect to the templates discovered by West et al. [17] and the relations in the paths we used, in particular the *nationality* relation, and the education-related relations */education/education/student* and */education/educational_institution/students_graduates*. While the *profession* relation exists in FreeBase, it does not appear in the paths we used. For k=3,10 no expansion works best, which reflects to a certain degree the results obtained with query logs, as this relation seems to perform well on its own.

5 DISCUSSION

The results of the search process are of course heavily dependent on the accuracy of the patterns induced from the data. While the PRA and its variations have made a breakthrough in discovering regularities in the knowledge graph, there is much space for improvement. The results of these experiments lead us to believe that improvements in the paths or other patterns associated with specific relations would contribute towards improving the targeted information extraction as well. There are also alternatives that could be used instead of the paths, for example the subgraph features, which describe the local neighbourhood of a node [6].

We find particularly interesting the fact that expanding the query only with path relation information, which would be the most easily accessible, leads to high results particularly when looking only at the first text fragment on the Web search results page. We rated each query expansion method separately, to understand the influence of the different types of information that patterns found in graphs provide. West et al. [17] note that the results of the extraction are better when the answers obtained separately are aggregated to produce an overall ranked list of answers. In future work we will investigate this procedure for the method we proposed. Another interesting avenue to explore would be to find criteria (e.g. the specificity/generality of the relations in a path) that would allow us to automatically choose the query expansion type that would lead to best results for a relation.

The application of our method has a few shortcomings with respect to forming natural language queries from incomplete facts. We only use the lexicalizations of relations obtained from ClueWeb, and only one for each relation. For the known argument of the relation and intermediary entities we only use the lexicalization associated with the node ids in FreeBase. We rely on redundancy of information on the Web and Google's internal indexing mechanism to partly compensate for these limitations. Because of this we consider that the potential of patterns found in knowledge graphs is yet to be fully realized for the task of targeted information extraction.

6 CONCLUSIONS

We have set out to investigate the potential of patterns in knowledge graphs, in the form of paths associated with different relation types, for adding information to the graph through targeted information extraction. We have used the information provided by these patterns to form different types of queries for filling in incomplete relation triples of the form $< source, relation, ? >$. The results of experiments performed on the FreeBase knowledge graph show that in the majority of cases, expanding the query leads to better results than using the baseline setting, which consists of "*source relation*".

There are multiple avenues to be explored in future work: from the patterns used to form queries, to lexicalizing this information, to experimenting with different methods for combining the search results. However, the results of the presented experiments show conclusively the usefulness of structural information in knowledge graphs for building queries for targeted extraction of new nodes.

REFERENCES

[1] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD '08)*. ACM, New York, NY, USA, 1247–1250. https://doi.org/10.1145/1376616.1376746

[2] Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka, Jr., and Tom M. Mitchell. 2010. Coupled Semi-supervised Learning for Information Extraction. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining (WSDM '10)*. ACM, New York, NY, USA, 101–110. https://doi.org/10.1145/1718487.1718501

[3] Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2016. Incorporating Selectional Preferences in Multi-hop Relation Extraction. In *AKBC@NAACL-HLT*.

[4] 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 *KDD*.

[5] Xin Luna Dong, Evgeniy Gabrilovich, Kevin Murphy, Van Dang, Wilko Horn, Camillo Lugaresi, Shaohua Sun, and Wei Zhang. 2015. Knowledge-based Trust: Estimating the Trustworthiness of Web Sources. *Proc. VLDB Endow.* 8, 9 (May 2015), 938–949. https://doi.org/10.14778/2777598.2777603

[6] Matt Gardner and Tom M. Mitchell. 2015. Efficient and Expressive Knowledge Base Completion Using Subgraph Feature Extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. 1488–1498. http://aclweb.org/anthology/D/D15/D15-1173.pdf

[7] Bhushan Kotnis, Pradeep Bansal, and Partha P. Talukdar. 2015. Knowledge Base Inference using Bridging Entities. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, 2038–2043. http://aclweb.org/anthology/D15-1241

[8] Ni Lao and William W. Cohen. 2010. Relational Retrieval Using a Combination of Path-constrained Random Walks. *Mach. Learn.* 81, 1 (Oct. 2010), 53–67. https://doi.org/10.1007/s10994-010-5205-8

[9] Ni Lao, Tom Mitchell, and William W. Cohen. 2011. Random Walk Inference and Learning in a Large Scale Knowledge Base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '11)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 529–539. http://dl.acm.org/citation.cfm?id=2145432.2145494

[10] Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-Value Memory Networks for Directly Reading Documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 1400–1409. http://aclweb.org/anthology/D16-1147

[11] Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity Linking meets Word Sense Disambiguation: a Unified Approach. *Transactions of the Association for Computational Linguistics* 2 (2014), 231–244.

[12] Vivi Nastase and Michael Strube. 2013. Transforming Wikipedia into a large scale multilingual concept network. *Artificial Intelligence* 194, Supplement C (2013), 62 – 85. https://doi.org/10.1016/j.artint.2012.06.008 Artificial Intelligence, Wikipedia and Semi-Structured Resources.

[13] Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional Vector Space Models for Knowledge Base Completion. *CoRR* abs/1504.06662 (2015). http://arxiv.org/abs/1504.06662

[14] Michael Nickel, Kevin Murphy, Volker Tresp, and Evgenyi Gabrilovich. 2016. A Review of Relational Machine Learning for Knowledge Graphs. *Proc. IEEE* 104, 1 (Jan 2016), 11–33. https://doi.org/10.1109/JPROC.2015.2483592

[15] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. YAGO: A Core of Semantic Knowledge. In *Proceedings of the 16th International Conference on World Wide Web (WWW '07)*. ACM, New York, NY, USA, 697–706. https://doi.org/10.1145/1242572.1242667

[16] Quan Wang, Jing Liu, Yuanfei Luo, Bin Wang, and Chin-Yew Lin. 2016. Knowledge Base Completion via Coupled Path Ranking. In *ACL*.

[17] Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge Base Completion via Search-Based Question Answering. In *WWW*. http://www.cs.ubc.ca/~murphyk/Papers/www14.pdf

[18] 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, New York, NY, USA, 605–614. https://doi.org/10.1145/2835776.2835795