



Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

Master thesis

Visual Information Management with Compound Graphs

Alvin Rindra Fazrie

4fazrie@informatik.uni-hamburg.de

Intelligent Adaptive Systems Master Program

Matr.-Nr. 6641834

First Reviewer: Prof. Dr. Chris Biemann
Second Reviewer: Dr. Alexander Panchenko
Advisor: Steffen Remus MSc.

Abstract

Traditional knowledge graph approaches for information management, which are based on a flat graph could impose a high cognitive load on the cognitive process of a user to understand concepts. In order to accelerate the cognitive process of a user, we design and develop a new prototype application for information management based on organizing concepts in a compound graph. An additional layer of abstraction is included to model generalized hierarchies of concepts and relations between concepts in a graph. This kind of graph representation is an added value for visualization purposes, where a user can have a better and faster understanding of the concept given by the information management. We also include a service-oriented architecture, in order to provide information in forms of subject-predicate-object triples. By conducting a usability test and gathering feedback from the users, the results show that compound graphs visualization can represent concepts better than normal graphs and the integration of semantic services can help a user to access information and to gain new knowledge.

Acknowledgement

First of all, I would like to sincerely thank Steffen for being my full-time thesis advisor. His assistance and precious guidance starting from the beginning of *autolinks* project and this thesis are always there for me. Whenever I need a help, he always gives his best and quick response, and I do really appreciate a lot for all his efforts.

I would also like to thank Chris for being my first reviewer and Alex for being my second reviewer. Their supports help me to complete this thesis and to smooth the process of the work from the beginning of *autolinks* project, thesis proposal, to the very end of the thesis.

Special thanks to the Language Technology Group members in creating a suitable and comfortable working environment during the completion of the thesis. We always have a "coffee" time, which means it is a time to play a kicker.

Finally, I would like to thank my parents and to my partner in life, Maria Ulfah, for their unfailing support and continuous encouragement throughout my years of study. This accomplishment would not have been possible without them. Thank you.

Contents

1	Introduction	7
1.1	Motivation	7
1.2	Research Questions	9
1.3	Contributions	10
1.4	Further Structure of this Thesis	11
2	Background and Related Works	12
2.1	Literature Review	12
2.1.1	Information Management	12
2.1.2	Ontologies in Computer Science	13
2.1.3	Semantic Web Technologies	15
2.1.4	Compound Graph	16
2.1.5	Information Visualization	19
2.2	Related Works	20
2.2.1	General tools for Information Management	20
2.2.2	Graph-based Information Visualization tools	21
2.2.3	Text Annotation Tools	24
3	System Overview	27
3.1	Architecture of the System	27
3.2	<i>autolinks</i> from User Perspective	28
3.3	Visual Information Management	30
3.4	Broker	37
3.5	Technology Stacks	37
4	Backend	40
4.1	Broker	40
4.2	Triples	46
4.3	Semantic Services	49
4.4	Information Extraction	50
5	Information Management Visualization	54
5.1	Knowledge Graphs	54
5.1.1	Building a Knowledge Graph	56
5.1.2	Merging Nodes Scenarios	58
5.1.3	Creating Relations Scenarios	59
5.2	An Algorithm for Knowledge Graph Extraction	60
5.3	Compound Graphs Motivation	63
5.4	A Layout Algorithm for Compound Graphs	66
5.5	Further Main Features in <i>autolinks</i>	69
5.5.1	Semantic Services	70

5.5.2	Query Exploration	70
5.5.3	Document Exploration	71
5.5.4	Annotation Exploration	72
5.5.5	Discontinuous Annotation	74
6	Qualitative Evaluation	76
6.1	Evaluation Setup	76
6.2	Feedback Details	77
6.3	Feedback Summary	80
7	Conclusion and Future Work	83
7.1	Conclusion	83
7.2	Future Works	84
	Bibliography	86
	Bibliography	86
	Appendices	91
A	Questionnaire	92
B	Feedback	96

List of Figures

2.1	DIKW Pyramid Model	12
2.2	Semantic Web Technologies Layer Cake [Domingue et al., 2011].	15
2.3	Compound graphs used in the data visualization on biological pathways. Figure taken from [Dogrusoz et al., 2005].	16
2.4	An example of a compound graph with adjacency and inclusion edges.	18
2.5	Reference model for visualization [Nazemi, 2014].	19
2.6	Storyfinder is a user based information management tool, which enables us to add entity from the web document [Remus et al., 2017].	23
2.7	<i>new/s/leak</i> is a data extraction tool, to provide important entities and the relationship based on the documents collections [Yimam et al., 2016].	23
2.8	<i>new/s/leak 2.0</i> annotates new entities, which were not detected by the NER [Wiedemann et al., 2018].	24
3.1	<i>autolinks</i> Architecture	27
3.2	<i>autolinks</i> User Interface with 'B Cell' as an example query	28
3.3	<i>autolinks</i> login interface	31
3.4	Knowledge graph canvas	32
3.5	Circlenav Button	32
3.6	Main navigation	33
3.7	Service list	33
3.8	Document list	34
3.9	Annotation Types	34
3.10	Upload a document	35
3.11	Document Lens	35
3.12	Bottom Grid Controller	35
3.13	Search bar	36
3.14	Properties of Nodes / Edges	36
4.1	Service endpoints	41
4.2	User endpoints	41
4.3	NLP endpoints	42
4.4	An annotation node 'T cell' with details view	43
4.5	Storage endpoints	45
4.6	Maintenance endpoints	46
4.7	RDF example in Turtle format [Harth et al., 2011]	47
4.8	A sentence example extracted from cTAKES- components, Figure taken from [Savova et al., 2010]	51
5.1	Entities as nodes	54
5.2	Relations as edges	55
5.3	An example of knowledge graph 'BCR'	57
5.4	Drawing relations to a node in different hierarchy	60

5.5	Hierarchies from flat graph	64
5.6	More complex hierarchies for compound graphs	65
5.7	Hierarchies based on properties	65
5.8	Two knowledge graphs generated by two queries in the search bar	71
5.9	Annotation nodes	73
5.10	Discontinuous annotation 1	74
5.11	Discontinuous Annotation 2	75
6.1	Does autolinks help you to access information?	77
6.2	Do you think that compound graphs represent concepts better than normal graphs?	78
6.3	Does the integration of semantics services help your research findings?	78
6.4	Do you think that autolinks hinders / blocks you to get information?	79
6.5	Compound graphs are ineffective compared to normal graphs	79
6.6	What are the best features shown in the autolinks?	80

List of Tables

5.1	Visual states of nodes and edges	56
5.2	Endpoints lists of semantic services	70
5.3	Sentences based on the annotation types	72

Listings

4.1	Server response of analyze endpoint	42
4.2	Server response of interpret endpoint	43
4.3	BCR resources example	48

1 Introduction

In this internet era, people could get information easily with search engines. They will give us a ton of hyperlinks clustered by multiple pages by entering a single query to the input, and then we could select a specific link we think the most relevant. The process of learning takes time sometimes. After the chosen web page rendered, we need to read through a page to get specific information related to the query given and sometimes we still have to deal with some hyperlinks to get further information. Even worse, many websites nowadays exploit the curiosity gap of the reader¹, offering just enough information to tease readers to be curious and not enough to satisfy the reader's curiosity, without clicking through another linked or related content. This clickbait phenomenon becomes so normal today² and it makes our time to study longer than before. To provide a quick researching platform, we develop *autolinks*³.

autolinks, 'automatic proactive researching', is a tool that provides a platform for quick researching based on a text or a sentence by visualizing the results together with their semantic relations. *autolinks* optimizes these concerns and is intended to make the learning process faster and more efficient. Instead of reading papers, websites, and other resources to understand a specific term, this machine will do it for us. From a text or a sentence given by the user, it will extract the information from multiple resources and visualize the core related information with the most convenient approach. The information is visualized by a force-directed graph, a graph, which contains nodes for the information and edges for the semantic relation so that it will ease the reader to understand how pieces of information correlate each other. Together with the implementation of compound graph concept, *autolinks* enables graphs to be inside of a node for the sake of a better categorization with an aim to improve and accelerate users' cognitive process in understanding concepts given in the knowledge graphs.

autolinks uses Natural Language Processing (NLP) components. NLP takes a responsibility to understand a given text and to comprehend which information from the sources have a relation to the given text and correlate each other. The reader could evaluate the results given and update them with the desired value. Bundled with this capability, *autolinks* accelerates the process of researching and understanding during the study.

1.1 Motivation

The motivation of this thesis comes from the idea to develop a compound graph-based information management prototype application, which provides different information needs of users by retaining their (global, personalized) knowledge, connecting their knowledge in context, and providing the source of information, called provenance. Provenance becomes vital in *autolinks* as it makes the difference in bottom-up information management from the

1. <https://thefinancialbrand.com/55342/content-marketing-strategy-curiosity-gap-clickbait/>

2. <https://theoutline.com/post/4716/how-everything-on-the-internet-became-clickbait>

3. <https://uhh-lt.github.io/autolinks/>

top-down information management tools. It acts as an external brain for users for the reason that users can remember what they have read and go back to it. It also enables the users to associate what they have created or edited with the original source of information.

The compound graph enables the data visualization to have an outer abstraction layer containing nodes as concepts and edges as relations between concepts in a knowledge graph, in order to accelerate the cognitive process of a user. This kind of graph representation has an added value for visualization purposes to enable users to have a better and faster understanding of the concept given by the information management. Knowledge graphs are composed of semantic triples relations, such subject-predicate-object triples so that users are able to explore and do research for certain entities and understand about their relations and categorizations in knowledge graphs.

Information Management with Compound Graphs

With Information Management (IM), users can manage their research findings to reduce complexity and organize information. Derived from ontologies concept, we provide knowledge graphs for knowledge representations with nodes to represent concepts and edges to represent relations between the concepts. The management enables users to edit, delete, or add new concepts and relations so that users can create their own concepts and environments.

One of the novelties in our information management is that we implement compound graphs for adding another abstraction layer to model concepts with generalized hierarchies. This graph model is applied in order to reduce the cognitive load of the user so that the user could get better and faster understanding of the concepts.

Triples

Resource Description Framework (RDF), as one of complementary technologies to express the semantics of the metadata in semantic web [Candan et al., 2001], is a general-purpose language [Domingue et al., 2011] initially designed as a metadata data model from a family of World Wide Web Consortium (W3C) specifications, to describe information from a web resource with supports both statement and resource-centric views. The RDF models can be treated as sets of triples and as sets of resources/properties [Candan et al., 2001]. RDF is used in knowledge management and information management applications.

Inspired by the RDF specification as sets of triples, we design semantics triples, which are basically RDF Triples with extensions. We call it triples to emphasize that we use it for drawing nodes and edges with an additional extension. This extension enables us to have recursive resources so that our resource could also contain another resource. Then by recursive resource, we are able to generate another abstraction layer of the concepts, which is called compound graphs. This compound graphs enable *autolinks* to have multiple levels of compound nodes as parents and normal nodes within compound nodes.

Query and Document Explorations

Doing research findings on the internet is a process that we want to identify as soon as possible all the relevant information from websites. In reality, this process takes time since there are tons of irrelevant information inside. It also happens when we read the documents.

Our main task here is to minimize the irrelevant information, so we have a better information access.

For this reason, *autolinks* provides query and document exploration, where we can model and enhance the information, which is contained in the document text. These features are provided in order to accelerate the speed of doing research findings, where we provide only relevant information in the knowledge graph. The knowledge graph can be created by entering a single query in the search bar or by clicking an annotated text in the main text field within the document lens. All annotations in the document lens are the result of information extraction from the processed documents. We need to upload documents beforehand, and this machine will analyze throughout document contents and return its analyzed results with entity annotations to be displayed in the document lens.

Information Visualization and Language Technology

Information visualization is a study in computer science, which focuses on constructing visual representation to amplify cognition [Card et al., 1999, p. 6], whereas the cognition here is further proposed as "acquisition or use of knowledge" [Card et al., 1999, p. 6], and the visual representation is typically extracted from large-scale textual information. The primary goal of information visualizations is to provide insights (discovery, decision making, and explanation) not only figure [Nazemi, 2014].

Information visualization is a vital thing for the research and data analysis because visual materials make data analyzing and understanding easier and prevent people from losing interest in a concept given. To visualize the concept, knowledge graphs are used for representing knowledge with nodes as entities and edges as relations between entities.

Language technology methods here are used to get entity annotations from the documents. One of them is Named Entity Recognition (NER). Since our case study is mostly in the medical domain, we use Apache cTAKES⁴ (clinical text analysis and knowledge extraction system) as Natural Language Processing (NLP) modules and semantic services [Savova et al., 2010]. Apache cTAKES is an open-source NLP system for extracting the information from electronic health records. Wiki media is used as an open knowledge base that acts as a basis for the structured data collection.

These points become the foundation of our research. The main goal here is to provide an information management tool that enables the user to get a rapid relevant information from the document contents and queries, so it benefits the user in research findings.

1.2 Research Questions

Derived from the motivation and the background of problems of this work, we conduct a usability test to evaluate whether the compound graph-based and information visualization methods' graph can be implemented to improve the general understanding of research findings.

The following research questions are examined in detail:

4. <http://ctakes.apache.org/>

1. Can compound graphs boost the user's understanding of the documents?
 - How can a user interface be devised that is non-intrusive, i.e., helping users solve their information needs faster instead of impeding them?
 - How can a compound graph be a better option to represent concepts?
2. Can the integration of multiple or generically semantic services help the research findings?
 - Which benefits or improvements that generic system, Service Driven Architecture, have compared to static design, where the information is given by one source, and it can not be changed?

To clarify these research questions, we will conduct a usability test and gather feedback to analyze how the compound graphs contribute to user understanding towards concepts. For this purpose, feedback questionnaires will be provided to the users after they test this prototype.

1.3 Contributions

The main purposes of this thesis are to answer research questions by developing and deploying *autolinks* 'automatic proactive researching' prototype to explore and analyze certain concepts and their relations. With compound graphs to model the knowledge graphs, this tool is intended to accelerate the cognitive process of users to understand their research findings via *autolinks*. State-of-the-art methods from information visualization and language technology are explored, adapted, and integrated to develop an information management tool with novelties extensions that benefit in research findings.

The novelties that we propose in this information management tool are the following:

1. Visualizing knowledge graphs with compound graphs model in order to give a better representation of the concepts so that users can get faster intuition and understanding towards the concept.
2. Extracting knowledge graph based on query and document explorations where we can construct and evaluate the information, which is contained in the document text and given by the semantic services.
3. Giving provenances in the extracted knowledge graphs in order to support users in retaining their global and personalized knowledge so that users can associate information in which they have already created or edited with the original source of information.
4. Integrating multiple semantic services to help users to have a quick access to information and to gain as well as to build knowledge from different sources of information.
5. Enabling discontinuous annotations, for example to annotate an entity from the main text by selecting words with different offsets so that users can annotate identical entities even though the entities have different offsets or text lengths.

We conduct a usability test to get user's feedback with an aim to evaluate our implementation. We find that compound graphs implementation gives a better concept representation than the flat graphs and integrating multiple semantic services help users to have quick information access in order to gain and build knowledge graphs from multiple sources of information.

1.4 Further Structure of this Thesis

In this thesis, we arrange the report into 7 chapters with some explanations in each chapter. The writing scheme is organized as follows:

- **Chapter 2 - Background and Related Work**

This chapter is consisting of related works and theories that we use to support this research.

- **Chapter 3 - System Overview**

Chapter three outlines the system overview, general architecture and components in *autolinks*.

- **Chapter 4 - Backend**

This chapter contains the description about the backend.

- **Chapter 5 - Information Management Visualization**

This chapter contains the description about the frontend part for the information management visualization.

- **Chapter 6 - Feedback**

We conducted usability test and gathered feedback to evaluate our implementation.

- **Chapter 7 - Conclusion and Future Work**

This chapter contains the conclusion that summarizes this work, and provides recommendations for the future works.

2 Background and Related Works

2.1 Literature Review

This chapter is divided into literature reviews and related works. Literature reviews' section explains various basic terms since they are required for further reference of the following chapters. Meanwhile, related works' section introduces some related tools to *autolinks* concerning information management tools and text annotation tools. In this section, we discuss and compare these tools with our tool: *autolinks*.

