Bauhaus-Universität Weimar
Faculty of Media
Degree Programme Computer Science and Media

Story Generation from Knowledge Graphs

Master's Thesis

Patrick Saad

1. Referee: Prof. Dr. Benno Stein
2. Referee: Prof. Dr. Norbert Siegmund

Submission date: May 26, 2019

Declaration

Unless otherwise indicated in the text or references, this thesis is entirely the product of my own scholarly work.

Weimar, May 26, 2019

. . . . . .
Patrick Saad

Genius might be the ability to say a profound thing in a simple way.

Charles Bukowski

Abstract

Providing users with tools to discover and explore semantic information is a hot research topic in exploratory search. Despite advancements in tools such as faceted search interfaces, a large amount of insights is still hard to obtain for users. Users are thus required to use general purpose database query languages such as SPARQL that require sophisticated technical skills, domain-knowledge, and manual effort. In this work, we make searching knowledge graphs like searching the web. This is accomplished by generating stories from a knowledge graph which contain insights about its entities, their influence, reach, and social activity. A search interface then allows users to explore the graph by viewing and searching the stories.

Our first contribution in this thesis is building a framework for generating stories containing analysis insights from knowledge graphs. The framework contains three different components: a data querying component, a data analysis and interpretation component, and a story generation component. In our experiments, it generated 450 stories from a knowledge graph with 500,000 academic papers from the Semantic Scholar Open Research Corpus.

Our second contribution is designing and implementing a web interface to view and search the stories using a novel search approach. It provides a keyword search component that allows users to search and navigate insights about stories, graph entities, and their interconnections. Finally we evaluate our interface with 5 participants taking a Computer System Usability Questionnaire (CSUQ) usability test. Participants positively rated our interface with an overall usability score of 1.70 on a 7-point Likert scale between -3 (strongly disagree) and 3 (strongly agree).

Contents

Chapter 1

Introduction

The need to provide non-expert users with simple yet powerful query interfaces arises in complex knowledge domains. Recent research in exploratory search has been anchored on using faceted search as an interactive solution to this problem [Kules et al., 2009]. Facets are key/value attributes that represent features for records in the data (e.g. a product may have two facets: color=red and price=100). Using a faceted search interface, users progressively filter the data according to their selection of facets and their values. Several such systems such as Faceted Wikipedia Search [Hahn et al., 2010], The Relation Browser [Capra and Marchionini, 2008], Freebase Easy [Bast et al., 2014] and SemFacet [Arenas et al., 2016] have been implemented for domains with semantic structured data. An overview of these tools is provided in Chapter 2.

Ehrlinger and Wöß [2016] define knowledge graphs as large networks of entities, their semantic types, properties, and relationships between entities. Tech giants such as Google, Microsoft, Facebook and LinkedIn use knowledge graphs as part of their infrastructure to augment search results and enhance various types of AI applications such as voice assistants, chat bots, and recommendation systems. This makes the connectedeness of the data easy to understand through simple non-verbose natural language. By representing data as a graph with nodes, edges, and weights that contain semantic information, several techniques from graph theory and network analysis can be applied to better understand the data. However, this information cannot be obtained from faceted search interfaces. Therefore, these untapped insights demand for more research on exploratory search applied to knowledge graphs.

One solution would be to create facets for all the inferred information. However, research has shown that providing users with facets can be overwhelming [Sinha and Karger, 2005]. When given too many filter options, the possible exploration paths increase tremendously, especially due to the total possible

combinations of facets and their values. As an example, only two facets with three possible values each have a maximum number of possible facet selections of $\binom{6}{2} = 15$. Scaling that concept to a large property model with multiple entities each containing a collection of facets with their possible values creates an overchoice effect [Gourville and Soman, 2005] and greater physical effort to reformulate queries for the available choices [Koren et al., 2008]. This problem can be solved by applying data analysis and data visualization to interpret all the possible search results for implicit insights. However, this approach is highly difficult for non-expert users, and requires time-consuming demanding work for expert users.
In this thesis, we make searching knowledge graphs like searching the web. A story generation framework is implemented with the goal of generating stories containing insights from a knowledge graph. It communicates with the graph using query languages, applies analysis techniques and statistical methods on the query results, and produces stories from story templates. In Chapter 3.1, our knowledge graph setup based on a subset of the Semantic Scholar Open Research Corpus [Ammar et al., 2018] is explained. In Chapter 3.2, methods for insight discovery based on social network analysis and graph theory are discussed. These techniques result in metrics that are used to reveal implicit information about entities in the graph. In Chapter 3.3, examples of story templates and they communicate insights to users in a clear manner are shown. For example, a story entitled "The Most Collaborative Authors of 2018" would contain a statistical summary as well as a list of top entities with regards to author collaborations in that year.

Another contribution of this thesis is creating a web interface to view and search the stories. Chapter 4 showcases the interface which utilizes a search approach that connects entity queries to stories containing insights and not to documents. Search results for graph entities retrieve information about the entity itself in a knowledge box, as well as a list of insight stories where the entity is analyzed and ranked. This can provide insights about queried entities that are otherwise too difficult to formulate for the casual user and not obtained from faceted search interfaces.

In Chapter 5, we evaluate our interface using a Computer System Usability Questionnaire (CSUQ) with 5 participants who are experts in the knowledge domain used in our experiments. The CSUQ contains 19 questions where participants answer every question on a 7-point Likert scale ranging from -3 (worst) to 3 (best) with 0 representing neutral feedback. Participants positively scored all usability aspects of the study, with a highest score of 1.70 for the overall usability of the system.

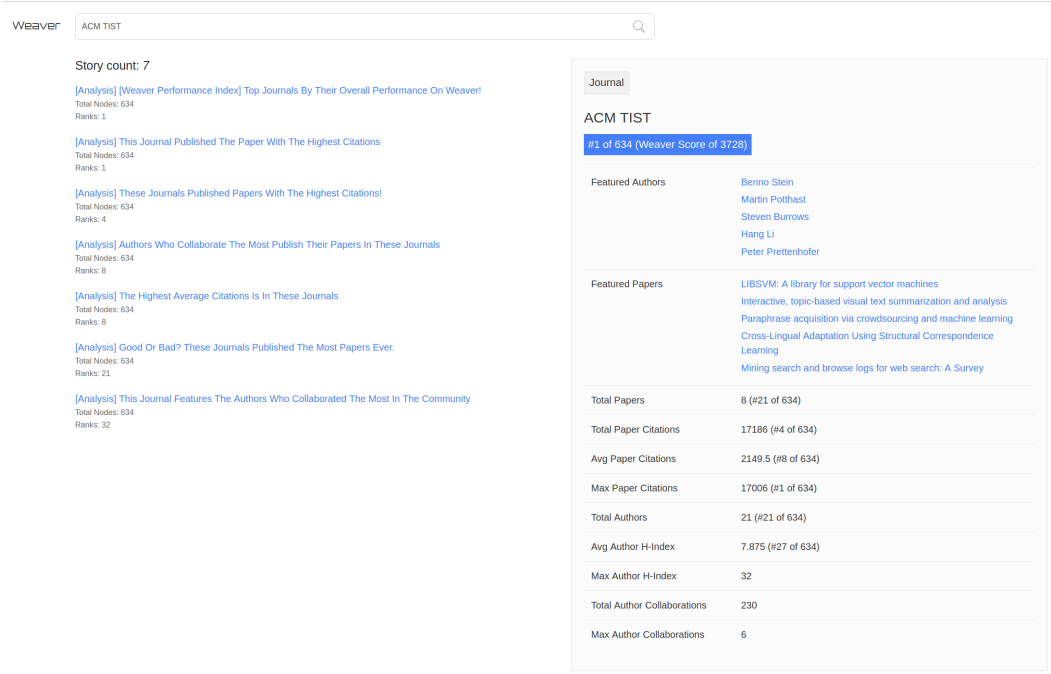

Figure 1.1: Our Weaver tool showcasing the search results for a Journal entity. On the left, the interface shows stories where the searched entity is ranked with regards to the analyzed facet(s) in each story. On the right, a knowledge box displays further insights about the queried entity such as its most influential connected entities, its total performance score, and its individual performance score for different facets - source: https://weaver.webis.de

Chapter 2

Related Work

Our approach in this work is related to research in fields such as exploratory search, distant reading, social network analysis, and automated journalism. We build our approach based on concepts, system architectures and approaches from these fields to generate stories out of knowledge graphs, and to communicate the results to end users using a search interface.

2.1 Exploratory Search

Exploratory search (ES) differs from traditional known-item search where a user wants to resolve a specific information need. It assumes both unfamiliarity with the domain and the lack of a clear information goal. ES systems provide users with a set of tools to better filter, search, visualize, and understand the data in a top to bottom manner.

One of the key motivations of ES is to understand the scenarios where traditional keyword-search or query language functionality are not sufficient to assist users in exploring large data. One of these scenarios is exploring knowledge graphs, where semantic information is hidden in the form of nodes, attributes, and relationships connecting these nodes. Navigating this complex information spaces presents big challenges to users, such as learning and applying query languages. To solve this problem, the concept of interactive search interfaces, and specifically faceted search interfaces (FSI) was proposed. Facets are metadata that provide hierarchical categories for documents in the information space. As an example, a dataset containing documents on clothing products can have facets such as manufacturer, type, price, and color. An FCI provides users with easy and interactive query refinement by providing facets which can be combined to filter the data, and satisfy information needs. Hearst et al. [2002] showed that users found flexible query modification using

faceted hierarchies more intuitive and easy to use than keyword-search. One of the problems FCI systems face are document domains with a large number of facets. Proposed solutions involve selecting subsets of facets, either ranked by alphabetical order or by their frequency in the document domain. Koren et al. [2008] proposed the use of a personalized interactive faceted search mechanism where facets and their values are automatically created based on the user's activity and preferences. Ruotsalo et al. [2013] also worked in a similar direction with predictive user modeling or interactive intent; a user's search intents are predicted and the interface can provide suggestions of possible exploratory directions based on previous actions by the user.

In this thesis, we build a tool to provide users with an ES approach to obtain insights from knowledge graphs. There exists several of such tools that use new research or increased user feedback to improve the exploratory experience. The following is a overview about some of these exploratory search tools.

