Contextual Search in Issue Based Information Systems
Towards Information Discovery in Complex Discourse Graphs

Magnus Lien
Master’s Thesis Spring 2016
With all navigation systems, before we can plot our course, we must locate our position.

LOUIS ROSENFELD
Acknowledgements

I would like to thank my supervisor Sasha Mile Rudan, and co-supervisor Dino Karaberg, for valuable feedback and guidance while writing the thesis. Their input also helped frame the work in the broad and extensive domain that is knowledge federation.

And I would also like to thank my fellow students whom I shared many long days with at the 9th floor of the faculty building. Their company and input on the writing process was an inspiration through the toughest times.

Last but not least, I would like I extend my deepest gratitude for all the support from my family and friends. I could not have done this without you, thank you!
Foreword

This research is based on a project initiated by Knowledge Federation, a frontrunner organization performing research on collective intelligence and knowledge federation. Recently KF released an early prototype of a knowledge domain mapping tool, named KnAllEdge, for organizing inter-personal discourse into digital knowledge maps. This tool is based on a Issue Based Information System, a method that first emerged during the 1970s. IBIS enable collaborative groups to organize and map their discussions into visual map structures. Discourse maps are non-linear information graphs where the discussions branch out as they develop.

IBIS is praised for its rhetorical nature of information organization, it enable visual and collective organization of resources and ideas relevant to complex issues - but those maps also become forbiddingly complex as they grow. When discourse maps expand, the many fine threads of thoughts and ideas become shrouded in large hierarchies of interweaving discussions. Collaborators that study the current state and surrounding context of a subject matter, struggle to provide precise and accurate contributions. A demanding task in a collaborative environment that can span many domains of expertise and knowledge.

In this thesis we study how contextual meta-data search can reduce information overload, and support discourse reasoning, for collaborators that federate knowledge within KnAllEdge IBIS. The outcome is an experimental prototype for contextual search in KnAllEdge. This prototype support different search methods, all based on text queries. It searches discourse maps using a selection of prefix filters, that collaborators can use to locate discussions based on their needs. The search parameters are visualized in the context of their history, to support reasoning, and to aid in discourse evaluation.

To gather novel insight about how collaborators interact with the prototype query language, a basic version of the prototype was tested in a workshop at the University of Oslo. The research shows that contextual meta-data search is a viable method for navigating large and complex discourse maps in KnAllEdge. Observations from the workshop indicate that the query language itself can be too ambiguous and difficult to use for those with less technical background, and this particular feature would benefit from further experimentation.
In the world with information overload on the one side, and global issues that need to be clearly understood on the other, IBIS is likely to emerge as the tool of choice. Hopefully this research illuminate the challenges of complexity when working with IBIS, and encourage researchers to experiment further with contextual IBIS navigation and visualization.

*Keywords:* KnAllEdge, IBIS, Navigation, Search
# Contents

1 **Introduction** 1  
   1.1 Research and Requirements 2  
   1.2 Scope 3  
   1.3 Chapter Overview 3  

2 **Background** 5  
   2.1 The Collective Mind 5  
   2.2 Three Generations of IBIS 6  
      2.2.1 gIBIS 7  
      2.2.2 QuestMap 8  
      2.2.3 Compendium and Compendium Next Generation 10  
      2.2.4 DebateGraph 11  
   2.3 KnAllEdge 12  
   2.4 Issue Based Information System 15  
      2.4.1 Collaborative Problem Solving 15  
      2.4.2 Discourse Mapping 16  
      2.4.3 Discourse Nodes 17  
      2.4.4 Node Relationships 18  
      2.4.5 Map Layout 19  
      2.4.6 Creating Discourse Contributions 20  
      2.4.7 Making Decisions 23  
   2.5 KnAllEdge Knowledge Space 23  
      2.5.1 Knowledge Consensus Mechanism 24  
      2.5.2 Knowledge Polarization and Argumentation 24  
   2.6 The Knowledge Federation 26  
   2.7 Summary 26  

3 **Meta-data Queries** 29  
   3.1 Node Searching in KnAllEdge 29  
   3.2 KnAllEdge Map Data 30  
   3.3 Meta-data Queries change the Data Layer 32  
   3.4 Submitting Meta-data Queries 32  
      3.4.1 Query Syntax 33  
      3.4.2 Prefix Mapping 33  
      3.4.3 Composing Queries 34
5.9 Summary .......................................................... 66

6 Results and Discussion 67
6.1 Information Overload .............................................. 67
  6.1.1 Map Size .................................................. 67
  6.1.2 Organizing Thoughts and Ideas ......................... 68
  6.1.3 Adjusting to Change ........................................ 70
6.2 Contextual Meta-data Queries ................................. 70
  6.2.1 Contextual Oversight ....................................... 71
  6.2.2 Support Reasoning by preserving Discourse Structure 72
  6.2.3 Consistency Checking and Error Detection .............. 73
6.3 Limitations .......................................................... 74
  6.3.1 Adding additional Query Prefixes ....................... 75
6.4 Future Research ................................................... 75
6.5 Improve the Query Language ................................. 75
  6.5.1 Multi-faceted Search Visualization ....................... 75
  6.5.2 Support Cognitive Mapping ................................ 76

7 Concluding Remarks .............................................. 77

A Prototype Source Code ........................................... 83

B Code for Topological Sorting .................................. 85

C Code for Shortest Path Algorithm ............................ 87

D Women in Technology IBIS Map ............................... 89
List of Figures

2.1 gIBIS Proto-Node Query ............................................. 8
2.2 QuestMap ............................................................... 9
2.3 Compendium ............................................................. 10
2.4 DebateGraph ........................................................... 12
2.5 KnAllEdge IBIS Nodes ................................................. 13
2.6 WhoAmI Connecting Actors and Interests ...................... 14
2.7 KnAllEdge IBIS Node Relationships ......................... 18
2.8 IBIS Discourse Structure [3] ................................. 20
2.9 Nature of Creativity Map ............................................ 21
2.10 KnAllEdge Knowledge Nodes and Decision Nodes .......... 24
2.11 WikiPedia Edit Wars .................................................. 25

3.1 Composing Meta-data Queries ................................. 33
3.2 Content Search .......................................................... 37

4.1 Segment of Women in Technology IBIS Map ............... 47
4.2 Workshop Participant Solving Questions .................. 48
4.3 Workshop Task Performance ...................................... 49

5.1 Prototype Components Overview ................................ 58
5.2 Query Input Interface ............................................... 59
5.3 Prototype Map Visualization ..................................... 62

6.1 Debategraph Digital Agenda Map .......................... 68
6.2 Map Visualized before Contextual Search ................. 71
6.3 Map Visualized after Contextual Search .................. 71
6.4 Simultaneous Discussions ......................................... 74
**List of Tables**

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Node Property Table</td>
<td>35</td>
</tr>
<tr>
<td>3.2</td>
<td>Node Types</td>
<td>35</td>
</tr>
<tr>
<td>4.1</td>
<td>Query Groups</td>
<td>43</td>
</tr>
</tbody>
</table>
Listings

3.1 Node Properties ................................................. 30
3.2 Link Properties ................................................. 31
5.1 Content Search ................................................. 60
5.2 Mapping Knowledge Prefix with Mapping Function ....... 62
5.3 Searching the Discourse Map ................................. 63
5.4 Building the Hierarchical Inheritance Tree ............... 65
Chapter 1

Introduction

In 1995 the American scientist Douglas Engelbart described a strategic solution where we form cooperative alliances or Networked Improvement Communities that would utilize new tools for developing and applying collective knowledge. Their work meant to harness the collective capabilities of mankind for dealing with complex problems, increasing our collective IQ, and enable efficient collaboration across communities and organizations [11].

Engelbart, Vannevar Bush, and John Seely Brown, have contributed much to research and development of collective knowledge management. They supported development of interconnected groupware for capturing informal and formal knowledge during the 20th and 21st century [26]. In the last decades there has been significant advances in the areas of data availability and data connectivity. We continuously push out new technologies that improve our data processing capabilities and deliver information faster, and at greater volumes.

Yet our capability for finding solutions to complex and broad challenges does not progress at the same rate, we are not utilizing the full potential of our collective intelligence. Engelbart pointed out that “we are moving towards a connected, structured and heavily tool supported work-flow, but this needs to become how the majority of people do all their work” [11].

However we have over the last decades progressed development of collective intelligence tools, and Issue Based Information Systems has emerged as as one of the most widely used systems for collective intelligence today. IBIS is a method for capture and organization of interpersonal discussion between groups of collaborators. Formal language is captured using nodes to represent questions, ideas and arguments. These nodes are then mapped to visual\(^1\) discourse maps. Nodes in a discourse map represent the discourse as is, without abstraction.

IBIS maps grow naturally over time, as more and more nodes are

\(^1\)Written or spoken communication
contributed. The discussions addressed in a map can be many, and these cause the discourse map to branch out ².

In large IBIS maps, it is difficult to scrutinize individual discussions efficiently, and to build upon existing contributions. The size of the map can quickly span several hundred nodes or more. The IBIS method promote information discovery, and is meant to aid in exploration of problem and solution spaces, yet does not suggest best practices to manage information overload for collaborators. Most IBIS applications support navigation of maps through a text search, used to locate contributions by searching labels and descriptions fields. But basic search and map filtering still does not enable collaborators to browse contributions relevant to their knowledge and interests, and they are not able to focus on individual discussions.

Each contribution in IBIS has a history leading up to it, and this history is defined by preceding contributions. While navigation through search can be made to efficiently locate specific contributions, we can also manage information overload by filtering out any contributions that are not part of the contextual history. So collaborators can focus their attention on contributions that are relevant to their work, putting them are a better position to understand the current state of any discussion.

1.1 Research and Requirements

This research investigate how map size, capturing thoughts and ideas with IBIS nodes, and adjusting to changes and updates in the map node structure, are connected to information overload in IBIS. The author propose an experimental search mechanism, named meta-data queries, for navigating KnAllEdge IBIS maps. With this prototype, we will investigate how contextual queries can support reasoning and evaluation of discourse history, and reduce information load for KnAllEdge collaborators. During the course of this research, we will then investigate the following:

1. What are some of the major factors that increase information load, for collaborators working with IBIS discourse maps?

2. To what extent can contextual meta-data queries reduce information load, for collaborators working with KnAllEdge IBIS discourse maps?

Meta-data queries introduce a text search interface and a simple but novel query language. A basic version of the prototype is developed, and later tested during a workshop held at University of Oslo. This basic version is used to evaluate comprehension and intuitiveness of the query syntax. Part of the research is a post-workshop analysis and discussion.

²Expand in different directions
1.2 Scope

KnAllEdge Version

KnAllEdge is undergoing current development, and only recently introduced versioning. This research have accounted for the state of the documentation and system as it were on February 27th 2016, see footnote 3.

KnAllEdge Knowledge Spaces

In chapter 2 the following concepts are introduced: Knowledge spaces, consensus mechanisms, and polarized views, these describe how KnAllEdge become a knowledge federation tool. This research does not evaluate knowledge spaces, consensus mechanisms and polarized views, in the context of the literature and research findings.

1.3 Chapter Overview

Chapter 2 - Background

This chapter begins with an introduction to collective mind. Then we move on to the last three generations of graphical IBIS systems4, including KnAllEdge. The author describe the IBIS method, and introduce the following concepts in the context of KnAllEdge: WhoAmI, knowledge spaces, consensus mechanisms and polarized views. The chapter ends with an introduction to the organization Knowledge Federation, and KnAllEdge’s role in supporting trans-disciplinary research groups.

Chapter 3 - Meta-data Queries

This chapter describe the specification for the meta-data queries prototype. Including the KnAllEdge map data structures, the query language, and context extraction.

Chapter 4 - Exploring Meta-data Queries

This chapter cover a workshop held at University of Oslo, to gather novel insight about the meta-data query language. We go through the planning, execution, data measurements and results. The results are later discussed in chapter 6.

3https://github.com/mprinc/KnAllEdge/commit/2bc29c256c44d18e6217de8e25ff61bbf6131c6
4These systems visualize IBIS discourse as hierarchical graph layouts
Chapter 5 - Prototype Development

The design and implementation of the KnAllEdge search prototype is covered in this chapter. We will go through the technologies used, the system architecture, and how the prototype components implement the meta-data queries specification.

Chapter 6 - Results and Discussion

This chapter begins with analyzes of information overload in IBIS graphs, then discuss the prototype research in the context of the findings. The chapter ends with recommendations for future work.

Chapter 7 - Concluding Remarks

Here the author summarize the research, and finalize it with concluding remarks.
Chapter 2

Background

Some of the great minds of the 20th century began collective mind research long before computers became a common household item. Enabling rapid evolution of mankind’s collective thinking capabilities is not a trivial challenge, but our current technological achievements can help us make progress.

2.1 The Collective Mind

Shortly after World War two, an American scientist and director of the office of scientific R&D\(^1\) Dr. Vannevar Bush, published an article *As We May Think* in the magazine *The Atlantic*. During his time, science had accomplished many impressive feats: the telephone, television, advanced calculation machines, electronic typewriters, and much more. In this article he expressed how the collective efforts of science should be refocused from extending mankind’s physical powers, to extend the powers of our collective mind [5, p. 1]. We have displayed great prowess in developing new technology for efficient communication and creating novel methods of contributing knowledge to society. But with insufficient efforts in organization and distribution of what we have learned. He saw that the expansion of knowledge was growing faster and faster, and it was difficult to research what has been done. Researchers were struggling to efficiently consult and extend existing research [5, p. 12].

One of the people who were inspired by this article was Douglas Engelbart. Engelbart was an accomplished and innovative American engineer, that would later pioneer advancements in computer software and hardware. Engelbart believed that connected digital media would enable us make use of our collective social nervous system, to solve increasingly complex and demanding problems. One of his well known contributions in this field is hyper-linked resources, in modern times

\(^1\)Research and Development
they are collectively known as hyper-links. And another is remote collaboration on shared electronic documents. He demonstrated these concepts, as well as several others\(^2\), during the Mother of All Demos in 1968, a Fall Joint Computer Conference in San Francisco.

But Engelbart did not rest on the laurels after these accomplishments. While he had recognized the need to harness the collective capabilities for dealing with complex problems, there was much work to be done towards efficient collaboration across global communities, and to increase our collective IQ. In retrospective, Engelbart dedicated much of his professional career towards researching technologies for augmenting mankind’s collective IQ.

In 1995, Engelbart published an article describing a strategic solution for forming cooperative alliances or Networked Improvement Communities (NIC), and utilizing new collaborative tools for developing and applying collective knowledge \([11]\). He recognized the need for people to have a larger role in global socio-technical solutions, that they are an important part of any machinery designed to augment collaborative documentation and distribution of knowledge.

### 2.2 Three Generations of IBIS

Through generations of research and development, Issue Based Information systems has proven itself to be a powerful knowledge federation tool. It is capable of capturing the collective thought processes of global organizations, societies and groups.

In 1987, Dr. Jeffery Conklin, headed a team of pioneers at Cognexus Institute that developed gIBIS. gIBIS marked the beginning of IBIS as a networked discourse mapping tool, and was the earliest system to support visual discourse maps with hypertext\(^3\). It was designed for capturing policy and design discussions with multiple computers connected through a local network.