2.1.1 Information Management

In these days, we know that knowledge and information are the most important assets to be managed. They have relations towards each other as described by the DIKW (data, information, knowledge, and wisdom) pyramid model, shown in Figure 2.1, and they have been characterized as a chain [Zeleny, 2005, Lievesley, 2006] or framework [Chisholm and Warman, 2006]. [Zeleny, 2005] describes DIKW pyramid with appealing metaphors, which are 'know-nothing' for data, 'know-what' for information, 'know-how' for knowledge, and 'know-why' for wisdom.

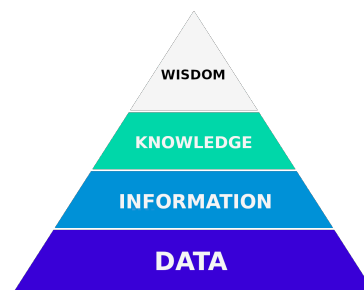


Figure 2.1: DIKW Pyramid Model

Humans can capture data or facts in information. When people confuse data with information, we can tell them that data is always correct, but information can be wrong. Data is the plain symbols such as numbers or letters, which are unorganized and lack any meaning or value [Rowley, 2007]. Information organizes data into a structure to give a meaningful purpose or value. With information, it also allows us to expand our knowledge. As a concept, knowledge is far more complex. Knowledge is the thing that we know and also contains beliefs and expectation. It is principally a result from cognitive processing and validation [Aven, 2013]. Wisdom is a shared understanding or as "knowing the right things to do" [Lievesley, 2006]. Wisdom enables us to do decision makings or informed actions effectively and also to reduce the decision risk. Wisdom also involves moral and ethics, such as human intuition, understanding, interpretation, and actions [Rowley, 2007].

Most people might see information and knowledge as synonyms. In fact, they have different definitions. In general, information is defined as organized data, data endowed with relevance and purpose, or interpreted data [Terra and Angeloni, 2003]. These definitions lead to the point where the information includes the participation from humans in order to organize the raw data to be more meaningful. Management is to organize and control the structure to process and deliver the information.

We can see that Information Management (IM) focuses on how we deliver the data to be meaningful information so that the people or user who uses the information management could get a better understanding or knowledge. The definition of knowledge, however, is much more complex than information. Unlike the information or data, which is more easily understood, classified and measured, knowledge is an asset or resource, which is invisible, intangible, and difficult to imitate [Terra and Angeloni, 2003].

Focusing on the individuals that create and own the knowledge is what makes Knowledge Management (KM) different from IM. So, it tends to be more people management, where individual human skills and behavior are changed and improved [Terra and Angeloni, 2003]. When we do research findings on the internet, we provide a specific query to be processed in the search engine. Then from the list of hyperlinks given, we want to know as soon as possible the relevant information related to our needs, which later on from this information, we can create relations among them to give a user intuition in order to get a better understanding.

The information management application is an application to process and organize the data that contains the information. IM can also be a database, concepts, mind maps, or simply a sheet of paper. In the case of *autolinks*, the information is arranged to be a collection of entities where they have relations as edges in a *node-link diagram* [Battista et al., 1998]. Node-link diagram is a common visual representation to represent entities as nodes and relations as edges or links for the sake of readability. The circles' shape or similar shapes for entities symbolizes an existence or a being, while the shape of line or link depicts a connection between existences. These entities should be editable so that a user can self organize the given results with better information. As a basic, to understand the principles in information management, we need to understand about the concept of an ontology.

2.1.2 Ontologies in Computer Science

From the history of the tower of Babel, we learn that people could not communicate since everyone spoke different languages. So, this is very important that understanding is only possible when speaking a common language. However what does it mean to speak a common language?

"People can't share knowledge if they don't speak a common language."

– Thomas Davenport (1997)

To speak a common language and to understand it, we need to share a common syntax, which has a meaning "setting out together or arrangement of the words" [Jurafsky and Martin, 2000]. According to [Manning and Schütze, 1999], syntax is the study of the regularities and constraints of word order and phrase structure. Moreover, on the other hand, there also must be a meaning agreement. So we need to define semantics. Semantics is a study of the meaning

of words, phrases, sentences, or documents. Taxonomy defines concepts classification; it is a collection of concepts organized into a hierarchical structure where it has a parent/child (broader/narrower) or super/sub concept relationships [Hlava, 2014].

Moreover, to have a formal knowledge representation, we need rules and knowledge about the concepts to support a common language of design [Kramer et al., 2015]. This rule is expressed in an ontology. Ontology is a central concept in philosophy, and it primarily denotes the study of existence. One famous ontology study is Aristotle's categories (384 BCE - 322 BCE) that enumerates all the possible kind of things that are possible to be a subject or predicate of a proposition. The compound word ontology comes from 2 Greek words, "Onto", which means the existence of being real, and "Logia", which means science or study [Knowles, 2005]. When people say "What is the ontological status of [category]?", it is just a way of saying "Is [category] real?".

The concept of ontologies as computational artifacts has recently been used in Artificial Intelligence and Computer Science. Especially in information systems, ontologies are a conceptual model of what "exist" in some domain [Gruber, 1993]. This conceptual model represents knowledge as a set of concepts within that particular domain with an aim to understand the relationships between these concepts [Hoekstra, 2009]. The concepts involve some formal namings, types, properties, and interrelationships (both for the taxonomic and non-taxonomic relationship) of the entities. To represent ontologies in CS, knowledge graphs can be used. We have nodes to represent concepts and edges for both directed or undirected edges to represent relations between the concepts. According to [Guarino and Giarretta, 1995], the same ontological theory may lead to different conceptualizations as the theory of distinctions among the entities (objects, events, regions...) or among the meta-level categories (concept, property, quality, role...) which may be relevant for knowledge representation and acquisition [Guarino, 1995].

The definition of ontologies in computer science are usually fixed by some authorities. In our case, they are created with a bottom-up approach, not the top-down approach. Traditional information management with top-down approach starts at the most general concept to specific concept and has some issues where the ontologies in the traditional information management are manually defined beforehand and fixed by design. This approach has a knowledge bottleneck where the concepts given in ontologies will never be complete. From time to time, there will always be new concepts or new insights into the ontology. These changes always happen, so that there are definitely issues with this kind of ontologies.

There is no system, which is perfect and we can conclude that there is a need for human involvement in order to self organize the data in our information management tool. So what we propose is a bottom-up approach, which is an entity-based. In opposite to top-down approach where everything has been defined and starts at the most general concept, our approach is that building an ontology can start from the text and create a kind of an entity graph directly from the given text. This process starts at the most specific concept, like a rabbit is a mammal, a mammal is an animal, and so on. We can build our own knowledge by defining it manually or we include existing knowledge bases for initialization then we can start to create entities and we can put relations to those entities.

2.1.3 Semantic Web Technologies

Web documents are now in the 3rd generation. From Web 1.0, which then we only had static web pages for information consumption, we moved to the 2nd generation, which is Web 2.0, where we have interactive web pages and also templates for accessing database information.

Various characteristic features provided by Web 2.0, which are distinguished from classical Web technologies are *Community* where it allows the contributors to have a collaboration and information sharing; *Mashups* where certain services from multiple websites can be extracted to experience and understand the data; *AJAX* - Asynchronous Javascript + XML where it is the basis of technology to introduce asynchronous communication for more responsive pages [Breslin et al., 2011].

In the 3rd generation, Semantic Web Technologies become a basis for Web 3.0 as an intelligent web where websites nowadays have intelligent infrastructure services behind the web. We have adaptive web pages, which implement netbots, information extraction, machine learning, natural language processing, etc. Semantic web technology also becomes a basis for modern information management. To understand semantic web technology, we can take a look at the semantic web technology stack [Domingue et al., 2011]:

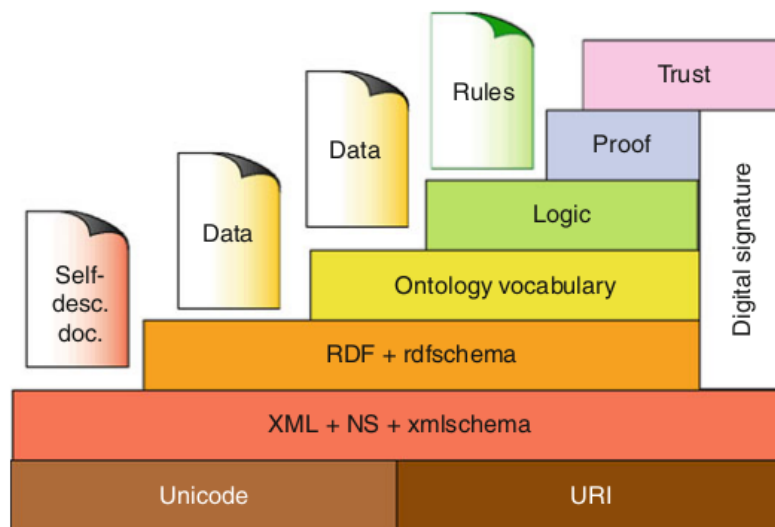


Figure 2.2: Semantic Web Technologies Layer Cake [Domingue et al., 2011].

From the figure 2.2, we can start from the lowest level where it has Unicode for text encoding and URIs (Uniform Resource Identifier) for the resource's reference and for defining a simple and extensible schema for unique worldwide identification of abstract/physical resources. The identification includes Web identifiers such as Uniform Resource Locators (URLs) or books and publications (ISBN, ISSN), and Object Identifier (DOI).

At a higher level, XML, namespace, and schema mechanisms provide syntactic descriptions for the structured objects. After this layer, five semantics layers are introduced: Resource Description Framework (RDF), Web Ontology Language (OWL), Rule Interchange Format (RIF), and layers of proof and trust. This layer type has significant functions to prevent

an upper layer from re-using functionality from the lower layer and allow an application, which only comprehends a layer below to interpret parts of definitions at a higher layer [Domingue et al., 2011].

RDF triples are implemented to emphasize that we use it for creating nodes, edges and the relations since RDF triples provide subject, predicate, and object tuples, which later on will be extracted for the visualization purpose. *autolinks* implements RDF triples with an extension, we call it semantic triples. The extension allows a resource to be again a list of resources. The recursion is necessary in order to facilitate compound graph implementations, so that we have graphs inside a node.

Graph theory in mathematics and computer science is the study of graphs. Using mathematical structures to model the pairwise relations between objects. The graph $G = (V, E)$ consists of a set of vertices ($v \in V$), nodes, or points, which are connected by a set of edges ($e \in E$), arcs, or lines [Trudeau, 1993]. The graph could be called an undirected graph, where there is no distinction from the edge direction towards two vertices, and it is called a directed graph, where its edges have directions towards a vertex.

Figure 2.3: Compound graphs used in the data visualization on biological pathways. Figure taken from [Dogrusoz et al., 2005].

For the implementation of source vertices and target vertices visualization, *autolinks* implements directed graphs. In a directed graph or digraph $G = (V, E)$, the direction of the edges is normally drawn as an arrow pointing towards the target vertices. The concept of the edges direction $e = (s, t) \in E$ is $s \rightarrow_E t$, where s is the *source* and t is the *target* of the edge $e = (s, t)$.

In the field of bioinformatics, the extracted data is required to be visualized in the state, which is not only coherently organized but also needs to be displayed by characteristics. The nodes will be clustered based on their strongest relations, their association to the same categories, or similar properties. To visualize this kind of abstraction, some graph models can be used. However, to represent complex relational information with the classical graph model is not sufficient because in the classical model we have large numbers of information visualized by nodes representation and this node numbers is the challenge that we need to reduce. Therefore, several classical graph model extensions have been proposed.

The clustered graph model is an extension of a classical graph model with partition. However, grouping or nesting partition in clustered graphs is only allowed for one level [Dogrusoz et al., 2009] so that the quality of results produced is unsatisfactory. Another model introduced by Harel [Harel, 1988] is the higraph model, which is a far more general model compared to the clustered graph model. The higraph model is a very general model for representing complex relationships [Harel, 1988]. The higraph model can represent inclusions, intersections, and adjacency relationships. This intersection relationship is the one that makes this model challenging to design and implement.

An intermediate model between clustered graphs and higraphs introduced by Sugiyama and Misue [Sugiyama and Misue, 1991] is called compound graphs. Compound graphs are more general than clustered graphs since this model allows to draw graphs with, both adjacency and inclusion relationships between nodes with multilevel nesting/abstractions [Bertault and Miller, 1999]. Figure 2.3 shows an example of a compound graph from the field of bioinformatics to display biological pathway information, which is a series of molecular reactions within a cell.

A *compound graph* is denoted as $C = (V, E, F)$ [Dogrusoz et al., 2009]. It consists of nodes V , the pair (V, E) forming the *adjacency edges* E or a (directed) graph, and the pair (V, F) , to which we refer as the *inclusion edges* I . The inclusion graph $T = (V, F)$ is a rooted tree, where the hierarchical structure of a compound graph is stored. It is assumed that $E \cap F$ is empty, which means a node cannot be connected to its children or parents by an adjacency edge.

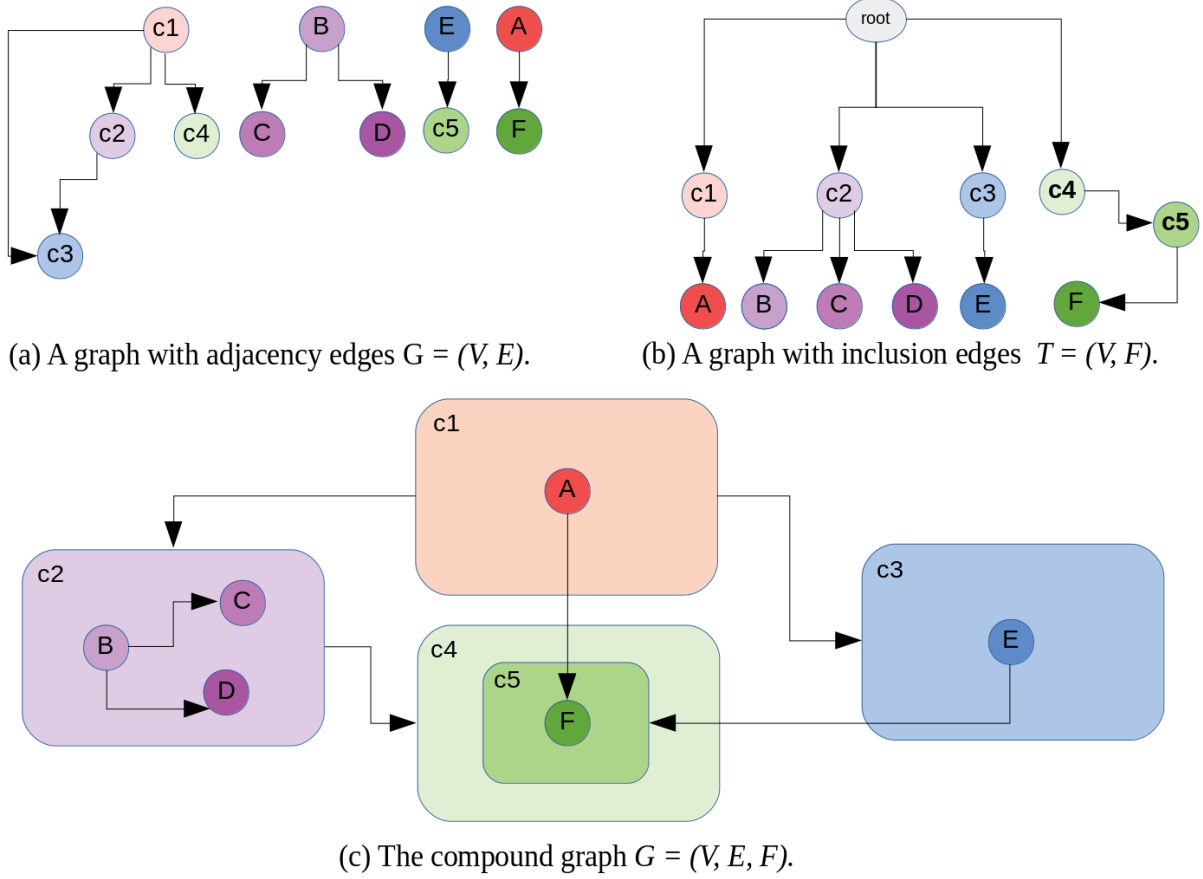


Figure 2.4: An example of a compound graph with adjacency and inclusion edges.

$$\begin{aligned}
 V &= \{c1, c2, c3, c4, A, B, C, D, E, F\} \\
 E &= \{\{c1, c2\}, \{c1, c4\}, \{c2, c3\}, \{B, C\}, \{B, D\}, \{E, c5\}, \{A, F\}\} \\
 F &= \{c1A, c2B, c2C, c2D, c3E, c4c5, c5F\}
 \end{aligned}$$

An example compound graph relationship with adjacency edges and inclusion edges is illustrated in Figure 2.4. Starting from the adjacency edges in Figure 2.4 (a): We can see, that edges denoted as $e = \{s, t\} \in E$ are drawn to indicate that there is a relation between the source node or *predecessor* denoted as s and the target node or *successor* denoted as t . This type of edges is direct adjacency relations in the compound graph illustrated in Figure 2.4 (c), that is why it is called as *adjacency edges*.