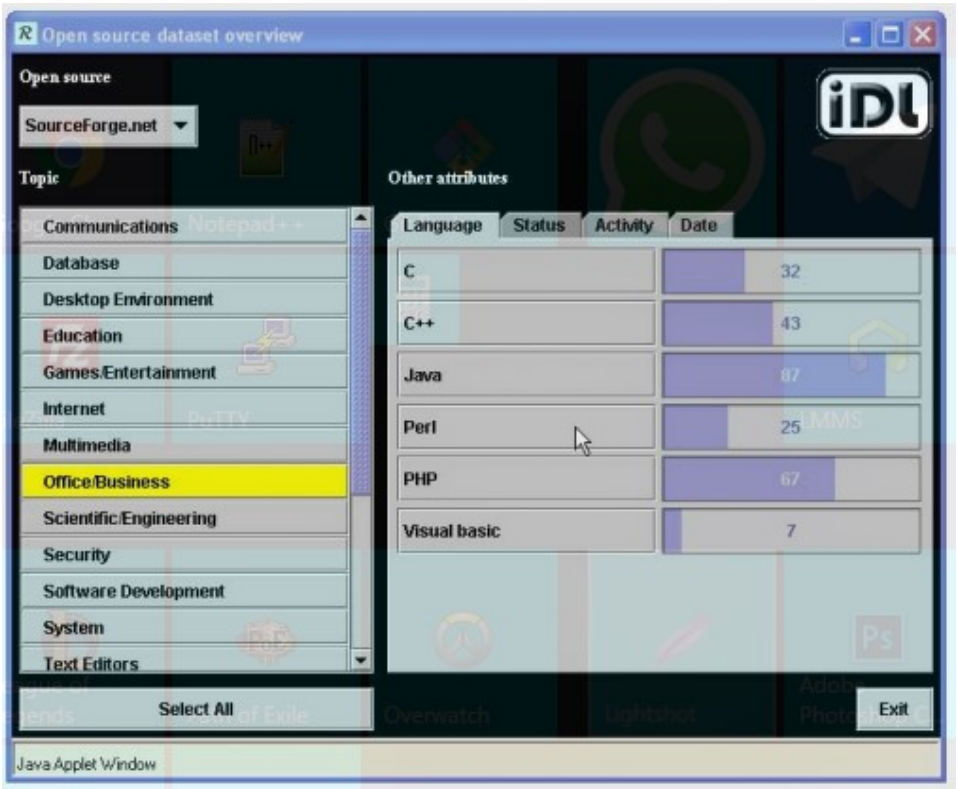

Figure 2.1: The Relation Browser (source: https://www.researchgate.net/figure/a-Relation-Browser-RAVE-after-clicking-on-Office-Business_fig3_2830968).

Figure 2.1 shows the Relation Browser (RB), a tool whose aim is to pro-

vide users with better understanding of complex data (i.e. data in knowledge graphs). This snapshot of one version of RB showcases the function of filtering topics according to their attributes. Influenced by the dynamic query perspective, the interface provides users with a direct manipulation control mechanism over the data's attributes and relationships. The aim is to provide slides of the data as a form of assisted suggestions which allow users to decide whether to search further into the selected slice. Optimal use cases for the RB consist of exploring sub-collections of very large databases, instead of individual items. This is a common goal in the fields of exploratory search and distant reading, as insights from aggregated data allow better understanding of the data than looking up individual entities.

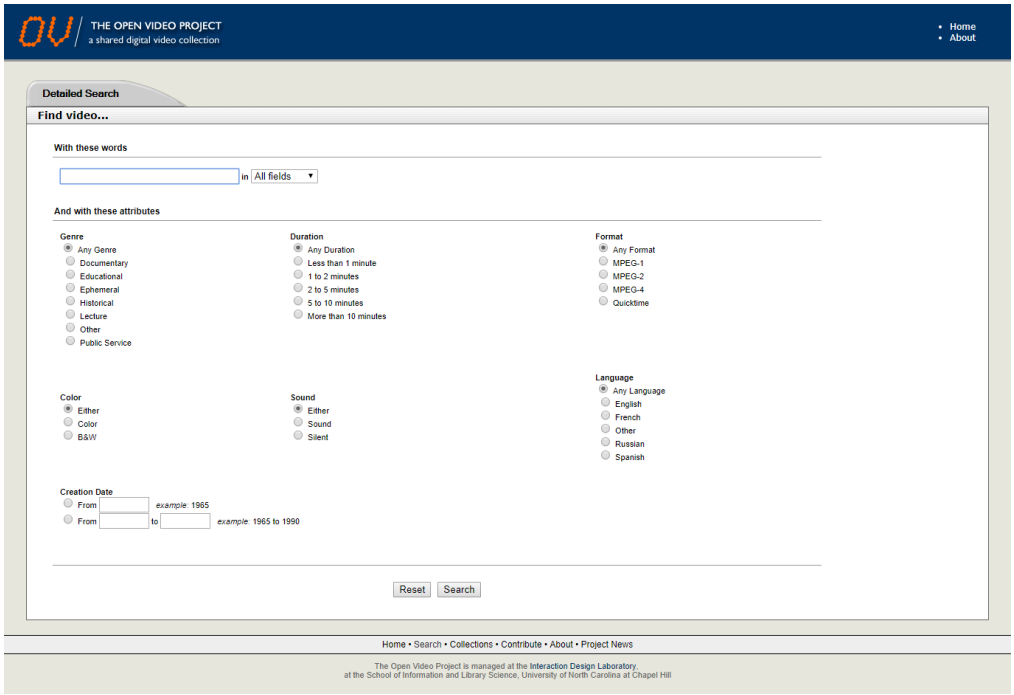

Figure 2.2: The Open Video Project (source: https://open-video.org).

Figure 2.2 shows The Open Video Project, another tool that follows the aforementioned faceted search concepts. Its interface presents metadata clusters such as genre (documentaries, educational, lectures, etc), duration (less than a minute, 1-2 minutes, etc), color (black and white, color), etc. The clusters also show users the total number of entries filtered by every option. Full text search of bibliographic records is also provided via keyword search. After the initial filtering action, users can again apply faceted search filters on the results.

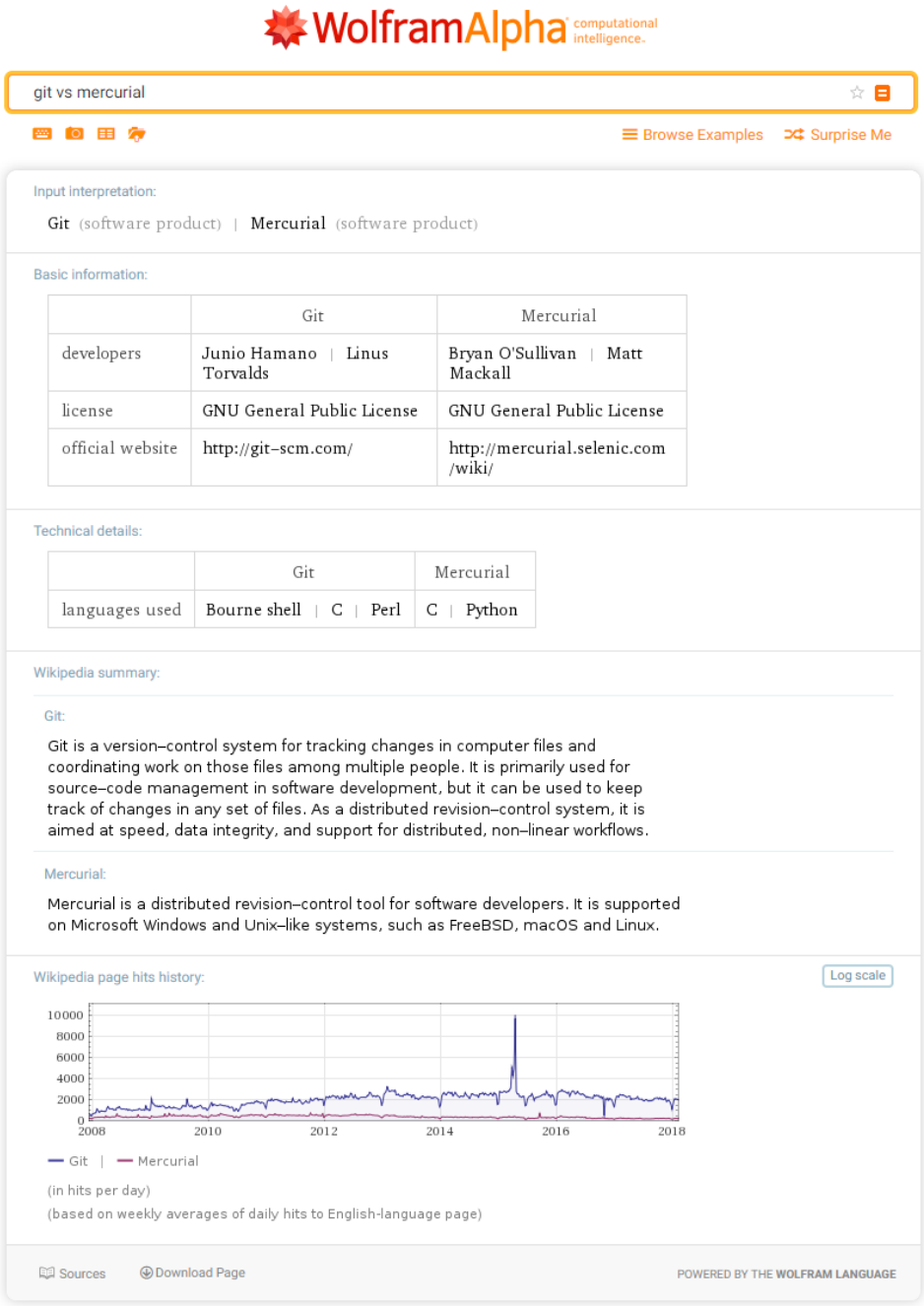

Figure 2.3: Wolfram|Alpha (source: https://www.wolframalpha.com)

Figure 2.3 shows Wolfram|Alpha, an online computational knowledge engine that uses data aggregation to provide users with query results in form of

insight stories. It answers factual queries with a story containing a summary of analysis results with statistics, timeseries graphs, entity definitions, etc.. This approach is different from traditional exploratory search retrieval methods that provide a filtered list of relevant documents to a search query. As one of its functions, this tool provides users connected via their Facebook accounts with a personalized story containing insights about their social activity, connections, and other social network analysis related insights. Statistics, data analysis results, and data visualizations such as plots and network subgraphs snapshots were used to communicate the insights in the stories.

Our approach is inspired from exploratory search in that it assumes no knowledge of the domain, and no prior information goals. Users want to understand the discourse in the knowledge graph by exploring its remarkable actors, communities, and activities. Exploratory search provides users with an interface that encapsulates query languages, which makes it easier to communicate with the knowledge graph without writing any queries, and learning the query language for the graph database in use. This encapsulates a layer of difficulty for users, making it easier to communicate with the graph using interfaces with advanced filtering such as faceted search. This new layer which goes on top of the query language layer is essential when dealing with user interfaces, replacing the required technical knowledge of directly interfacing with a knowledge graph.

When it comes to insights, faceted search serves only an initial step to analyzing knowledge graphs. In a typical scenario, users filter the data with their faceted search selections, and see the filtered documents according to their faceted query. Besides the immediate information need that users request via their selections, the returned query results can hold hidden insights that require additional data analysis or statistical computation.

Given a knowledge graph with a large amount of attributes and node relationship types, the possible faceted search combinations that users can explore are exponential. This creates a problem of having too many exploration paths to follow, and a bigger problem of not knowing whether the current facet combination would return interesting results. In our approach, we want to eliminate all this costly repetitive work from the user's side. With the increase in computing power, machines are more capable of cycling through all possible facet combinations, and querying the knowledge graph for results from these combinations. A machine could also apply data analysis and data interpretation on these results. Given that users will find it hard and costly to perform all of these analysis and interpretation functions, we propose that a framework can automate and encapsulate that functionality. A system can be programmed to perform the same exploratory search steps as human analysts do, and report

insights according to some criteria of what is considered interesting.

Similar to most exploratory search interfaces, we add keyword-item search capability. However, our query matches will show stories instead of documents that are relevant to the query. For example, searching for an author's name would not reveal an author's profile page or a list of papers where they appear, but rather a list of stories containing relevant insights about the author. We believe that this search approach allows for an improved exploration of the data for users with or without an information need.