The researchers at the institute later developed QuestMap, that had graphical hypertext and extensive groupware\(^4\) capabilities. A few years later, Verizon research labs then developed the Compendium tool, it was based on gIBIS and QuestMap, and licensed as open source software. Compendium supported discourse mapping in both individual and group work scenarios. Today compendium is no longer maintained \([29]\), but a community edition named Compendium Next Generation is still in use.

The DebateGraph system is a fairly new IBIS tool, it contains many features, and it is still widely used today. It is not an offshoot stemming from gIBIS and QuestMap.

---

\(^2\)Engelbart demonstrated a prototype that would eventually become the modern computer mouse

\(^3\)Visual content that was linked to other related content

\(^4\)Software that supports collaboration
from gIBIS, QuestMap or Compendium. DebateGraph offer comparably better support for large collaborative groups than the previous generations of IBIS, with highly connected discourse maps that exchange knowledge, and many support tools for discourse mapping.

2.2.1 gIBIS

Graphical IBIS was introduced in 1987, as one of the first hypertext compatible IBIS systems. It was originally meant to be an experimental prototype of a system named DesignJournal, with the purpose of studying computer mediated teamwork, and navigation of large unstructured information spaces, particularly addressing “interface problems inherent in capturing large amounts of informal design information, and in providing effective methods for the indexing and retrieval of this information.” [10, p. 247-248]. DesignJournal itself was intended as a tool for system designers, to support all aspects of system design, including documentation, requirements specification, design sketches, design solutions and more. The DesignJournal project started in the labs of the Microelectronics and Computer Technology Corporation’s (MCC) Software Technology Program [6][8, p. 6].

gIBIS enabled smaller teams to use IBIS to capture design rationale⁵, collaborating on shared computers and through local networks [9][6, p. 314]. The developers made novel efforts to incorporate map visualization with gIBIS. And gIBIS supported in total three, initially two, different map visualizations integrated in the same interface. The primary view was a large and detailed view of the hierarchical map, showing visual nodes connected by relationship links. A newer revision of the tool added a smaller inset mini-map, that displayed an overview of the full map structure, but much smaller and with less details. In addition to the map visualization, users had a control panel on the right side of the interface. On the top of this panel there was a scrollable list view of the map, it displayed map nodes as labels with special indexes to indicate their position in the primary view.

From user testing with gIBIS, Conklin noted that the size of the map visualization often were a limiting factor for the amount of nodes a user wanted to work with during a single session⁶. Even when node descriptions were hidden, gIBIS collaborators only felt comfortable working with 40-50 nodes, as many as they could view without moving the view of the map [9, p. 212] [6, p. 329]. However, because gIBIS came with a query system based on proto-nodes for performing IBIS map search, users could quickly navigate through the map, by searching for nodes they were interested in.

A proto-node is a pseudo node that contain the same properties as

---

⁵Reasons behind a system design or artifact
⁶A coherent time period of discourse mapping
other map nodes, figure 2.1 show the proto-node. Users could input custom values into these properties, and gIBIS would search the map for nodes with the same values. Any map node that contain the exact same property values as the proto-node would be a match. This did impose some limitations, for example it was not possible to search for two authors simultaneously, but it was possible to search for a node with a specific author and a specific set of keywords [9, p. 206][6, p. 314]. gIBIS and the proto-node search feature was not designed to handle large and complex IBIS maps, and was not deemed suitable for larger teams. However the simple query engine worked well for the smaller teams it was designed for.

2.2.2 QuestMap

Corporate Memory Systems Inc (CMSI) licensed gIBIS from MCC and created a new graphical IBIS tool named CM/1, later renamed to QuestMap. The beta of QuestMap was licensed to the company South California Edison (SCE) in 1992, as novel tool for synchronous capture of
operation decisions, and the rationale behind them. QuestMap took the role of supporting an asynchronous organizational memory, and being a file management tool\(^7\) [8, p. 6]. CMSI went out of business due to lack of sales a few years later, in 1996, but the software remained in use for more than a decade.

Figure 2.2: QuestMap

QuestMap had graphical hypertext and extensive group-ware\(^8\) capabilities, manifesting as two major system features. Figure 2.2 shows the interface of the QuestMap editor.

One feature was a file tracking system that supported shareable maps, that could embed other maps, for keeping personal notes and references. The other was support for group discourse mapping, where a facilitator would use the tool to capture discourse and rationale with IBIS during meetings. Facilitators were responsible for leading discussions and map participants’ thoughts and ideas accurately to the relevant discourse maps. Dr. Jeffrey Conklin, a QuestMap project manager at the time of introduction, performed a ten year industrial strength study of QuestMap at SCE. CMSI had trained staff and facilitators in how to use the tool, and how to use the argumentation method of IBIS most effectively. It was

---

\(^7\)Preserving notes, decisions and rationale that could be reviewed at a later time  
\(^8\)Software that support collaboration
later revealed that the commercial IBIS application struggled with limited adoption amongst staff members and managers.

The training was posit at one of the primary reasons QuestMap failed to gain user adoption. QuestMap required a fairly good understanding of the software to be used correctly, and many of the less technically skilled collaborators never learned how to use the tool as it was intended, struggling with the concept of hyper-linked nodes, and how to manage data [8, p. 7]. Staff members did not understand well enough the capture process, and the IBIS language, in order to articulate their thoughts, and did not discover what QuestMap was capable of as hyper-text software tool. The tool itself was complex and difficult to use, and users did not receive experience short term rewards from using it, as the tool slowed down the communication processes.

2.2.3 Compendium and Compendium Next Generation

A few years later, Verizon research labs then developed the Compendium as open source software, based on gIBIS and QuestMap [26]. Figure 2.3 show the interface of Compendium. On first glance it looks very similar to QuestMap. It is based on the same interpretation of IBIS as QuestMap. Meaning that it follow the same the rules for governing creation and use of IBIS nodes, and the possible relationships between them. Compendium is primarily designed as a group discourse mapping tool supported by facilitators, but supported discourse mapping in both individual and group work scenarios.

![Figure 2.3: Compendium](image)
Compared to gIBIS and QuestMap, Compendium is designed with
more emphasis on user experience. Meaning it is easier to integrate with
other IBIS systems. It allow sharing maps between IBIS systems, and
automatic import and export of data. Map items in Compendium can
also be shared between multiple map views, as well as individual maps.
Sharing items mean that updates will affect multiple maps simultaneously,
supporting the mapping processes of other groups. Compendium also
supported systematic modeling, using map templates to quickly describe
known problem scenarios, making it easier to outline the problem when
dealing with familiar situations.

Development on Compendium has been active for several years, but
now the software is no longer maintained. A community developed
edition named Compendium Next Generation [29] is still available, and
it is considered one of the better open source\textsuperscript{9} tools for discourse mapping
with IBIS. There is however little evidence indicating development activity
around Compendium and Compendium NG at this time.

2.2.4 DebateGraph

The DebateGraph project was started in 2005 by David Price and the
former Australian Prime Minister Peter Baldwin. DebateGraph has been
used in major political and news organizations such as Amanpour CNN,
The Independent. Perhaps most famously, as part of political participation
to advice the American President Barack Obama by the staff at the White
House.

Discourse in DebateGraph is based on the core IBIS nodes: Questions,
ideas and arguments, described in detail further into the chapter. Interest-
ingly each node also serves as blog or wiki page, for annotating map
nodes with detailed comments or notes. Collaborators can even rate argu-
ments, and get a visual indicator if the community considers the argument
to be weak or strong. Unlike the aforementioned IBIS system, Debate-
Graph is an application developed for the web, supporting useful features
such as embedding IBIS maps in external pages. It support both private
map spaces, for sensitive information, and public ones that are open.

Further DebateGraph has a many features for tuning and customizing
the visual output, supporting multiple layouts, for different use cases, and
user needs. It has basic search functionality, to search node labels and
descriptions, the search extends across all public maps, and private maps
where the user has access. This search interface works similarly to a text
based search engine. The input is typed in as text, and this text is matched
to a set of attributes existing on the maps and map nodes. The output is
a list of nodes, grouped according to the map they belong to. By selecting
an node from this list, the map moves to the area where the node is found.

\textsuperscript{9}Application code freely distributed, with minor restrictions
Figure 2.4 is an example of a map related to a search on solar energy. On the right side of the interface, the control panel with search input and a list of results grouped under the map they belong to.

One very powerful feature of DebateGraph is the sharing of IBIS nodes with other public maps, aggregating knowledge across different domains of discourse. Participants can get value from the contributions of others, and can deliver value themselves. For example a farmer owning land near a wind farm can view research and debates regarding wind mills, and the effect they have on the local community, even if the farmer is not the initiator, or a participant, of the discourse.

2.3 KnAllEdge

The KnAllEdge project was initiated by Sinisha Rudan and Sasha Rudan in 2010. KnAllEdge’s collaborative groups discuss knowledge, they raise important questions on knowledge topics, to contribute ideas and attribute new perspectives to existing knowledge truth\(^\text{10}\), and to evaluate knowledge.

KnAllEdge knowledge topics are inspired by the Topic Maps standard, a brief introduction to this standard is given further into the chapter. KnAllEdge use the Issue Based Information System to organize discussions about knowledge, capturing informal and formal knowledge, through discourse.

Each discourse map in KnAllEdge is based on a central topic. For example a group of collaborators that wish to discuss the outcome of

\(^{10}\)Different viewpoints on what is considered established facts
a recent research project. They can create a new topic map to discuss findings, compare them with related work, evaluate new results, and attribute the knowledge topic with what they found. The existing research can then be used to guide and support future research efforts.

It is also possible to embed related topics inside existing discourse maps, and embedded topics are seeds for their own discourse maps. Figure 2.5 shows a fragment of KnAllEdge IBIS map, showing the base knowledge topic on the left side, and three IBIS nodes on the right side. The figure is an example of how a knowledge topic, here annotated with %kn:KnAllEdge, and related discourse nodes, exist harmoniously in the same visual space. The fragment is a part the Nature of Creativity discourse, a topic map seeded on ideas from the famous Serbian inventor Nikola Tesla.

Figure 2.5: KnAllEdge IBIS Nodes

**Connecting Collaborators in KnAllEdge**

KnAllEdge connect collaborators with others that share common interests, through its own social network WhoAmI. WhoAmI is responsible for finding suitable matches for collaborative knowledge discourse mapping. WhoAmI uses special objects to form a relationship between an actant, human or non-human actor, and an interest. Several of these objects

---

11WhoAmI does not discern subjects, i.e. people, from objects. It follows the principle
grouped around an actant form a WhoAmI profile. The WhoAmI interests are descriptors that can describe any entity, a project, a task, an organization, and more.

Through WhoAmI, KnAllEdge is able to connect people together to perform knowledge mapping work, and it can connect collaborators to topics that are relevant for them. Looking at figure 2.6 we see an example of how two collaborators, marked with ampersand symbols, are associated with the technologies Linked Data and Resource Description Framework (RDF). We also see that viewpoints, experience and skill are defined using different semantic descriptors. Both actors are interested in these topics, with different perspectives, and they could both contribute with valuable knowledge on regarding future development, and use cases for these technologies.

WhoAmI is still in the early design and experimentation phase, as of February 27th 2016. But the roadmap outline social relationship suggestions, knowledge association suggestions, and augmenting collaborator interaction with the system, as the primary areas of future development and experimentation.
**Topic Maps**

Topic Maps is an industry standard\textsuperscript{12} for describing information as topics and associating these with information resources [22]. A way to describe intangible concepts with tangible topics. Topics are subjects of information that can be used to describe all conceivable concepts and things. In KnAllEdge, topics are represented as knowledge nodes, as we previously saw in figure 2.5. These knowledge nodes are an essential part of KnAllEdge knowledge spaces, described further into this chapter.

A strong benefit gained from topic maps is that the topic map layers refine information subjects without modifying them. They describe superficial data about them, and they are not directly linked. In the case of an external information resource, for example a topic that reference a research finding, the changes are not propagated to the research document, but form a superimposed descriptive layer within KnAllEdge.

### 2.4 Issue Based Information System

IBIS was proposed in 1970 by Werner Kunz and Horst W. J. Rittel as a tool for supporting coordination and planning of political processes. A documentation system for tackling wicked problems, ill defined problems with unclear and changing requirements, and widening the scope of discussion [9, 17, 23]. It has been applied to various projects in areas of planning, administration, teaching, design and development of software [23, p. 277][26].

IBIS aid in existence and communication, two important domains of concern [9]. Existence means it make information visible when needed, and the history of reasoning behind its current state is preserved. And IBIS support communication through the simple and powerful language elements.

#### 2.4.1 Collaborative Problem Solving

IBIS is modeled on the conversational approach that collaborators use for solving wicked problems, mapping discourse between collaborators into IBIS maps. In problem areas where many people are encouraged to follow one ideology or behavior, such as research, politics, religion, and environmental planning, one often encounter wicked problems. A wicked problem is by nature vaguely defined and there are no optimal solutions, just good and bad solutions. Nancy Roberts outlined three strategies that can be used to manage and act on wicked problems: Authoritative; Put responsibility in the hands of few, Competitive; Weight opposing views against each other, and Collaborative; Invite collaborators to engage in

\textsuperscript{12}Topic Maps is formally standardized as ISO/IEC 13250:2003 [15]
joint problem solving [24], IBIS relates to the last strategy. Discussion and collaboration allows collaborators to debate on how to manage these kinds of problems, and formally agree on a solution [9, 17].

### 2.4.2 Discourse Mapping

The initiation of a discourse process is the identification of the problem domain, a root topic. Collaborators can then create issue nodes that define the scope of discussion within this domain. They can assume positions to the issues, and further support or oppose each others positions through arguments. The discourse can be extended with new or related issues, and more ideas and arguments can be added if needed.

Let us look at a simple example of how IBIS discourse unfolds: A female politician in the Norwegian government initiates a IBIS discourse with the topic *Minimum wages*. The initiator also contribute an issue related to this topic, *Should we raise minimum wages for single-parent households?*. This narrows the scope of the topic and facilitate discourse on this specific issue.

Ideas are brought in when different positions are assumed, for example another collaborator could submit the idea that *Minimum wage is not sustainable for single-parent households*, and supports this idea with a strong argument. For example a reference to recent research on single-parent families. The number of problems related to minimum wages is likely much larger than just single parent households, so we can see how many different questions, ideas and arguments could be appended to this discourse.

The mapped discourse allows collaborators to study what has been discussed before, as issues, positions and arguments are preserved. The history is also evidence for decision-makers, they can support earlier decisions by pointing to the state of the discourse at the time the decision was made.

**Discourse Mapping in Group Sessions**

In this paper we refer to discourse mapping as the process of contributing to discourse maps. *Dialog mapping* is the process of discourse mapping, supported by facilitators, during group meetings [8]. Dialog mapping involves one or more facilitators that manage the capture process and facilitate meeting sessions. The facilitator has two roles, the first involves facilitative tasks such as leading the discussion, administrating time and tasks. The second role is to listen and write down key points in the discussion, e.g. opinions, challenges, questions, ideas and arguments.

Dialog mapping is a managed IBIS mapping session, addressing the problem of agreement on level of detail and abstraction in articulation. In the context of IBIS this problem can emerge as disagreement over
which language style to use when formulating node labels, or specifics around how to describe the problem domain when mapping problems and solutions [8, p. 5]. The facilitator greatly negates this issue. And further enables the participants to focus on the problems at hand instead of how to describe them. Participants can still validate that their contributions are heard by studying the mapping process on a shared monitor or projector. They observe changes in real time as the facilitator types them in. If participants disagree or feel that their contributions are not adequately mapped, they can voice their concern.