Edge $e = \{s, t\} \in E$ is called as *intra-graph edge*, if $s \in V^{G_i}, t \in V^{G_j}$, and $i = j$. If $i \neq j$, then we call e as an *inter-graph edge*. Figure 2.4 (b) illustrates the another type of relations, which is an inclusion relations, where $e = \{s, t\} \in I$. It represents a nesting relations between source and target nodes meaning that source s contains target t and we call this kind of edge type with *inclusion edges*. This type of edge draws compound relations. When a node contains a graph, it is called a *compound node*, otherwise, it is called a *leaf* or *non-compound node* [Dogrusoz et al., 2009]. Children nodes are the graphs that are located inside a compound

node and if a compound node still has other nodes, which are located inside its children nodes, we call them as descendant nodes. In opposite to *descendant* nodes, parent graphs that are higher than one level above of the current nodes hierarchy are called *ancestor* nodes. In Figure 2.4 (c), we can see that node F is the descendant node of node c4 and vice versa node c4 is the ancestor node of node F.

In Figure 2.4 (b), root graph (G_0) is a graph, which locates in the top of hierarchy. This graph only contains nodes, which have no owner/parent graph and it is resided at the 0^{th} level of the parent-node-child-graph tree structure. These nodes are called outermost compound nodes since they have the minimal depth and the compound node, which are located as a subgraph with maximal depth are called an innermost compound node. This iterative parent and children nodes' concept is what compound graphs differ from normal graphs. Normal graphs need more nodes and relations to make parent-children level relations than the compound graphs. Compound graphs reduce unnecessary nodes and edges and replace them with a single parent node to represent a higher / lower level concept in the ontology.

To realize the compound graphs visualization, we need to implement the layout algorithm for knowledge graphs, which is compatible with compound graphs. We chose Compound Spring Embedder (CoSE) as a layout algorithm, which is based on a force-directed layout scheme. This layout algorithm will be further explained in the chapter of Information Management Visualization.

2.1.5 Information Visualization

According to [Card et al., 1999, p. 6], information visualization is about transforming data into interactive graphical representations, which afford to amplify human cognition and support the process of information acquisition. In information visualization, human perception's aspect and the processing of visual information play a significant role.

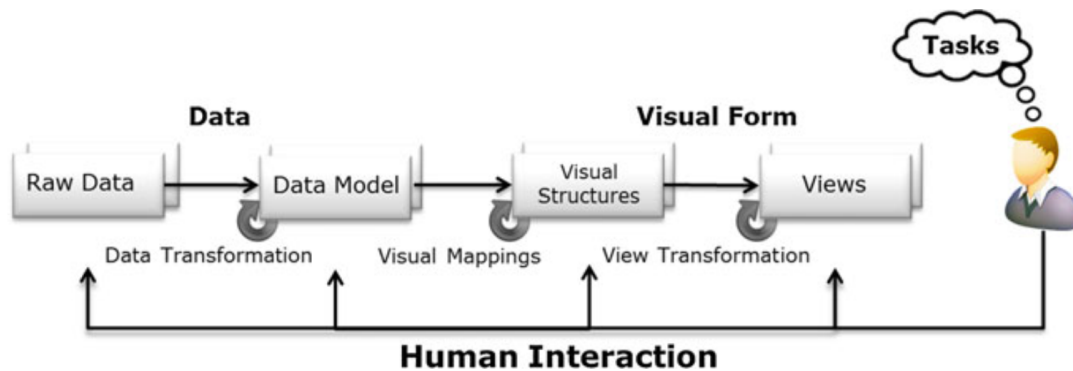


Figure 2.5: Reference model for visualization [Nazemi, 2014].

Data, task and the human perception and interaction involvements are the abstract views of information visualization structure [Nazemi, 2014]. Figure 2.5 illustrates the visualization reference model with its steps of transformation. This model is a primary foundation to comprehend the model of information visualization.

Data transformation is the first step of the reference model. Starting from the series of raw data formats with their structured sets of relations as a data table, it will be easier to visualize [Card et al., 1999]. A data table, which is represented by rows of variables as sets of values combines relations with their metadata description. The next step is the visual mappings to map the data tables to visual structures. In this visual structure, the visualization should support the human to interpret faster, to differentiate graphical entities, or to make fewer errors [Card et al., 1999, p. 23] so that the critical information will be well perceived by the human.

The final step after mapping the data model to visual structures is the loop of interaction between human and visual forms [Card et al., 1999, p. 31]. This step involves the manipulation of static graphical presentation to create different views of visual structures. This reference model introduces the processes of transformation starting from raw data to visual structures, the view manipulations involving location probes, viewpoint controls, and distortion, and human operations back the steps of transformation [Nazemi, 2014].

The work of information visualization is definitely highly related to visual analytics and it is a sub-discipline of visual analytics, but they apparently have some differences and gain their own established definitions. Visual analytics proposed by Thomas and Cook is the science of analytical reasoning facilitated by interactive visual interfaces [Thomas and Cook, 2006]. [Keim et al., 2010] augment the definition of visual analytics to be a combination of automated analysis techniques with interactive visualizations for an adequate understanding, reasoning, and decision making on the extensive and complex datasets. Based on this definition, the use of analysis techniques and interactive information visualizations are needed, giving a higher priority to data analytics from the beginning and through all iterations of the sense-making loop [Keim et al., 2008].

2.2 Related Works

We introduce some of the related works to *autolinks* regarding information management tools and text annotation tools. As it has already been mentioned in the information management section, information management can be anything as long as it is used to organize any information to be more understandable and accessible. Information management ranges from offline approaches such as concepts maps or mindmaps [Remus et al., 2017] to online approaches such as Content Management Systems, Wikis [Kaufmann, 2017], and Document-Cloud⁶.

2.2.1 General tools for Information Management

Learning new things is our nature as humans. The process of learning takes daily and we also memorize the information from the learning process for later reference. Forgetting things is also a human nature since the human memory also has limitations and sometimes the memory is not maximally utilized in an effective way and the information is not well-organized. In order to improve the performance of learning, there are some methods needed

6. <https://www.documentcloud.org>

to improve the human cognitive process. One of the methods is to implement an information management approach in order to organize the information ordered and easily accessible with attractive visual representation.

Concept Maps and Mind Maps

A concept map [Novak, 1998, Novak et al., 1984] is a top-down knowledge representation tool showing the relationships between concepts, a perceived regularity in objects or events which are designated by a label. The more general concepts are placed at the top and the more specific concepts are placed at the bottom, with 'cross-links' relationships among concepts [Novak, 1998]. A mind map [Buzan, 1974, Buzan and Buzan, 1976] is a color-rich and image-centered diagram to visually organize information. This map uses a radial diagram to represent semantic and branches to show relations between of learned material [Eppler, 2006]. Other approaches to information management are available such as conceptual maps and visual metaphors.

Concept maps, mind maps, and other tools such as conceptual maps and visual metaphors, [Eppler, 2006] are some offline approaches in information management. These approaches use information visualization to represent concepts and have been compared regarding their application parameters to get the advantages and disadvantages of these four visualization formats. Mainly, the advantages of these tools are to organize and to understand information faster and better, see [Eppler, 2006] for further comparisons.

DocumentCloud

Document cloud is a web service for creating and publishing document collections. This tool is designed for journalists to annotate and to scan text documents. The documents are analyzed using Named Entity Recognition (NER) provided by OpenCalais⁷, a web service for giving access to extensive information about entities in the text.

Content-Management Systems and Wikis

Content Management Systems (CMS) enable websites to become editable in the browser so that the content of the websites can be easily changed and supplied with new pieces of information. Wikis is one of the essential examples of CMS for information management. This feature provides collaborative editing of web pages within the browser, and by this feature, wikis are often to be a project to build a collection of articles on the topic of knowledge management, e-learning, and so forth [Stein and Blaschke, 2009].

2.2.2 Graph-based Information Visualization tools

Some of the related tools for bottom-up construction of a problem-oriented ontology we have are *Storyfinder* [Remus et al., 2017, Kaufmann, 2017], Networks of Names [Kochtchi, 2013, Kochtchi et al., 2014], or *new/s/leak* [Yimam et al., 2016]. Other tools such as FacetAtlas, PivotPaths, and SaNDVis, which are graph based information visualization tools to explore and analyze large network data, are also presented in this subsection.

7. <http://www.opencalais.com/>

FacetAtlas

FacetAtlas [Cao et al., 2010] enables the multifaceted visualization technique for visually analyzing rich text corpora. Multiple facets/topics of information which are extracted from an article such as symptom, treatment, cause, diagnosis, prognosis, and prevention are clustered by internal (in-facet) and external (cross-facet) relationships. The clusters with its relationships are intended to make users understand the cross-document and inter-corpus relationships as well as to organize the results of information retrieval. It also provides rich interactions such as highlighting, filtering, and context switching. The capability of the system in the healthcare domain conducted by the authors proves that this system is able to accurately cluster and visually deliver the diseases dependence, symptoms, and treatments.

PivotPaths

PivotPaths [Dörk et al., 2012] is an interactive visualization to explore faceted information resources. The strolling method is purposed as the use of interactive visualization to navigate various information spaces, which contain multiple facets and relations, such as authors and their journal articles in relation to publication date and category, via visual cues and interactive pathways. Then the viewer can explore them to make comparisons among facets and resources, and follow animated transitions among pivot operations.

SaNDVis

Social Networks and Discovery Visualization (SaNDVis) presented by [Perer et al., 2011] is a visual analytics tool that supports relationships among discovery tasks. This tool uses SaND and SanDGraph for data mining and model building, and to retrieve the results for queries of the social graph. As a people-centric visual analytics tool, the relationship discovery in this tool focuses on searching and finding people-centric tasks instead of documents. The tasks involve expertise location, team building, team coordination in the enterprise.

Storyfinder

Storyfinder is a user based information management, which enables us to add entities from web documents. This application consists of a browser plugin and a web server backend with the combination of natural language processing techniques and visual analytics. It aims at highlighting and managing the information from a web page [Remus et al., 2017]. Natural Language Processing (NLP) components play a role to analyze the web pages and to extract named entities and keywords from the web pages and store them for further reference.



new/s/leak

new/s/leak, the *network of searchable leaks*, is a data extraction tool, to provide important entities and the relationship based on the documents collections. This tool is a powerful tool for searching and exploring large amounts of data in a short time so that it can help journalists to get quick access to essential entities such as people, organizations or place, and their relationships. *new/s/leak* uses NLP preprocessing steps such as named entity tagging, time expression extractions, entity networks, relations, and metadata [Yimam et al., 2016].

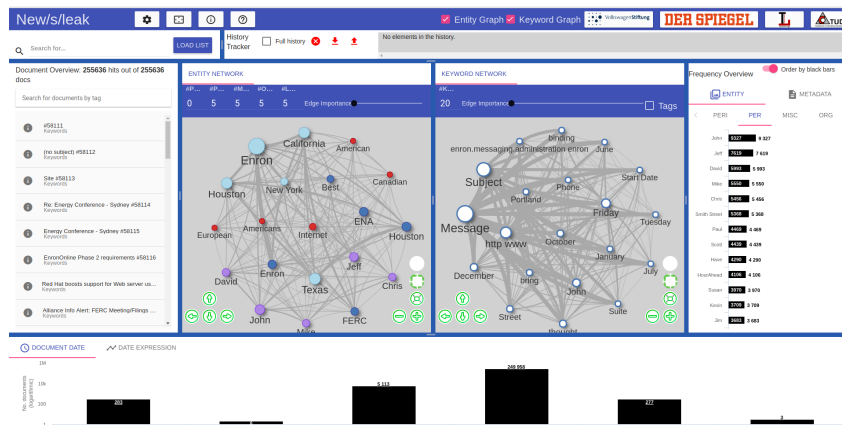


Figure 2.7: *new/s/leak* is a data extraction tool, to provide important entities and the relationship based on the documents collections [Yimam et al., 2016].

The new version of *new/s/leak* 2.0, an open-source software for content-based searching of leaks to supports journalist investigating information from the unstructured target text collections with multi-lingual data supports [Wiedemann et al., 2018]. In this version, *new/s/leak* 2.0 supports automatic language detection and language-dependent information extraction for about 40 languages. Not only entities are visualized in this version but also keywords for efficient exploration. *new/s/leak* 2.0 also supports manual and automatic text annotation. Manual annotation enables whitelisting new words to be annotated entities and it also could

blacklist unwanted annotations, which have been automatically preprocessed or manually annotated.

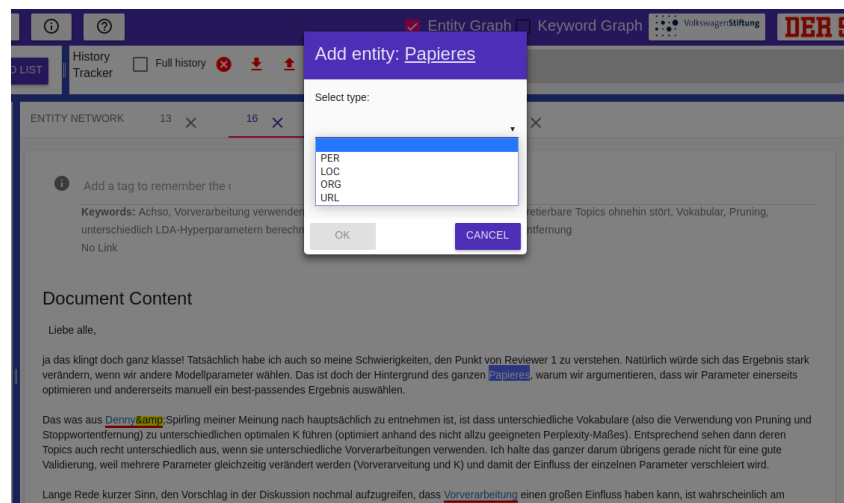


Figure 2.8: *new/s/leak 2.0* annotates new entities, which were not detected by the NER [Wiedemann et al., 2018].

Network of Names

The Network of Names [Kochtchi et al., 2014] identifies and explores relationships between individuals and organizations based on information extracted from a text corpus of newspaper articles. This tool extracts the names and the relationships between elements from the unstructured text. The identification process of people and organizations is performed in the preprocessing step using named entity recognition, and the relationships will be displayed in a web-based interactive visualization. Similar to *autolinks*, the visualizations use a node-link diagram in which nodes are the representation of entities and the edges between nodes are the representation of entities' relationships. The relationships are labeled in order for the system to learn the patterns and apply the labels to the similar relationships in other texts.

Network of the Day

Network of the day [Benikova et al., 2014] is a web-based interactive visualization that presents important terms of a single day incorporated with the information from the days before in a graph-based visualization. The extraction of important terms in Network of the Day is also processed using Named Entity Recognition and comes from online sources such as tweets from Twitter and articles from news sites. Similar to the Network of Names, the extracted element utilizes a clustering algorithm to cluster the related terms via sentence-level co-existence. The related terms are then displayed in the web-based visualization.