To summarize, exploratory search is a major related work as our system. Our work proposes to provide users with insights based on exploratory analysis concepts. This means users can then explore insights from an automated exploratory search process. Our work differs from the research covered in this section in many ways: (1) we do not provide users with facet lists to filter documents with, (2) we provide users without specific information needs with stories containing insights about the data for them to explore (3) we provide an interconnected search navigation between stories, entities, and search results

2.2 Distant Reading

Distant reading is a paradigm in the field of digital humanities which analyzes and aggregates data to reveal insight in the form of maps, trees, and graphs [**?**]. Boot [2014] describes distant reading as "understanding literature not by studying particular texts, but by aggregating and analyzing massive amounts of data.". This recognizes that massive amounts of data are hard to manually explore and therefore understand. The aggregation and analysis of the data is conducted by skilled experts by describing the data when looked at as a whole. This approach is based on the fact that machines are able to analyze raw data and discover patterns significantly faster than humans. This allows for the discovery of patterns, trends, and outliers from analyzed documents. Such patterns provide insight on how the data is connected, biased, or structured.

When used with an exploratory search system, the analysis results from distant reading can potentially suggest new search facets by previously applying analysis on the data. Since this analysis was already conducted and its results interpreted to be significant, the users' exploratory search process is therefore assisted with results from a distant reading process. In our research, we significantly rely on the concept of distant-reading by aggregating and analyzing knowledge graphs with the goal to assist users in understanding the

data as a whole. Knowledge graphs are known to contain massive amounts of data with semantic information, which are difficult for humans to manually analyze using any traditional exploratory search methods. Therefore, we use distant reading as a main inspiration in the approach of this thesis.

2.3 Social Network Analysis

Social Network Analysis studies patterns of relationships that connect nodes in a social network graph [Scott, 1988]. Several metrics can be calculated to classify, group, and evaluate the importance or influence of nodes in a network, as well as overall characteristics of the network itself. In our research, we apply algorithms such as PageRank and Betweenness Centrality on a knowledge graph to find social network related insights. We use the relationships between nodes in knowledge graphs to obtain complex insights about the nodes, the activity, and the collaboration patterns in our graph. We use concepts from social network analysis to extract insights about communities as well as individual entities in the graph. In our experiments, these insights are highly important for understanding the community, its activity patterns, and the top performing entities.

Network analysis is a rapidly growing field, and there are now a number of libraries available that provide a wide range of analytical tools. The methodology in these packages falls into three general classes: descriptive techniques, permutation methods, and generative models. The classes range roughly along a continuum, from capturing static regularities in network structure to testing models for the emergence of that structure.

Based on descriptive techniques by [**?**], many tools and libraries provide access to insights from networks. Packages like UCINET [1], Pajek [2], and statnet [3] perform statistical inference associated to methods from social network analysis.

2.4 Automated Journalism

According to Carlson [2015], automated journalism are natural language generation (NLG) systems that transform data into specifically news text. News domains that deal with structured data are used as use cases for automated journalism. Examples of such cases are stock reports (ANA Kukich [1983]),

[1] http://www.analytictech.com/
[2] http://vlado.fmf.uni-lj.si/pub/networks/pajek/
[3] http://www.statnet.org/

weather forecasting (Sripada et al. [2004]), and sports and finance (Nesterenko [2016]). Earlier works of data-to-text are based on NLG architectures that consist of several modules; numeric data is processed for patterns, data interpretation converts these patterns into messages, document planning filters the messages to be mentioned, and micro-planning handles how to express those messages in concise texts. However, these systems need to meet several requirements to be used in the domain of news journalism. Leppänen et al. [2017] et al. identified six such requirements that are important in journalism and must be reflected in journalistic NLG: transparency, accuracy, modifiability and transferability of the system, fluency of output, data availability, and topicality of news. They used a templating language that consists of sentences with slots filled by information from facts. Figure 2.4 shows the automated stories generated by their NLG news generation bot Valtteri. Their NLG system generated over 750 000 web articles on the 2017 Finnish Municipal election results. Their interface allows search queries based on an entity and a location. This allows users to filter facts based on these selections, acting as a relevance filter. Their system sorts the stories by newsworthiness, with every fact in a story having a newsworthiness score. For every story, the system adds facts until either reaching 5 facts, or until the newsworthiness score of the next fact candidate reaches below 20 percent of the most newsworthy fact in the story.

In our work, we build a framework that automates the communication of insights from knowledge graphs. However, we focus on providing an interconnected exploratory search experience for end users. We use an information retrieval approach with stories as the retrieved documents to entity search. We rank the performance of entities in all stories and provide an interconnected exploratory search experience for users. Our goal is to provide explicit insights about the most relevant entities in a knowledge graph based on concepts and metrics from social network analysis and graph theory.

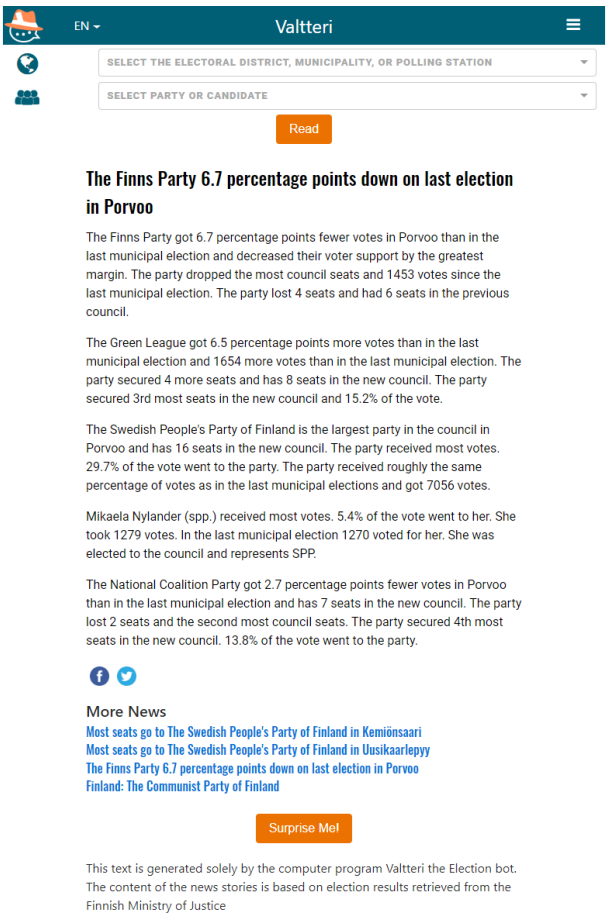

Figure 2.4: The web interface for Valtteri, a news generating system that produced over 750 000 news articles on the 2017 Finnish Municipal election results - source: https://www.vaalibotti.fi

Chapter 3

Story Generation Framework

We develop a story generation framework that transforms a graph database into an index of stories containing analysis insights (see Figure 3.1). The framework contains several components, starting with setting up a knowledge graph, discovering insights, generation stories with these insights, and indexing the stories. In the following sections, we discuss the framework components and their procedures and experiments. In Chapter 3.1, we discuss our knowledge graph setup using Neo4j[1] as our graph database, and how the knowledge graph was created from a subset of the Semantic Scholar Open Research Corpus [2] (SSORC) of scientific publications in Computer Science.

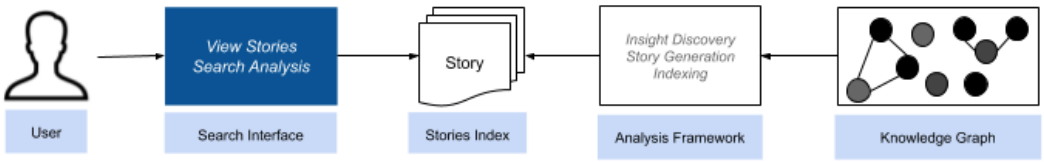

Figure 3.1: The story generation framework generates stories from a knowledge graph.

In Chapter 3.2, the insight discovery component uses Neo4j's query language Cypher [3] to query the direct and indirect relationships of nodes in the graph for insights based on concepts from social network analysis and graph theory. These query results are stored as facets, which are then analyzed for insights about entities in the graph. In Chapter 3.3, the story generation

[1] http://neo4j.com
[2] https://api.semanticscholar.org/corpus/
[3] https://neo4j.com/developer/cypher-query-language/

component analyzes entities' graph activity and performance based on these facets. Summary statistics, top performing entities, trend detection, and outlier detection techniques are used to rank entities in the graph with regards to these facets. A templating approach is implemented for the story content, which uses these aforementioned analysis methods to fill slots containing HTML text, images, plots, and tables. In Chapter 3.4, we index the stories, entity information, and entity performance ranks in the stories. This index is used in an exploratory search tool which provides users with access to insights about entities in the knowledge graph.

3.1 Knowledge Graph Setup

One of the aims of this thesis is to assist users in understanding large amounts of data. To achieve that, we take advantage of the semantically rich data representations provided by knowledge graphs. For our experiments, the SSORC corpus containing 39 million published research papers in Computer Science, Neuroscience, and Biomedical is used. The framework runs on a subset of SSORC containing 500,000 papers due to time and resources limitations. The final generated stories and the interface feature a main subset of 4488 paper records. However, the experiments require all 500,000 records as they form the directly connected nodes to the main subset that is used in the insight discovery component. Requirements for working with bigger knowledge graphs are discussed in Chapter 6.2.

For the paper selection process, using our domain knowledge, we manually chose an influential seed author whom we have direct access to, and fetch his 258 papers. This author selection was driven by our usability study in Chapter 5, where we target users who are directly involved in the knowledge graph we built. These users could give better feedback about the novelty and interestingness of the insights using their domain knowledge.

After selecting the initial paper subset P1, we then fetch the paper subset P2 which includes all the connected papers (incoming and outgoing citations) to P1. P2 contained 4488 papers in total, an increase of 1527% from P1. We repeat the process one more time on P2, obtaining the final paper set P3 which contains around 500,000 papers. We note that our search interface and stories feature papers from only the P2 set, even though our knowledge graph contains all P3 papers. This is due to our insight discovery component (Chapter 3.2) which requires an additional level of connected papers (P3) to analyze the nodes in P2.

After obtaining all the records, we setup a graph database using Neo4j. We constructed graph queries using Neo4j's Cypher query language to insert

these records into our knowledge graph. Cypher provides an expressive way to write queries using ASCII-Art syntax. We follow graph database concepts to prepare a graph model that describes our data. Figure 3.2 shows the graph model we used in our experiments. The labels, relationships, and facets we used organize the data into complex inter-connected structures.

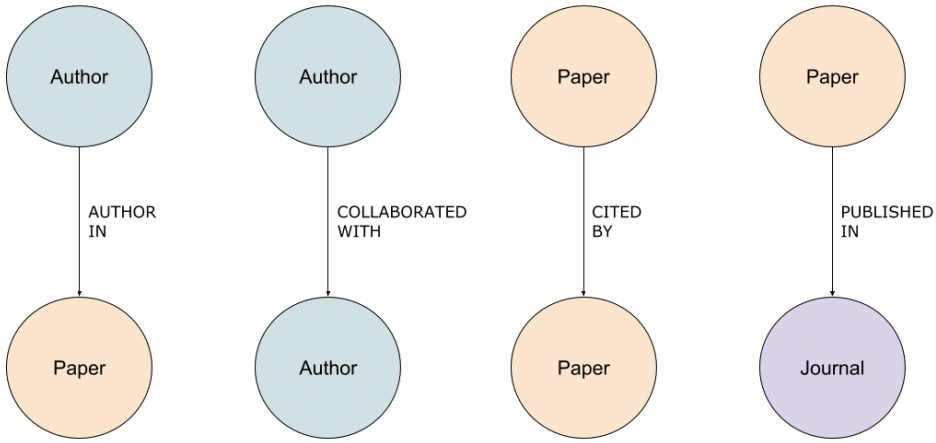

Figure 3.2: Our graph model showing entity types and their relationships.

