Problem Description

Working title:
1: Top-k relevant spatio-temporal retrieval on knowledge graphs using keyword search
2: Spatio-temporal keyword search on RDF graphs.

Knowledge graphs allows for easy extraction of facts from an entity, and the structure makes it efficient to find multi degree relations between different entities and facts. The task is to study indexing and search techniques on graphs containing spatio-temporal data, and propose effective ways to retrieve the data contained in the graph based on keyword searches.

Summary

Write your summary here...

Preface

Spatio-temporal keyword search on RDF graphs.

Table of Contents

List of Tables

List of Figures

Abbreviations

Fact	=	Subject or object in graph, containing some small piece of information
KB	=	Knowledge base
KG	=	Knowledge graph

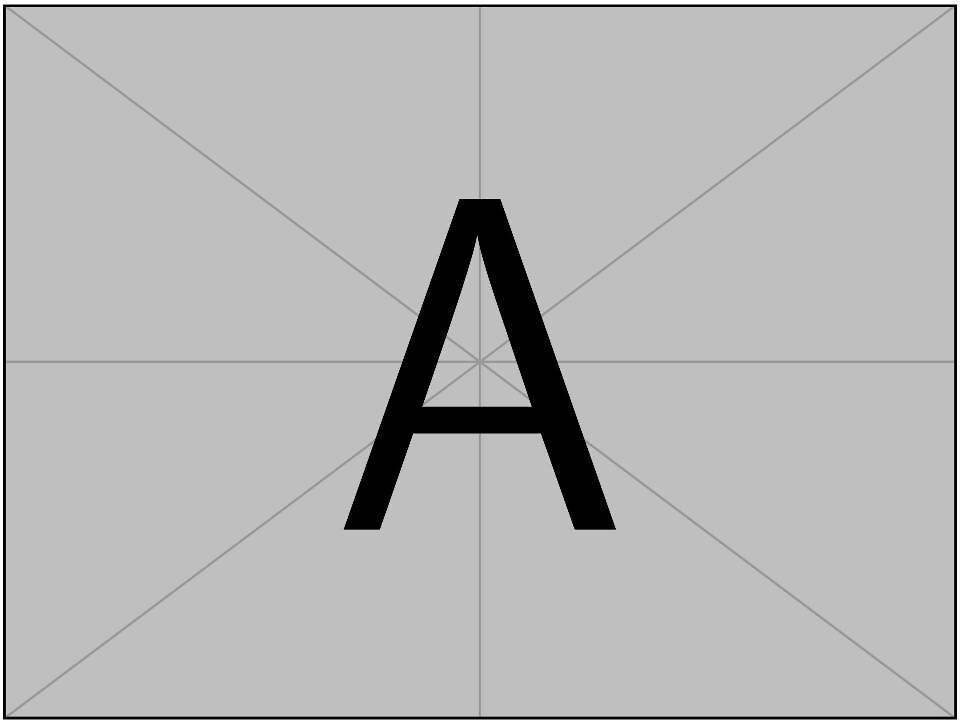
Chapter 1

Introduction

A knowledge base is a collection of information. This collection is usually centered around a topic, but there also exists large, general, knowledge bases. Smaller knowledge bases are typical for any type of community, such as a company, school, or fan community, that seeks to preserve and share information. The larger knowledge bases have the same goal as the smaller ones, but the scope will be much larger. A common type of knowledge base is a wiki, a collection of linked articles where any one with the correct privileges can edit, add or delete articles. Knowledge bases can however be any type of collection. The information can be both structured, as in a wiki, or unstructured, with no link between the information. The information in a structured collection can be used to create a graph with information in small chunks representing a fact, as well as the relation between such facts. There exists multiple such graphs today most are based on Wikipedia, such as Yago[7] and DBpedia[1].

When a set of articles are linked they form a graph with each article being a entity in this graph[5, 4]. Most articles falls into one or more categories, which is in turn used create a taxonomy of the entities. Since a category can be a subset of a different category, this creates a hierarchy of different entities based on what categories an entity falls under. Using the entity and category, combined with the linking of articles it is possible to create a directed graph of entities.