2.2.3 Text Annotation Tools

Text annotation tools are the tools that annotate text with annotation types such as part-of-speech, named entity (person, organization, location, etc.) or grammatical dependencies in order to analyze and synthesize the information within the text. At the moment, *autolinks*

provides annotation types in the biomedical domain and extends the tools that will be mentioned in this chapter, which only provide standard annotations. The annotation extension in *autolinks* is discontinuous annotation feature where annotating texts are possible within different spans.

WebAnno

Annotating documents is a time-consuming task, and it is not a low-cost process. Some tools of linguistic annotations in the past were developed using proprietary formats for data exchange and require local installation effort. WebAnno was developed [Yimam et al., 2013] to provide a web-based annotation tool so that users can use directly with internet access. It is interoperable in multi-data formats, supports annotations in a various level of linguistics, and provides quality management using agreement of inter-annotator.

GATE Teamware

GATE Teamware [Bontcheva et al., 2013] is a web-based annotation framework. It is an open source and supports collaborative annotation of texts, where it enables users to work on complex corpus annotation projects collaboratively. Different user roles such as annotator, manager, and administrator are provided. This cloud-based text annotation service will do a preprocessing step for documents, and the user can start with the pre-annotated text.

BRAT

[Stenetorp et al., 2012] developed the first web-based open source annotation tool, which is based on a client-server architecture and supports collaboration in the annotation for multiple annotation layers on a single document [Yimam et al., 2013]. In the beginning, this tool was developed as an extension of the stav text annotation visualizer [Stenetorp et al., 2011] to visualize the annotations for the BioNLP shared task 2011.

tagtog

tagtog is a web-based annotation framework and a multi-user tool that supports both manual and automatic annotation. Users can use dictionaries or train machine-learning models to annotate using the web editor or the API in the automatic annotations mode. It also supports different document formats and also supports biomedical domain articles from known repositories such as PubMed so that it can be utilized to annotate biological entities (such as genes) and concepts (such as Gene Ontology terms) from a complete text document [Cejuela et al., 2014].

The afore mentioned information management and text annotation tools implement language technology methods, so does *autolinks*. We implement NLP modules such as medical NER provided by Apache cTAKES. With service-oriented-architecture in our architecture, we deliver multiple wiki services to provide data for the knowledge graph creation. *autolinks* differs from the aforementioned tools mainly concerning visualization layout and extension of annotation feature. Compound graphs as the visualization layout give a better hierarchy visualization and a better concept representation.

Discontinuous annotation is an extension of traditional annotation feature where it enables to annotate entities manually in different text offsets. This feature at some point is necessary since standard annotations cannot always work if some identical terms are located in different offsets or have different text length. A simple example might be B cell and B - cell terms in a sentence and these terms are basically the same things, but in normal annotations, they are different entities.

Besides, provenance is also a feature that distinguishes our bottom-up information management tools from the other top-down ontology tools. In this bottom-up approach, we can build and update our own knowledge from the sources such as semantic services and uploaded documents. With provenance, a single concept entity will have its traceable sources where we can associate the information within the entity with its original source of information.

In this chapter, we have discussed the information management tools with visual representations in the node-link diagram form with force-directed graph layout. The layout is where concepts are represented by nodes and relations are represented by edges. Finally, we extend systems like the tools that have been introduced in this chapter, where flat graphs are the standard of visualization, with compound graphs implementation and discontinuous annotation implementation for the text annotation tools.

3 System Overview

This chapter describes the general system overview of the *autolinks*. We commence this chapter with the system architecture section and then explain *autolinks* from the user perspective section. After that, we briefly introduce our visual information management system and its components, and then we also briefly introduce the backend, which comprises of the broker, which handles the communication between a client and the services, and the database. In the last section of this chapter, we describe the technology stack used behind *autolinks*.

3.1 Architecture of the System

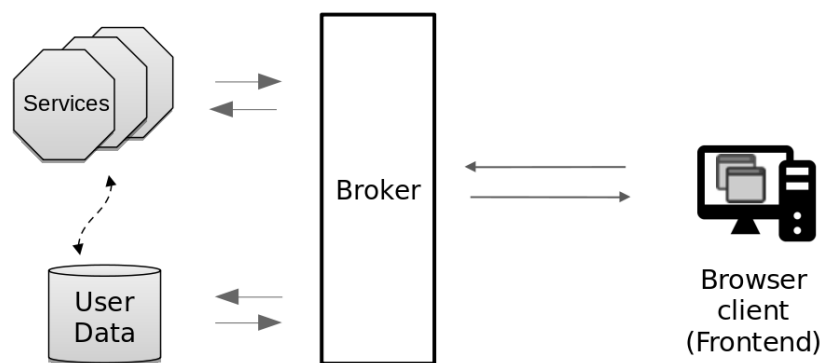


Figure 3.1: *autolinks* Architecture

The architecture of *autolinks* is depicted in Figure 3.1. The system architecture has two major parts: backend and visual information management (frontend). The backend consists of broker, services, and data storage.

A broker component in the middle of Figure 3.1 acts as a bridge to coordinate the communication between frontend, for forwarding requests, and services and data storage, for transmitting results and exceptions. With broker components, service-oriented architecture could be enabled in *autolinks*; thereby, we can plug many semantic services and other services for NLP modules, data extraction, and data management. Semantic services in the backend generate results of triples based on a query or entity exploration in the document, and then the results will be stored first in the database and will be forwarded to the frontend. Once the triples have already been generated by the semantic services and have been stored in the database, the next communication is done only between frontend and database for the resource that has already been extracted.

Meanwhile, the visual information management frontend provides the visualization of knowledge graphs and manage the knowledge graphs to be editable or user-based. The resources extracted by semantic services become the foundation to build the knowledge graph, and the recursive pattern of nested resources becomes a basis of the implementation of compound graphs.

3.2 *autolinks* from User Perspective

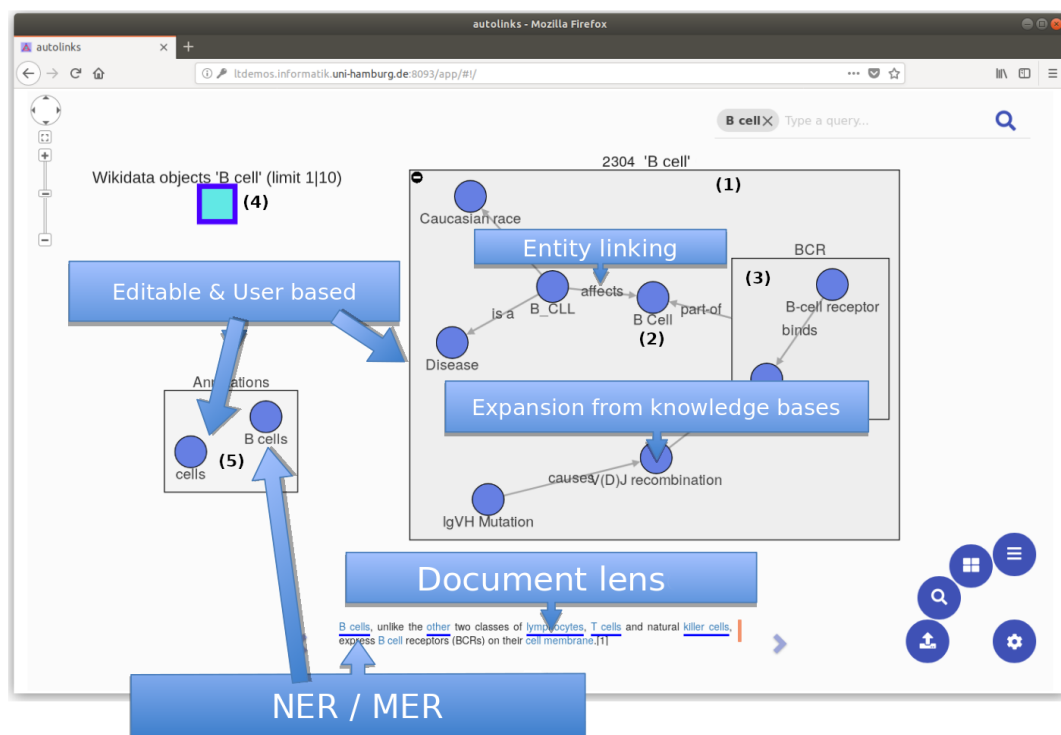


Figure 3.2: *autolinks* User Interface with 'B Cell' as an example query

From the user perspective, *autolinks* will lead the user to the main interface after login (see Fig. 3.2), which displays a canvas for the knowledge graphs. The Figure above shows two example graphs 'B cell' on the top left and on the right generated by two different semantic services. In the lower left, we can see an annotation container, which contains two annotation resources, 'B cells' and 'cells' visualized as annotation nodes.

We can see in the example, there are nodes as entities and edges as relations, which construct the knowledge graphs. One of the nodes, 'B cell', is a concept or an entity, which has a part-of relation to BCR entity. The BCR itself is a compound node since it contains children nodes such as B-cell receptor and Antigen. From the example given, we can describe some of the node types starting from (1), which is an outermost service compound node. This node is principally a compound node, which contains the knowledge graphs generated by a particular semantic service. Every knowledge graph provided by semantic service always has an outermost service compound node, since this node is extracted from the outermost resource identifier. The label of service compound node characterizes the semantic service.

We can see that in the graph view, we have two outermost service compound nodes, "2304 'B cell'" and "Wikidata objects 'B cell' (limit 1|10)". The '2304' number is the number of resource identifier generated by the demo semantic service, and "Wikidata objects" is the name of endpoint generated by SPARQL semantic service. Each semantic service has its own characteristics.

The number (2) is a normal node, which we call it as an entity 'B Cell' in this example. This standard node is the children of "2304 'B Cell' service compound node and it is in the lowest hierarchy of the service compound node since it does not contain anything. However, this node is a vital component to represent concepts and to draw relations among the concepts. The 'BCR' node (3) is also another compound node in the lower hierarchy level. This compound node is the children node of "2304 'B cell'" service compound node and in the same hierarchy of the 'B Cell'. Since this node has the same hierarchy level with the 'B Cell' node, the direct relation or the edges can be drawn from the 'BCR' to 'B Cell'.

The "Wikidata objects 'B cell' (limit 1|10)" node (4) is another outermost service compound node in the collapsed mode of the compound node. In *autolinks*, the implementation of compound graphs has a unique feature in which it is possible to expand and collapse the compound nodes. This feature helps a user to expand and collapse the compound parent nodes for better management of knowledge graphs complexity [Dogrusoz et al., 2018]. The results of knowledge graphs may vary depending on the source of semantic service. From one query or one entity in the text, a semantic service may produce different knowledge graphs with the others. Also, knowledge graphs in *autolinks* is editable and user-based so that the results also may vary for each user.

The nodes in number (5) are the annotations resources visualized as annotation nodes inside an annotation container. These nodes are special nodes and different from other nodes since these nodes are not allowed to have relations or to merge with other nodes. This annotation container and resources will be explained in further details in the chapter visual information management. From the user perspective, we have some other core implementations besides knowledge graphs in *autolinks*, which are described as follows:

- **Document lens**

A document that is uploaded in the *autolinks* is extracted and analyzed in the backend, and the backend will give its interpretation to the frontend. Once the document is already interpreted, the frontend then will show the document lens at the bottom of the graph canvas. The main text component in the document lens is then shown in sentencewise.

- **Named-entity recognition / Medical entity recognition**

A document text that has already been interpreted will have annotated texts in each sentence. The offsets of annotation are received from the annotation properties that are extracted by the NLP interpret endpoint. The properties contain the offsets, types, analyzer, and entity properties such as id, code, preferred text, and others. In Figure 3.2, we can see that annotated entities are highlighted with blue color in the document lens. These entities are recognized by Named-entity recognition (NER), a subtask of information extraction, to classify categories of the entities such as a person, organization, and locations. The NER also detects the location of entities index from the string position. In *autolinks*, we use a Medical Entity Recognition (MER) provided by Apache cTAKES.

- **Entity linking**

To build knowledge graphs, we need two vital components, which are nodes and edges. The edges have a significant role to draw relations and to give meanings among entities. Drawing relations in *autolinks* has some different scenarios and will be discussed in further details in the chapter of visual information management.

- **Editable and user based**

Every entity and its relations in *autolinks* have properties. The semantic services in the backend provide properties on every entity with metadata that contains a label, a description, an alias, and others. One of the metadata, label, is used for changing the name of the nodes; Not only label but also other metadata are fully editable. This user-based approach enables users to have their own personalized knowledge graphs in *autolinks*.

- **Source of Information / similar sources**

autolinks gives an extra knowledge to the knowledge graphs. Each entity is bundled with provenances (traceable sources) so that the user knows any source of information, which has a relation to that entity. This knowledge enables the user to jump to a particular document or knowledge graphs, which are related to the entity and have been previously seen by the user.

- **Expansion from knowledge bases**

Some entities in the knowledge graphs are the results of expansion from the semantic services. It means that even though those entities do not exist both in the document and in its interpretation, they are still shown in the knowledge graphs since they have relations to the entities, which exist both in the document and the knowledge graphs.

3.3 Visual Information Management

In this section, we briefly introduce the visual information management as a frontend client in the *autolinks* architecture. This frontend client is bottom-up information management and becomes the core part of the compound graph implementation. A bottom-up approach is chosen for the reason that *autolinks* helps users to construct their own personalized ontologies. At the moment, *autolinks* works in the biomedical domain and since *autolinks* is based on a service-oriented architecture where the service can be added or removed, thereby, *autolinks* may also work in other domains in the future.

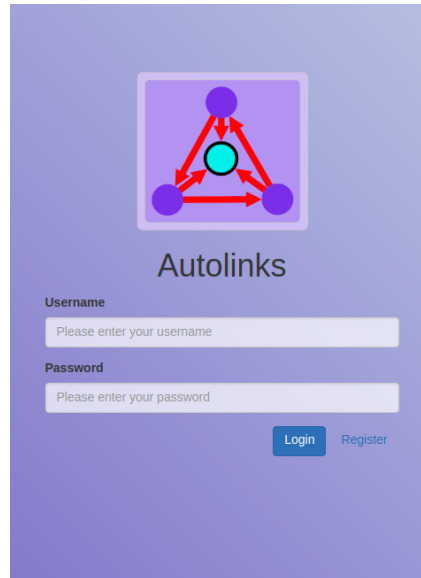


Figure 3.3: *autolinks* login interface

From the user’s point of view, *autolinks* is a web-based information management application. For demonstration purposes, this application is available at the server⁸ of Language Technology Group, Universität Hamburg, and can be run locally in user’s machine via docker. In order to make *autolinks* work, a user needs to run at least five docker containers, which are MySQL, broker, cTAKES-NLP, elasticsearch, frontend container, and at least one semantic service container. More semantic service containers can be added in order to make the results of the knowledge graph more diverse. After running those containers, users can open the application via a web browser with the port provided by the *autolinks* frontend container.

Figure 3.3 displays the login screen of *autolinks*. Since the *autolinks* architecture is designed as a multi-user system, user registration in *autolinks* is required in order to enable personalized information management and to ensure privacy purposes that only a particular user, which has privilege has access to his own knowledge graphs.

The authentication of users in *autolinks* is executed by using a username and a password. If a user has already registered to *autolinks*, he could immediately log in; otherwise, he can create a new user account on the registration page. The access data will be saved in the browser session, and thereby, users do not have to re-login if the browser is closed. After login, *autolinks* displays an empty graph canvas and users can start to build their own personalized knowledge graphs.