The following are two examples of Cypher insert queries we used to construct nodes with their labels, relationships, and facets.

(1) Insert a node having a label and some properties (facets)

CREATE (p:Paper {paperId: "1", title: "The Probabilistic Relevance Framework: BM25 and Beyond", year: 2009})

(2) Insert a relationship between two nodes

MATCH (a:Paper paperId: "1")
MATCH (b:Paper paperId: "2")
MERGE (a)-[:CITED_BY]->(b)

We used Neo4j's Python driver [4] to process the JSON records from the original corpus, and use Cypher queries to insert nodes with their labels, relationships, and facets into the graph. Table 3.1 shows the node totals by their labels in our knowledge graph.

[4]https://neo4j.com/developer/python/

Paper	Author	Journal
4488	8124	634

Table 3.1: Our knowledge graph node totals by label

Figure 3.3 shows a snapshot from the knowledge graph with nodes and their relationships.

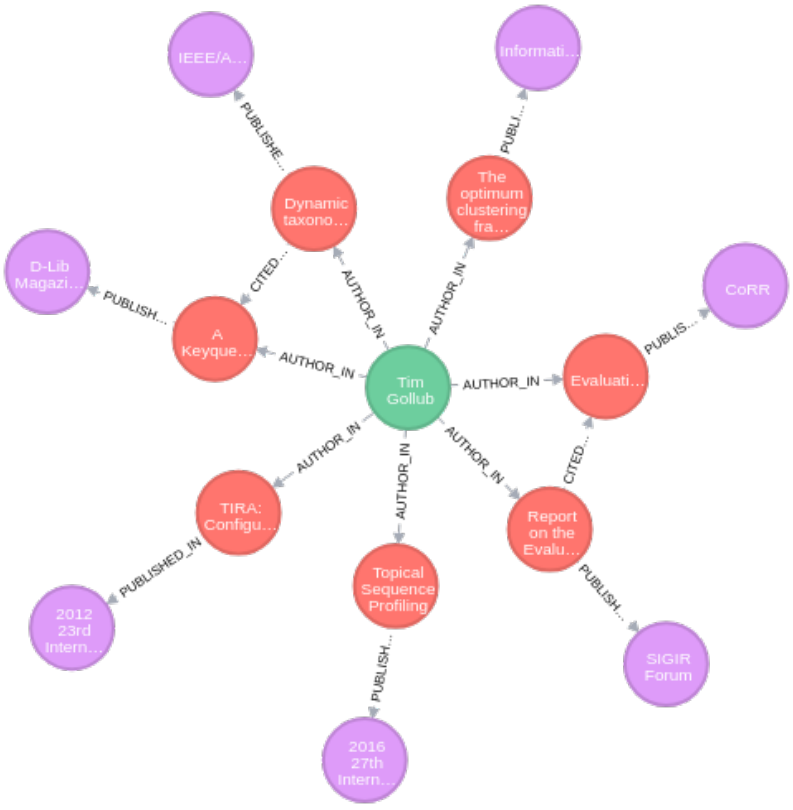

Figure 3.3: A snapshot from the knowledge graph. Node labels are color coded (red for Paper, green for Author, and violet for Journal).

For our experiments, we did not assign weights or properties to relationships themselves, and we avoided duplicate relationships in opposite directions.

As a summary, we extract a subset of records from the Semantic Scholar Open Research Corpus starting from an influential author whose community we have direct access to, based on a usability study decision which is explained in Chapter 5. We design a graph model based on the available data attributes and assisted by our domain knowledge. We use Cypher queries to insert the records into a graph database as nodes with labels, relationships, and facets.

3.2 Insight Discovery

The insight discovery component aims to process the nodes, their relationships, and their facets to generate metrics that can be used as insights in the story generation component. The framework stores these metrics in the graph itself as additional facets. In the following, the concepts and algorithms used to process the graph and extract these implicit insights are explained.

Our knowledge graph contains a complex structure of nodes, relationships, and facets. Relationships between nodes refers to explicit semantic information about their interaction. Concepts from social network analysis and graph theory that measure the relationships between entities in the graph were first applied. These metrics were then stored in the graph itself as facets for nodes. These facets represent insights about the nodes' impact, reach, and activity in the graph.

Many metrics could be generated for nodes in knowledge graphs, such as PageRank, Betweeness Centrality, Harmonic Centrality, Closeness, etc. In our work, we generated some of these metrics as well as some from our own interpretation of graph and network theory. For example, Neo4j's graph algorithms library [5] was used, which contains optimized ready-to-use algorithms to produce a PageRank facet for Paper nodes. Cypher queries were constructed to generate other insights from the graph based on concepts of direct and indirect relationships and facet inheriting. Examples of such facets include the "incoming nested citations" fact for Paper nodes, which contains the total number of indirect outgoing relationships of type CITED_BY between papers. The generated facets can be categorized by the following methods:

A- Count the total direct relationships of every node

In Figure 3.3, our graph model with all relationships and their directions between different node labels is shown. Therefore every node in the graph will have at least one relationship to other nodes. Queries were constructed to count all the different relationships connected to each node in the graph, and the resulting metrics were inserted as facets for the nodes. This approach can provide a different single facet for each two nodes that are connected by a relationship. Algorithm 3.1 shows our approach.

Table 3.2 shows the facets generated by this approach. We decided to remove one of these facets which describes how many journals every paper is connected to. This is because the type of relationship between Paper and Journal nodes is a many-to-one relationship. Thus inserting this facet for

[5] https://neo4j.com/docs/graph-algorithms

Algorithm 3.1: Generating facets using method A

```
Input: label
Output: facets
if the label has at least one relationship and one node then
    foreach relationship in label.relationships do
        Get the label name lc of the node connected by this relationship;
        foreach node in label.nodes do
            Count the total number of connections T between this node
             and lc nodes ;
            Insert a new facet for the node with T as the facet value
        end
    end
end
```

papers would have the values 0 (a paper is not connected to any journal) or 1 (a paper is connected to 1 journal). For our experiments, this provides insignificant insights, since all papers in our data were published in a journal.

Facets generated by method A for every label		
Paper	Author	Journal
Total Paper Citations	Total Papers	Total Papers
Total Authors	Total Author Collaborations	

Table 3.2: Facets from the facet generation method A

The insights revealed by this method are relatively easy for users to obtain using faceted search interface. In our data domain, users use the total citations of papers for reputation assessment. Other facets are less familiar to users, such as author's total collaborations or the total papers published in a journal. Using this simple method, the most basic and intuitive insights about nodes in the graph are revealed.

B- Aggregating facets values from directly connected nodes

After obtaining the facets from method A, a concept of facet inheritance to generate additional facets in our graph was used. This approach applies for nodes connected by many-to-many relationships. The direction of the relationship between these nodes is important as well. Facet inheritance is only used for nodes that are on the receiving end of a relationship direction. Algorithm

3.2 shows our approach.

Algorithm 3.2: Generating facets using method B

```
Input: label
Output: facets
if the label has at least one relationship with an incoming direction and
 one node then
    foreach relationship in label.relationships do
        Get the label name lc of the node connected by this relationship;
        foreach facet in lc.facets do
            foreach node in label.nodes do
                Get the connected nodes by the relationship;
                Get their facet values for the current facet;
                Compute the total, average, minimum, and maximum of
                 these values;
                Insert new facets for the node with these aggregated
                 values;
            end
        end
    end
end
```

As an example, authors are connected to papers via an outgoing relationship. We use this relationship information to generate facets for papers and authors. These facets use an aggregation of facet values from all connected nodes. A paper connected to many authors can be assigned facets whose values represent the cumulative total, average, maximum, or minimum of any facet of these authors. Vice versa, an author connected to many papers can be assigned such facets by applying the same operations on any facet of these papers. Figure 3.4 shows how facets can be generated from other facets of connected nodes of type many-to-many or one-to-many.

Table 3.3 shows the facets generated by this approach. We note that we did not generate all possible facets using this method. For example, an additional facet for authors could be generated from their papers' PageRank facet. An author can be assigned facets that describe the maximum, minimum, or mean of all their papers' PageRank values. Similarly, a journal could be assigned a facet that communicates the average PageRank of its papers.

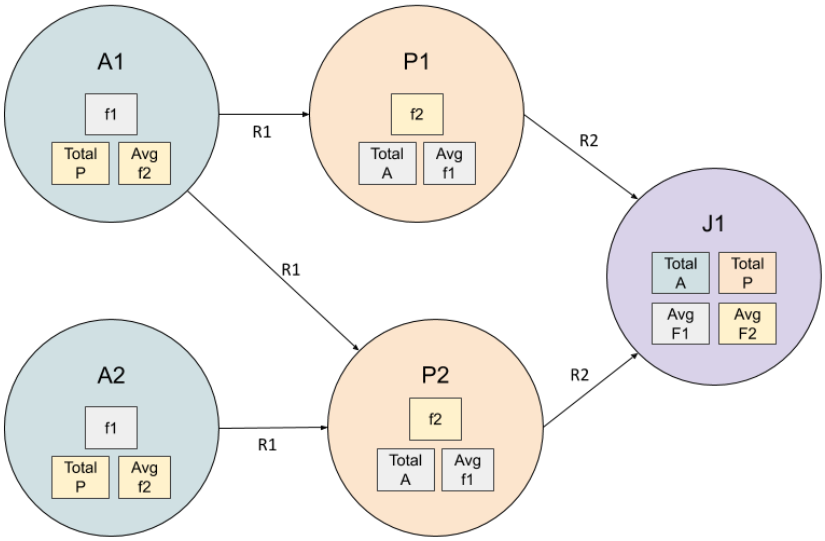

Figure 3.4: An example of many-to-many node connections. Methods A and B are applied on these nodes to generate facets.

Facets generated by method B for every label		
Paper	Author	Journal
Average Author h-index Maximum Author h-index	Maximum Paper Citations	Average Paper Citations

Table 3.3: Facets from the facet generation method B.

C- Measuring the total indirect relationships of a node

Indirect connections between two nodes are normally associated to a "friends of friends" social relationship. However, they can extend up to multiple degrees of separation (friends of friends' friends' friends). In our work, we generate facets that describe these connections as major hidden insights in knowledge graphs. These types of insights are difficult to obtain using human cognition alone as well as with state of the art search interfaces. The facets that are generated by this approach rely heavily on such relationships.

In Figure 3.4, the node J1 is indirectly connected to nodes A1 and A2 via their direct connections to nodes P1 and P2. These indirect connections are used to count the total distinct nodes of type A that are connected to J1. Since the nodes of type P could have common A nodes, queries with the distinct keyword were constructed to eliminate all duplicate connections to

the same nodes. This gives us precise numbers for J1 as to how many unique indirect connections to nodes of type A it has. Using this concept, we also provide facets to nodes such as J1 based on the aggregation of facets from indirectly connected nodes. Table 3.4 shows all facets we generate with this method:

Facets generated by method C for every label
Paper
Total Authors Collaboration
Total Nested Citations
Total Authors Papers
Author
Total Paper Citations
Total Nested Author Collaborations
Journal
Total Paper Citations
Total Authors
Total Author Collaborations
Average Author h-Index
Maximum Author h-Index

Table 3.4: Facets from the facet generation method C.