Knowledge graphs is usually structured with a subject, predicate, and object. Both the subject and object are entities, or facts, and the predicate describes the relation between two facts. This structure makes the graph a directed cyclic graph. An example of the subject predicate object can be ”Nidaros Cathedral” ”is located in” ”Trondheim”.



One of the uses of knowledge graphs today is to find and display a info box in search engines. This information is a compact set of facts that tries to fit the search query. Because of the graph structure of knowledge graphs the information in the info box can be adapted to the query by choosing the predicates and related facts closest related to the query. This makes it possible to create a set of information that can give the user a quick overview of the information retrieved by the query.

This paper will propose methods of integrating time and date facts in existing search methods.

1.1 Previous work

There has been done some work keyword search on RDF graphs. This includes [9] and [2]. None of these takes spatial or temporal data into consideration. The methods for keyword queries outlined in the papers are based on a combination of indexing the graph and traversing subgraphs. The indexing is similar in both papers, but retrieval, scoring and traversal methods varies. One method of retrieval and scoring that is used as a baseline is a breadth first search for traversal and a minimal subgraph for scoring. These methods are easy to implement and understand, but often inefficient, and inaccurate.
In paper [6] some methods for indexing, searching and ranking keyword searches on spatial RDF graphs is proposed. These methods builds upon the methods outlined in [9, 2].

The most substantial difference is the use of R-trees to index the spatial dimension of the graph. This is done so that a subgraph containing the keywords can be retrieved given a location. This method starts by finding the closest spatial vertex and uses that as the root for the subgraph. Subgraphs rooted in a spatial node requires different ranking, and there is proposed a set of rules in the paper.
R-trees are designed around the spatial dimension, but some work on modifications to include a temporal dimension has been done. Some research done on spatio-temporal tree structures include [8, 10]. Most of the spatio-temporal trees are however created to be able to query and index moving or frequently updated objects. To include a temporal dimension some differences in the tree structures are made. Most approaches builds on the R-tree, but modifies the tree in different aspects. For the purpose of this thesis a different tree structure is not needed.

Research questions?
RQ1: How can spatio-temporal data be integrated into exiting keyword query methods for RDF data.
RQ2: What methods can be used to create more effective queries on RDF data.
RQ3: How do spatial and temporal RDF query methods differ from from other query methods.
RQ4: What methods can be used to query RDF data based on a users keyword search.

Chapter 2

Background

- More on existing KB
- Structure of Yago?
- What makes Yago special
- Describe DBPedia

2.1 Knowledge Graphs

Knowledge graphs is a representation of complex data that describes facts and relationship between facts. Facts can be placed into different categories, and can have different properties. Combining facts from different categories or with different properties can give a quick overview of a topic. Based on a query facts can be combined to give users a quick answer.

- More about graphs in general

Graphs theory is a prominent part of computer science. Graph structures is a common way to create a dataset that can be efficiently explored for a variety of purposes, and with many different algorithms.

- Type of graphs used in knowledge graphs

Because of the linking of articles in common knowledge bases it is possible to derive a graph that represents this linking. This graph will be a cyclic directed graph.