Components in the Frontend

To build knowledge graphs, users need to comprehend how to operate this application. There are some components in the visual information management needed to be understood and performed by the users. Those components in the frontend are described as follows:

8. <http://ltdemos.informatik.uni-hamburg.de:8093>

- **Graph Canvas**

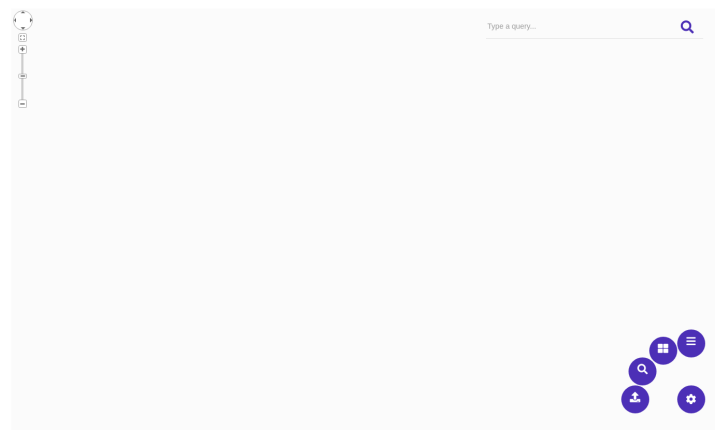


Figure 3.4: Knowledge graph canvas

A library of the network graph for analysis and visualization, Cytoscape.js, triggers the graph canvas. The Cytoscape.js core needs to be initialized by calling a Cytoscape function, *cytoscape()*, then a DOM container element is required as a document element parameter. This canvas is the HTML element where the knowledge graphs are visualized.

- **Circlenav Button**

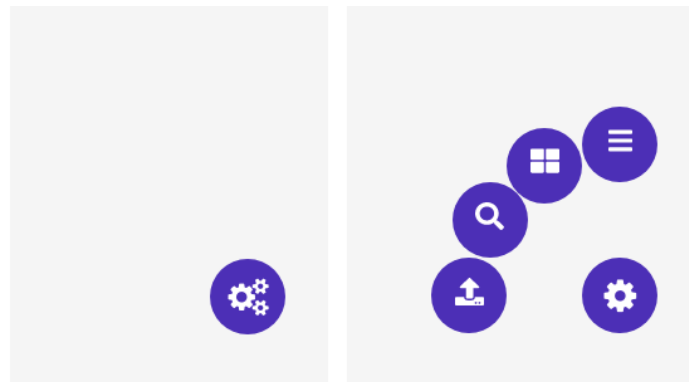


Figure 3.5: Circlenav Button

Circle navigation is the starting button that users see after login and it is laid in the right corner of the graph canvas. This navigation button will expand once clicked and display buttons for toggling the main navigation, bottom grid controller, search bar, and document upload modal.

- **Main navigation**

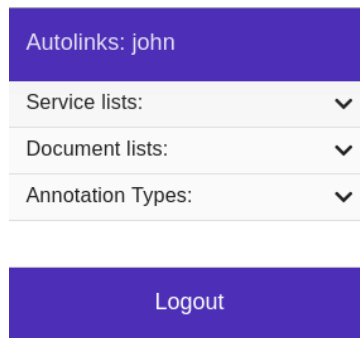


Figure 3.6: Main navigation

The main navigation is the main component to operate *autolinks*. This component has three main menus, which are vital for the query and document exploration features. The first menu is the service lists, which show the current active semantic services. The second menu is the document lists, which display the list of documents that have already been analyzed. The third menu is the annotation types menu, which is only activated once users select a document in the document list menu. Users can analyze what entities type in the text by enabling one or more annotation types in the annotation types menu.

- **Service List**

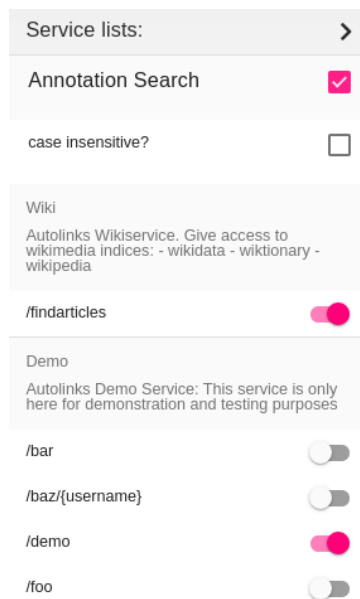


Figure 3.7: Service list

Service list is a menu that displays the active semantic services and the annotation search. The services that are currently active in the *autolinks* are Wiki service and Demo service. Wiki service provides access to Wikimedia indices such as Wiki data, Wiktionary, and Wikipedia, meanwhile, The demo service is only for demonstration and testing purposes.

The annotation search service helps users to retrieve the annotation resources that are related to a query given. With an annotation resource, users could call other related

annotation resources with the same label, but different offsets in a document or users could call knowledge graphs constructed by a particular semantic service, which have the same label to the annotation resource.

- **Document List**

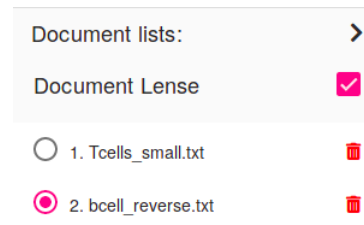


Figure 3.8: Document list

In the main navigation, documents are stored and listed in the document lists menu. There is an option also to enable or disable the document lens in the middle of graph canvas. Selecting a document in the document list will trigger the annotation types menu to display. This annotation types menu is located below the document list menu.

- **Annotation Types**

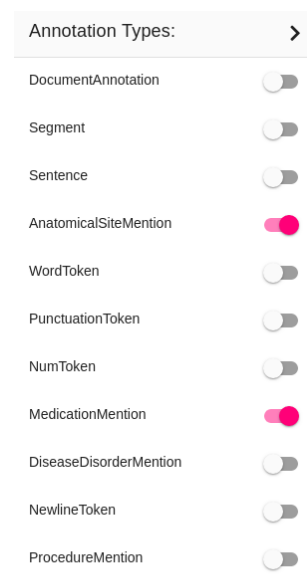


Figure 3.9: Annotation Types

Annotation types are the result of document exploration feature. The uploaded document is analyzed to extract the medical entity information from texts in the document by using Medical Entity Recognition (MER). MER is a type of Named Entity Recognition that works in detecting the words, which are related to the biomedical domain. We can enable a specific annotation type or enable multiple annotation types at a time for a single sentence shown in the document lens.

- **Upload document dialog**

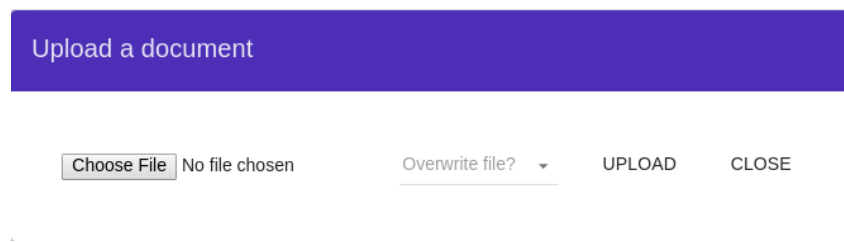


Figure 3.10: Upload a document

This upload document is a feature to upload a document. At the moment, we limit the size of the document to 10kb, which composes around 1-10 sentences. *autolinks* currently only receives .txt file extension and it has an option of overwriting a file via document upload modal. This option is to overwrite a same-named document that has already been previously uploaded with the latest document. If we choose to overwrite a file, the old file will be replaced; otherwise, *autolinks* will give a warning that the document has already been uploaded. Once the document is successfully uploaded, it will appear in the document list in the main navigation.

- **Document Lens**

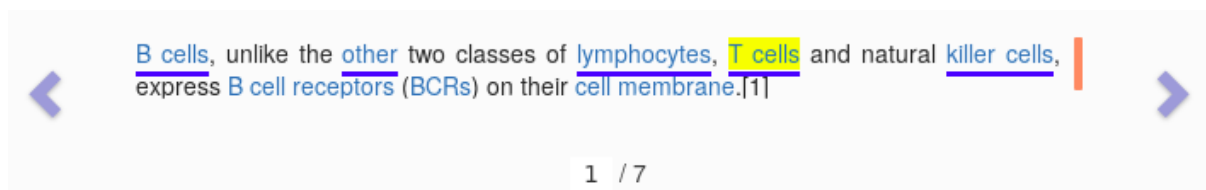


Figure 3.11: Document Lens

The document lens contains the main text, which becomes the user focus. The extracted and analyzed documents are shown sentencewise. Then the user can navigate to a particular sentence number by clicking the left/right arrow navigation or enter the selected sentence number in the indicator input. A yellow highlight on the text 'T cells' indicates that a user is hovering on the annotation nodes related to 'T cells' annotation on the text.

- **Bottom Grid Controller**

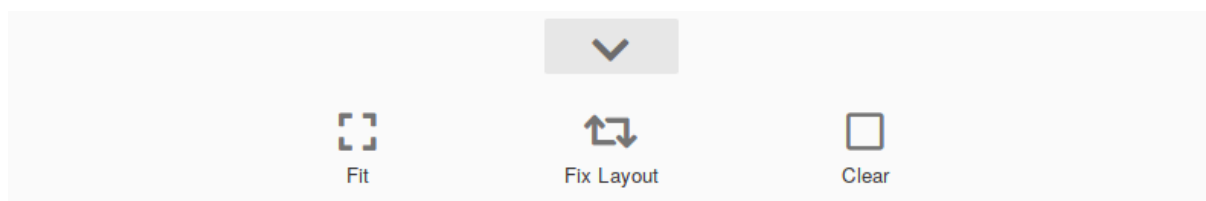


Figure 3.12: Bottom Grid Controller

Bottom grid controller is a component in *autolinks* that allocates buttons for external functionalities. These external functionalities can be view manipulations (Location fitting, Location Fixing), clearing canvas, and viewpoint controls (magnifying or changing the viewpoint by zooming or panning). Extra space in the bottom grid controller is meant as an area for other future features buttons.

- **Search bar**

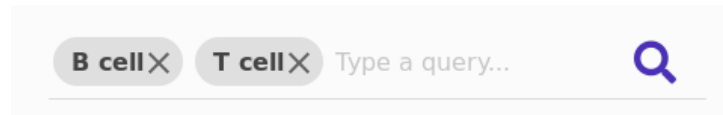


Figure 3.13: Search bar

Search bar enables query exploration. By entering a query, a knowledge graph can be visualized based on their selected semantic services. *autolinks* search bar does not only support for a single query from a single semantic service but also multiple queries from multiple semantic services.

- **Side navigation (Nodes/Edges details)**

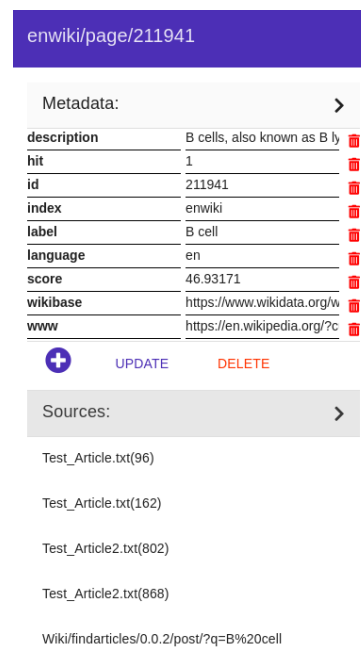


Figure 3.14: Properties of Nodes / Edges

The nodes/edges details are placed in the side navigation, which is meant to display the details or the nodes/edges including metadata and sources. The metadata in the side navigation is entirely user-based and editable. In the metadata menu, users can add new metadata values or change the label of the node by adding label text as metadata key and desired value as metadata value and then clicking the update button. The sources menu is principally the provenances that we have already mentioned in the introductory chapter; we can select sources of nodes/edges information from the uploaded documents or semantic services.

3.4 Broker

Broker is responsible for forwarding information between a client and services and to handle user data and authentication. A single broker can manage multiple services, and these services provide information as resources triplets through a broker. *autolinks* implements Service-oriented Architecture or Service-Driven Architecture. With this architecture, it enables the user to plug in or remove services in the *autolinks* runtime.

Modules in the Broker

Modules available in the broker are service, user, NLP, storage, and maintenance.

- **Service**

In general, service module provides a list of active semantic services and returns a list of resources for the knowledge graph extraction.

- **User**

User module manages user authentication from the registration of a user, information of a user, deletion of a user, and user update.

- **NLP**

NLP module analyzes the text from uploaded documents and extracts the information within the text to obtain significant entities such as people, places, or organizations and their relations.

- **Storage**

The storage module is vital to read and write of user data. The process of editing resources and documents is also done in this service. To annotate a new entity, we can use addannotation endpoint via this service.

- **Maintenance**

Maintenance module is the property to handle the availability of the broker and to handle the error of the prototype. User data and the storage can be reset via this service.

3.5 Technology Stacks

In this section, we introduce the programming languages and frameworks used in the *autolinks*' architecture both the backend and the frontend. We focus only on the main components without minor libraries and explain the reasons upon their selection.

Backend: Broker and Semantic services

We use **Node.js**⁹ for the *autolinks* backend implementation. By its event-driven, non-blocking I/O model, Node.js was chosen since this model makes Node.js lightweight and efficient. The input/output model is normally a blocking I/O model where in this blocking method, a scenario from a user needs to be finished first before performing another scenario from another user. In a non-blocking I/O, we can initiate a scenario to be independently carried out, thereby, the server could handle multiple requests at the same time.

Node.js is built as an asynchronous event-driven Javascript runtime on Chrome's V8 Javascript engine and bundled with **npm**¹⁰ (Node package manager) that loads modules or packages from the third parties, which make the development efficient and faster. This package manager for Javascript allows to manage dependencies of the project and modularize the project.

For the development of *autolinks* API, we use **swaggerUI**¹¹ for the design, development and the first approach of documentation. This tool supports multiple programming languages for the client and server code and manage to have proper documentation for REST services testing. This tool enhances the experience of the developer with interactive documentation of API. This tool also enforces to have a standardization for the API architecture; thereby, our service implementations must follow interfacing policies such as parameter and return type.

Since at the beginning of development, *autolinks* focuses on the medical domain. For this reason, Natural Language Processing module that could recognize information from the medical perspective needs to be used. In this case, we chose **Apache cTAKES**¹².

Apache cTAKES (clinical Text Analysis and Knowledge Extraction System) is an open-source natural language processing module for information extraction from electronic medical record clinical text. The reasons why we choose Apache cTAKES are not only that it is an open source and 100% free project but also that it has a great performance since it could process 50,000 clinical records per hour in the database or file-stored batches to create rich linguistic and semantic annotations that can be used for the decision support systems in the medical research.

Apache cTAKES was built upon **Apache UIMA**¹³ (Unstructured Information Management Architecture), a framework that facilitates the analysis of unstructured content such as text, audio, and video. This framework enables the modularization for Apache cTAKES so that it can be customized to use any component, which is compatible in Apache UIMA.

We store resource triples data and annotation data into **MySQL**¹⁴ database and for the user data, we use **SQLite3** due to experimental reason. Storing a large number of data will affect the database performance. In this case, we need to use indexing tools to make indexing data much quicker. An indexing tool that we use is **Elasticsearch**¹⁵.

9. <https://nodejs.org/>

10. <https://www.npmjs.com/>

11. <https://swagger.io>

12. <http://ctakes.apache.org/>

13. <https://uima.apache.org/>

14. <https://www.mysql.com/>

15. <https://www.elastic.co/products/elasticsearch>

The backend services, visual information management client, and the database are containerized by **Docker**¹⁶. Docker is a platform to build, ship and run applications in local and cloud. It provides scalability and independence of Operating System and actual implementation.

Visual Information Management

The client, visual information management, is developed fully in JavaScript with frameworks and library both for the development of front-end web application and for the development of a compound graph in knowledge graphs.

AngularJS¹⁷ was chosen to be the front-end web application framework since this framework has already been in a stable release compared to the newer version of angular, which was in inconsistent releases. Node.js is implemented for the front-end server as a middleware to utilize the API from the backend with npm as the package manager.

autolinks was built as an HTML5 canvas-based application for drawing the knowledge graphs for performance reason, unlike other web applications, which are based on SVG/DOM objects to draw knowledge graphs. We need to make sure the framework that we used is compatible with the utilization of HTML5 canvas. Even though the newer version angular is getting more popular now and also its contender, which is ReactJS becomes the most popular front-end framework to be used. It does not mean they are the best choice for developing an HTML5 canvas-based application since they have different natures such as the implementation of Virtual DOM in ReactJS. Compared to the newer version of angular, AngularJS has a better learning curve so that it makes the development of *autolinks* project faster.