It is the facilitator’s responsibility to ensure all members are heard, and that viewpoints, opinions and facts are mapped adequately. The facilitator has the responsibility of listening, making sure the participants stay on track, capturing all conversation that is relevant, and to follow correct use of IBIS terminology.

But dialog mapping is not without fault, for example the facilitator can fail to capture information with appropriate IBIS nodes, or capture information without sufficient level of detail. In some cases the group does not pay enough attention to the map and have no ownership to the conversation there, and then miss errors of capture as they emerge. In other cases the discourse lacks structure and serves better use as a meeting summary.

However there are also significant benefits of capturing IBIS discourse this way. Facilitators relieve the collaborators of the responsibility to fully learn IBIS, the facilitators are trained in the use of the necessary systems and IBIS itself [8, p. 9]. Part of this training is learning how to transcribe meeting conversations using the IBIS terminology, becoming fluent in IBIS. As well as learning how to listen to conversations, and map the questions, ideas and arguments to the discourse map with a high level of accuracy. While facilitators play an important role in dialog mapping, they are not strictly required for the mapping process. As members of the group can also become fluent through training [8, p. 119], just like one would learn a new language, and they do become more more proficient over time. Having group members with IBIS experience can be useful for smaller sessions, or if the facilitator is unavailable, in these scenarios the group can do the mapping themselves.

2.4.3 Discourse Nodes

Discourse maps are built with discourse nodes, and three core node types make up the IBIS language. The original IBIS specification named these issues, positions, and arguments [17]. But using KnAllEdge terminology, we refer to these as questions, ideas, and arguments. KnAllEdge also include topic and comment node types, making five node types in total. We are familiar with the topic type from earlier in this chapter, and the comment
node is for notes. It can be used to manage cases where collaborators might not find a suitable node for adhering a piece of information to the map, or for drafting ideas and thoughts. Comments can be placed anywhere on the map, and refer to any other node. But comment nodes are not part of the discourse structure, and outside the scope of this study.

### 2.4.4 Node Relationships

The supported rhetorical moves, the possible relationships between discourse nodes, can vary between the different IBIS implementations. Conklin defined nine kinds of node relationships to support the requirements of gIBIS, adding four relationships to Kunz and Rittel’s original IBIS specification [9][8, p. 2][6, p. 305].

In figure 2.7 we see the relationships supported by KnAllEdge, these relationships dictate how nodes can be linked in KnAllEdge maps:

- Questions may generalize or specialize other questions
- Questions may question other questions, ideas and arguments
- Ideas can respond to questions
- Ideas can generalize or specialize other ideas
- Arguments can support or oppose ideas
- Arguments can generalize or specialize other arguments

![Figure 2.7: KnAllEdge IBIS Node Relationships](image)

Questions can specialize other questions to provide partial answers, or it can generalize them to expand the domain of the question to cover more generic cases. Questions can also question a question, idea or argument. This helps avoiding negative socio-technical translation [31], collaborators that misuse ideas and argument nodes to question other ideas and arguments. Like questions, ideas and arguments can generalize and specialize other nodes of the same type. This supports fine grained
discussions in a specialized area of the discourse, and discussions covering a larger area of the topic domain. Collaborators can narrow down or expand on existing discourse, based on their domain of expertise and their needs. IBIS has received praise for this rhetorical nature of information organization [9, p. 210].

2.4.5 Map Layout

An IBIS discourse map is essentially a hierarchical information graph. A graph is a set of nodes interconnected by links, an implementation of the abstract mathematical model of nodes or vertices that can be connected together via edges or links. Computers can easily read and parse node objects, and links, and when visualized the graph is human readable as well. And IBIS maps convey discourse information to collaborators visually, i.e. topics, questions, ideas and arguments have labels, descriptions and symbols for visualizing information.

Links convey information about which nodes are connected, defining the node to node relationship. In KnAllEdge IBIS, links are objects, and they contain several properties for describing the relationship between two nodes. All links are also directed, they have a beginning and an end, and they make it possible to tell where a discourse begins and where it ends. Links are important because they provide the semantics between discourse nodes, enabling computers, and collaborators, to reason about the structure of the graph. For example if a question X is a specialization question Y, then the link identifies the source and target, and we learn that X specialize Y. And a collaborator can visually trace the discourse from the root of the discourse, to each leaf node, the last node on a branch, as we see in figure 2.8.

The layout in figure 2.8 shows a common tree structured IBIS layout [8], KnAllEdge use a similar layout, as seen further into the chapter. With this layout, nodes are positioned beginning with the root node on the left, and ending in one or more leaf nodes on the right. The discourse branches out like a tree towards the right.

In this example the links point from the child to the parent nodes, but the layout direction is direction is left to right. In KnAllEdge the link points from the parent to the child, but the link direction is not explicitly marked.

Figure 2.9 shows an overview of the Nature of Creativity map in KnAllEdge, it has a clear visual resemblance to the previous figure. We find that IBIS often looks similar in different systems, sharing similar question, idea and argument marking, and a tree like graph layout.

13Positioning of nodes in the visual space
2.4.6 Creating Discourse Contributions

IBIS does not impose strict rules for contributing nodes to a discourse. Ideally one asks questions with well worded text, and perhaps include references for why the question is important enough to warrant collaborators’ time, making sure the question is well defined and accurate according to the particular situation where it is asked. Combining the research of Kunz and Noble, we define well formed questions as belonging to one of five categories [17, 23]:

- Factual (Observation) - Is X the case?
- Deontic - Should X become the case?
- Explanatory (Causality) - Is X the reason for Y?
- Instrumental - Is X the appropriate means to accomplish Y in this situation?
• Conceptual - What do you mean by X?

In order to make sure a question fits in the correct category for the problem domain it belongs to, it is recommended to only contribute when one has a reasonable level of understanding the problem domain [17]. It is not always beneficial when collaborators, who does not understand the topic well enough, attempt to contribute on a best effort basis. Usually experienced collaborators will have more experience with similar questions or problem cases, and is in a better position to ask the important questions and give reasonable answers.

To avoid confusing those who read the questions, Kunz recommend to describe or label questions with the precise state of the problem [17, p. 3], to accurately describe the context and scope of the question in the label text. However, Conklin recommended to gIBIS collaborators to write discrete questions, to make them separate from their surrounding context [10, p. 250]. But these are recommendations, and not strict rules, both highly contextual and discrete questions is likely to be found in the same discourse. If collaborators always write questions with the proper context, it would be easier for other’s to understand what they are asking, but for questions, ideas and arguments that is a specialization, one should not be too explicit. For example a discourse map with the topic Income Taxes. There is a question Should we reduce income taxes?; and a another question that is a specialization of the previous one, Should income taxes be reduced for family households?; the third question Should income taxes be reduced for single parent family households?. While the reader would get a very precise idea of what the question is about, the label could eventually become very long.
Good ideas often come from open ended questions\textsuperscript{14} [8, p. 120]. And a good understanding the problem domain is also beneficial to have, to be able to contribute with creative and reasonable ideas. And when constructing arguments to sway opponents in the decision process.

While arguments are the least common contribution to IBIS maps [8, p. 3], it's important to understand that arguments strengthen or weaken, questions and ideas, and incite one to create new questions. They are the most important node in forming the testimonial body of evidence that influence the decision making process. Without arguments, decisions made with the evidence at hand, will be difficult to justify. The difference between an idea and an argument is not always clear, as arguments do not necessarily express an opinion on the matter. But the argument is declared as a supportive or opposing argument, and it common for IBIS software to visualize the distinction using colors or symbols.

Looking back at the discourse stemming from the Nature of Creativity topic. Branching out from the NoC topic a collaborator has submitted a explanatory question labeled models of creativity, and in turn this question seeded the ideas: Phenomenological model, ideological model, and biophysical model. The ideological model has one argument, indirectly creative mode of consciousness. This argument also links to related research, reflected in the argument label. This argument does not express the opinion of the creator, as one might expect from an argument. It does however express support of the idea through the argument link, in this case the research as an argument validates the existence of the idea.

Note that despite this, IBIS gives a lot of freedom in terms of labeling questions, ideas and arguments. It is up to collaborators to contribute in accordance with established norms and rules of the domain of discussion and the collaborative environment. Ideally a question should only ask one thing, ideas only contain one response, and arguments only contain one piece of evidence in support of an idea, or to reject it [9, p. 207]. The context of individual nodes should be discreet from their surrounding context, it is difficult to anticipate where the reader came from, and their previous understanding of the matter [10, p. 250]

However, IBIS has spawned a family of similar yet different systems, where each implementation is tailored to the needs of particular domains. IBIS guidelines does not dictate that contributions must follow a formal template or best practice, but it is recommended [23]. Consequently grammar and syntax rules are not generalized across systems, and this responsibility is left to the implementors.

\textsuperscript{14}Questions that require more than a yes or no answer


2.4.7 Making Decisions

IBIS discourse is sometimes referred to as a decision space, because solution in IBIS are ultimately decided upon before action is taken to implement or reject proposed solutions. How decisions are made and discourses settled in IBIS is decided by the collaborators involved, or by an external decision process in the IBIS system.

Even if the discourse at hand is actively updated and changed, decisions can be made based on current progress. It is common to make early decisions on questions that have a sufficient pool of strong ideas and arguments, even though the discourse is actively growing.

Settling questions means to convince the deciding authority, often the collaborative group itself, that certain ideas are better or worse than others. The authority make a decision by studying the supportive and opposing arguments. This can also be achieved with a formal decision procedure [17], considered a more efficient approach when many collaborators are involved. This process can also be fully or partially automated. In an case, the evaluation process to determine the strength of arguments must be agreed upon.

2.5 KnAllEdge Knowledge Space

The KnAllEdge team is currently researching and developing KnAllEdge knowledge space, to extend the existing IBIS decision space. While unfinished, it is very much an essential part of knowledge federation in CollaboScience, and we will end this chapter with a brief introduction.

As we have learned, KnAllEdge is utilize IBIS for a more than managing problems through discourse. CollaboScience aim for highly integrated knowledge sharing between KnAllEdge and the other services that exist in the CS toolkit. In a sense, KnAllEdge is evolving IBIS to become issue based knowledge transformation procedures. It preserves the discourse core of the IBIS system, that means the nodes and discourse mapping process from IBIS is preserved. But it also adds nodes representing knowledge space. So KnAllEdge is not only transforming discourse to decisions, but also transforming discourse to knowledge.

KnAllEdge also implements knowledge extraction from IBIS. In standard IBIS it is common that knowledge artifacts are generated during the discourse process, but IBIS does not have constructs for preserving knowledge data. The problem is exposing the knowledge embedded in the discourse and embedded sub-discourses, and gathering consensus of discourse on topic. Note the distinction between marking an idea as solution to a problem and consensus. Consensus is a higher level agreement on what knowledge to extract from the discourse.

In a dialog mapping scenario, extracting knowledge could be done by
facilitators or other mediators that are tasked to transcribe discussions to IBIS discourses, but then we add additional overhead work for managing information maps. For IBIS discourses that are not mediated, then the collaborators have to do it by themselves. But this is detrimental to reducing the cognitive overhead of using IBIS for organizing information, and cognitive overhead has historically been a substantial obstacle for IBIS adoption [26].

KnAllEdge does not exclude the traditional IBIS decisions space, as debating conflicting views regarding any topic is an intrinsic part of the discourse, but deciding if idea A is stronger than idea B is not the only goal, aggregating the knowledge they contribute with is just as important.

2.5.1 Knowledge Consensus Mechanism

Using KnAllEdge as a knowledge consensus mechanism is one important use case. If a collaborator feels that some particular topic is not adequately solidified\(^\text{15}\), or that a consensus has not been reached on the truth of knowledge behind the topic, he or she can create a discourse to resolve this issue. KnAllEdge describes this as an orthogonal discourse, it is separated from the knowledge space until consensus has been reached. At this point the consensus is merged into the knowledge topic.

![Figure 2.10: KnAllEdge Knowledge Nodes and Decision Nodes](image)

As we see in figure 2.10, the IBIS nodes are tagged with \%ibis tags. This means that the node is part of the IBIS name-space, semantically separated from the \%kn knowledge space.

2.5.2 Knowledge Polarization and Argumentation

In some cases it is not possible to get consensus on a knowledge topic, or the knowledge cannot be expressed with facts and evidence. In other cases this is due to the nature of the knowledge domain, dealing with a wicked problem, as previously described in this chapter. Collaborators

\(^\text{15}\)Fuzzy or lacking detail
can distance themselves from an existing knowledge truth, and construct partially or fully disjoint discourses, exposing a different view on the topic.

Polarized views and opinions on such topics can be filtered based on what KnAllEdge knows about the collaborator, such as interests and viewpoints. Effectively personalizing the knowledge truth. In the case of collaborators with strong religious affiliation, it makes sense that topics regarding the evolution of mankind emphasize and consolidate discourse supporting views of a metaphysical nature\textsuperscript{16}. KnAllEdge preserves these multiple discourses as disparate truths, a measure to prevent edit wars due to conflicts between communities and individual content authors.

Figure 2.11 is an illustrative example of author edit wars in WikiPedia, heated discussions regarding what should be written on a wiki page. Content authors on wiki pages can sometimes fiercely oppose the views of other authors, leading to long arguments and aggressive methods to establish their own view as the one and only knowledge truth [30]. One such method is to overwrite other authors’ contributions, and this process gets repeated every time someone with a different view try to alter the wiki document.

![Figure 2.11: WikiPedia Edit Wars](image)

\textsuperscript{16}Viewpoints supported by divine or spiritual evidence
2.6 The Knowledge Federation

Amidst the wake of these discoveries during 20th century, the organization Knowledge Federation has emerged as a trans-disciplinary research prototype on how to achieve knowledge federation, and augmenting the capabilities mankind’s collective mind. KF recognized that we already have the technology for developing work spaces for collective thinking, and even more efficient knowledge management. This organization has dedicated much effort towards augmenting collective mind through trans-disciplinary research and projects, continuing Dag Engelbart’s unfinished revolution.

The advancing technological development during the last decades, has enabled KF to streamline the collective mind re-evolution. And KF are now successfully merging multiple disciplines into trans-disciplines, and enabling knowledge federation across trans-disciplinary domains. Being trans-disciplinary means that researchers, entrepreneurs, journalists, information architects and other collaborators work together to perform innovation work, and federate knowledge across their respective disciplines.

They created the CollaboFramework for this purpose, a toolkit of many parts that can be organized and combined to form different systems according to what is needed to support different trans-disciplinary groups. The systems support self-organization of work, and federation of knowledge as a collective mind. Much of the work at KF is devoted to systemic innovation. Supporting research, testing and implementation of new systemic solutions in science, education, finance and other societal systems. And KF use itself as a sandbox for developing and evaluating the solutions. KnAllEdge is a knowledge mapping layer that support the CollaboFramework toolkit, the trans-disciplinary groups use KnAllEdge to manage and extend knowledge through discourse, and this knowledge is made transparently accessible to the other tools and services.