2.1.1 Yago

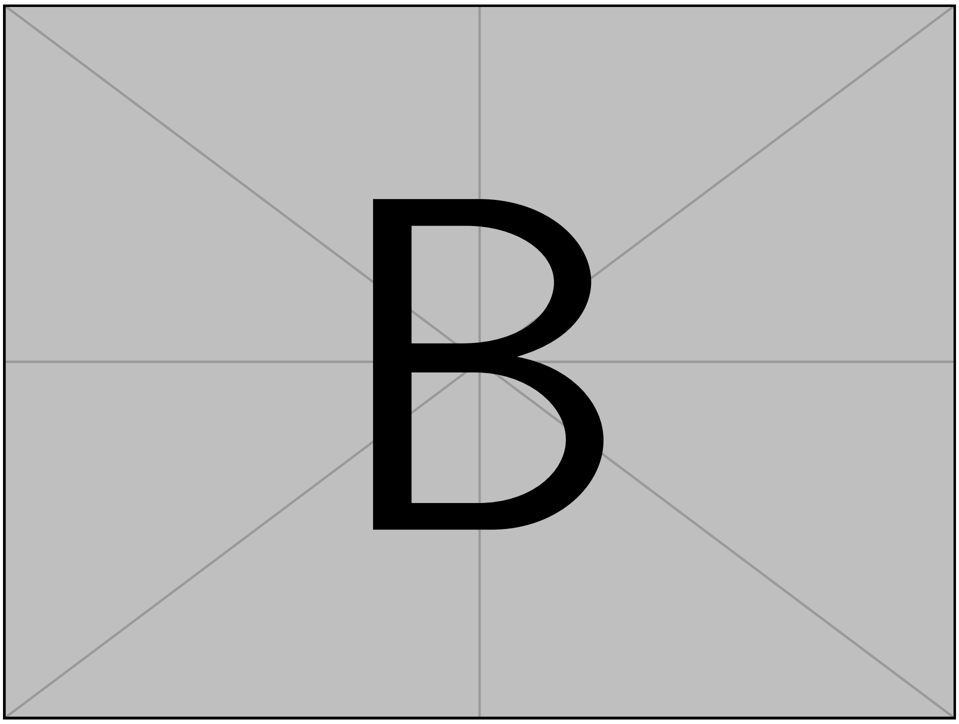
Yago[7] is a knowledge graph created from a set of different sources. The graph is built from Wikipedia, WordNet, GeoNames and Wikidata. This combination of sources makes it possible to get information in multiple languages, and spatio-temporal data.

- Describe the RDF N3/ triplet format

All facts in Yago consists of a subject, predicate and object. This fits perfectly into the

RDF N3, or Notation3 format.

- Use of triplets.



- Graphic describing triplet format and how it forms graphs

- Temporal data in Yago.

Predicates form the link between facts, and can be divided into many sub-groups. This grouping helps when finding possible temporal data for a fact. Facts in the categories ‘People’, ‘Groups’, ‘Artifacts’, and ‘Events’ will usually also contain a link to a time or timespan.

- Spatial data in Yago.

- Different data sets in Yago.
- Data sets used in this thesis.
- Yago Date Facts
- Yago Facts.
- Yago Geonames Only Data.
- Pseudo-hierarchy of facts.
- Use of persistent TDB dataset.