The core of visual information management is the compound graph. This kind of graph was enabled to be extracted for the creation of knowledge graphs by the implementation of **Cytoscape.js**¹⁸. Cytoscape.js [Franz et al., 2016] is an open-source JavaScript-based graph library. Its most common use case is as a visualization software component, so it can be used to render interactive graphs in a web browser. It also can be used in a headless manner, useful for graph operations on a server, such as Node.js. Cytoscape.js provides a JS application programming interface (API) to enable software developers to integrate graphs into their data models and web user interfaces. What makes Cytoscape.js different from any other visualization libraries is that Cytoscape.js supports numerous graph theory use-cases such as directed graphs, undirected graphs, mixed, loops, multigraphs, compound graphs, and other kinds of graphs.

Most network visualization libraries have a limitation to draw a more generalized hierarchies model and only supports clustering edges and nodes in one level. On the other hand, Cytoscape supports compound graph, which becomes the basis of *autolinks* visualization. This kind of graph enables multi-level of grouping or generalization.

16. <https://www.docker.com/>

17. <https://angularjs.org/>

18. <http://js.cytoscape.org/>

4 Backend

In this chapter, we explain the backend of *autolinks*. The backend is responsible for Information Extraction (IE) and data storage. The information extraction is processed by NLP modules managed by the broker and the results of information extraction will be stored in the database. The broker also returns the server response with triples format to be visualized in the information management for the purpose of knowledge graphs visualization. In the following sections, the purposes and the features of the backend will be discussed in more details.

4.1 Broker

autolinks implements a service-oriented/driven architecture, and to enable that we use a broker component. The broker component is responsible for the coordination of communication among multiple services, such as for forwarding requests, as well as for transferring results and exceptions. Our broker is also responsible for managing user authentication, services, NLP modules, storage management for resources and documents, and also maintenance. These modules are explained as follows:

Service

The service module manages the connected semantic services and notifies the visual information management about a list of active semantic services. These active semantic services then generate a list of triples' resources for the knowledge graph extraction in the frontend. Other functionalities of this module are distributed at some module endpoints and they can be seen at Figure 4.1. From these endpoints, only some endpoints are used by the visual information management. Starting from GET service/listService endpoint, it gives the active semantic services to the visual information management. The active semantic services will then be visualized in the *autolinks* main navigation.

The POST service/call endpoint is the key to build the knowledge graphs. This endpoint returns the triples to the visual information management, and the information management will extract the triples to be nodes as entities and edges as relations. This endpoint requires the service name, version, path, method, offsets detail and context. We need to specify the offset length of the context in the offsets data value and the query given by the user in the text value of the context. Every time a user enters some queries in the search bar, each query will be posted to the context value in a single request to the server and this endpoint will be called according to the number of queries given.

POST	/service/call	🔒
GET	/service/get/{service}/{path}/{version}/{method}/	🔒
POST	/service/details	
POST	/service/registerService	
POST	/service/deregisterService	
GET	/service/listServices	
POST	/service/pingService	
GET	/service/pingService	

Figure 4.1: Service endpoints

User

User module manages user authentication from the registration of users, information of users, deletion of users, and update of users. GET user/info endpoint gives the details about the current logged in user starting from the id, name, active status, registration time and last seen time. For the user registration, at the moment *autolinks* only requires name and password and it is done via POST user/register endpoint. For deleting a user, it can be executed via GET user/delete endpoint and for updating a user authentication data, it can be performed via POST user/update endpoint.

GET	/user/info	🔒
POST	/user/register	
GET	/user/delete	🔒
POST	/user/update	🔒

Figure 4.2: User endpoints

NLP

NLP module analyzes the text from uploaded documents and extracts the information within the text. A standard NLP module extracts significant entities such as people, places, or organizations, In *autolinks*, we extract more specific entities in the medical domain such as anatomical site mention, medication mention, procedure mention, and also token annotations such as punctuation token, numeric token, and word token. There are two main endpoints in the NLP module, which are GET /nlp/analyze/did and POST /nlp/interpret/did. Analyze endpoint requires a document identifier to extract the information from the chosen document. This process involves Named Entity Recognition (NER), which will be explained more details in the information extraction subchapter. After the NER and relation extraction processes, this

endpoint returns the annotation responses, which are composed of properties, discontinuous offset (doffset), type and analyzer of entities. It starts from splitting the document text in sentencewise, sentence by sentence then it returns the annotations of that sentence, entity by entity.

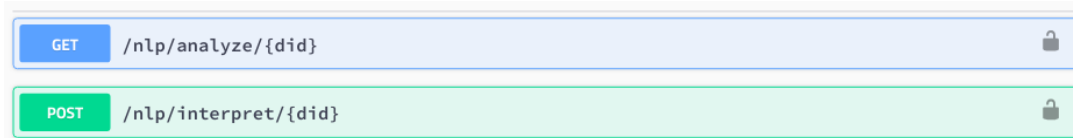


Figure 4.3: NLP endpoints

The properties provide details of the entity including the identifier, concept unique identifier, unique type identifier, preferred text, and others. Below is the example of analyzing endpoint response from the 'T cell' annotation.

```

1      {
2          "properties": {
3              "id": "0",
4              "cui": "C0039194",
5              "tui": "T025",
6              "code": "57184004",
7              "score": "0.0",
8              "typeID": "6",
9              "generic": "false",
10             "polarity": "0",
11             "historyOf": "0",
12             "conditional": "false",
13             "uncertainty": "0",
14             "codingScheme": "SNOMEDCT_US",
15             "disambiguated": "false",
16             "preferredText": "T-Lymphocyte",
17             "discoveryTechnique": "1"
18         },
19         "doffset": {
20             "offsets": [
21                 {
22                     "from": 54,
23                     "length": 7
24                 }
25             ]
26         },
27         "type": "AnatomicalSiteMention",
28         "analyzer": "CtakesNLP"
29     }

```

Listing 4.1: Server response of analyze endpoint

Listing 4.1 displays the details of 'T cell' entity. From the example given, 'T cell' has properties such as a concept unique identifier = C0039194, a type unique identifier = T025, a preferred text = "T-lymphocytes", etc. For the offsets, it starts from the string position 54 with length of the string 7. Type key here is the annotation type and this key plays a significant role

for the visual information management since with this type key, the frontend could identify which annotation types that the user wants in the document.

From the example above, the 'T cell' has an annotation type 'Anatomical Site Mention' given by the apache cTAKES Natural Language Processing as the analyzer. With this type value, the user could enable the 'Anatomical Site Mention' as the annotation type and the 'T cell' annotation will then be shown/highlighted in the document text.

The NLP interpret endpoint provides the process of information extraction at the annotation of entity by providing the annotations at a given offset/doffset. It requires not only the document identifier but also the chosen text offsets for both offset-from and text length. The process of this endpoint extracts the information from the offsets given and returns its interpretation to the information management. The interpretation result produces the annotation resources with metadata and provenances (traceable sources). This annotation resources will then be visualized as annotation nodes in the graph view and they will be located inside a special container namely annotation container. An example of annotation node 'T cell' with opened details (metadata and sources) view is shown in Figure 4.4. The annotation nodes and the annotation container will be discussed further in the chapter visual information management.

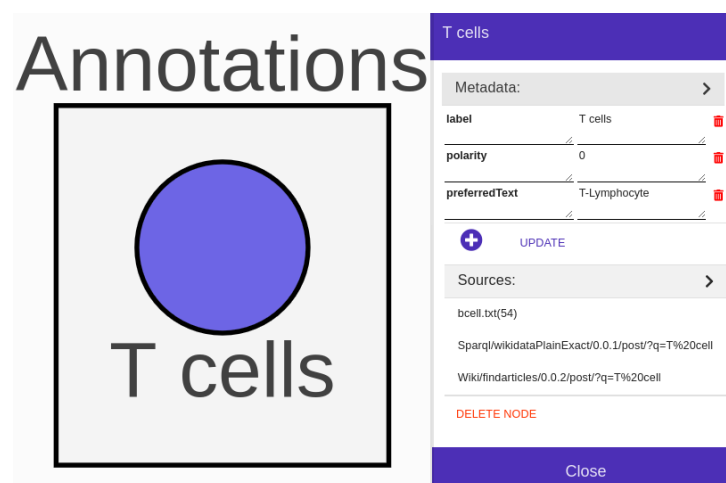


Figure 4.4: An annotation node 'T cell' with details view

Metadata in annotation resources are fully editable. It consists of the properties of the annotation nodes given by the cTAKES NLP and a label, which is used for node labeling. Provenances, which are the traceable sources in *autolinks* support users to have an extra knowledge where the information comes and has relations. The concept of provenances in *autolinks* is mainly to provide the source of origin of the nodes and it is located on the sources menu in the side navigation of the nodes details. With sources, a concept / a node has an ability to access its sources of information for both documents and semantic services. Users can jump to the related document with its specific sentence or users can rebuild the knowledge graphs from the related semantic service.

```
1 | "sources": [
2 |   "annotation::CtakesNLP:AnatomicalSiteMention:1:54:61",
```

```

3 || "service::Demo/demo/0.0.2/post/?q=T%20cells"
4 || ]

```

Listing 4.2: Server response of interpret endpoint

Sources from semantic services and sources from a sentence in the document have different structures key given by the interpret endpoint, shown in the Listing 4.2. Semantic service source key starts from "service::" string, followed by the name of the service, path, version, method, and closed by the value of the query given by the users. The document source key starts from "annotations::" string, followed by name of the NLP analyzer, annotation type, document identifier, starting offsets, and closed by the end value of offsets.

Storage

The storage module is a vital part of reading and writing of user data. The process of editing resources and documents is also performed via this module. GET /storage/document endpoint is used for getting the list of uploaded documents with or without the details of the documents. The details of the documents consist of a document identifier, a filename, a mime-type, an encoding, and an analyzed status. To upload a document, we can use POST /storage/document endpoint with an option to overwrite or to not overwrite the old uploaded file.

After a new document is uploaded, the document is analyzed by GET /nlp/analyzed/did endpoint and interpreted by POST /nlp/interpret/did endpoint. After these process finished, we can modify the document and the content with the rest of available endpoints. GET /storage/document/did is used for getting the content, info, or analysis of the documents. The content parameter provides the document to be downloaded; the info parameter shows the details of the documents; and the analysis parameter returns the analyzed responses about the annotations type, offsets, and properties of an entity in the document content. Finally, to update the analysis of a document, we can use POST /storage/document/did endpoint and to delete a document, it can be done via DELETE /storage/document/did.

POST	/storage/resource/edit	🔒
GET	/storage/resource/search	🔒
GET	/storage/resource/get	🔒
POST	/storage/document	🔒
GET	/storage/document	🔒
DELETE	/storage/document/{did}	🔒
GET	/storage/document/{did}	🔒
POST	/storage/document/{did}	🔒
POST	/storage/document/{did}/addannotation	🔒
GET	/storage/info	🔒

Figure 4.5: Storage endpoints

The storage module handles not only documents but also handles the triple resources composed by the subject, predicate, and object for the knowledge graph creation. The resources can be added, edited, or deleted by only a single endpoint, which is POST /storage/resource/edit endpoint. This endpoint specifically requires an old resource and a new resource for a change in the model. For creating a new resource, the parameter "before" as an old resource needs to be set to null in the request body. For removing a resource, the "after" parameter needs to be set to null in the request body, and for editing, both parameter "before" and "after" needs to be set to old resource and new resource with same resource id. The editing process in this endpoint is typically applied to update the compound/parent id and the metadata of the resource.

To support the implementation of provenances concept, GET /storage/resource/search and GET /storage/resource/get endpoints are provided in *autolinks*. GET /storage/resource/search is an endpoint to find resources based on a search query and return their full resources or only sources properties. The similar approach also can be performed via GET /storage/resource/get endpoint. With this endpoint, we can get the desired resource with a source key. This source key is important to find the related knowledge graphs and related sentence in the desired document so that the users know the sources of information from a single concept or node.

Annotating a new entity in the document text with an existing annotation type or a new annotation type can be performed via POST /storage/document/did/addannotation endpoint. This endpoint requires a document identifier, an annotation type, an offset, properties and an analyzer in the request body. With this endpoint, a user can have more annotated entities in the document text with more affluent annotation types.

Maintenance

Maintenance is the module to handle the availability of the broker and for error handling.

Whenever the error comes produced by the bug from both backend and visual information management, it can be handled by this module. This module is mainly to reset the *autolinks* database.

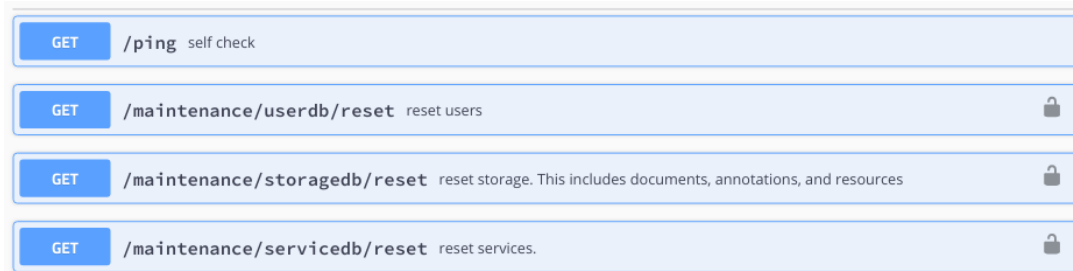


Figure 4.6: Maintenance endpoints

4.2 Triples

Semantic triples or simply triples is an ordered 3-tuple of <subject, predicate, object> in Resource Description Framework (RDF). The subject could be a URI or a blank node; the predicate is a URI; and the object could be a URI, a blank node or a literal. This blank node is an unnamed resource, where it is regularly applied for a structured value representation. In *autolinks*, we specifically only use a subset of RDF(triples) and we do not use a blank node, list or properties.

As it has already been mentioned in the previous paragraph, the representation of triples is in the Resource Description Framework (RDF). The RDF is a general-purpose language for information representation in the web. This data model consists of a statement set, wherein each of it contains a subject, a predicate, and an object.

Resource Description Framework

The Resource Description Framework (RDF) is a language for the representation of resources. This resource can be anything such as a web page, an entire website or anything that contains information on the web [Candan et al., 2001]. It enables the exchange, reuse, and encoding of structured metadata and allows the interoperability of metadata to support common conventions of semantics, syntax, and structure.

Variety of applications that implements RDF concepts involve Resource Discovery where RDF enables search engines to identify resources on the web efficiently. Cataloging the RDF enables users to describe the content and the relationships in particular web site or digital library; Intelligent Software Agents where RDF facilitates knowledge exchange and sharing and supports software agents to filter, find, and merge data intelligently [Candan et al., 2001].

RDF has many formats, and one of the popular formats is turtle format. Terse RDF Triple Language (Turtle) format is an RDF serialization syntax, which constructs a Notation 3 (N3) syntax subset, a shorthand non-XML serialization for RDF models[Domingue et al., 2011]. Since N3 is designed with human-readability in mind, it is far more readable and compact than XML RDF notation.


```

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix mo: <http://purl.org/ontology/mo/> .

<http://www.bbc.co.uk/music/artists/d87e52c5-bb8d-4da8-b941-9f4928627dc8#artist>
  rdf:type mo:MusicArtist, mo:MusicGroup ;
  foaf:name "ABBA" ;
  foaf:homepage <http://www.abbasite.com/> ;
  mo:image <http://www.bbc.co.uk/music/images/artists/542x305/d87e52c5-bb8d-4da8-b941-9f4928627dc8.jpg> ;
  mo:member <http://www.bbc.co.uk/music/artists/042c35d3-0756-4804-b2c2-be57a683efa2#artist>,
    <http://www.bbc.co.uk/music/artists/2f031686-3f01-4f33-a4fc-fb3944532efa#artist>,
    <http://www.bbc.co.uk/music/artists/aebbb417-0d18-4fec-a2e2-ce9663d1fa7e#artist>,
    <http://www.bbc.co.uk/music/artists/ffb77292-9712-4d03-94aa-bdb1d4771d38#artist> ;
  mo:musicbrainz <http://musicbrainz.org/artist/d87e52c5-bb8d-4da8-b941-9f4928627dc8.html> ;
  mo:wikipedia <http://en.wikipedia.org/wiki/ABBA> ;
  owl:sameAs <http://dbpedia.org/resource/ABBA> .