KF has positioned itself as an evolving answer on how to perform knowledge federation. Based on Engelbart’s bootstrapping strategy, the tools developed for improving collective IQ are also used to improve the effectiveness of the development process itself. The outcome of these processes are new methods and new tools, always pushing their own boundaries and challenging established knowledge work paradigms.

2.7 Summary

In this chapter we learned of the beginning of the collective intelligence and knowledge federation movement, and how Knowledge Federation
and CollaboScience is utilizing KnAllEdge to model knowledge for transdisciplinary knowledge work. Through the different generations of IBIS, we have seen how problem solving and discourse management is becoming an important supportive structure for sustainable knowledge creation and federation spaces. We have seen how collaborative teams engage in discourse mapping with KnAllEdge IBIS, and the rules and requirements that govern the mapping processes. The last part of this chapter gave an introduction to the next step for KnAllEdge, knowledge spaces, the bespoke mechanisms for augmenting knowledge with knowledge consensus, and polarized knowledge truths.
Chapter 3

Meta-data Queries

This chapter define a specification for a prototype capable of performing contextual queries in KnAllEdge IBIS discourse maps. The contextual query mechanism is given the name meta-data queries 1.

3.1 Node Searching in KnAllEdge

The meta-data query specification is an experimental approach to performing KnAllEdge IBIS search. To navigate KnAllEdge discourse maps collaborators can pan around the map, and expand and retract nodes. But it does not have discourse search capability. Meta-data queries fill this role, providing contextual search and a simple query language.

To support the traditional text search interface that we find in modern IBIS tools like Compendium NG and DebateGraph, meta-data queries also support query-less content search. Content is a reference to any kind of visible node text, labels, notes, and descriptions. In visual data maps, we find that searching for descriptions and labels are common strategies that users go to, if they have a text search interface at their disposal [28, p. 803]. Collaborators familiar with earlier IBIS systems that support text search, will likely appreciate that KnAllEdge support this familiar feature.

The query language itself is used to target specific properties and values of KnAllEdge discourse nodes, such as date, author, node type and more. These queries use a prefix:suffix syntax, where the prefix define the scope of discourse nodes to search. A scope is number of discourse nodes, with specific properties and values, that are evaluated in the context of a given prefix. The suffix filter out nodes from this scope, using a mapping function, that also decide which properties to filter on. Most prefixes match a single generic property that all map nodes have. In this case it is common that the suffix is matched to the value of this property, but this is not always the case, as we will see later in this chapter.

1The origin of the name meta-data queries come from the word meta, it means to reference oneself or itself.
A single query can match multiples nodes in the discourse, and multiple queries can be submitted as part of a single search. The output of the search process is a collection of map nodes, containing a contextual history of contributions, related to the search parameters.

### 3.2 KnAllEdge Map Data

At the time of writing, KnAllEdge use a single database for storing discourse maps. When a collaborator opens the KnAllEdge map selection screen, KnAllEdge display a collection of maps, from a map collection called *kMaps*. When the user have selected a map, KnAllEdge will fetch the collections that contain the map data and visualize the map. Two of these collections are important for the meta-data query mechanism, the *kNode* node collection and the *kEdge* link collection. The node and link objects in these collections share the same alphanumerical identifier property, that is a reference to the parent *kMap*.

#### kNode Collections

A *kNode* collection contain all the nodes for a given *kMap*, and each node in the collection share the same set of properties. Listing 3.1 show some of the properties that a node has, a complete summary is given further into the chapter.

```
1
2   
3   "id" : "5552df907ffddcd74096d0f1",
4   "name" : "Interaction Types",
5   "type" : "type_ibis_idea",
6   "mapId" : "5566f25867a6d01e65beddde",
7   "iAmId" : "5548fee2e59efb4e0bb8478",
8     ...
```

Listing 3.1: Node Properties

Each node property and value is essentially a key and value pair. All nodes contain the same properties, the same keys, but contain different values. For example the *type* property in listing 3.1 has the value *type_ibis_idea*, and if we look at any other node we will find the same *type* property, but the value could be *type_knowledge* or a different type value. There are also unique node values, for example the value of the *id* property is unique for every node in the KnAllEdge database.

---

2A combination of alphabetic characters and numbers.
kEdge Collections

A kEdge collection contain all the links for a given kMap. Link objects are the glue that enable KnAllEdge to associate nodes with each other, so they can be organized correctly. Links contain different properties than the nodes, listing 3.2 show the most essential properties, the unique identifier, the map it belongs to, and the source and target identifiers. Like the kNodes, a link has an unique identifier.

```
{
    "_id" : "5552d7c17ffdcdd74096d0d0",
    "mapId" : "5566f25867a6d01e65beddce",
    "sourceId" : "5552df907ffdcdd74096d0f1",
    "targetId" : "5552d7c17ffdcdd74096d0cf",
    ...
}
```

Listing 3.2 : Link Properties

From chapter 2 we learned that nodes can specialize, generalize, support and object, to other nodes. This is based on the properties of the links. Even though, at the time of writing, the KnAllEdge developers had not explicitly defined how IBIS and KnAllEdge node relationship would be implemented, it is likely that these relationships would be defined in the link objects. In their current state, we find that they already define one type of relationship, specifically how nodes should be organized in the map.

A node to node relationship is always directed in KnAllEdge IBIS, it is a pointer from one node to another, from the parent to the child. For example if a collaborator connect an idea to a question, the idea is then the child and the question is the parent. To identify the parent we look at the identifier on the sourceId property, seen on line 5 in listing 3.2. The value of this property match the _id property the node in listing 3.1. The same way the value of the targetId property will match an _id property on the child node.

All nodes in a map has a relationship with at least one other node, and two nodes are only related if there is a link between them. Collaborators contribute new nodes, and read existing nodes, in the order parent-to-child. The latter is the link direction.

Visualizing a Discourse Map

To construct a map from the kNode and kEdge collections, KnAllEdge connect all the nodes using the link objects. The resulting map is an acyclic directed graph. Because a relationship between two nodes is always directed, the resulting map is also directed. There are no link loops, it
is not possible to follow a directed link, and end up at a previously visited node, hence it is acyclic.

**Map Layers and Visual Nodes**

KnAllEdge use visualization layers instead of simply visualizing the map data itself. Visual layers can be used for many purposes, for example to visualize abstractions of the data, or visualize the map with different information perspectives. This does not require any change in the data structure, and no manipulation of node properties in the data layer. \( kNodes \) and \( kEdges \) exist in the data layer. The visual layer is composed of \( vkNodes \) and \( vkEdges \), these represent the \( kNodes \) and \( kEdges \) when visualizing the discourse map, and can apply many different types of visualization modifications.

### 3.3 Meta-data Queries change the Data Layer

The prototype is essentially a filter for the \( kNode \) collection, it fetches this collection from KnAllEdge, process it according to the submitted queries, and produce a new \( kNode \) collection. No changes are made to the properties of any \( kNode \). The \( kEdge \), \( vkNode \), and \( vkNode \) collections are also left untouched.

The query processes that change the node collection are idempotent. Successive queries with identical input, will produce the same output. Assuming that the data layer has not been modified by another process between queries. However the visual output of identical queries is not necessarily idempotent. KnAllEdge can potentially apply different visualizations, or reconfigure the layout, depending on the outcome of a query.

### 3.4 Submitting Meta-data Queries

A text input area makes up the query interface for the prototype. The field is where the collaborator can submit perform searches and submit queries. The input is required to be text, a string of characters that follow a specific syntax. Additionally collaborators can search for node content using simple text search, because the prototype append a special prefix to queries without a pre-selected prefix.
3.4.1 Query Syntax

A meta-data query has three parts: A prefix, a colon (:) separator, and a suffix. For example Type:type_ibis_question. This is the same JSON\textsuperscript{3} format that KnAllEdge use internally, to represent a single property and value on an IBIS node.

Figure 3.1 show a composed query, where each part follow this prefix:suffix syntax. The prefix is the group of characters leading the colon separator, the suffix is the character group trailing the separator.

Figure 3.1: Composing Meta-data Queries

3.4.2 Prefix Mapping

A prefix represents an abstraction for a set of queries that target any number of properties and values, and define the scope of meta-data that will be searched.

For example with the query "Name:Directly Creative Mode of Consciousness", the Name prefix means that the query match the name property of the node, and the prototype evaluate the suffix Directly Creative Mode of Consciousness to the value of this property. The scope of this query is the whole discourse, as the property exist on every node.

In this second example query "Argument: Classical Science", the Argument prefix is bound to the value of a node’s type property, and the

\textsuperscript{3}JavaScript Object Notation
suffix is compared to the node’s name property. In this case the scope of the query is argument nodes.

As we see from these examples, the suffix value, the second part of a query, specifies the information we are interested in within the prefix scope.

### 3.4.3 Composing Queries

The search string in figure 3.1 is two queries composed as a single string, and we see that each query is separated by a delimiter, the comma character. Each query in a composed string is independent, and does not influence the outcome of any other query in the string. And each query is based on the same prefix and suffix syntax as a individual query.

From the example in figure 3.1, with the composed query “Knowledge:TNC Online, Question: Collective Mind”. In this case the collaborator was interested in finding out more about collective mind, and its relation to the question of collective mind.

### 3.5 Node Properties and Prefixes

Here we will look at the properties and values of KnAllEdge kNodes, what their purpose is, and the prefixes that will represent them in the prototype search interface.

Table 3.1 show all the properties that exist on a node, and the first column is their actual property name. The prefix in the second column list the human readable label that is mapped to the property. The camel case notation is the naming convention used for kNode property names, compound phrases where each word, except the first, begin with a capital letter, giving the impression that the phrase has humps, hence the name. The property names are meant to be understood by the system developers, and not the users. The prefix that represent each property was labeled to be more succinct and easier to understand.

The third column include a brief description of the property, and extended with a more detailed description later.

Some of descriptions mark the property as not used. These properties are either deprecated, remnants from earlier beta phases, or contain content that is not query-able. For example the sub-properties of dataContent property, except property, contain additional meta-data such as images. The prototype only allows queries on content that serve a purpose for a collaborator, so these properties are ignored for the time being.
<table>
<thead>
<tr>
<th>Property Name</th>
<th>Prefix</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name &amp; Text</td>
<td>Node label</td>
</tr>
<tr>
<td>type</td>
<td>Type</td>
<td>Node type</td>
</tr>
<tr>
<td>iAmId</td>
<td>UserId</td>
<td>Id of the node creator</td>
</tr>
<tr>
<td>updatedAt</td>
<td>Updated</td>
<td>The date and time of last modification</td>
</tr>
<tr>
<td>createdAt</td>
<td>Created</td>
<td>The date and time it was created</td>
</tr>
<tr>
<td>ideaId</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>__v</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>version</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>activeVersion</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>mapId</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>dataContent</td>
<td>Text</td>
<td>Additional meta-data and node content</td>
</tr>
<tr>
<td>visual</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>isPublic</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>_id</td>
<td>-</td>
<td>Not used</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Origin</th>
<th>Property Value</th>
<th>Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>KnAllEdge</td>
<td>type_knowledge</td>
<td>Knowledge</td>
</tr>
<tr>
<td>IBIS</td>
<td>type_ibis_question</td>
<td>Question</td>
</tr>
<tr>
<td>IBIS</td>
<td>type_ibis_idea</td>
<td>Idea</td>
</tr>
<tr>
<td>IBIS</td>
<td>type_ibis_argument</td>
<td>Argument</td>
</tr>
</tbody>
</table>

Table 3.1: Node Property Table

**Type**

The *Type* prefix is mapped to the *type* property. This property define the node type, i.e. knowledge, question, idea, or argument.

At the time of writing, only four values can be placed here. They follow different naming conventions, depending on their origin. For KnAllEdge nodes the pattern begin with *type_* and then the type name. For example *type:type_knowledge* for knowledge nodes. IBIS nodes begin with *type_* followed by *ibis_* and the type name. For example *type:type_ibis_idea* for idea nodes.

Each value from table 3.2 is mapped to its own prefix, with a name that represent its type. Collaborators can use these prefixes to query the node label on either one of these four types.

Table 3.2: Node Types
Name

The name property represents a node’s visual label, and is represented by a Name prefix. This label visually cover a significant area of each node. This property can also be searched with the Text property, as mentioned earlier in this chapter.

Author

The Author prefix is mapped to the iAmId property, and represents the author or creator of the node. This identifier is associated with a specific WhoAmI user profile.

The Author prefix is a placeholder, a temporary solution. The iAmId property value is an identifier and not the name of the author as the prefix suggest. This can be resolved if the prototype is given access to the list of WhoAmI user names, in order to resolve these identifiers, or KnAllEdge appends the actual name of the author as a different property.

Updated

The Updated prefix is mapped to the updatedAt property, and represent the date and time at which the node was last updated.

Created

The Created prefix is mapped to the createdAt property, and represent the date and time at which the node was created.

Text

There are two node properties in kNodes, at the time of writing, that describe node content. The name property, and the property sub-property of the dataContent property. The name property is what the collaborator see as the node label, and the property sub-property is a more detailed description that can be appended to a node.

When a collaborator submit a search without an explicit query, the prototype appends the Text prefix, and the original search string becomes a suffix. The prototype will then compare the value of name and property sub-properties to this suffix.

In figure 3.2, we see an example of how a content search looks like. In this example the search found a matching string in the name property of an argument node. In this figure, the idea Quantum system can behave as memory is included, to illustrate the branching structure. But this node is not part of the discussion history leading up the the searched node, and would be excluded from a contextual meta-data search.
3.6 Context Extraction

So far we have covered syntax, structure and use of meta-data queries, the relevant data properties in KnAllEdge and how the queries match these. This section describe the process of extracting contextual history, or paths, from the discourse maps. The specification declare a topic as the root of the context, and extracts all nodes between each node that match a query and a the context root. With this method, collaborator will receive the full history of individual discourse branches.

Meta-data queries extract the history between a topic root and the query, and does not allow questions, ideas or arguments to be the context root. Only a topic can be the root context, it is the single most important piece of context [17, p. 7] [7, p. 363], the problem domain on which the discourse is formed. Questions shape or expand the discourse in different directions, and they raised in the presence of topics [17, p. 4-5] [7, p. 363]. Subsequently an idea can only be raised on a question and arguments can only be presented on ideas.

KnAllEdge allow related topics to be embedded in the map, and these topics can have their own maps with separate discourses. These topics are also valid context roots, but the benefits of allowing collaborators to only query specific embedded topics is uncertain, and would be more suitably implemented after user experiments with the current state of the specification.

A single linear history from the context root is extracted, with a process that is designed to extract the path between two arbitrary nodes. Because this process does not discern what nodes it works with, it will be easier to accommodate for changes, such as root context switching or reducing
the length of the context history. This process extract one path for each discourse node that returned a match from a search. Each time this process runs, a collection of nodes is created if a path can be found. This collection contain a sequence of nodes, the two nodes submitted to the process and all nodes that connect these two in the discourse. This process is executed for each node that match a query or set of queries. So if the query Author: Sally Smith yields five matching nodes, the process run five times.

3.7 Summary

The meta-data queries specification describe a search tool that can be used to search for combinations of properties and values. Meta-data queries map all useful properties and values, so they can be readily searched. But the mapping functions can also match specific combinations of properties and values to a single prefix, this enable only search within desirable discourse scopes. The current specification includes a mapping for all relevant prefix values, but it is expected that more prefixes will be added in the future.
Chapter 4

Exploring Meta-data Queries

It is important to gather novel insight into how potential future collaborators interact with meta-data queries, as this approach use a different type of input than traditional text search. There is always flaws in a system, and sometimes the users expect other functionality than what they are given.