As a summary, all these facets provide significant insights that were previously hard to obtain for users using exploratory search interfaces. For example, we generate facets that show the average and total paper citations, the average and maximum author h-index, and the total distinct author collaboration for every journal in our graph. In the next component in the framework, this information is used to rank all journals by every one of these facets, giving users several angles to assessing the quality, activity, and influence of these journals in the community. From this approach, we also generated facets for papers such as the total papers of their authors, which users could use to understand which papers are written by highly active authors in the community. Also, the total nested citations of a paper will reveal its influence and reach in the community, rather than its direct connections which is mostly used to judge a

paper's quality or importance. Also, the total author collaboration in a paper reveals how collaborative its authors are, thus providing insights about papers published by authors who adopt a teamwork mentality in their activity in the community. The facets we generated for authors using this approach provide insights as to their influence and importance in the graph. Their unique total paper citations describes how influential their papers were in the graph. Their total nested author collaborations facet reveals how engaged they are in their community of authors.

In the next component, we provide a descriptive summary for all the facets we generate, and we rank the nodes in the graph by every facet, and generate stories containing these insights. We discuss more ideas for facets based on social network analysis in the future work section.

3.3 Story Generation

In the previous components of our story generation framework, we constructed a knowledge graph and used concepts from social network analysis to generate different facets for different entity types. These generated facets describe implicit insights about the connectedness, influence, and reach of entities in the graph. The story generation component analyzes these facets and uses the analysis results as content for different story templates.

Stories should find and verify important or interesting information and present it in an engaging way (The American Press Institute [6]). In journalism, good stories exhibit reportorial effort, verified information from experts and target a topic that is relevant or significant to readers. We use these definitions to establish a guideline to what kind of stories we can generate from knowledge graphs. In Chapter 3.2, we generated facets that describe several aspects of influence, reach, and behavior of entities in the graph. Therefore, we define our stories as documents that contain insights with regards to a single or multiple numerical facets.

The story generation component produces stories containing insights based on the analysis of a single numerical facet or a combination of 2 or more facets. For our experiments, we generated 4 types of stories:

1) Numerical facet analysis

For each entity type, we produce a story for every one of its numerical facets. Templates for these stories contain descriptive statistics, entity rank-

[6]https://www.americanpressinstitute.org/journalism-essentials/makes-good-story/

ing, and timeseries with statistics and outlier detection.

2) Numerical facet correlation analysis

For each entity type, we produced a story for every 2-combination of its numerical facets. These stories provide an overview about any dependency, correlation, or pattern between two numerical facets. Templates for these stories contain correlation analysis results from applying Pearson's r on the 2-combinations. We use the Pearson correlation coefficient to discover correlation insight from two sets of univariate numerical data. In our experiments, we set a binomial coefficient $\mathbf{r} = \mathbf{2}$, which return combinations of 2 attributes. The story content would communicate the correlation results both as text (correlated, no correlation, unpredictable) and a graph plot. These types of stories would give end users insight in how attributes influence each others' performance in the graph.

3) Time-filtered numerical facet analysis

We generated these stories for entity types having facet(s) of type time (e.g. Year). Stories that communicate insights for a specific time unit (i.e 2018, 2019, Today, etc.) are highly used in journalism. For our experiments, these time-filtered stories provide the largest amount of stories. For every time facet, we generate a story for every combination of a numerical facet with every facet value of a time facet. Templates for these stories include descriptive statistics and entity ranking.

Our graph contains a time facet for paper entities whose values communicate the year a paper was published. The total distinct number of year values in our graph is 56. Therefore, the time-filtered numerical facet analysis stories for entities of type Paper contain a total of 56 distinct time values * 8 numerical facets = 448 stories. Instead of generating one story which contains analysis for every value of these 56 time values, we opted to generate separate stories for each time value. We found that this is a good journalistic decision since it is highly common to find journalistic stories on the web that target a single year (i.e. Top Authors of 2019, Top Papers of 2018, Top Journals of 2017).

4) Weaver performance analysis

For every entity type, we generated one of this story type which communicates the global performance and influence of entities from the entity type in the graph. This story type represents the aggregation of all individual entity

performance insights that are produced in numerical facet analysis stories.

In table 3.5, we show the total number of stories generated by our story generation framework grouped by entity type and story type:

	Paper	Author	Journal	Total
Numerical facet analysis	8	5	9	22
Time-filtered numerical facet analysis	448	0	0	448
Numerical facet correlation analysis	28	10	36	74
Weaver performance analysis	1	1	1	3
Total	485	16	46	547

Table 3.5: Total stories by story-type for different entity types

We use quantitative data interpretation methods to communicate insights from a knowledge graph. We do not use any natural language generation (NLG) techniques to generate text as story content. Although NLG would be great to improve the story text, NLG is beyond the scope of this thesis. In our work, we use both a manual and a template approach to assigning titles to stories. The manual approach consists of a list of titles that correspond to different numerical facets, or their combinations. Examples of such titles include "The Top Authors That Influenced The Community The Most!". Since we have more than 500 stories, we did not compose manual titles for all them. For the majority of our stories, we used a template approach to assign story titles based on the entity type and the facet(a) being analyzed by the story. This automated approach produces story titles that can be harder to understand than the ones written by human journalists. While story titles are highly important to attract users' attention from a journalistic perspective, we do not focus our research on that area. We mention possible ways to generate more interesting titles in the future work section.

De Mast and Kemper [2009] argue that comprehensive exploratory data analysis would: display the data, identify the most prominent features and interpret those features. Since we do not use any NLG or data-to-text architectures to produce text for story content, we use data interpretation methods such as detecting outliers, descriptive statistics, entity ranking and variable correlation to communicate the analysis results to users. We also use peak detection to describe to users any points of interest in any graph plots we use in

story templates. Therefore we go beyond simply displaying the empirical data and actually provide users with a straightforward interpretation of the results in form of assisted automated analysis. We do not present any hypotheses, subjective statements, or text arguments about the computed insights. Therefore, our automated approach satisfies the concept of journalistic integrity and verified facts from reliable sources. Our stories are based on computationally-verifiable facts that cannot be mistaken as fake news..

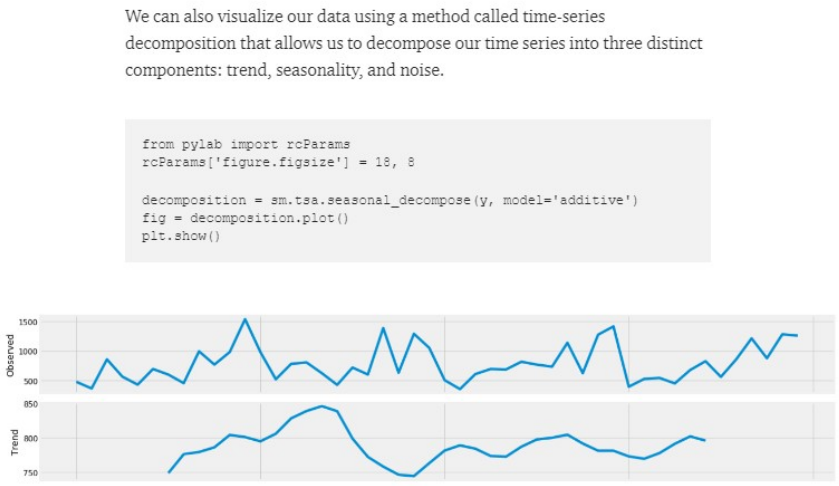

Figure 3.5: A modern data analysis article layout has a combination of text, code snippets and graphs - source: https://towardsdatascience.com

For the design and layout of the story templates, we follow common design concepts used by data analysis articles written by humans. We focus on making the stories look familiar and presentable to users. In general, data analysis stories follow a common layout as shown in Figure 3.5:

1) An introduction to the topic being covered
2) An overview of the dataset being used
3) A detailed coverage of the analysis steps and their results
4) A conclusion with summary of the results

We found that automating the first two steps in a template was straightforward; we use a description of the dataset and show the facets being analyzed. However, we do not show detailed coverage of the analysis steps that typically involve components such as code snippets.

Figure 3.6 shows an example story template that is used in our experiments.

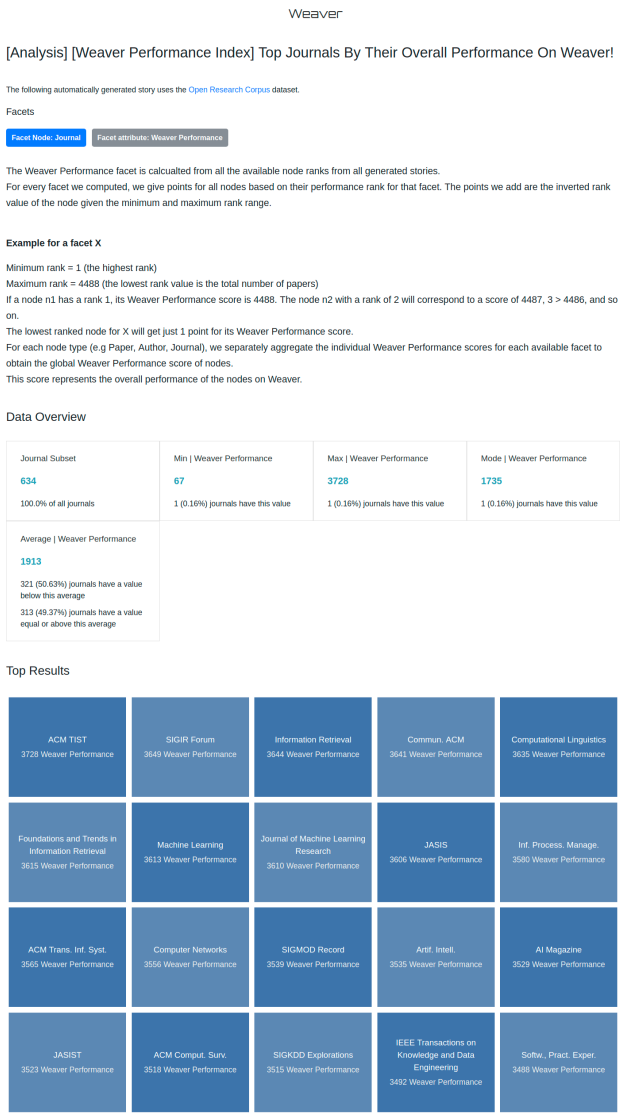

Figure 3.6: An example story template showing the layout, design, and content.

The story starts with a title and information about the dataset used by the knowledge graph, as well as the current node label (e.g. Paper) and node facet (e.g. PageRank) featured in the story. In general, the numerical facets we generated in Chapter 3.2 tackle different angles to measuring influence, reach, and activity in our knowledge graph. To explain complex insights in a story, a section with a description of the method used to generate the analyzed facet is provided.
The main data overview in the template comes from a statistical overview that

communicates statistics about the analyzed facet such as its mean, mode, minimum, and maximum values. The number and percentage of entities related to these statistics are also shown. For example, for a numerical facet F, the total number of entities N having a facet value below or equal to the mean value or above the mean value of F is shown. These components form both the introduction of the topic being covered as well as a detailed overview of the dataset being used. The final section in this template summarizes the main insights in the story. These insights are shown as a list of top entities ranked by their performance in the story. For each entity, its display name as well as its facet value for the facet analyzed in the story are shown. These entities are hyperlinked to their respective search results pages, creating an interconnected navigation experience between stories, entities, and the entities' search results.