Chapter 3

Basic retrieval methods

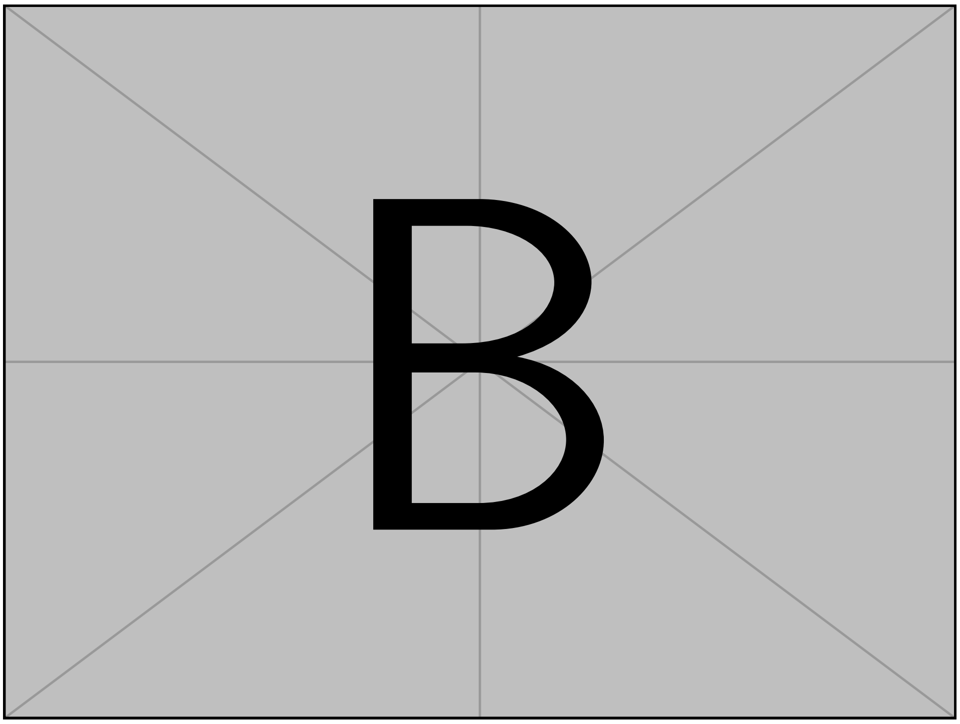
3.1 Indexing

- General indexing, need for index.
- Keyword index

3.1.1 Spatial indexing

- Different types of indexes.
- Uses for index in general and thesis.
- Querying(?).

Most spatial indexing is done using R-trees. R-trees are based on a B-tree, but is optimized for indexing spatial dimensions. R-trees are able to effectively index spatial data by dividing space into smaller and smaller sets, or bounding boxes, and having the leaf nodes of the trees representing the smallest unit of space used in the data set, usually a small area or a point.



- Image Explaining R-tree index/ bounding.

3.1.2 Temporal indexing

In the dataset used in this thesis objects do not move, or move rarely. This makes indexing time unnecessary and it can be treated as a fact in the object with a start, and possible end time. This allows time to be retrieved as a fact when traversing the graph the same way as other facts.

3.2 Search

3.2.1 Bottom up BFS search

The most basic method of finding a match for keywords is using a bottom up breadth first search (BFS) [3]. For each keyword in the query the algorithm will find all vertices that contain the keyword. From that set of vertices the BFS search finds the first vertex that can connect all the vertices in the set. The vertex that connects the the rest of the set is the one that best fits the keywords in the search. There is however no guarantee that this set of vertices contains any spatial or temporal data. If this is the case, the search will continue until the a vertex containing spatial and temporal data is found. The first set of vertices containing all the keywords, a place, and time will be the best result using this search. The BFS search can however contain multiple times, or multiple places. To determine what time or what place best fits the query, a separate ranking algorithm can be used. BFS is also a time consuming algorithm and is used as a proof of concept and baseline in this thesis.

Algorithm 1 BFS

```
1: procedure BFS nodes = getRootNodesFromPlace() maxDepth = 3
        while nodes is not empty: currentNode = node[0] if node.depth ¿ maxDepth:
    break if node contains query word: add node to list of hits if all query words are
    found: set maxDepth to current node depth add neighbors of node to nodes list
```

- Traversing the main graph
The first part of the search is to find a set of root nodes for sub-graphs. This is done by collecting all neighbors from the place queried. The neighbors can be connected by any predicate in the first developed algorithm, but as explain in 4.0.1 this selecting specific predicates can greatly increase speed by limiting the amount of nodes traversed. The BFS algorithm described above is used for each of the possible roots of subgraphs, stopping when all keywords are matched, or the depth of the graph exceeds the three or exceeds the currently shallowest subgraph that has matched all keywords. The reason for limiting the depth of subgraphs is to ensure at least a partial hit within a reasonable time. Limiting the depth of subgraphs to the current shallowest subgraph is done because no a deeper graph cannot be a more accurate hit. - Tokens - Objects When traversing the graph and finding a match, a node object is created. This object is used to keep track of the pseudo hierarchy in the graph. All objects contain information on the depth of the node, matched query terms, and relation to parent and children, if any. In addition root objects contains a list of all query terms hit in the sub graph. If a child node is found within multiple subgraphs, a new object will be created representing the node for each subgraph. This creates some objects that are nearly identical, but with different relations and possible different depth.