This chapter is dedicated to a workshop held at the University of Oslo, where a group of participants were invited to try out meta-data queries. The workshop was semi-structured, this meant that participants engaged individual problem solving as well as open discussions.

The workshop was divided into two parts, where the first part was dedicated to testing a minimum viable product (MVP), a basic testable implementation prototype. The participants followed a set of information seeking tasks, and used query terms to find nodes in a IBIS map. The second part engaged participants in a group discussion, the author asked a few questions to set the topic of the discussion and let the participants speak freely.

4.1 The Prototype

This workshop used a paper prototype, a simple construct that was not expensive to prepare and make. This kind of testing yielded data much faster compared with the time it would take to develop the advanced prototype. And further unearthing flaws in the approach, the workshop itself, and potentially making new discoveries that are not part of the design. The prototype did not include any computer tool or handbooks, and compared to a software prototype, participants could not learn through trial and error 1.

In the process of gathering data, test participants were tasked to create queries in order to answer questions. Another aspect of this testing was to study the outcome participants expected from queries, what kind of

---

1Testing with a live query interface would allow the user to test and learn based on the output received
results are expected from the prototype during use. These were the individual tasks that the participants solved.

The group tasks engaged participants in discussion regarding the information context of contributions, we also see the term discourse context for this, and how meta-data queries could solve cases where nodes did not describe the information context in their label.

**Experimental Query Syntax**

As part of the meta-data query tasks, the participants were given the option of using an experimental syntax for meta-data queries. This syntax was not described in chapter 3. These queries had a `nodetype:prefix:suffix` syntax. The nodetype could be either Question, Idea or Argument, followed by Person and the person’s name. This would allow shorter and more specific queries targeting only a node with a specific type and creator.

In the case that the participant found this syntax comprehensible and easy to use, then they could use these in their answers, but they were not required to do so, and this was explicitly communicated before they started working on the tasks. The goal was to observe if participants understood how to use this syntax, and if they found it useful. Later in this chapter we will see the full list of queries that the participants could use when solving tasks.

### 4.2 Descriptive Investigation

Observations made during the workshop aims to provide important details about what is happening when users engage with the prototype. The observation aspect is important due to emotional and physical responses that are expressed from participants during the test, this provides valuable knowledge about participants’ personal experience when interacting with the prototype.

This kind of descriptive investigation does not provide data about casual relationships, i.e. whether meta-data queries are more or less effective, and intuitive, than other methods, and why or why not [18, p. 20]. But we observe the participants, looking for patterns of behavior, similarities and differences in task solving, results, and in participant opinions. Further because the workshop would involve multiple minds debating on the same issues, it could lead to discovery of novel ideas and solutions.
4.3 Semi Structured Focus Group

Study groups are essentially group interviews [18, p. 192]. Unlike one on one interviews, study groups are usually semi-structured or unstructured. Compared to a fully structured interview, the semi-structured focus group gives room for questions, and devote time for clarification and discussion. Both semi- and unstructured study groups enable discussions to form arenas where conflicting opinion can give rise to new ideas. However unstructured study groups can not have any form for structure, no list of tasks to solve, so the semi-structured version was a better fit for testing the query parts of the prototype.

If the workshop had been arranged as one on one interview, it would be difficult to gather honest matters of opinion, where participants agree and disagree, compared to what we get from study groups. In an interview, there would be greater risk that the author could accidentally introduce a bias of opinion, the participant could be more passive and task focused, and it would take much more time interviewing many participants in sequence. There would also be a higher risk that the participant would get uncomfortable when solving tasks alone, knowing they were observed during the process.

4.4 Preparation

In the early stages of preparation the author prepared an IBIS map to be used for testing the prototype. The KnAllEdge tool would ideally be used to create the map, but it was not a simple task to export the visual output of the map on paper. The interface is not designed for this purpose. Instead the map was much simpler to draw in a model editor, to ensure the structure of the map would not be broken when printed on paper.

A hour long panel debate named Women In Technology from The White House’s YouTube channel was transcribed by the author to create an discourse map. This debate between highly ranking leaders in several technology domains, was held at Facebook headquarters in Palo Alto in 2011, see appendix D for the entire map.

In total eight people participated in the debate, five from the debate panel and three from the audience. The map was written with the dialog mapping method, and the author took on the role as facilitator [8, p. 10], writing down questions, ideas and arguments as they were presented by the panel and audience. The net result was 12 questions and 27 ideas. An interesting note, and a good case for using IBIS: During the recording, the author noticed that the ideas presented by the panel were altogether poorly supported by factual arguments. In an IBIS map, weak and unsupported positions in a debate are clearly exposed.

The arguments were mostly grouped in a few specific areas, as some
debaters were better at backing up their claims than others. This made certain areas of the map very easy to read, as there were few nodes there. The author added 12 arguments, in total 39, adding more complex visual representation in those areas.

4.5 Participants

In total four participants were part of the workshop at UiO. All of the participants were master students on their second year at the Department of Informatics in 2015. Their ages spanned from 27 to 34 years old. Two from the interaction design branch of Informatics, and two from the programming and networks branch.

For this testing the assumption was made that the participants did not know the discourse corpus. Meaning that the participants were not familiar with the women in technology debate, and the opinions expressed by panel members and audience. The participant should not recognize the questions, ideas and arguments in the map when looking at it.

4.5.1 Relevance

Highly qualified academics, researcher and business professionals, constitute the majority of KnAllEdge collaborators [31]. They are above average proficient with information management tools, but many does not have experience with using IBIS systems.

Similarly, none of the test participants had previous knowledge or experience from other IBIS systems, but all were familiar information management tools such as email, shared text editors and calendar planners.

One of the programmers reported being highly proficient with the database query language Structured Query Language (SQL), and had several years of developer experience. The other reported a few years of developer experience, mostly school projects and assignments based on Java, C++ and JavaScript programming languages. Both of the interaction designers reported experience in the area of user interfaces and interaction, they had previously participated in projects that involved visual software design, and development of intuitive software tools during various projects.

All in all their experience was diverse within the computer software domain, and they all had some experience with information management tools, and they all had knowledge from both academic and non-academic projects. It was interesting to study how their previous experience would affect problem solving, and the differences in approach.
4.6 Material

The participants were given two sheets during the workshop introduction. One sheet contained the IBIS discourse map from the women in technology debate, the other contained tasks and related information:

- Table of query groups with prefix and suffix values.
- List of tasks to be individually solved using queries.
- List of questions for group discussion.

4.6.1 Query Groups

Five groups of meta-data queries were available to use when answering the tasks. The groups were extracted from questions, ideas and arguments on the IBIS map. Each group contained a single prefix and several suffix values the participants could use when solving the query tasks. The full length of the panel debate lasted for one hour, so none of the tasks related to finding nodes with a given date or time.

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Suffix</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Panel</td>
<td>All, Valerie, Sheryl, Jocelyn, Theresa, Jessica</td>
<td>Members of the discussion panel</td>
</tr>
<tr>
<td>Audience</td>
<td>All, Amy, Shelly, Kelly</td>
<td>Members of the audience who asked questions</td>
</tr>
<tr>
<td>Question</td>
<td>All, Text, Person:(*)</td>
<td>Questions asked from both panel and audience</td>
</tr>
<tr>
<td>Idea</td>
<td>All, Text, Person:(*)</td>
<td>Ideas suggested to questions</td>
</tr>
<tr>
<td>Argument</td>
<td>All, Positive, Negative, Text, Person:(*)</td>
<td>Arguments supporting ideas</td>
</tr>
</tbody>
</table>

*: Suffix value is either name of panel or audience member

Table 4.1: Query Groups

The Person suffix is synonymous to names of panel and audience members. The Person group was divided into two groups, reflecting the distinction between the audience and the panel members. The reason behind this division was to make labels more distinguishable on the map, and to make room for asking questions targeted at either group. This change was introduced after a few students reviewed the papers for the workshop during the planning phase.
There is one important difference between the query syntax here and the one described in chapter 3. As we see in table 4.1, some of the suffix values have a colon, parentheses and an asterix attached to them. This is the experimental query syntax described earlier in this chapter, where the participants could join two queries in a single query statement. In this case the name of an audience or panel member, a person, would be inserted here. For example the query "Argument:Person:Shelly" matches an argument node made by Shelly.

Writing Queries

The participants were informed that any query, created by combining a prefix and suffix from the query group, would always match at least one node on the map. And the author gave a three example queries: To find all questions by the audience member Amy, the participant could write Question:Amy or Audience:Amy; to find a specific question by label, for example Female role models, they could write Question:Female role models.

The author explained that, in the case of the first example, the Question prefix would give only questions, and the Audience prefix would give all nodes created by Amy. The author explained that if they picked All as the suffix it meant they wanted to query for all nodes that had the selected prefix.

The author also informed the group that the experimental syntax would let them query for a question, idea or argument node in addition to an author name, all in a single query. In the case of the experimental syntax, the suffix would be Person:Author Name.

The participants were allowed to combine as many queries as they felt were necessary, including combinations of basic query syntax and the experimental syntax, and to answer each query question using only queries.

The author gave one hint to get the participants going, to first decide on one type of node they wanted to find, picking one of the values from the prefix column of table 4.1. Then pick one of the values in the suffix column in table 4.1.

4.6.2 Query Tasks

The participants were asked to work on a total of six tasks with one question per task, where each question had to be answered by only using meta-data queries. The solution to answering a question was to make a set of queries that would produce the nodes they were looking for as the result. After writing down an answer, a participant would outline the expected outcome on the IBIS map. The tasks were as follows:

1. Find all questions from all audience members.
2. Find all questions by audience member Amy.

3. Find all ideas by panel member Sheryl and questions by panel member Sheryl.

4. Find all questions and ideas after the question “How can gender balance be achieved at home?”.

5. After the Idea: “Make IT attractive to young girls”, find all ideas after the question from panel member Jessica.

6. Find all ideas after the idea “Make IT attractive to young girls”.

For evaluation, each question that was answered correctly would give the participant one point. A correct answer would reveal all nodes that were asked for in the question. No penalties were given for broader answers, that would display more nodes, meaning that each question had multiple answers. But if a given answer would not yield the correct nodes, this answer would get zero points. Using basic prefix:suffix, the above questions can be answered as follows:

1. Audience:All

2. Audience:Amy

3. Idea:Sheryl + Question:Sheryl

4. Idea:Allow meetings via Skype + Idea:Use Technology + Idea:Give more, get more + Idea:Men needs to step up

5. Question:What are the challenges? + Panel:Jessica

6. Question:Make IT attractive to young girls + Panel:Jessica + Panel:Theresa + Panel:Jocelyn + Panel:Sheryl

And using only the experimental nodetype:prefix:suffix, one possible answer for questions would be the following:


2. Question:Person:Amy

3. Idea:Person:Sheryl + Question:Person:Sheryl


5. Idea:Person:Jessica

4.6.3 Group Questions

After solving the individual tasks, the participants would move on to group questions. These questions were related to preservation of context. Examples of question nodes that did not describe their own context explicitly. I.e. the more contextual “Why are there so few female leaders in technology?” compared to the less descriptive “Why are there so few female leaders?”. If the latter question was provided without the context it can be more difficult to reason about. Giving the wrong impression that the author suggest that female leaders are underrepresented in all work domains. The author showed these node questions to the participants, and asked the following:

- In what way do you believe you could create your queries, in order to understand the context of these questions?
- Do you see any potential pitfalls when using meta-data queries for IBIS search?

The participants were asked how they would use queries in order to acquire the sufficient comprehension of the context of this question. A tough question, but as we will later see, they gave good answers to this question. The author also asked how would they proceed in order to understand the discourse context when looking at the map itself.

4.7 Execution

4.7.1 Introduction

Participants was introduced briefly to IBIS, and shown some examples of IBIS maps in KnAllEdge. Then participants were handed a copy of the discourse map. They were introduced to how meta-data queries are composed, the query groups, and how to create queries. This was a lighter version of the query syntax specification described in chapter 3, including the experimental syntax described earlier in this chapter. Afterwards the participants were given a few minutes to study the task list and ask questions.

4.7.2 Solving Tasks

The participants started with the task list, and solving the one set of tasks. They could study the map at any time, but it was not required to write their answers. Some used the map to draw what they believed the query would match.
If, for any reason, a participant wanted to skip a task, they were allowed to do so. However, they were encouraged to give each task a try before skipping to the next one. No solutions were given on how to solve tasks, participants were encouraged to answer as best they could, but if the task was too difficult they could always try the next tasks instead.

4.7.3 Group Tasks
The second set of questions was regarding the syntax of the questions and ideas in IBIS. Given a question labeled without explicitly defining a context. The participants were asked reflect on how to use meta-data queries to gather enough information, and how they would approach to gather this information by just studying the map itself. Figure 4.1 shows a segment from the discourse map, where some nodes describe their context more accurately, and others do not.

4.8 Gathering Data
At the beginning of the session, the participants were asked to sign a consent form to acknowledge the use of audio, notes and pictures. Names were the only personal identifiable information gathered in the consent form.
4.9 Results

4.9.1 Solving Questions

The overall accuracy of answers was not consistent. One of the participants, with developer background and several years of experience, gave the impression of understanding the concept quickly, and finished all question in a few minutes. This participant revealed after the workshop...
that the meta-data queries seemed similar to database queries, and this type of logic was fairly easy to understand. The participants followed up with this assessment: “I believe if each query term returns a set of nodes, then two queries would probably give me the nodes between these sets”. The other participants with developer experience expressed that the use of queries was not so difficult, but the expected output was ambiguous. This individual frequently asked for confirmation when attempting to draw the output on paper.

Overall two participants three or more, seen in figure 4.3.

4.9.2 Comprehending the Query Syntax

After the initial session, where the author explained the concept of meta-data queries, two participants were quick to express confusion about what a query is, indicating that the explanation given was inadequate, or that the logic of meta-data queries is not as intuitive as expected. So the author followed up with two example queries: Audience:Amy and Panel:jessica, and circled out the nodes on the discourse map. The author explained that the nodes preceding the query results would be revealed. The two participants nodded in agreement, giving the impression that they understood.
However, when the group started solving the questions on the sheet, one of the two aforementioned participant exclaimed that “there is too many to choose from, I do not know which one to use”, and pointing to the table of queries on the sheet. The rest of the group, except for one, nodded in agreement to this statement. One of the other participants then pointed out that due to larger amounts of alternatives provided on the paper, the participant assumed that they were only allowed to use specific query terms for a given question.

The Experimental Query Syntax

Considering that the participants had some trouble with the basic syntax, it was less surprising that it was difficult to understand the more sophisticated \textit{nodetype:prefix:suff} syntax. When answering the questions two of the participants attempted to use these queries, and both approached the solution by combining two queries into one. For example one participant, a junior programmer student, wrote \textit{Question:Person:Audience:Amy}. After the workshop was finished the author asked about the reasoning behind this answer, and one participant answered: “I thought Audience:Amy was the correct suffix because it references the person Amy in the map”. The other participant, one of the HCI students, stated that the answer was a guess, and expressed preference towards using the basic suffix:prefix syntax.