Note that a big part of what makes a good story is how clear it is presented to users. The layout, presentation and content of the stories are highly significant to the overall story quality. Therefore any automation of a good human journalist would need to use a wide collection of visualization and presentation tools. Coupled with NLG techniques, stories based on scientific analysis can also use visual communication techniques in fields such as data visualization, digital humanities, and graph network visualizations ([Kosara and Mackinlay, 2013]).

In our search interface, graph plots are used for such visually rich elements, with a focus to providing clear and easy-to-use navigation between the entities, the search results, and the stories. This is because our search interface is based on the ranks of entities inside stories, which explain how they perform with regards to each facet, and thus their overall significance in the graph.

For the flow of the stories, an author-driven approach is used for the story's narration as described by [Segel and Heer, 2010]. This means our stories use a linear path to first provide an overview of the data analysis numbers, then provide the results for top performing nodes. Our automated stories are not interactive, but rely heavily on efficient communication. We use static visualizations, but we believe our framework could also incorporate embedded interactive visualizations to highlight specific insights and to allow the user to view changes over time such as in timeseries graphs. Segel et al. also point out that users could be provided with free exploration visualizations. We believe these could be incorporated into our framework as graph visualizations of the entities highlighted in the story as well as their immediate connected surrounding. We discuss this further in the future works section.

3.4 Indexing

The indexing component is responsible for saving the stories, the entities, and the rank of entities in stories. For every story involving single numerical facets, this component saves the rank of each node's facet value performance in that story. After all single numerical facets are processed, each node would have its own indexed document with a list of the stories it appears in as well as its rank among all nodes in every story. Figure 3.7 shows how this component works with other components in the framework.

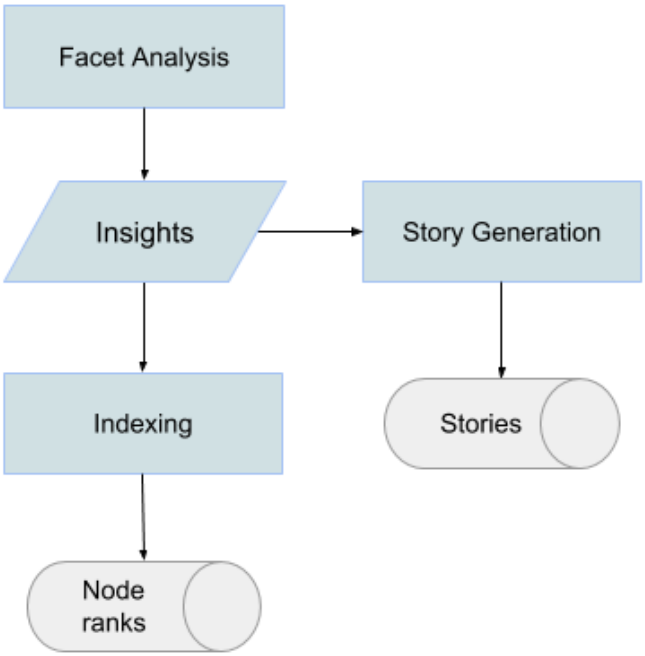

Figure 3.7: The indexing component saves raw insights such as nodes' ranks in stories to be later used in the search interface.

The ranking of nodes in their stories is used to portray their significance and performance in the graph. Nodes that rank higher are more important in the graph and are given higher performance scores that are shown in the user interface's knowledge box (see Chapter 4).

This concept is taken further by saving the ranks of connected nodes with regards to the nodes in a story. For example, when ranking a story involving papers, the rank obtained by each paper is assigned to all of its authors. This means that when searching for stories about an author, stories that feature the

authors as well as their papers are retrieved. We use this concept of connected nodes and stories as the key exploratory search function in our search interface.

We use Elasticsearch[7] to index basic information about nodes in the graph. This includes their type, their unique ids and their display name. We index this information to provide autocomplete suggestions for the nodes in the graph (e.g. journal name, paper title, author name). We heavily depend on this index alongside the story and node ranks indexes for running our tool. These indexes are essential to our approach to transform knowledge graphs into stories containing insights. Note that this approach is different from traditional faceted search interfaces that sends queries to databases to filter documents according to a search selection. In our case, we process the knowledge graph once, and then we only interface with the indexes generated by the indexing component and the stories themselves.

[7]https://www.elastic.co

Chapter 4

Weaver User Interaction

In this work, we develop Weaver, a search interface tool that allows users to explore the generated stories as well as entities from the knowledge graph. Weaver accesses the indexes saved by the story generation framework to perform search retrieval. The interface's features can be summarized by the following tasks:

1- Searching for a specific entity via a keyword-search input
2- Retrieving stories as search results for the queried entity
3- Retrieving additional aggregated insights for the queried entity

Figure 4.1 shows how we established a continuous exploratory search experience by retrieving relevant stories as well as relevant entities for a queried entity.

One of the contributions of this work is using stories as documents for a search retrieval task. This is different from the retrieval technique used in faceted search interfaces where documents from the data itself are filtered by users' facet selections. In order to do that, two search strategies that filter the stories are explored:

(A) using keyword search to filter stories by term matches in their titles and content
(B) using an entity search to filter stories by the entities in their content

We decided that our interface would not work well with method A. This is because our automatically generated stories lack dynamic text, which makes it harder for users to find interesting search matches. Instead, we opted for method B which uses entities as search queries. Users would then search for

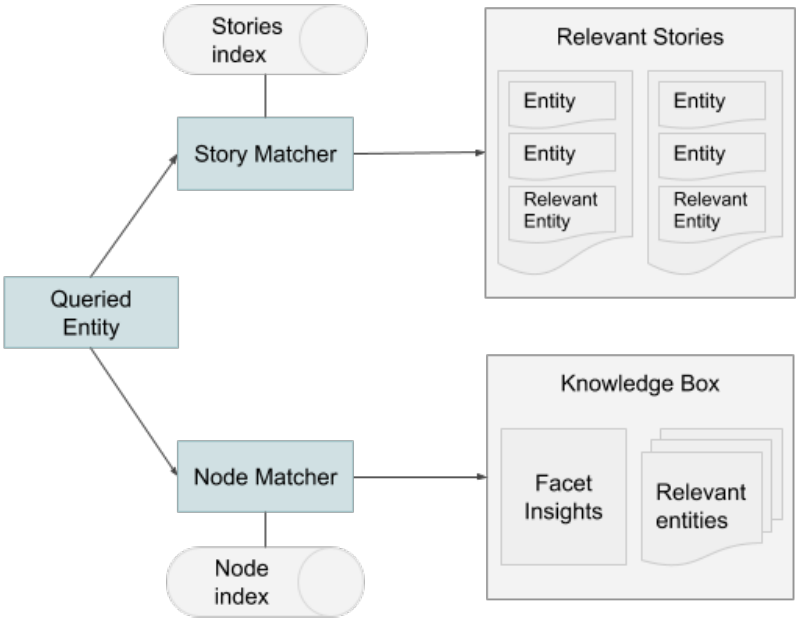

Figure 4.1: The search retrieval technique we used matches relevant stories as well as nodes to a queried node. All entities are hyperlinked to their search results pages, creating an easy-to-use and interconnected search experience.

a specific entity and obtain a list of relevant stories for that entity. For our retrieval technique, an inverted-index containing a list of references to stories for each entity in the graph (see Chapter 3.4) is used.

In addition, hyperlinks are added for entities to connect them to their search results pages. This provides users with semantic links between stories and entities, improving the user experience where users would focus less on refining queries, and more on navigating the interconnected stories and entities.

Figure 4.2 shows how every story in the search results shows information such as the total number of nodes that were analyzed in the story, as well as the ranks of the queried node in the story, sorted by the ascending values of the ranks. For our search results ranking strategy, stories are ranked by their top node ranks. An important point to be made is that relevant stories can be indirectly related to the queried entity via its connected entities. For example, the search results for an author might return stories that are directly connected to the author's papers. Figure 4.2 shows how search results for an author can contain several ranks for each of their papers contained in a story. Since the stories' content only shows a limited number of top nodes, an additional box is added to a story showing information about the queried node such as its rank, connected entities and the analysis value with regards to the facet analyzed

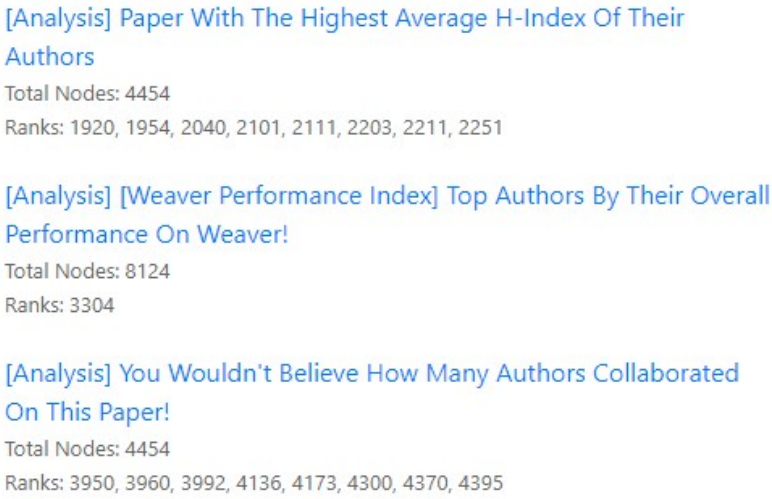

Figure 4.2: Search results for an author contain a list of stories with the ranks of the connected entities to the queried entity.

in the story. This is only visible when users access stories filtered by some search results for a queried entity. Figure 4.3 shows this essential feature that explains to users how every story got selected as a relevant match to a queried node using the aforementioned connected entities technique.

Another main feature is providing additional insights about a queried node via a knowledge box. When a user searches for an entity, relevant stories connected to the entity are retrieved. However, the amount of stories that can be retrieved could be overwhelming for users to explore. Also, we look to connect nodes together in the similar way which we connect entities to stories. Therefore, a knowledge box is added that displays additional insights about the queried node in addition to the retrieved story matches. The knowledge box contains the following insights:

(1) **Facet values and global facet value ranks of the queried entity**: for every facet generated by the story generation framework (see Chapter 3.2), we rank the performance of the queried node with regards to this facet among the entities with the same label. The higher the facet value, the higher the rank. Users can then see the individual ranks for every facet that the framework used to generate stories.