Algorithm 2 minimum spanning graph

```
1: procedure MINIMUM Algo go here
```

- Start by finding all trees with the same minimum depth for each node (Roots for different trees) related to the place in the query. After traversing the main graph we have found all subgraphs containing at least one query term. These subgraphs contain many nodes which hit the same terms. Before ranking the subgraphs the minimum spanning graph needs to be found. The minimum spanning tree is found using a greedy algorithm that iterates through all nodes in the subgraphs, then keeping the nodes containing the most terms, and lowest depth. The minimum tree will contain as many terms as possible, a term will only be found in one node, and the graph will be as shallow as possible.

- Find the minimum spanning tree for each of the root nodes containing the maximum amount of the query words. - Rank based on 1. Query words hit 2. Nodes in the tree 3. depth of the tree The final piece is to rank the minimum subgraphs. This is done using the nr. 2 formula found in [6].

Chapter 4

Optimized methods

- Alternative search methods/ algorithms.

4.0.1 Pruning

When using the BFS search method, all possible spatial vertices close to the queried place will be explored. This is expensive and many of the vertices will be irrelevant. Pruning the potential place vertices will reduce the amount of subgraphs traversed, and will in turn reduce the overall time used to find results for the query.

- predicate pruning

When traversing the graph a lot of unnecessary predicates are followed, resulting in many extra nodes added to the search. Specifying a set of predicates that contain the relevant information can greatly increase the speed of the algorithm, and also keep the memory requirements a lot lower. When selecting predicates for traversal the information expected from the search should be the top priority. Because of this, all predicates that may contain spatial or temporal data should be kept.

- unqualified place pruning.

For any spatial search, a lot of places will be found. The graph also contains information on the relationship between places. To find the best possible matches for a spatial query, only places of a higher resolution should be chosen. This means that places of similar or smaller expanse should be allows to be queried, e.g. a query of Boston can return the entirety of Boston, close to Boston, or within Boston. Massachusetts has multiple predicates linking it to Boston, but should not be queried because the information returned would be less detailed than what is queried.

- Bounding.

When selecting places or times for a query, the boundaries should be set so that they encompass the entirety of the queried time and or place.

Chapter 5

Methodology

5.1 Datasets

5.1.1 YagoFacts

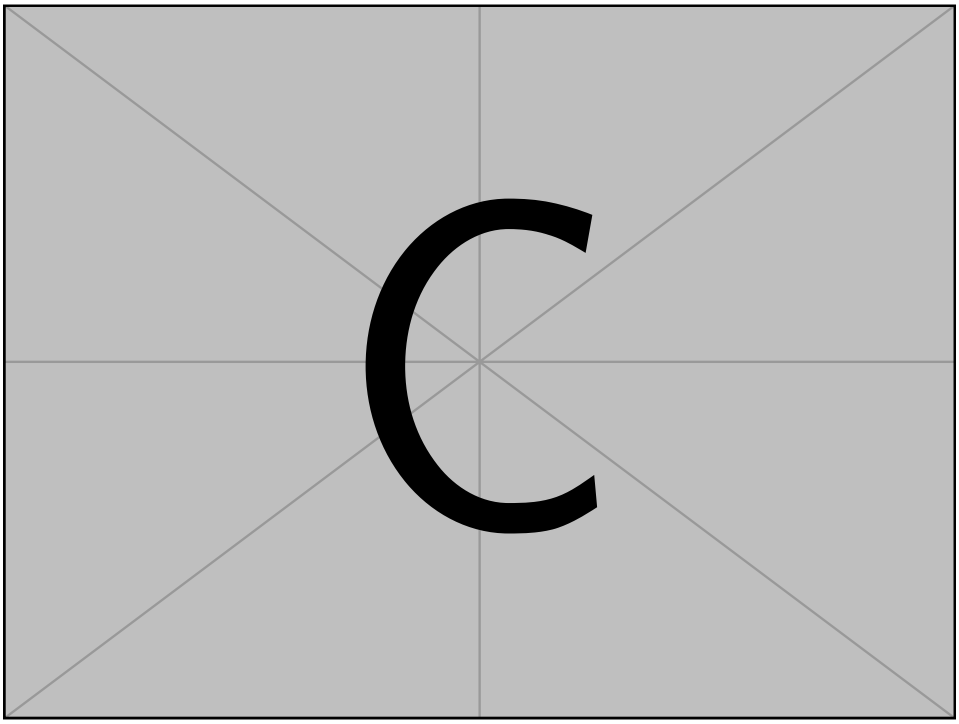
Data set containing all facts in Yago that hold between instances. This data set is the largest that is used, and holds all information.