4.9.3 Group Discussion

After the task assignments the participants were engaged in discussion about discourse contexts in IBIS, and how poorly worded questions and ideas would implicate their comprehension of the results. This scenario emerge if a query produce a sub graph with a root question that has no context. The author purposefully ensured some questions in the map had no context. E.g. one question had the the label: “How can this be achieved?”. The participants discussed this issue and converged on two reasonable solutions. The first was to always let the dialog chain trace back to the root. In the context of the previous example, the question would have a dialog chain all the way back to the root node in the map. All participants agreed that this would be a good solution. One participant suggested to support incremental, or step by step navigation, from one node back to the topic. This participant argued that the user could then choose how far back in the history he or she needed to read.
4.10 Possible Drawbacks of the Study

This workshop is one of many user studies that needs to be conducted. The size of the study group was small, and the use of a paper prototype impose limitations that impacts how users interact with the prototype.

Using these queries should be simple enough that the users can work with the software supported by intuition and minimal preparation. But the observations from the workshop indicate that less technically skilled knowledge workers are much less comfortable with writing meta-data queries, than the author anticipated.

4.10.1 Study Groups

A small study group, like this workshop, is useful when performing in-depth studies on individual user behavior, yet it introduced many elements that were foreign to the participants. The observations made during this study remains valuable for future experiments, highlighting weak areas of the workshop itself, and the query method. Future experiments can learn from this experience, and conduct more specific studies on how well users comprehend queries. Performing tests with controlled experiments, comparing meta-data queries to other IBIS and non-IBIS query mechanisms, would yield measurable data to validate the observations [18, p. 192] in this study. This workshop was performed without control groups.

4.10.2 The Women In Technology Discourse Map

During the workshop design process the author transcribed an hour long panel debate about women in technology from Facebook’s headquarters. As we have learned, in business settings, transcribing debates to IBIS maps is typically done by professionally trained mediators. The author does not have the required training to accurately transcribe panel discussions. But as an untrained IBIS mediator, this does reflect a real scenario with discourse contributions from untrained contributors.

Feedback from the participants indicated that the language and wording of writing on several other labels was somewhat opaque, making the overall discourse harder to comprehend and follow. Some questions was poorly written, ideas not always associated with correct questions, and misplaced arguments. This introduced uncertainty in the study, and this is likely to influence participant’s ability to solve tasks. Future experiments should avoid similar uncontrolled variables, for example by having a trained mediator examine and correct the map.
4.10.3 Testing with a Paper Prototype

Solving tasks with the high fidelity prototype would give a more realistic scenario for testing. As we saw earlier the participants struggled with understanding what results their queries would produce on the map. If they had been tested with high fidelity prototype, they would immediately get results from a query. This would be a very different way of interacting with IBIS, yet the low-fidelity prototype gave novel understanding of how users would use meta-data queries, as well as exposing flaws that would cost time to mend in later prototypes.

4.11 Type of Research

During the workshop we performed descriptive investigation of how users interact with meta-data queries, describing what is happening and observing behavior. But in order to state claims about the correlation of meta-data queries to performance one should perform an experimental investigation [18, p. 20]. A properly executed experiment of this type stands much stronger in justifying claims about casual relationships. That is, to measure if meta-data queries are responsible for a positive or negative change in recorded task performance, compared to other IBIS systems.

One possibility is to execute an experimental investigation, and comparing performance between other navigation systems to meta-data queries. For example one comparing performance to gIBIS, and another comparing performance to KnAllEdge. The first case compares a control condition, gIBIS proto nodes, versus a treatment condition, meta-data queries. In the second case we are comparing KnAllEdge to itself, a control condition without meta-data queries, and a treatment condition with meta-data queries. Assigning participants randomly to control and treatment in this case would produce very accurate data, but non-random assignment could also be used [18, p. 24]. The instructions given to participants would be replicated to ensure equal scenarios. The ideal scenario is that the only difference between control and treatment conditions, is the navigation system.

4.11.1 Relevance of Test Group

The participants domain of knowledge lies within academia, research, interaction design and development. They are potential future members of KF, and they have relevant experience. It is within reason to assume that expert users within KnAllEdge are capable to comprehend the use of meta-data queries, from years of experience with IT systems and collaboration software.
The test groups using KnAllEdge current consist of a few members, in a sandbox \(^4\). The primary users today are largely researchers, and individuals affiliated with the academic community. But further down the road, it has been indicated that KnAllEdge will be published as an open service for use in various organizations, and smaller interest groups or individuals. The exact target groups for KnAllEdge in the future is uncertain at the time of writing. But many organizations have interests and can benefit from this type of information management system, and doing pro-active testing with organizations outside of academia would smoothen future transitions, and ensure their needs are recognized.

**4.1.12 Summary**

The primary focus for the workshop was the meta-data query specification, and how the participants would use it when asked to search for discourse elements in an IBIS map. The participants were introduced to IBIS, KnAllEdge and meta-data queries. The observations made, and answers given, indicate that participants did not comprehend meta-data queries easily. The syntax of queries resonated to a higher degree with participants that had previous experience with similar concepts. The connection between a search in the graph and the expected output was at times confusing for the participants, and the participants with more developer experience fared better with predicting the outcome.

The discussion part of the workshop expanded on the problem of missing discourse context. The questions, ideas and arguments in the map had varying degree of preciseness in their description. As the author transcribed the video directly, close to word by word, this reflect the variations found in everyday discussions.

The participants were challenged to think of how to find the discourse context of these kinds of contributions. They brainstormed solutions to how meta-data queries can amend such cases. The participants suggested tracing the discussion backwards, towards the first contribution, would be a good solution in this case.

\(^4\) Testing within a limited area or scope
Chapter 5

Prototype Development

Before meta-data queries is ready for demanding collective knowledge mapping projects, it must be evaluated in smaller KnAllEdge experiments and workshops. This is important as fast and frequent feedback is paramount, to develop quality software iteratively. What we are about to see is the first iteration of a contextual meta-data queries prototype, that is capable of performing IBIS discourse search in KnAllEdge. The output of this process is a web application, based on the web browser scripting language JavaScript, and supported by several framework and utility libraries.

During this chapter we will go through the important areas of the prototype. We will look at the tools and libraries used, we will study the three main components of the application. The implementation of prefix mapping functions, and the processing steps involved in a search, are explained in more detail.

The prototype contain quite a bit of code, and much was left out from this paper. A link to the prototype code repository can be found in Appendix A.

5.1 Tools

All code was written using Sublime Text version 3 [27], a code and text editor that supports a wide variety of markup and programming languages. To check code validity, such as detecting syntax errors, the JSHint[16] linter plug-in was used.

For code maintenance and preserving code history, the author use a distributed revision control system named Git [32]. Git use both local and centralized repositories for storage. Updates are committed locally, and pushed to the central repository. Following this pattern, iterations of the prototype code was written and pushed to the private centralized repository at Bitbucket [2]. Updates were committed and pushed at regular intervals, to maintain a coherent history of changes.
5.2 Languages and Code Libraries

**JavaScript**

In 1995 JavaScript, a lightweight scripting language, was first released by Brendan Eich at Netscape for the Netscape Navigator web browser. It was standardized at European Computer Manufacturers Association (ECMA) a few years later. Today JS exists in many environments; it is supported by all major browsers; and can run in non-browser environments such Mongo[21] databases, and the Node.js server environment. JS is a versatile language and supports object oriented, imperative and functional programming paradigms.

The prototype application runs in a web browser environment. It use JS to manipulate and control all application elements, changing the page structure, through the document object model (DOM). The DOM is an object containing a hierarchical representation of the page, and is the entry point we used for displaying output to the browser window. This is required when altering visual elements on any web application without refreshing the page.

**Hypertext Markup Language**

HTML is a markup language for creating the user interface of web pages. Unlike a dynamic language, such as JS, it only declares the structure of a page and the semantic meaning of HTML elements. For this prototype, HTML serves two primary purposes: It is used for delivering the application logic i.e. JavaScript scripts, and to visually represent items from the application’s graphical templates, such as nodes from IBIS maps. After the scripts are loaded and the application starts, the HTML is accessible through the DOM. Drawing and moving elements in the browser viewport is done by manipulating their DOM counterparts.

5.3 JavaScript Libraries

This section cover the core JavaScript libraries: Angular, D3, Backbone and QueryEngine. Starting with with a description of the relevant components of Angular, the application framework used for the KnAllEdge application, as well as this prototype. Angular define the application structure, adds functions for internal and external communication, and comes with HTML templates for visualization of components.

---

1. Live changes in a page while is it open
2. What the element is, and how it relates to other elements
For storing map data we use Backbone collections [1], and QueryEngine facilitates data queries for nodes and map data. D3, a descendant of the popular Protovis[4] tool, is used to process map data into a graph data structure, so they can be visualized.

Angular

Angular is a popular framework, developed by Google, for building web applications. It encapsulates application structure, sophisticated JS business- and rendering logic, and interfaces for external communication. Because Angular ships with a base architecture the developer does not have to write any intricate interface logic, or worry about synchronizing internal components of the application. Angular is designed for decoupling the DOM, the the application visual output, from the logic of the application. The interface Angular provides is much more powerful than then browser DOM interfaces, making the development process faster, with less room for mistakes, and providing utilities for solving difficult tasks.

All the visual components of the prototype application are built using Angular directives, directives is a very powerful feature in Angular. Directives are Google’s version of HTML5 web components, customized and reusable HTML element that bundles HTML markup, styling and business logic. Directives are small applications that contain encapsulated code logic. Directives can declare custom HTML tags, optionally supplied with a custom HTML template, to represent themselves in the HTML layout.

Data-Driven Documents - D3

D3 is a JS library for visualizing data that can be used in most new browsers [4]. D3 enable data to graph transformations, and it is used here to transform the map data to graph layouts, positioning nodes and connecting edges. D3 does support visualization as well, but as Angular gave more explicit control, it is used instead.

QueryEngine and Backbone

QueryEngine is a library that facilitate data queries in object collections, JavaScript arrays and Backbone collections [19]. Backbone is essentially a web application framework, but here only the data collection functions are used. In the prototype we use these collections for storage of map data.

---

3 Backbone.js - Application framework library
4 Rules of behavior, looks and performance
5 D3 is supported by browsers newer than Internet Explorer 8
6 Data search
7 Serialized lists
data. Collections are lists of Backbone models, JS objects extended with utility functions for easy processing and retrieval. Backbone collection are compatible with QE, supporting powerful search functions. Backbone and QE serves as the base for building the prototype meta-data query engine.

5.4 Modular Components

The prototype consist of the follow Angular directives and services: A search directive, a visualization directive, a data service, and a external dependencies service. The search and visualization directives handle user input, composing meta-data queries, and map visualization. The data model service does all processing involving map data, including processing meta-data queries and contextual search. The external dependencies service load additional libraries required by the core libraries when the application runs. As they are similar in their design we reference directives and services as components.

Figure 5.1 outline the component architecture, visualization and data components. The external dependency service is a support component and is excluded.

5.4.1 Search Component

The search component is the intermediate connection between users and the search engine. It has an interface that take input from a user, and process the words and symbols of this input into data queries. It continuously transmit query data to the data component, while a user is typing in the input, and compare the query input to the map data. The search component communicate back to the user, the nodes it found, as query suggestions.
Query Suggestions

One important issue that this component must deal with, is to tell users what queries it can accept. KnAllEdge does not communicate the metadata properties of map nodes to its users, and they need to know what prefixes and suffixes they can use for searching the map.

During the workshop described in chapter 3, the participants were given a sheet with available prefixes and suffixes. In the prototype we read these values from the graph nodes and output them as suggestions while the user is typing.

Query prefixes are suggested at the beginning of a new query, and the postfix or value suggestions associated with a given prefix are suggested after a prefix is selected. For example if a user types in the letter n, a drop down list of suggestions is then presented. Lets say he or she select the name prefix from this list, then a new list is displayed, with the names of all the people who submitted contributions to the map.

The suggestions are filtered as the user types, eliminating non-matching alternatives from the suggestion list. See figure 5.2 for an example, here we see the drop down suggestions presented when submitting a Question prefix.

![Figure 5.2: Query Input Interface](image)

Grouping Queries

The component support single and composed queries. To distinguish individual queries, each query is visually compartmentalized with a small box, and put inside the search field.

Content Search

The query-less search is an alternative for use cases where the user just want a simple search for content in the map. It does not use standard prefix or suffix formatting in the input. Users that want to perform a content search can simply start typing text in the input field.
The data component still expects a prefix:suffix syntax, so a text prefix is added when the context search is submitted. A text prefix is bound to a mapping function, as seen in figure 5.1. It supports partial and mixed-order searching, that is, any node name that partially matches the input is returned. For example submitting Models will match Models of and of Models. This search is also case-insensitive, so both uppercase and lowercase characters are considered equal. Content search does not return suggestions, and this way we can avoid users having to remember the exact text in labels and descriptions.

```javascript
1. setFilter('search', function(model, searchString) {
2.   var searchRegex = queryEngine.createSafeRegex(searchString);
3.   // Search collection for models with a name attribute matching the text input
4.   var pass = searchRegex.test(model.get("name"));
5.   return pass;
6.});
```

Listing 5.1: Content Search

### 5.4.2 Submitting a Search

A user can submit a single query, a composed string of queries, or a string of text. For simplicity we refer to all of these as a search. Note that a colon separator is automatically inserted after a prefix, when it is selected using the drop down list.

Submitting a search is done by pressing the enter key, or after selecting a suffix from the drop down list with the mouse. To distinguish when a query ends, and when the next begins, the component mark the end of a query suffix to be when a search is submitted. This is also done if a suffix is selected from the suggestion list, or if the user click on empty space in the input field with the mouse. Existing queries can be edited and resubmitted, by using the mouse to click and edit the prefix or suffix, or by navigating back and forth using the arrow keys on the keyboard.

### 5.4.3 Data Component

The data component is a service is used for storage and processing. This component encapsulates all logic that works on the KnAllEdge map data. It accepts IBIS map data as input from KnAllEdge, and produce an internal data collection to represent a given IBIS map. The collection is further processed to create a view model, after a search is submitted and processed. The view model is a ready-to-render\(^8\) data collection, used to visualize the IBIS map with D3. When not running in the KnAllEdge

\(^8\)Can be visualized as-is with D3
environment, this view model is sent to the visualization component instead. Note that KnAllEdge creates its own view model, so it is not strictly necessary to include it there. It is done because it is needed for local testing, and the difference in data size is negligible.

The data component also support the aforementioned query suggestions. The data component have many roles, and internal components. We do not include the details of these functions here, expect for a few important ones. Appendix B has a link to the full source code. Here a list of the operations that the data component perform:

- It fetches map data from KnAllEdge.
- It processes KnAllEdge map data to create a search-able data collection.
- It performs node queries and contextual searching.
- It performs on-the-fly node searching for input suggestions.
- It creates a view model that is either sent to KnAllEdge, or to the visualization component.

5.4.4 Visualization Component

The visualization component is a lightweight component that renders the map to the application, designed to be de-coupled easily when the application runs inside KnAllEdge.

This component requires a view model from the data component as input. The component has has a D3 processing step that reads view model and applies layout modifications, essentially positioning nodes and links based on a set of configuration parameters, and a selected graph layout. When this is finished the data is embedded into a HTML template, and the finished template is rendered in the browser. The standard layout used in the prototype is a tree structured node-link graph, the standard layout in KnAllEdge, see figure 5.3.