(2) **The Weaver score and global Weaver rank of the queried entity**: for each ranked facet from (1), we give points to each entity equivalent to the inverse value of its rank with regards to the total number of entities analyzed for that facet. For example, for a given facet, the number 1 ranked

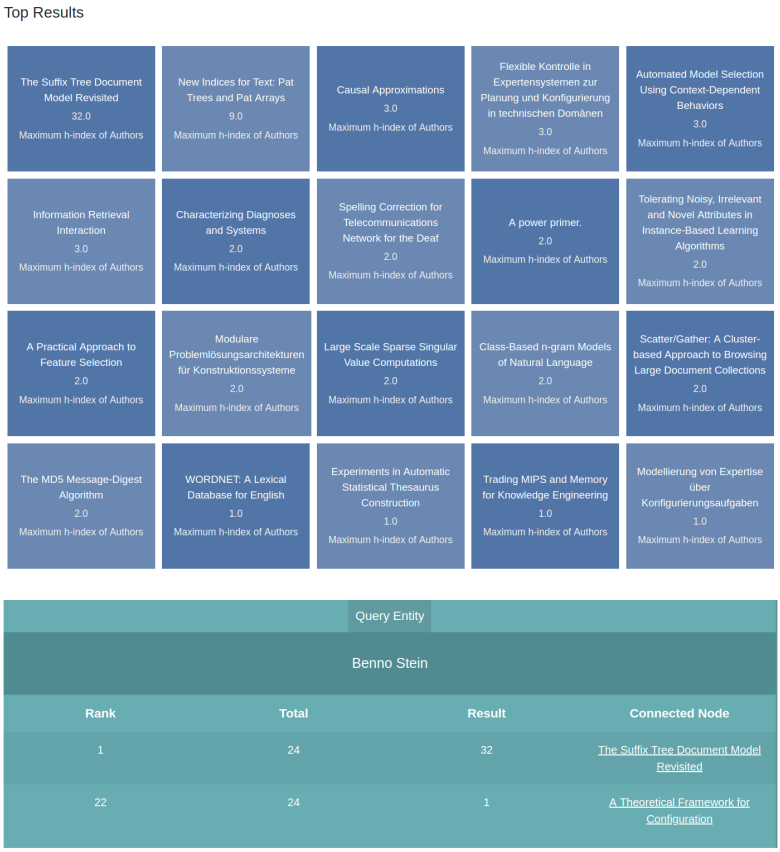

Figure 4.3: An example story content when users enter a story via the search results of a queried entity.

entity from a total of 5000 entities receives 5000 points, the number 2 ranked entity receives 4999, etc. We then aggregate all the points accumulated after processing all facets for the queried entity's node type into a global Weaver score, and we then rank the scores to obtain the final Weaver rank.

(3) **The top connected entities to the queried node**: we query the graph for the connected entities to the queried entity via the available node relationships of the queried node. For every relationship type, a list of connected entities is obtained and ranked based on a manually-selected facet from the connected entities' available facets. This ranking facet is selected based on manual judgment of its ability to show the influence of these specific entities. The top ranked entities are then selected as the featured connected entities to the queried entity.

As an example, if the queried entity is an author, separate lists containing

its featured papers, authors, and journals are shown. For every one of these lists, a different facet is used to rank the entities in it: "incoming citations" for papers, "h-index" for authors, and "total citations of papers" for journals. It's important to note that every entity displayed in the knowledge box is hyperlinked to its search results page. Figure 4.4 shows how the knowledge box reveals further insights about the queried node.

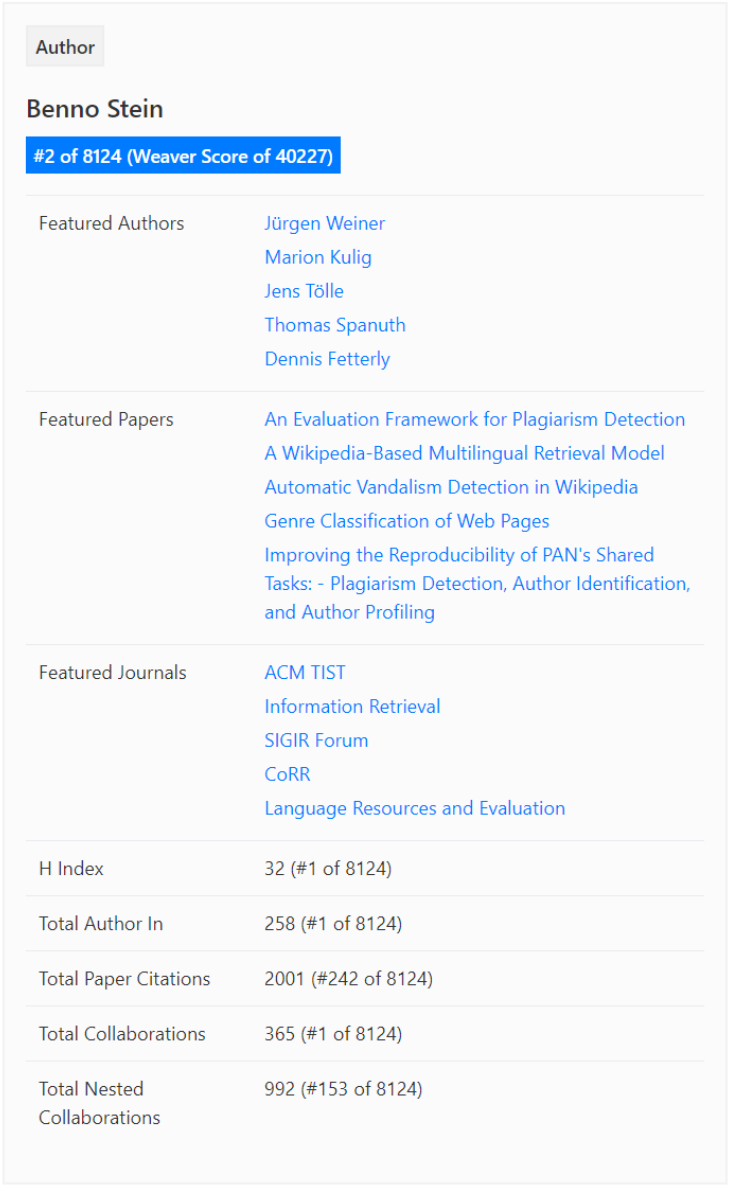

Figure 4.4: The knowledge box retrieves insights about the queried node.

As a summary, we enhanced our search interface by connecting stories, entities, and search results together in an easy, intuitive, and interconnected exploratory search experience. This allows users to navigate to new searches and explore new entities and their stories by clicking on any hyperlinked entity throughout the interface. The knowledge box in particular provides highly relevant search suggestions to the current queried entity via its lists of featured entities. In a way, the knowledge box acts as self-contained compact story for the queried node, showing its insights which were used in the story generation framework, as well as its most connected entities from the graph.

Chapter 5

Evaluation

In this work, we built a knowledge graph from a subset of a large dataset of scientific publications. The graph is constructed starting from a community of authors which we can easily recruit as test subjects for a usability study. This strategy was supported by one of our research questions regarding the usefulness, interestingness and novelty of the insights the stories provide. We believe that these questions and the overall usability of our interface can be best evaluated by targeting end users directly connected to the knowledge domain we used in our experiments.

A usability study is conducted with 5 participants who are experienced in the knowledge domain we used in our experiments. This small test size is recommended by usability expert Jakob Nielsen [1]. Our goal was to evaluate several aspects with regards to the user experience, based on expert user experience research by Peter Morville [2]. The participants are PhD researchers actively involved in research in the field of computer science. The user study took form of face-to-face interviews lasting between 30 to 45 minutes in our lab, with one workstation displaying our search interface. Participants would not be compensated in any way for their participation. We explained the research problem, our proposed solution, the user interface we developed, and the research questions we wanted their experienced feedback on. As a summary, we had a discussion with the participants about the following user experience points:

Useful: we discussed whether the capabilities we provided via our search interface would be useful for their work. The question here was to understand whether we are providing an innovative solution that provides users with new

[1] https://www.nngroup.com/articles/how-many-test-users/
[2] http://semanticstudios.com/user_experience_design/

insights that help them with their work. All participants expressed that they would use our tool to assist in their work.

Usable: even though we did not perform a user-centered design study earlier in our work, we wanted end-users to be able to quickly and easily access the automated insights our stories provide. All participants found using the interface to be intuitive.

Desirable: we discussed whether our stories, search results, and the knowledge box were visually attractive enough to provoke a good experience for participants. Most participants particularly enjoyed the knowledge box and search results, but commented on the story titles and content.

Findable: participants were asked whether navigation was an issue in our interface. We specifically focused on their feedback about how entities, stories and search results are all connected from a navigation perspective. All participants reported that the navigation in the interface was easy to use and straightforward.

Credible: participants were shown the knowledge graph itself and explained about the story generation framework. This influenced their decisions that information provided by the tool is trustworthy. This is because the insights are based on statistics and data analysis. The participants' knowledge domain confirmed the reported insights about the most influential authors and papers, which gave the insights credibility.

Valuable: participants expressed how valuable the interface was in reducing the effort of discovering insights from knowledge graphs. Our interface provides an automated solution for the amount of work required by users to obtain such insights from knowledge graphs. Indeed, all participants found that our approach with the search interface plays a valuable role in solving their problems along with using other tools.

All participants conducted a Computer System Usability Questionnaire (CSUQ) [Lewis and R, 1993] having 19 questions with 7 answer options starting with "strongly disagree" and ending with "strongly agree", with an additional N/A option. The answers are used to calculate the test scores for every question on a seven-point Likert scale between -3 (strongly disagree) and 3 (strongly agree). For each question, all non N/A scores are averaged to obtain the average score for the question.

Weaver was published online [3] for participants to independently complete the user study after using the interface for a period of time. This strategy gives participants more time to get acquainted with the tool to provide a more accurate feedback. The results of the CSUQ which provide information about

[3] https://weaver.webis.de

the overall system usefulness, information quality, interface quality, and overall satisfaction were then collected. Table 5.1 shows the participants' mean and standard deviation scores for individual questions in the CSUQ. The column **n** represents the total number of participants that answered the question.

From these results, we can first say that some participants did not rate questions 9, 10, and 11. We collected comments from the participants, which stated that the system does not encounter any mistakes that require error logs, which justifies their N/A responses to these questions.

The highest standard deviation rates are for questions 7 and 10. Question 7 received an average score close to 0, meaning a neutral response from the participant about the ease-of-use of the system. The high standard deviation values shows disagreement about these questions. Question 10 is related to mistake recovery using the system; the high number of N/A (2 out of 5 participant answers) meant that the question was unclear or the interface did not provide situations where users can apply such a recovery.

The highest question scores with averages close to 2 were related to the participants using the interface to efficiency and effectively complete their work. These questions also targeted how easy it is for participants to find the information they need, the interface of the system, and the organization of the information itself. This tells us that our interface has a good design, and the information we presented in the stories are well-organized and accessible. Since no experienced user interface designers assisted in our project, we are satisfied with this feedback.

Question 8 had a slightly negative score; in their comments, participants explained that the interface could provide more functionality, i.e. add interactive design elements to the stories, where it is possible to tweak some filters within the stories, or to incorporate more interactive visualizations.

The participants gave positive feedback about the system use, with an average of 1.28, with little to no opinion disagreement. The lowest performance was for questions related to information quality, with an average score of 0.72. While still considered as positive feedback, participants had a relatively neutral opinion to this aspect of the interface. We interpret this as feedback about the insights themselves, relating to our research question about whether automatically generated stories can be viewed as interesting by users. Throughout this work, we discussed ways to improving the content of stories to match some level of scientific journalism that users are used to. Since we did not use any text-generation methods to enrich our stories, we relied on providing statistics, graph plots, and ranking entities as story content. Some of our stories featured complicated facets that show different insights about influence and reach in the graph, which the participants did not easily understand, and hence may explain the 0.60 score for question 13. With easier to understand facets

established in social network analysis, the meaning that users extrapolate from the insights might make the stories easier to understand and hence increase the information quality.