```

Figure 4.7: RDF example in Turtle format [Harth et al., 2011]

An example of RDF in turtle format is depicted in Figure 4.7, describing group music "ABBA". We can see that in turtle, one can use abbreviations after the @previx. This previx is used to abbreviate the URIs and is aimed to have a better readability of the graph pattern. We can see that the syntax is @previx abbr ':' <URI>, where the URI is in an angle brackets, and each triple is separated by a dot '.' and for the new property in the repeating subject, it can be separated by semicolon ';'.

From the example given, we can see that the subject is 'http://www.bbc.co.uk/music/artists/d87e52c5-bb8d-4da8-b941-9f4928627dc8#artist', which has some properties or predicates 'rdf:type', 'foaf:name', 'foaf:homepafe', 'mo:image', 'mo:member', 'mo:musicbrainz', 'mo:wikipedia', and 'owl:sameAs'. We can notice that each property is seperated by ';' and each of them belongs to the mentioned subject. The objects are 'ABBA', 'mo:MusicArtist', 'mo:MusicGrozp', 'http://www.abbasite.com/' and the rest of the URLs. To understand that this RDF is explaining about group music 'ABBA', we can take a look at the <subject, predicate, object> relation wherein 'http://www.bbc.co.uk/music/artists/d87e52c5-bb8d-4da8-b941-9f4928627dc8#artist' is a subject, 'foaf:name' is a predicate, and 'ABBA' is an object in which the 'ABBA' object is a partial content of the mentioned subject. The other properties such as 'rdf:type' -> 'mo:MusicArtist', and 'rdf:type' -> 'mo:MusicGroup' also contribute for the explanation.

Service Resources

As it is already mentioned in the chapter of Background and Related Works, *autolinks* implements a subset of RDF with an extension to facilitate compound graph implementation. This schema allows the resources of triples to accommodate other resources inside a parent resource. The parent resource is the one that acts as the compound node where graphs could be put inside this type of node. Resources in *autolinks* are as follows:

- Resource: list-of-resources | <subject, predicate, object> triple | descriptor
- Subject: Resource
- Predicate: Resource
- Object: Resource

One resource, which has a value of <subject, predicate, object> triples represents two nodes with the relations. Subject forms the source node, object forms the target node, and one full triples resource is principally a three resources, which have a string value or a descriptor, and a triple <subject, predicate, object> shapes its relation, and this triple is basically the edges, so the edges can be created once it knows where the source node is and where the target node is.

A resource again can be a list of resources, a string descriptor, or a triple resource. Once it has a value of resources list, this resource is called a parent resource, and it shapes the compound node. The recursive process of resources forms children nodes and their relations inside this compound node. A single string resource, which has a descriptor or string value is interpreted as a single node without any relation.

To give more explanation about how a resource with a list of resources value could facilitate the compound graph implementation, we present a simple example of 'BCR' resources provided by the demo service:

```

1  {
2  "rid": 11357,
3  "value": [
4    {
5      "rid": 11355,
6      "value": {
7        "subject": {
8          "rid": 11347,
9          "value": "B-cell receptor",
10         "cid": 11357,
11         "metadata": {},
12         "sources": [
13           "service::Demo/demo/0.0.2/post/?q=B%20cell"
14         ]
15       },
16       "predicate": {
17         "rid": 11349,
18         "value": "binds",
19         "cid": 11357,
20         "metadata": {},
21         "sources": [
22           "service::Demo/demo/0.0.2/post/?q=B%20cell"
23         ]
24       },
25       "object": {
26         "rid": 11340,
27         "value": "Antigen",
28         "cid": 11357,
29         "metadata": {},
30         "sources": [
31           "service::Demo/demo/0.0.2/post/?q=B%20cell"
32         ]
33       }
34     },
35     "cid": 11357,
36     "metadata": {},
37     "sources": [

```

```

38         "service::Demo/demo/0.0.2/post/?q=B%20cell"
39     ]
40 }
41 ],
42 "cid": 11359,
43 "metadata": {
44     "label": "BCR"
45 },
46 "sources": [
47     "service::Demo/demo/0.0.2/post/?q=B%20cell"
48 ]
49 }

```

Listing 4.3: BCR resources example

From the Listing 4.3, we present the structure of semantic triples in Javascript Object Notation (JSON) format. Each resource structure has four primary keys, which are a resource identifier (rid), a compound/parent identifier (cid), a value, and metadata, and also an additional key, which is sources. Everytime a user calls a service, unique rids are generated, and cids are generated according to rids of resource parents.

In the example, we could see that the parent resource identifier has a numeric value 11357. What makes this resource represent the compound node is in its value. It has a value key where the value is an array or a list of resources. Moreover, we can take a look at the bottom part of the Listing where we can see a cid, metadata, and sources. It has a cid of particular value since this example is only a subset of more extensive resources. Moreover, the structure of the whole resources if we show all of them could be even more profound from the ancestor resources to the descendant resources.

This resource forms a compound node called 'BCR' since it has a label value 'BCR' in the metadata. The source key provides the provenances (traceable sources) where we can identify the relation of this resource to any related service or document that has been created or edited by the users. Inside of 'BCR' resource value, we have one child resource since it only contains one array value. In this resource value, we have a subject, predicate, and object triple format that forms the source node 'B-cell receptor', the target node 'Antigen', and the relation 'binds' between those nodes. From the example given, we can see that the resource principally describes 'B-cell receptor binds Antigen'.

4.3 Semantic Services

Recently, service-oriented architecture (SOA) has progressively been adopted as the most approaches for developing advanced and complex distributed systems by combining reusable parts and likely distributed components known as services. This software engineering approach demands procedures to be automated and semi-automated to accomplish. Due to the requirements for services location and searching, also choosing the appropriate ones and assembling them into complex processes in the SOA, semantics is used as a key to obtain a more advanced level of service-orientation automation in order to enable the benefits of data search, integration, and analysis [Pedrinaci et al., 2011]. Those aspects consequently led to the standardization of Semantic Web Services' research and development. The standardization

involves ontologies and algorithms' developments to discover, select, compose, and execute services automatically or semiautomatically.

Semantic web service or semantic service have been seen as tools to provide an infrastructure for semantic web agents. It is a software system powered by semantics that delivers a piece of business functionality through standardized protocols and computer networks [Domingue et al., 2011]. *autolinks*' semantic services identify entities and their relations from available service sources, such as Wikimedia indices, then extract and annotate those entities with their relations in RDF format. The service also produces semantic metadata for each resource by using Natural Language Processing components. The annotation resources or nodes given by semantic services provide provenances (traceable sources), the ability to connect related documents or related knowledge graphs.

Semantic services, which are available in *autolinks* at the moment are wiki service and demo service. Wiki service gives access to Wikimedia indices: such as Wikidata, Wiktionary, and Wikipedia. Demo service is only for demonstration and testing purposes.

4.4 Information Extraction

One of the backend server responsibilities is to do an information extraction. We extract the information from the documents uploaded by the users and provide the analysis and interpretation. The analysis process extracts the text from the uploaded documents and returns the whole analysis including annotation types, properties such as canonical form, capitalization, num position, part of speech tagging, and token number, discontinuous offset where it has multiple offsets giving the start and the length of the offset, and the analyzer.

The interpretation process interprets each offset inside the uploaded document and returns the annotation resources based on the selected offset. These annotation resources will be inserted into a special container called annotation container. This particular container is a non-editable container and relations cannot be performed towards this type of container. Each annotation resource affords unique properties such as resource identifier, value, metadata, and sources. The value is a descriptor composed by the analyzer, annotation type, document id, start and end offsets in a string.

In the field of semantic annotation, Information Extraction (IE) becomes one of the most frequent methods to use. IE is a natural language analysis form to explore the structured information from unstructured text and formal knowledge expressed in ontologies. The main tasks performed during the process of information extraction are [Bontcheva and Cunningham, 2011]:

- Named Entity Recognition (NER), to identify and classify different types of entities such as people, places, or organizations from natural language text.
- Coreference resolution, the process to resolve referents and to have a decision in measuring the similarity between two language expressions to the same entity.
- Relation extraction, to identify relations between entities from natural language text.

Besides those tasks other Natural Language Processing (NLP) components such as Part-Of-Speech (POS) taggers, semantic interpretation, and morphological analyzer are also involved in information extraction.

Medical Entity Extraction

Since the beginning of the development of *autolinks*, *autolinks* focuses on the biomedical domain and that is why Medical Entity Recognition (MER) is implemented. MER is responsible for identifying and classifying entities type in the biomedical domain such as Anatomical Site, Disease Disorder or Medication.

For the MER implementation, Apache cTAKES is chosen to be the tool. Apache cTAKES (clinical Text Analysis and Knowledge Extraction System) is a natural language processing system to extract the information from electronic medical record clinical free-text. This tool is built on the existing open-source technologies such as Apache UIMA (Unstructured Information Management Architecture) framework and OpenNLP natural language processing toolkit [Savova et al., 2010].

Apache cTAKES is composed by a pipelined components in a modular system by combining rule-based and machine learning techniques with a purpose to enable information extraction in the clinical narrative. Some components or annotators that are available in the Apache cTAKES are Sentence boundary detector, Tokenizer, Normalizer, Part-of-speech (POS) tagger, Shallow parser, and Named entity recognition (NER) annotator, including status and negation annotators [Savova et al., 2010].

We present an example of a sentence extracted and analyzed by cTAKES in Figure 4.8 with a sentence text 'family history of obesity but no family history of coronary artery diseases.' where Fx is family history.

An example of a sentence discovered by the sentence boundary detector:

Fx of obesity but no fx of coronary artery diseases.

Tokenizer output – 11 tokens found:

Fx of obesity but no fx of coronary artery diseases .

Normalizer output:

Fx of obesity but no fx of coronary artery disease .

Part-of-speech tagger output:

Fx of obesity but no fx of coronary artery diseases .
NN IN NN CC DT NN IN JJ NN NNS .

Shallow parser output:

Fx of obesity but no fx of coronary artery diseases .
NP PP (NP) (NP) PP (NP) NP

Named Entity Recognition – 5 Named Entities found:

Fx of obesity but no fx of coronary artery diseases .
obesity (type=diseases/disorders, UMLS CUI=C0028754, SNOMED-CT codes=308124008 and 5476005)
coronary artery diseases (type=diseases/disorders, CUI=C0010054, SNOMED-CT=8957000)
coronary artery (type=anatomy, CUI(s) and SNOMED-CT codes assigned)
artery (type=anatomy, CUI(s) and SNOMED-CT codes assigned)
diseases (type=diseases/disorders, CUI = C0010054)

Status and Negation attributes assigned to Named Entities:

Fx of obesity but no fx of coronary artery diseases .
obesity (status = family_history_of; negation = not_negated)
coronary artery diseases (status = family_history_of, negation = is_negated)

Figure 4.8: A sentence example extracted from cTAKES- components, Figure taken from [Savova et al., 2010]

From the example above, we could comprehend about the role of each component in apache cTAKES. Starting from the cTAKES sentence boundary detector, it predicts whether

a question mark, a period or exclamation mark in the sentence is in fact at the end of that sentence.

Tokenizer component of cTAKES has two subcomponents. The first subcomponent is basically responsible for splitting the internal sentence text between the punctuation and space, even though the process is actually more complicated. The second subcomponent is responsible for merging tokens for the creation of data, fraction, measurement, person, title, and time tokens with the implementation of finite state machines for each of the types [Savova et al., 2010].

The component of the SPECIALIST Lexical Tools¹⁹, which we call it "norm" is wrapped around by the cTAKES normalizer. The normalization gives a representation to words that have been normalized with respect to a lexical properties numbers, which include alphabetic case, inflection, punctuation, spelling variants, genitive markers, diacritics, stop words, symbols, and ligatures²⁰. The normalizer normalizes each word to the base form. Yet in *autolinks*, we do not use the normalizer component.

The cTAKES Part-of-speech tagger annotator processes the text and assigns each word its part of speech types such as noun, verb, adjective, preposition, adverb, and others. This annotator together with the cTAKES shallow parser is the wrappers around OpenNLP's modules. The shallow parser annotates the text based on their types of phrase such as Noun Phrase (NP), Verb Phrase (VP), Prepositional Phrase (PP), Adjective Phrase (AP), or Adverb Phrase (AdvP).

One of the essential components in cTAKES is cTAKES NER component. This component uses a terminology agnostic dictionary look-up algorithm inside a noun-phrase look-up window. Every named entity is connected to a particular concept through from the terminology the dictionary look-up, where the dictionary used is a Unified Medical Language System (UMLS) subset [Philip Ogren and Chute, 2008]. This dictionary includes Systematized Nomenclature of Medicine Clinical Terms (SNOMED CT) concept²¹ where it is used for the electronic conversion of clinical health information and also as a standard in interoperability specifications for the U.S. Healthcare Information Technology Standards Panel, and RxNorm concept²², which presents normalized names for clinical drugs and connects its names to many drug vocabularies commonly used in drug interaction software and pharmacy management. These concepts are guided by the consultations from practitioners and clinical researchers. Figure 4.8 shows NER annotates obesity with disease/disorders type, UMLS CUI, and its SNOMED-CT codes, and it annotates corona artery with anatomy type.

The cTAKES negation annotator performs the NegEx algorithm. This algorithm is a pattern-based approach to find words and phrases denoting negation near named entity mentions. Similar to the cTAKES negation annotator, the cTAKES status annotator implements a comparable method to find connected words and phrases that show the status of a named entity.

Each component performed in cTAKES has various performances. From [Savova et al., 2010], it shows that the cTAKES sentence boundary detector and cTAKES tokenizer have an accuracy

19. <http://www.SPECIALIST.nlm.nih.gov/>

20. <http://www.lexsrv3.nlm.nih.gov/SPECIALIST/Projects/lvg/current/docs/userDoc/index.html>

21. <https://www.nlm.nih.gov/healthit/snomedct/>

22. <http://www.nlm.nih.gov/research/umls/rxnorm/>

0.949; the cTAKES part-of-speech tagger has an accuracy 0.936; the cTAKES shallow parser has an F-score 0.924; the cTAKES named entity recognizer and system-level evaluation have an F-score 0.715 for exact spans and 0.824 for an overlapping spans; the cTAKES concept mapping has an accuracy 0.957 for exact and 0.580 for overlapping spans; the cTAKES negation annotator has an accuracy 0.943 for exact spans and 0.939 for overlapping spans, and the cTAKES status attributes have an accuracy for exact 0.859, and for overlapping spans 0.839.

5 Information Management Visualization

This chapter explains concepts of visualization in information management that allows users to access information and gain new knowledge. The information management in *autolinks* mainly focuses on how to visualize the data given by semantic services and deliver the knowledge to the users in knowledge graphs.

5.1 Knowledge Graphs

The foundation of visualizing knowledge graphs in *autolinks* is based on a network concept called a *node-link diagram* [Battista et al., 1998]. This visual representation is a common representation for the knowledge graphs to draw vertices as circles or similar shapes and edges as lines. Nodes or vertices have additional information such as entity name, organization, location, or any other information properties. Meanwhile, links or edges have relational attributes among nodes and can contain an arrow for the direction in the directed graph model, or vice versa in the undirected graph model.

Compared to other alternatives for networks visual representation such as matrix-based representations, [Ghoniem et al., 2005] said that node-link representation tends to be more intuitive, readable, and less abstract, so that this representation is suitable for our knowledge graph visualization.

Entities as Nodes

Nodes represent entities. In the ontology concept, an entity is something that exists [Gruber, 1993], and it could be as a subject or as an object. In Figure 5.1, we can see that we have two entities, which are 'B-cell Receptor' and 'Antigen'.

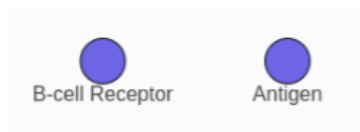


Figure 5.1: Entities as nodes

The shape of circles for entities is, in fact, an ellipse shape with the same width and height values so that it draws a circle shape. We chose the circle shape since this shape is a convention in the node-link diagram. Other shapes that are provided by *cytoscape.js* are a triangle, rectangle, round rectangle, barrel, rhomboid, diamond, pentagon, hexagon, octagon, star, and others.