Mapping Prefixes

Mapping functions reside in the data component, one function for each prefix described in chapter 3. For example, listing 5.2 is the mapping function used to bind knowledge nodes to the Knowledge prefix. In this case, the type property value is mapped. In the case of a prefix-to-property mapping, it is the same type of function, but with different internal logic.
Figure 5.3: Prototype Map Visualization

```
1. setPill(‘Knowledge’, {
2.   prefixes: [‘Knowledge:’], // The prefix this function
3.     is bound to
4.   callback: function(model, value) {
5.     // This block of code is executed for every node in
6.       the collection
7.       var _val = model.get(‘type’);
8.       if (model.get(‘type’) === ‘type_knowledge’)
9.         return (model.get(‘name’) === value);
10.     else return false;
11. }})
```

Listing 5.2: Mapping Knowledge Prefix with Mapping Function

The function labeled `function(model, value)` is executed once for every node in the map, and once for every query in a search. Each time it is executed it has access to the current map node, including all node properties and values, and the current query suffix. This function can either return true or false.

The function return true if its criteria for success is met, or false otherwise. Returning true means it found a a node that has a property and value that matches the query, the success criteria is defined by the block of code in the function body.

### 5.5 Performing a Search

Searching map nodes means to search through the map node collection stored in the data component. The search component have indirect access to the data collection, by calling functions on the data component that control parts of the processing steps.
Locating the Map Nodes

The search component initiate a node search by submitting a string of queries to a `searchCollection`, seen in listing 5.3. The function convert the string to a list of queries. Each object in this list contain one query. The function performs a search through the data collection for a node that match each of these queries. The prefix mapping functions are executed as part of this process.

```javascript
// Perform search on collection - return querynodes
function searchCollection(searchString) {
    var queries = searchString.split(",");
    var queryNodes = []; // List of models/query nodes
    _.each(queries, function(searchTerm, index, list) {
        if (/\S/.test(searchTerm) && searchTerm !== null) {
            // Check that search string contains at least 1 character
            var queryNode = projectSearchCollection
                .setSearchString(searchTerm)
                .query()
                .toJSON();
            queryNodes.push({
                "type": "queryNode",
                "queryString": searchTerm,
                "label": searchTerm,
                "nodes": queryNode
            });
        }
    });
    return queryNodes;
}
```

Listing 5.3: Searching the Discourse Map

The output of this process is a list of query nodes, each contain a reference to a matched node in the map.

Prerequisite Sorting

Because the map data is a directed acyclic graph, calculating shortest paths between two nodes in this graph, can be done efficiently in linear time if one performs a topological sort on the graph first [12]. The algorithm was implemented as a recursive function\(^9\) that checks each node, and then each adjacent node until it finds a node with no outgoing links. This node is marked as visited, and the node neighbor with the shortest link is found and put in the cost mapping, this neighbor is set as the current node, and the cycle continue until all nodes have been checked. Each node is only

\(^9\) Recursive function - Function that executes itself in a controlled loop
checked once, and when it is finished the cost mapping list is complete. For details on the algorithm and implementation, see Appendix B.

**Context Extraction**

Context extraction is done in the data component. From chapter 3. The context of a query node is defined as the path from a given query node to the map topic node. This process involves parsing the map data, checking its validity, and sort the map nodes. The outcome is a cost mapping, that we can lookup for any desired node, and find its parent. By iterating through this list, for each node produced by each query, we can extract all the context paths.

We first check that the map graph is in fact a DAG \(^{10}\), and create a cost mapping by performing a topological sort. The cost mapping is used to find the path between two nodes, and for cases where one or several IBIS nodes in the context path has multiple predecessors. If the discourse branches out, and then merges back together, there is more than one path to take. This sorting is necessary for cases where a node is linked to more than one parent. This is not currently the case with KnAllEdge, but the idea has been discussed, and might be implemented in the near future. One use case could be that an embedded topic is connected to two separate contributions, the current KnAllEdge version replicates the node instead.

The algorithm use link weights to calculate distance between nodes, as part of creating the cost mapping. KnAllEdge links are not yet weighted, so each link is set to have the weight of 1. The context path with the shortest overall length is found by summarizing all the weights. For example with the root node A and the query node B. Cost mapping gives the shortest path from A to B, in terms of summed link distance or weight.

The context history between query nodes, and the topic, is found in the cost mapping. For each query node, we follow the parent chain in the cost mapping, and store each node, including the query node and the root node, in a list. The full algorithm for context path traversal is described in Appendix C.

After all queries have been processed, each query is associated with a list of nodes. These lists are then merged together to one list, and duplicate nodes are deleted. It is common that some of the context history between different queries is the same, that is, the same nodes. We now have a list of nodes that is ready to be visualized.

---

\(^{10}\)Directed Acyclic Graph
5.6 Creating the View Model

Creating the view model is done in the data component. The view model is a collection of nodes, similar to the map collection, but where each node has a parents and a children property. These properties are both lists, and initially they are empty.

A `buildInheritanceTree` function iterate over the node list produced by the context extraction process, and all the links in the map data collection. It compares the source and target identifiers on each link object, and see if it can find these ids on any of the nodes. If it does, the node with the source id is added to the parents list on the destination node, and the destination node is added to a children list on the source node. Listing 5.4 describes the steps of this process.

```plaintext
function buildInheritanceTree (Links L, Nodes N){
    For each node n in N
        Add to n, empty list properties: children, childrenLinks, parents, parentsLinks

    For each link l in L {
        Node s = FindNode(l.sourceId)
        Node d = FindNode(l.destinationId)

        Add d to s.children [list of nodes]
        Add l to s.childrenLinks [list of links]
        Add s to d.parents [list of nodes]
        Add l to d.parentLinks [list of links]
    }
    return N
}
```

Listing 5.4: Building the Hierarchical Inheritance Tree

This step is performed once when the application starts, and once for every search. Ideally the view component of the prototype, should contain as little logic as possible, because this component is not active when the application is part of the KnAllEdge system. By performing all view model processing in the data component, we can simply send the view model data to KnAllEdge without any additional processing needed.

5.7 Testing

For testing in stand-alone mode, a file containing extracted map data from KnAllEdge was used, and the prototype visualization component is included in the program. The author performed tests using two common browser environments, the Chrome and FireFox web browsers.
5.8 KnAllEdge Integration

During integration the prototype visualization component is replaced by the KnAllEdge visualization engine, and the map file is replaced with live KnAllEdge map data. The application gain access to KnAllEdge through a client plug-in service. And it must be integrated into the KnAllEdge software repository.

KnAllEdge requires plug-ins to request access by submitting a identification object. This object contains basic data about the plug-in, and which services it will offer and which services it needs data from. This code is not part of the prototype code repository, but was added by the KnAllEdge developers, when the plug-in was integrated.

5.9 Summary

This chapter described the prototype for performing contextual meta-data queries in KnAllEdge, following the specification from chapter 3. This prototype implement the full specification, and can be tested outside the KnAllEdge system environment.
Chapter 6

Results and Discussion

In this chapter we investigate the major factors that influence information overload in IBIS. We will discuss the outcome of the KnAllEdge search prototype, looking how contextual meta-data queries can manage information overload in the KnAllEdge IBIS discourse map.

6.1 Information Overload

What are some of the major factors that attribute to information overload for collaborators working with IBIS discourse maps?

6.1.1 Map Size

As a information visualization tool, KnAllEdge should facilitate understanding and exploration [13, p. 2], but the size of the viewable area in the KnAllEdge interface is finite. It is not a trivial task to organize and present large visual graphs in finite spaces [25]. Additionally the KnAllEdge performance can deteriorate when there are too many nodes and links to visualize, and it is reaching the capability limits of the system. But usability becomes a problem much earlier than this [14, p. 24]. If a part of the map is outside the field if view in the interface, collaborators easily get disoriented [9, p. 212], loosening the oversight of where they are in the discourse.

For example the DebateGraph map *Digital Agenda for Europe*, as seen in figure 6.1. This map discuss strategies for a digital economy boom for Europe by 2020. This is a good example of a discourse that involves many collaborators, and at the time of writing, this map had 281 registered ideas alone. In order to maintain performance, the visualization size of the Digital Agenda map is restricted to only show three levels, but it is actually much larger. And the DG map for *Nuclear Politics* had over 1200 registered ideas.
It is a strain on cognitive effort to read ideas and thoughts of others through a lengthy discussion [9, p. 211][6, p. 327]. Our short term memory can only hold roughly seven to ten pieces of information [20], with some variation. But even if we exclude everything but the relevant content for one single discussion, it would in most cases be more than ten contributions.

But it is also important to consider how many contributions a collaborator actually can manage. Observations from one gIBIS study by Conklin, indicated that it is preferable for collaborators to work with all relevant content for one discussion in the same visual space [9, p. 212]. In this case the collaborators would group relevant contributions according the problems they were working on, and exclude the rest.

Consider that in a discourse map, there can be any number of contributions linked to other contributions. Collaborators that browse discourse looking for questions, ideas and argument relevant to their current discussions, have to evaluate contributions that are not related as well. Contributions that give little to no useful information in the context of any given discussion, attribute to the size of the overall structure, and waste the limited cognitive resources.

### 6.1.2 Organizing Thoughts and Ideas

Communication of thought with the non-linear\(^1\), and fine grained structure of IBIS, require a lot of effort in organization and writing. It is not common for most individuals to rigorously break down thoughts into questions, ideas and arguments [6, p. 325].

Conklin attributes lack of IBIS fluency, collaborators finding it difficult

---

\(^1\)Collaborators does not read discourse as a strict linear text, visual imagery and structural cues complement the readers interpretation of text content
to organize thoughts into IBIS nodes, as the major factors that cause issues of premature segmentation [6, p. 324].

Lack of IBIS fluency can be compared to lack of grammar in natural languages. If collaborators lack the necessary knowledge and experience to properly organize thoughts using IBIS, they are not fluent in the IBIS language. Not only do they spend time and focused efforts on organizing their thoughts, but readers are more likely to fall off the trail of thought, misunderstand what the authors are trying to say, and miss the big picture.

Facilitators, that organize discourse through dialog mapping sessions, are less likely to lack fluency. But when mapping in individual sessions, there are no facilitators to guide the process. Autonomous collaborators have little choice but to learn the process. But, over time they will eventually become proficient enough to naturally segment thoughts [8, p. 120].

Yet not all ideas can be communicated succinctly, and in some cases there is a need for a thorough explanation to communicate a higher meaning. In these cases, premature segmentation issues happen more frequently [10, p. 250]. The problem domain could be very complex and difficult to describe, or the authors understanding of the problem could be too vague.

In this case it could be tempting to contribute one idea, and put a page’s worth of details in the description field. But then other collaborators can not easily contribute their own thoughts to this idea. Collaborators need to be able to evaluate each others work, and be able to attribute other’s work with their own ideas and opinions. Thoughts should be mapped explicitly in IBIS [9, p. 211], to make communication as clear and open as possible.

In cases where collaborators use IBIS for drafting unfinished thoughts, it is common to experience a similar segmentation problem, e.g. having a great idea and evidence of why this is a great idea, but without knowledge of what problem that this idea might solve [9, p. 211]. In this case the problem is that the collaborator need a platform drafting their thoughts before contributing. Great ideas need to mature, and we can allow them to do so, with proper tool support. For example, in KnAllEdge and DebateGraph, collaborators can draft using comment nodes. This way collaborators can outline and work on early thoughts before contributing them, but they are still required to use the IBIS grammar when they do.

Collaborators that struggle with expressing themselves also experience cognitive overhead [8, p. 120]. As collaborators are responsible to find out how their disorganized thoughts can support the existing discourse and contributions. They must evaluate what others have asked and contributed before them, in order to ensure their own work fits the context. But if a contributor is confused about what to write, and how to write it, it can also lead to mistakes, and improper use of the nodes to circumvent the restrictive language. If the collaborator has written down notes, using the comment node or other methods, they are responsible to keep up to date

69
with what is being discussed, so that they can submit their contributions at the right time. In these cases it lies a responsibility on them to study other collaborators work, even if it might not be related to their ideas.

6.1.3 Adjusting to Change

IBIS discourses change and evolve over time. Contributions are adjusted and corrected, and old discussions can expire [9, p. 212]. Discourse maps must be kept up to date for IBIS to be effective as an information organization tool [7, p. 360]. While flagging old content can be done via automated software, it is still humans that must take the task of reviewing contributions, and evaluate when the discourse need to be updated or corrected.

Reviewing discourse in the context of contemporary understanding and experience, is a natural part of the discourse mapping process. The old, but still accurate or important information, should be preserved alongside the new [9, p. 212].

But collaborators can and will make mistakes, it is in human nature after all. Depending on the type of mistake, the amount of work required to fix it can span from a simple word correction to dedicated effort from many collaborators.

For example, node labels contain a few phrases or sometimes even a single word, and this does not leave much room for hiding spelling errors and bad grammar. Even if it is not corrected immediately, as long as the collaborator made himself or herself understood correctly, it is a fairly easy task to correct the label. But other type of mistakes are not so easily fixed. Collaborators’s can submit questions, or ideas, in the context of the wrong problem, or sneak in assumptions accidentally. If one remove an assumption from a question’s label, the contributions that follow this question must be re-evaluated, checking if they are still valid. If they need to be changed, then we must re-evaluate their children as well.

In the end, change due to mistakes is the type of work that can be managed with less effort, if collaborators can detect these mistakes early. And it would lessen the overhead of having to re-evaluate contributions, trying to understand how the mistakes have influenced them, and what the correct action to take is.

6.2 Contextual Meta-data Queries

To what extent can contextual meta-data queries reduce information load for KnAllEdge IBIS collaborators?

The meta-data queries prototype enable KnAllEdge collaborators to see the full discourse history starting from a chosen root topic, and leading
up to any question, idea or argument node. If collaborators submit queries that yield a small result set, the overall size of the map is reduced. And the context supplied with each query allow collaborators to study the entire history of any discourse node. It takes less cognitive effort to review a smaller context history, and it is easier to detect incompleteness, inconsistencies and mistakes. Each search produce a new map, that filter out nodes that are not part of the search. The discourse nodes in this map are connected identically to the original map, preserving the original discourse relationships.

6.2.1 Contextual Oversight

With search and filtering that reduce the overall size of the graph layout one must consider that the user is moved from one view of the graph to another.

In figure 6.2 we see part of the TNC map. This map cover all the available space and goes outside the field of view. By comparison if we search for the knowledge topic classical science, as seen in figure 6.3, we still have a full oversight of what has been discussed between the nature of creativity topic and the classical science topic. Note that it is the same knowledge topic on the two branches.

![Figure 6.2: Map Visualized before Contextual Search](image1)

However this is a relatively small map, there are 12 topics, 7 ideas, 5 questions, and 8 arguments. Meta-data queries does not impose a limit on the total count of nodes included in the context between a query node and the root node. The configuration of the layout map, how much
space each nodes take, and the size of the map interface, determine if the search output will be bigger than the boundaries of the viewable area. However, this oversight only show what is relevant based on the discourse contribution(s) that were searched for, and it is still possible to browse the map like one normally would, after a search. By visualizing a focused map, it can be browsed more orderly, and enable collaborators to work more efficiently. The collaborator is assured that all content there is relevant to understand the discussion or contribution they searched for. This means they do not have to browse other un-related contributions.