As mentioned earlier, we received a favorable opinion about the interface quality (1.07) with no disagreement. Still, much work could be done in this area, such as acquiring the expertise of user interface design (UI) and user experience (UX) specialists.

Finally, participants rated the overall usability of our interface as 1.70 with no disagreement. This is a major positive feedback that confirms the validity of our approach, as well as its potential for further research and improvement. All in all, participants expressed an above average satisfaction both in their verbal and written comments, as well as their CSUQ test scores for our work.

Table 5.2 shows the CSUQ score averages and standard deviation by category.

Question	**n**	**Mean**	**S.D.**
1. Overall, I am satisfied with how easy it is to use this system	5	1.80	0.84
2. It is simple to use this system	5	1.80	0.84
3. I can effectively complete my work using this system	5	1.20	0.45
4. I am able to complete my work quickly using this system	5	1.20	0.45
5. I am able to efficiently complete my work using this system	5	1.40	0.55
6. I feel comfortable using this system	5	1.40	0.89
7. It was easy to learn to use this system	5	0.40	1.67
8. I believe I became productive quickly using this system	5	1.00	0.71
9. The system gives error messages that clearly tell me how to fix problems	2	-1.00	1.15
10. Whenever I make a mistake using the system, I recover easily and quickly	3	1.33	1.54
11. The information (such as on-line help, on-screen messages and other documentation) provided with this system is clear	4	0.50	1.29
12. It is easy to find the information I need	5	1.40	0.55
13. The information provided with the system is easy to understand	5	0.60	1.34
14. The information is effective in helping me complete my work	5	0.80	0.84
15. The organization of information on the system screens is clear	5	1.40	1.52
16. The interface of this system is pleasant.	5	2.00	0.71
17. I like using the interface of this system	5	1.60	1.14
18. This system has all the functions and capabilities I expect it to have	5	-0.40	0.89
19. Overall, I am satisfied with this system	5	1.60	0.89

Table 5.1: CSUQ results by question.

Question Category	**Mean**	**S.D.**
System Use (questions 1-8)	1.28	0.40
Information Quality (questions 9-15)	0.72	0.33
Interface Quality (questions 16-18)	1.07	0.22
Overall (questions 1 and 19)	1.70	0.04

Table 5.2: CSUQ results by question category.

Chapter 6

Conclusion and Future Work

In this thesis, we designed, implemented, and evaluated a framework that generates stories containing implicit insights from knowledge graphs. We first build a knowledge graph from a subset of the SCORC containing 4488 papers. We apply our story generation framework on the graph to generate 540 stories containing insights based on data analysis, social network analysis, and statistical methods. Finally, we built an exploratory search interface that uses these interconnected stories to provide end-users with access to hard-to-obtain implicit information from our knowledge graph. In this chapter, we summarize our contributions and briefly discuss ideas to improve the results from our story generation framework as well as the functions of the exploratory user interface.

6.1 Contributions

We began the thesis with the following research questions:

1) Can we automatically-generate interesting stories containing analysis insights from a knowledge graph?

2) How can we provide end users with easy-to-use access to these stories using an exploratory search interface?

The first research question is answered in Chapter 3; we setup a knowledge graph and developed a story generation framework to generate stories containing analysis insights from the graph. We enriched the graph with additional facets that describe implicit information such as the informal connections, social performance, and the interdependence of the nodes. In Chapter 3.3, we described how these insights can be communicated to users in a clear manner

in form of story templates containing HTML text, tables and graph plots.

The second research question is answered in Chapter 4, with the implemented exploratory search interface that allows users to explore the graph insights via the stories. It uses an information retrieval method that retrieves stories instead of documents related to the searched entity. It also provides users with an additional knowledge box with graph insights about the searched entity.

Finally, we conducted a CSUQ usability study to get feedback about our work and specifically our search interface Weaver. We obtained an above average score of 1.70 for the "overall" question category from participants. While our framework does not completely answer the question of what makes interesting stories, it received positive feedback from participants who are experienced users in the knowledge domain used in our experiments.

6.2 Future Work

In this work, our contribution to using insight stories in an exploratory search interface could be used in the future to facilitate the discovery of insights in complex knowledge domains for end users. One of the tasks that can be investigated is how to integrate our work with traditional document search engines to provide users with both document search capabilities as well as the implicit insights which our story-based approach provides. This would give users more exploratory power while not sacrificing cases where specific information needs exist.

In Chapter 3.1, we create a dataset from a subset of the SCORC containing 4488 papers. We believe future work can focus on scaling this setup to bigger datasets. This would require several changes in terms of the graph management software used, as well as modifications to the framework itself. Every component in the framework will have to be able to perform its operations on big data. The knowledge graph setup as well as the insight discovery components will require more resources in terms of memory and possibly computing power and storage to work within reasonable amounts of time. Similarly, the insight discovery component will require increased computing costs in terms of time and resources.

In Chapter 3.2, we enriched the graph with additional facets to provide

more insights to be used in stories. However, we do not claim that we covered all the possible facets that could be used to enrich the graph. Future work could investigate using techniques from graph theory, social network analysis, or distant reading to generate additional facets that can provide more insights into the data. For example, future work could use more centrality algorithms such as betweeneess centrality, harmonic centrality, or other facets that can be used for analysis.

The use of automatically-generated stories that communicate information in an interesting manner is a hot topic in automated journalism. Natural language generation and data-to-text research would highly improve the quality of the stories that are generated. While our stories contain rich HTML with plots and tables, future work can incorporate techniques from digital humanities such as rich interactive visualizations. In our approach, we make searching knowledge graphs like searching the web. In search engines like Google, the search results are the stories from the websites that are crawled by Google. In our case, the stories are self-generated, therefore future work can focus on the quality of the stories to make them more user-friendly and interesting to read. An important point for future research in any automated journalism-based approach is also being able to automatically generate interesting dynamic titles for stories. During our experiments, we considered using clickbait research to generate titles that would better capture users' attention. Future work could investigate in this direction or use some other natural language generation techniques to produce quality titles for stories.

In Chapter 4, we presented our Weaver web tool that looks familiar to search engines, but with our stories as the retrieved documents. In addition to stories, the search results show a knowledge box with more insights about the queried entity and its featured connections in the graph. We believe future work can better assess this story-based approach with a bigger user study, possibly performed by Human-Computer Interaction research. Also, this type of online user interface should be tested for online analytics, organic traffic, and SEO scores after being available on the web for some reasonable amount of time to assess the public need for the provided insights.

Bibliography

Waleed Ammar, Dirk Groeneveld, Chandra Bhagavatula, Iz Beltagy, Miles Crawford, Doug Downey, Jason Dunkelberger, Ahmed Elgohary, Sergey Feldman, Vu Ha, Rodney Kinney, Sebastian Kohlmeier, Kyle Lo, Tyler Murray, Hsu-Han Ooi, Matthew E. Peters, Joanna Power, Sam Skjonsberg, Lucy Lu Wang, Chris Wilhelm, Zheng Yuan, Madeleine van Zuylen, and Oren Etzioni. Construction of the literature graph in semantic scholar. In *NAACL-HLT*, 2018. 1

Marcelo Arenas, Bernardo Cuenca Grau, Evgeny Kharlamov, Šarūnas Marciuška, and Dmitriy Zheleznyakov. Faceted search over rdf-based knowledge graphs. *Journal of Web Semantics*, 37:55–74, 2016. 1

Hannah Bast, Florian Bäurle, Björn Buchhold, and Elmar Haußmann. Easy access to the freebase dataset. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 95–98. ACM, 2014. 1

P Boot. *Distant Reading. Franco Moretti.*, volume 30. 03 2014. doi: 10.1093/llc/fqu010. 2.2

Robert G. Capra and Gary Marchionini. The relation browser tool for faceted exploratory search. In *JCDL*, 2008. 1

Matt Carlson. The robotic reporter: Automated journalism and the redefinition of labor, compositional forms, and journalistic authority. *Digital journalism*, 3(3):416–431, 2015. 2.4

Jeroen De Mast and Benjamin Kemper. Principles of exploratory data analysis in problem solving: What can we learn from a well-known case? *Quality Engineering*, 21:366–375, 09 2009. 3.3

Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, 48, 2016. 1

John Gourville and Dilip Soman. Overchoice and assortment type: When and why variety backfires. *Marketing Science*, 24:382–395, 08 2005. doi: 10.1287/mksc.1040.0109. 1

Rasmus Hahn, Christian Bizer, Christopher Sahnwaldt, Christian Herta, Scott Robinson, Michaela Bürgle, Holger Düwiger, and Ulrich Scheel. Faceted wikipedia search. In Witold Abramowicz and Robert Tolksdorf, editors, *Business Information Systems*, pages 1–11. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-12814-1. 1

Marti Hearst, Ame Elliott, Jennifer English, Rashmi Sinha, Kirsten Swearingen, and Ka-Ping Yee. Finding the flow in web site search. *Commun. ACM*, 45(9):42–49, September 2002. ISSN 0001-0782. doi: 10.1145/567498.567525. URL `http://doi.acm.org/10.1145/567498.567525`. 2.1

Jonathan Koren, Yi Zhang, and Xue Liu. Personalized interactive faceted search. In *WWW*, 2008. 1, 2.1

Robert Kosara and Jock Mackinlay. Storytelling: The next step for visualization. *Computer*, 46(5):44–50, 2013. 3.3

Karen Kukich. Design of a knowledge-based report generator. 01 1983. doi: 10.3115/981311.981340. 2.4

Bill Kules, Robert G. Capra, Matthew Banta, and Tito Sierra. What do exploratory searchers look at in a faceted search interface? In *JCDL*, 2009. 1

Leo Leppänen, Myriam Munezero, Mark Granroth-Wilding, and Hannu Toivonen. Data-driven news generation for automated journalism. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 188–197, 2017. 2.4

James Lewis and James R. Ibm computer usability satisfaction questionnaires: Psychometric evaluation and instructions for use. *International Journal of Human-Computer Interaction*, 7:57–, 01 1993. doi: 10.1080/10447319509526110. 5

Liubov Nesterenko. Building a system for stock news generation in russian. In *Proceedings of the 2nd International Workshop on Natural Language Generation and the Semantic Web (WebNLG 2016)*, pages 37–40, 2016. 2.4

Tuukka Ruotsalo, Kumaripaba Athukorala, Dorota Głowacka, Ksenia Konyushkova, Antti Oulasvirta, Samuli Kaipiainen, Samuel Kaski, and

Giulio Jacucci. Supporting exploratory search tasks with interactive user modeling. In *Proceedings of the 76th ASIS&T Annual Meeting: Beyond the Cloud: Rethinking Information Boundaries*, page 39. American Society for Information Science, 2013. 2.1

John Scott. Social network analysis. *Sociology*, 22(1):109–127, 1988. 2.3

Edward Segel and Jeffrey Heer. Narrative visualization: Telling stories with data. *IEEE Transactions on Visualization and Computer Graphics*, 16:1139–1148, 2010. 3.3

Vineet Sinha and David Karger. Magnet: Supporting navigation in semistructured data environments. pages 97–106, 01 2005. doi: 10.1145/1066157.1066169. 1

Somayajulu Sripada, Ehud Reiter, and Ian Davy. Sumtime-mousam: Configurable marine weather forecast generator. *Expert Update*, 6, 02 2004. 2.4