5.1.2 GeoNames

All facts in Yago containing geographical coordinates and names. This data set contains all the place information and is used to index places and is used for the spatial indexing and used when finding a place that fits with the query or used as a starting place for spatial queries.

5.1.3 DateFacts

All facts in Yago containing dates.



5.2 Evaluation

There are multiple possible methods for evaluating the indexing and search methods. A good method for evaluating the retrieval methods is time. The faster information can be retrieved the better the retrieval method should be, assuming the information retrieved is correct. For the purpose of this theses time complexity will be the main evaluation method, along with evaluation of how well the retrieved information fits the query.

5.2.1 Time complexity

When evaluating the time complexity of a solution, a simple timer is sufficient to compare. In addition the big O notation of a solution should be described and explained.

BFS

A BFS search has the complexity of O(V+E) [**?**] meaning that we simply add the nodes and edges. The best case is O(1), meaning the query terms are found on the first node. Worst case is still O(V+E), but a worst case will not find the terms, returning a empty result.

BFS with pruning

Time complexity with pruning is the same as without pruning, but because there will be less nodes the real time taken will be lower.

5.2.2 Space complexity

BFS

BFS space-complexity = ??? [**?**]

BFS with pruning

5.2.3 Information match

All methods for retrieving information should find the same results. When ranking the results the ranking should also be the same for all methods implemented.

BFS

BFS with pruning

Chapter 6

Results

6.1 BFS search

6.1.1 Time

6.1.2 Accuracy

Chapter 7

Discussion

Chapter 8

Conclusion

Bibliography

[1]

[2] S. Elbassuoni and R. Blanco. Keyword search over rdf graphs. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 237–242, New York, NY, USA, 2011. ACM.

[3] H. He, H. Wang, J. Yang, and P. S. Yu. Blinks: Ranked keyword searches on graphs. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, SIGMOD '07, pages 305–316, New York, NY, USA, 2007. ACM.

[4] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61, 2013.

[5] F. Mahdisoltani, J. Biega, and F. M. Suchanek. YAGO3: A Knowledge Base from Multilingual Wikipedias. In *CIDR*, Asilomar, United States, Jan. 2013.

[6] J. Shi, D. Wu, and N. Mamoulis. Top-k relevant semantic place retrieval on spatial rdf data. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, pages 1977–1990, New York, NY, USA, 2016. ACM.

[7] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A Core of Semantic Knowledge. In *16th International Conference on the World Wide Web*, pages 697–706, 2007.

[8] Y. Tao, D. Papadias, and J. Sun. The tpr*-tree: An optimized spatio-temporal access method for predictive queries. In *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, VLDB '03, pages 790–801. VLDB Endowment, 2003.

[9] T. Tran, H. Wang, S. Rudolph, and P. Cimiano. Top-k exploration of query candidates for efficient keyword search on graph-shaped (rdf) data. In *2009 IEEE 25th International Conference on Data Engineering*, pages 405–416, March 2009.

[10] S. Šaltenis, C. Jensen, C. (codirector, M. Böhlen, C. Dyreson, H. Gregersen, D. Simonas, J. Skyt, G. Slivinskas, K. Torp, R. (codirector, B. Moon, M. Soo, A. Com, and A. Timeconsult. R-tree based indexing of general spatio-temporal data. 01 2000.

Appendix

Write your appendix here...