Further the prototype does not require collaborators to know what properties and data that exist in the map. As described in chapter 5, the prototype give search suggestions in a drop down list, to provide oversight for collaborators. They expose the prefixes so collaborators know about them, and they give an oversight of what is being discussed in the map. For example, if a collaborator wants to see all the questions that have been submitted, they can simply type in the question prefix to see what questions have been asked. The prototype also narrows down the list by filtering suggestions as the collaborator types. The suffix works the same way. For example with the given prefix *Name*, the directive will suggest the names of people who submitted content to the map.

This could potentially make the search interface feel more intuitive. In the workshop experiment we saw that participants got confused by having too many prefixes and suffixes to choose between. While they could only pick one of five prefixes, it is possible that the combination of prefixes and suffixes in the same table made them uncertain about how the queries worked.

Currently the prototype supports nine prefixes mapped to node properties and property values. Only after selecting a prefix do they get suggestions for suffixes, and both prefixes and suffixes are filtered as the use types.

### 6.2.2 Support Reasoning by preserving Discourse Structure

Asking questions is at the heart of IBIS method [8, p. 120]. But it is also a matter of asking the right questions, and asking them the right way.

Even the careful author is in danger of not anticipating all the various routes by which a reader may reach a given node, and so may fail to sufficiently develop the context necessary to make the node’s contents clear, if not compelling. We have as yet found no solution for this problem [10, p. 250]

Both discrete and contextual questions can co-exist in discourses, and both approaches has advantages.
Consider a discourse map on the topic *Google DeepMind*\(^2\), and the difference between the question *Intelligence?* versus *How intelligent can we make DeepMind?*. The former question is much broader, it does not describe the context of the surrounding discourse or topic, while the latter is more precise. If a collaborator evaluated these questions, the latter would give a clearer picture of what the author was asking.

Collaborators are not forced to be explicit, but they can be. If we read discourse structures through the explicit visual context of a meta-data query, the visual history would provide the necessary context comprehension in both these cases. As each node is visualized in the same view, collaborators can see how the context change as they read through the history. One can write questions like *Should we reduce income taxes?*, and *Family households?*, and *Single Parents?*. While one could say that we should not write questions like this, because they are closed questions, and that can lead to yes or no answers. But they are nonetheless common in everyday discussions [8, p. 120], and it is likely that collaborators will frame some IBIS questions the same way.

With meta-data queries, it is less significant whether the question is state the context explicitly or without any context at all. In both scenarios, meta-data queries enable collaborators to ask important questions such as: Is the premise for this solution sound?, and does this discussion address important questions?. Collaborators can reason about the context of any discussion. Each question, idea, or argument, that precede a searched contribution is exposed.

### 6.2.3 Consistency Checking and Error Detection

Some errors in IBIS discourse mapping can be less visible than others. For example if a KnAllEdge collaborator accidentally assign the wrong relationship to a node, or write a label that goes outside the scope of the discussion. To elaborate, we go back to the previous example *Should we reduce income taxes?* and contribute *Should taxes be reduced for family households?* as a specialization of the former question. This question exclude the word *income*, so it could refer to any tax type, making it a generalization. But it also narrows the scope to be *family households*, so it is also a specialization.

Because these two nodes must be linked together, and relationships links are clearly drawn in KnAllEdge, one could argue that the collaborator would easily detect this. But this assumes the map layout visualize the nodes in close proximity, and that the collaborator actually notice that there is an error. Representing the history explicitly like the prototype does, benefit collaborator’s by making these kinds of mistakes easier to detect.

---

\(^2\)Google’s artificial neural network, originally developed by DeepMind Technologies
Node that the severity of mislabeled nodes and wrong semantic descriptors might not cause much harm in some cases. If the mistakes does not influence the following discussions, and the group does not have to spend their time and energy to rewrite discussions. But a seemingly small mistake could potentially cause problems for software that analyze the map, i.e. for KnAllEdge’s automated tools that extract discourse from maps and merge it into topics. Ideally we should detect these mistakes and fix them as soon as possible, and meta-data queries support this process.

6.3 Limitations

Discourse Exploration

Meta-data queries is not an ideal tool for discourse exploration. From the query specification in chapter 3, we know that the history for each individual query is limited to its ancestors. And for each ancestor that is evaluated, no neighbor nodes are evaluated. This means that any questions, ideas, or arguments that are contributed at the same time as one of these ancestors, the ancestors’ siblings, are not included. For example see figure 6.4, here the collaborator submitted a Idea:Robotics query, the neighbor discussions related to the Google DeepMind topic is hidden.

Figure 6.4: Simultaneous Discussions

All nodes that are not part of the prototype search context are excluded from the visualized result map. There is no way to expand the nearby discourse nodes, as they are not part of the map as it is created. KnAllEdge developers expressed that they consider changing the plug-in data interface, so that the prototype can send commands to hide map nodes, instead of re-creating the map. This would be ideal, as collaborators would have a greater freedom to explore. If exploration is the goal, collaborators should study the original discourse structure instead of using the contextual search prototype.

Associating Prefixes with Node Meta-data

Observations from the workshop indicate that the prefix:suffix syntax was not intuitive for all participants. The set of prefixes implemented by the prototype are conceptually simple, perhaps even simpler than the ones
used in the workshop, they group nodes to meta-data properties such as author, creation date, and node type.

But if prefixes are mapped to more advanced or abstract queries in future updates, which is possible thanks to the mapping function hidden behind each prefix, it is likely that collaborators will not be able to associate the prefix with specific properties intuitively. Extra care must be taken to name a prefix according to what collaborators expect it to represent, or to include additional documentation to describe them.

### 6.3.1 Adding additional Query Prefixes

All supported prefixes are explicitly mapped in the source code, to create a new prefix it must be added in the source code, and the prototype application must be restarted for these to take effect.

### 6.4 Future Research

### 6.5 Improve the Query Language

The use of a query language for searching IBIS has its drawbacks, as we saw in the workshop. Text search is a common feature to have, as nodes have labels that contain only text. But the workshop results indicate that the query language can be difficult to understand. Half of the participants found it difficult to use, stating that it was ambiguous what a query meant. But the data from the workshop does not give substantial evidence, as to why the queries were complicated to understand. The author has recommended to the KnAllEdge developers, that they study this issue further during later workshops and experiments.

The author still recommend queries as a viable method for IBIS search, because they give a great deal of freedom regarding what type of discussions to search for, while being specific enough to search smaller groups nodes based on useful meta-data such as author and date.

### 6.5.1 Multi-faceted Search Visualization

The meta-data queries prototype extract IBIS nodes from discourse maps, and display them as a single map, where all the different discussions are merged together. However there have been studies on the benefits of visualizing information maps using separate views. Using multiple facets, where each show the same data from different angles. This type of multi-faceted views can better enable participants to better contextualize their surroundings when navigating large information networks [33, 34], and promote specific features of the information visualized.
Meta-data queries can visualize the search results in a separate view. As each query creates its own context, they could be visualized separate from each other. This way collaborators can get full overview of the map, and surrounding discourse, in addition to a overview of select discussions of their choice.

6.5.2 Support Cognitive Mapping

The prototype implemented in KnAllEdge create a new map each time a search is submitted, and large areas of the original discourse get hidden. This re-organization of the visual output is not a good approach for supporting cognitive mapping. Cognitive mapping is to utilize the brains spatial memory to remember information [14]. For example cognitive mapping explain why we are able to remember named locations on a map, given that we have seen them before.

Meta-data queries can potentially support cognitive mapping better by visualizing queries in the context of the original map, for example by outline the output of queries using annotations or colors on map nodes. This approach would likely yield more noticeable results on IBIS maps where there are several hundred nodes or more, cognitive mapping is often more beneficial when working on large visual graphs [14, p. 29]. One possible drawback is that the current prototype is specifically designed to avoid visualizing some areas of the map, those that are not relevant to a search. It might be sufficient to simply hide the content of the non-relevant nodes in this case, making them a part of the landscape but carry no labels or symbols, but this is uncertain.

To make cognitive mapping part of the map search, would naturally involve support from the KnAllEdge developers, as they would have to make changes to their visualization interface. It is recommended to do user experiments before dedicating research and development time, to investigate how much benefit cognitive mapping can potentially give.
The emergence of socio-technical solutions for collaborative knowledge work, enabling organizations, communities and researchers to collectively understand and resolve complex issues, both local and global, sets the stage for this thesis. Issue Based Information Systems augment our collective memory by remembering knowledge for us, and support reasoning about knowledge through its rigid and simple discourse structure. But the IBIS method is still only a partial solution, since also interconnecting knowledge and documents may increase the cognitive overload. In this thesis we have explored contextual queries as a way to overcome this deficiency.

We investigated how contextual meta-data queries can support discourse reasoning, and aid collaborators to manage information growth in KnAllEdge discourse maps. As an experimental IBIS search prototype, we also studied how a group of test participants interacted with the query language.

To achieve this, the author designed two prototypes. One basic prototype to gather novel learning about the queries, and a contextual search prototype that is currently integrated into KnAllEdge.

The results show that as maps expand, collaborators loose oversight of the scope of information that are relevant to their work, and lack of IBIS fluency require extra effort in studying the discussion context and surrounding discourse, before contributions are made. Collaborators must evaluate discourse that is often outside the scope of their own work, attributing to information overload. Meta-data queries is a viable method for searching IBIS discourse, it expose errors and mistakes in the structure, and facilitate discourse reasoning in complex IBIS maps. Meta-data queries reduce information load by filtering out discourse that is not within the context of the search parameters.

From the workshop we learned that the query language was difficult to use, for those with less technical experience. Unlike the basic prototype, the more advanced software prototype support input suggestions, and visualize output immediately. The author posit that further experimentation
with prototype embedded in KnAllEdge, will likely produce different results, warranting a more in depth user study.

Based on the results we conclude that:

- Lack of discussion oversight and IBIS fluency, increase information load for collaborators working with IBIS discourse maps.

- Contextual meta-data queries reduce information load by providing a complete discussion context within a discourse scope, as specified by meta-data search parameters.

KnAllEdge continuously evolve the practices and paradigms for knowledge mapping based on validated learning from the existing solutions. Contextual meta-data queries is now implemented in KnAllEdge, and is now part of the CollaboFramework toolkit. Upcoming workshops and collaborative projects, will run experiments that gather more validated learning about the prototype. This process is already under way with the first workshop experiment scheduled for the second week of May 2016.
Bibliography


Appendix A

Prototype Source Code

To get the source code for the meta-data queries prototype, visit https://bitbucket.org/Mgns87/meta-data-queries/overview. This is a Bitbucket repository and is based on git, git must be installed in order to clone the project. Alternatively the code can be downloaded as a zip file and extracted. Instructions for running the prototype locally on a computer, can be found in the README file inside the project folder.
Appendix B

Code for Topological Sorting

This appendix cover the source code to achieve topological sorting of the KnAllEdge IBIS graph. Definition of the algorithm: Given a directed graph \( G = (V, E) \), where \( V \) is vertices or nodes and \( E \) is edges or links, find a linear ordering of nodes such that for all links \((v, w)\) in \( E \), \( v \) precedes \( W \) in the ordering. Whenever there is a link from node \( A \) to \( B \), the ordering always visits \( A \) before \( B \).
```javascript
var topologicalSort = (function() {
  function topologicalSortHelper(node, visited, temp, graph, result) {
    temp[node] = true;
    var neighbors = [], nodeIndex = 0, nodeInstance = {};
    for (var i = graph.length - 1; i >= 0; i--) {
      if (graph[i]._id === node) {
        nodeInstance = graph[i];
        neighbors = graph[i].getIds('children');
        nodeIndex = i;
      }
    }
    for (var i = 0; i < neighbors.length; i += 1) {
      var n = neighbors[i];
      if (!visited[n]) {
        topologicalSortHelper(n, visited, temp, graph, result);
      }
    }
    temp[node] = false;
    visited[node] = true;
    result.push(nodeInstance);
  }
  return function(graph) {
    var result = [];
    var visited = [];
    var temp = [];
    for (var node in graph) {
      if (!visited[graph[node]._id] && !temp[graph[node]._id]) {
        topologicalSortHelper(graph[node]._id, visited, temp, graph, result);
      }
    }
    return result.reverse();
  }();
});
```
Appendix C

Code for Shortest Path Algorithm

This appendix cover the source code for finding shortest path between two nodes the KnAllEdge discourse map. Before this can be run, the input have to be a topologically sorted directed acyclic graph. Each node in the graph except the source node is set to have an infinite cost (distance). Then the code iterate through the nodes, and check if an adjacent node following after a given node, has a greater cost than the given node, plus the cost of the distance between them. If this is true then the cost of the adjacent node should equal the cost of the given node plus the cost of the link, and the parent for the adjacent node becomes the given node.

To find the shortest path between node A and B one can lookup the cost mapping array, as seen in listing C, and backtrack the parent chain from B to find shortest path to A. This implementation is a variant of algorithm 6, as described by David Eppstein [12].
```javascript
var shortPathDAG = (function () {
  var cost = {};
  function findPath(topSortGraph, sourceNode, destNode) {
    for (var i = 0; i < topSortGraph.length; i++) {
      cost[topSortGraph[i].id] = {};
      cost[topSortGraph[i].id].cost = Infinity;
      cost[topSortGraph[i].id].parent = null;
    }
    cost[sourceNode.id].cost = 0;
    for (var u = 0; u < topSortGraph.length; u++) {
      var edges = topSortGraph[u].childrenLinks;
      for (var y = 0; y < edges.length; y++) {
        var weight = edges[y].weight ? edges[y].weight : 1; // Set weight to 1 if unweighted.
        if (cost[edges[y].targetId].cost > cost[topSortGraph[u].id].cost + weight) {
          cost[edges[y].targetId].cost = cost[topSortGraph[u].id].cost + weight;
          cost[edges[y].targetId].parent = topSortGraph[u];
        }
      }
    }
    return function (topSortGraph, sourceNode, destNode) {
      findPath(topSortGraph, sourceNode, destNode);
      var subgraph = [], target = cost[destNode.id],
          temp = target;
      subgraph.push(destNode);
      while (temp.parent) {
        subgraph.push(temp.parent);
        temp = cost[temp.parent.id];
      }
      return subgraph.reverse();
    }()
  }
  return function (topSortGraph, sourceNode, destNode) {
    findPath(topSortGraph, sourceNode, destNode);
    var subgraph = [], target = cost[destNode.id],
        temp = target;
    subgraph.push(destNode);
    while (temp.parent) {
      subgraph.push(temp.parent);
      temp = cost[temp.parent.id];
    }
    return subgraph.reverse();
  }()
};
```
Appendix D

Women in Technology IBIS Map

This appendix include a transcribed IBIS map based on an hour long panel debate named *Women In Technology* from The White House’s YouTube channel. This map was transcribed by the author, to create an IBIS map for the workshop described in chapter 4. This debate between highly ranking leaders in several technology domains, was held at Facebook headquarters in Palo Alto in 2011. Follow this link, to see the full debate from the White House’s YouTube channel (Verified 15.5.2016): https://www.youtube.com/watch?v=T44XdGH5s-8.