In *autolinks*, we include additional information such as a label as a name of the entity or other attributes such as metadata and traceable sources in the nodes.

Relations as Edges

In the node-link diagram, the relations between nodes are represented as edges/links. They are drawn with arrows shape with a direction connecting two nodes to specify which one is the source nodes or target nodes. It indicates that relations between these entities exist.

In Figure 5.2, we can see that the edge has a label text in the middle with text "binds" to determine the semantics of edges. It implies that the 'B-cell receptor' has a binding relationship to 'Antigen'.

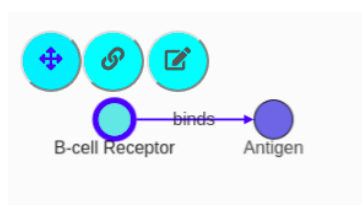


Figure 5.2: Relations as edges

The edges in *autolinks* are drawn as a straight line, but the edge curve type that *autolinks* uses is bezier edges. This edge type enables *autolinks* to have a straight edge for a single relation between two entities and to have separated perfect curve edges for two or more connections between entities. In the cytoscape.js library, there are other supported edges types such as unbundled-bezier edges, segments edges, or haystack edges.

The unbundled-bezier edges type is another type of bezier edges but with manual control points so that the curve point can be set manually with value 0.25, 0.5, 0.75. Moreover, even smaller or larger amounts can also be arranged. The segment edges type is a series of straight lines, and haystack edges type is a type of edges that draws bundled consecutive edges.

Visual states of nodes and edges

Nodes and edges in *autolinks* are designed to have four visual states. These four visual states, which are 'Normal', 'Focus', 'Highlighted', and 'Same Entities highlighted' can be seen in Table 5.1.

- "Normal" state is a typical representation for nodes and edges.
When there is no interaction by the user towards nodes or edges, they are displayed without any specific effect of highlighting. Nodes are colored blue, and edges are colored light grey.
- "Focus" state is the visual representation for a node or an edge, which becomes a focus by a user.
This condition happens when a user clicks a particular node. Then the node will turn to light blue with border color blue. Once the selected node is in the focus state, the connected edges of the node will also turn to blue color. However, if the user clicks an edge, only that edge that shifts to blue color and the connected nodes are not affected.






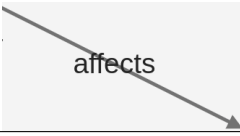

Visual States	Node	Edge
Normal	 B cell	
Focus	 B cell	
Highlighted	 B cell	
Same Entities Highlighted	 B cell	

Table 5.1: Visual states of nodes and edges

- “Highlighted” state is triggered when the user hovers over a node or an edge. This visual state is needed to visually highlight a part of knowledge graphs that are currently investigated by a user. Similar to the focus state, the highlight state on nodes also drives the highlight state on edges, but not the other way around.
- "Same entities highlighted" is a visual state that is activated when two or more identical entities in the knowledge graphs exist.
This visual state helps a user to discover same entities/nodes, which are placed in different hierarchies or knowledge graphs generated by different semantic services. This visual state is really beneficial for users to explore nodes, which have the same labels but in the different hierarchies.

5.1.1 Building a Knowledge Graph

As a bottom-up approach information management, ontology in *autolink* is defined starting at the most specific concept. It can be done by building knowledge graphs manually or by developing knowledge graphs that are generated by semantic services.

At the moment *autolinks* cannot retrieve the data of knowledge graphs, which are created outside a service compound node. It stores the update knowledge graphs based on resource container generated by semantic services, which later on can be retrieved from query exploration. Nodes, which are created outside a service compound node, are stored but not yet retrievable. As a bottom-up information management tool, users are prioritized to build their own ontologies with knowledge graphs provided from the source of information or start from annotations extracted from documents. However, users are still able to start building the knowledge graphs outside a service compound node, and later on, merge them to a particular service compound node in order to make them retrievable.

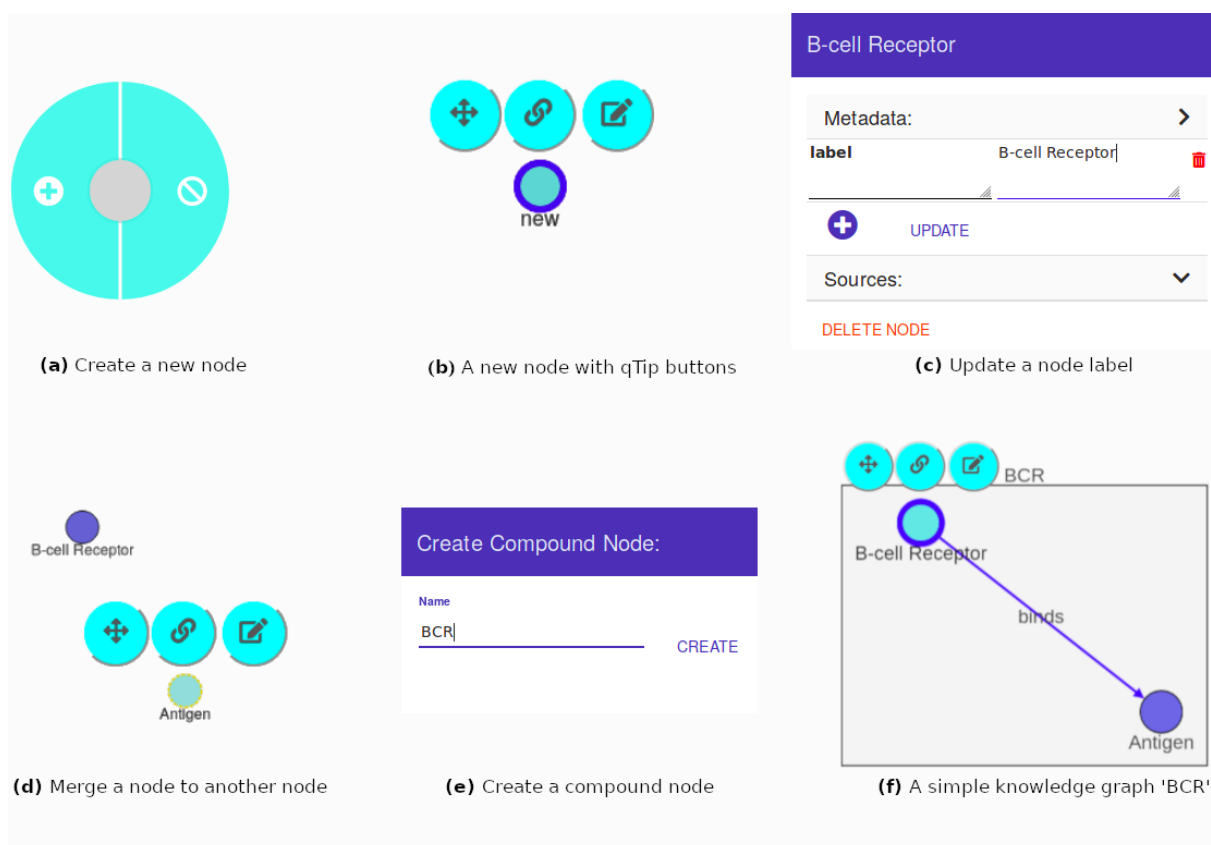


Figure 5.3: An example of knowledge graph 'BCR'

To build a simple knowledge graph, we can start by creating a single node in *autolinks* with procedures that can be seen in Figure 5.3. After login, users can begin to have interactions towards *autolinks* environment. A user can click somewhere in the graph canvas and hold the cursor for a second. It will display a green circular context menu, which is used to create a new node in the graph canvas by selecting an add icon (see Figure 5.3 (a)). Once an add icon is selected, a new node is created immediately in the position of the cursor.

Each node that is created and generated by semantic services is equipped with qTip buttons. A qTip is a library of the tooltip for Cytoscape framework that can contain a collection of buttons for a single node. These qTip buttons are shown when a node is in a focus visual state. A user clicks a node then buttons in qTip are shown and can be selected. For a new node, we can update the node label by clicking an edit icon button. This button will toggle the side navigation to appear on the right side of the screen. The side navigation here is designed to display the node details including metadata and sources.

In metadata, we can change a label value for a node to the desired value. In example 5.3 (c), we can see that a label value 'new' is replaced by 'B-cell Receptor'. Once a user clicks the update button, the node label will immediately be changed, and the updated value is stored in the database.

To create another node, we can again follow the procedure from Figure 5.3 (a) to Figure 5.3 (c). Once we have at least two entities, we can start to build a more general concept of entity. It can be done by selecting the arrows icon of one node, then select another node as a target. After clicking the target node, this procedure will display a modal asking for merging

two nodes. In the example of Figure 5.3 (d), 'Antigen' is merged with 'B-cell receptor'. This will eventually create a compound node that contains both, 'Antigen' and 'B-cell receptor'.

Side navigation for creating a compound node will appear from the right side of the screen after the user confirms to merge. We need to specify the label of the more general concept of the knowledge graph by filling the input value with the desired value and clicking a create button. In the example of Figure 5.3 (e), we choose 'BCR' as a compound node since B-cell Receptor is a BCR itself and Antigen is a part of 'BCR' that refers to an antigen receptor of the adaptive immune system.

After the confirmation of creating a compound node, the merged nodes will be surrounded by an additional layer of abstraction. This layer is a node with a rectangle shape, which is called a compound node. This particular node has a rectangle shape and contains a number of child nodes. The dimensions of a compound node are determined by the bounding box of the children nodes. The expanded version of a compound node will show the children nodes around the bounding box, and the collapsed version of the compound node will only show a small single node with a rectangle shape.

After a merging scenario, we can start to draw a relation between the nodes. Figure 5.3 (f) shows the edge connecting 'B-cell receptor' and 'Antigen' inside the 'BCR' compound node. This relation can be drawn by selecting the chain icon in the qTip buttons and draw the edge towards another node as a target where in this case 'Antigen' node is a target node. There are some scenarios to draw relations and to create compound nodes in *autolinks*. These scenarios are explained more details in the next subsection.

5.1.2 Merging Nodes Scenarios

In this subsection, we will explain about merging nodes scenarios. As it is already explained in Chapter 3, in *autolinks*, there are limitations of drawing relations towards nodes that have a different level of hierarchies, so that these scenarios become important since merging nodes would lead to different results especially when involving edges in the process of merging. The scenarios are mentioned as follows:

- Similar to the procedure of merging in the previous subsection, merging a single node without any relation to another single node, which also has no relations will raise side navigation to create a new compound node.
- Merging a compound node without any relation as a source to a single node as a target will also create a new compound node so that a more general compound node will be created and contain the source compound node and a single target node.
- Merging a single node / a single compound node without edges as the source node to another compound node as a target means that the source node, either a single node or a compound node, will move the source node inside that target compound node. As the target compound node is already a general concept and contains children nodes such as regular or compound nodes, we just need to change the compound identifier of the source node with the resource identifier of the target compound node.

- Merging a single node with relation to a single node with relation will create duplicates of those nodes inside a compound node. If either a source node or a target node has any relation, it will be duplicated. This scenario happens because if it is not duplicated, these nodes will be merged inside a new compound and keep the relations connecting nodes that have different hierarchies, which are not possible in *autolinks*.

5.1.3 Creating Relations Scenarios

Scenarios of edge creation are discussed in this subsection. *autolinks* has a standard rule for drawing relations between entities: drawing relation is only possible in one level hierarchy. To make it more understandable, we can take an example from the previous subsection 'Building a knowledge graph' in Figure 5.3. B-cell receptor node can draw relations to Antigen node and vice versa. Both of B-cell receptor and Antigen node can not draw relations towards BCR compound node since BCR compound node is in a more general level of a hierarchy.

When we create another node 'V(D)J recombination' as an example outside BCR compound, this node is able to draw relations to BCR compound node since they are in the same hierarchy, at the root level of the graph. However, this new node can not 'directly' have relations towards B-cell receptor node and Antigen node since these nodes are the children nodes of BCR compound node and in the lower level of BCR and V(D)J recombination hierarchy. Instead, it will lead to a different scenario that will be explained as follows:

- In the same level of a hierarchy, a single node / a compound node can draw relations to another single node or a compound node.
- Drawing relations, from a node directly to its parent node or its ancestor nodes, is not allowed.
- Drawing relations to a node/compound that is in a different level of hierarchies and ancestors from a source node will create a duplication of the target node. This duplicate will be placed in the same hierarchy of the source node so that the source node will still have a relation towards a target node but in the same hierarchy level.

Figure 5.4 shows an example of drawing a relation from a source node to a target compound node, which has a different hierarchy than the source node. An edge is drawn from the 'B-cell Receptor' node to the 'B Cell - Antigen' compound node, which is inside 'BCR' compound parent node. This relation will produce a duplication of the 'B Cell - Antigen' compound node and it places the duplicate to the same hierarchy of 'B-cell Receptor'. As it is already explained that it is not possible at the moment to create a direct relationship between two different hierarchy nodes since triples <subject, predicate, object> cannot be stored in the different resources container. Triples needs to have the same compound identifiers where it is only possible to be inside a single resource container with a single resource identifier.

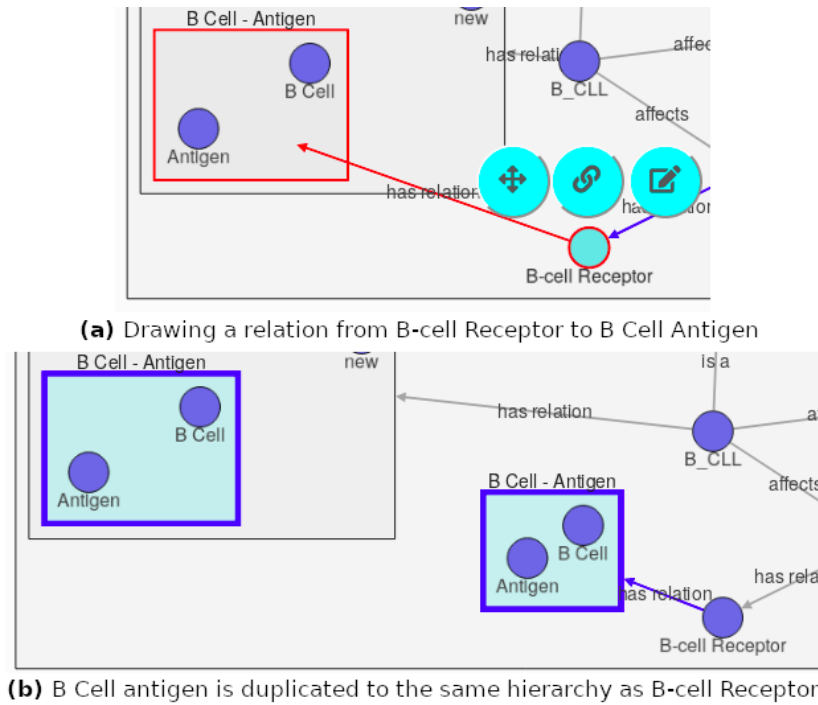


Figure 5.4: Drawing relations to a node in different hierarchy

5.2 An Algorithm for Knowledge Graph Extraction

We discuss in this subsection about an algorithm that extracts the service resources given by semantic services to be visualized as knowledge graphs in the visual information management. Using a search bar, a user can do query exploration. From the query given by the user, the visual information management will ask a request to get resources. The server responds with service resources that contain triples for knowledge graph extraction.

In the previous chapter, we have already seen an example of BCR resources in Listing 4.3. The structure of the resource starts from `rid`. This identifier becomes the outermost service compound that will contain children nodes either normal nodes or compound nodes. The process of extraction can be seen in Algorithm 3.

The resources given by the semantic services are in JSON format so that the following algorithm is using JSON convention to access a value from the data object. First, we declare two global variables for the nodes and the edges as empty arrays. These empty arrays are used by Cytoscape library for adding a collection of nodes and edges. Then it will check whether the resource given is empty or not.

Algorithm 1: CreateKnowledgeGraphs($r := Resources$)

```
1:  $newNode := []$ 
2:  $newEdge := []$ 
3: if  $r \neq null$  then
4:   if  $r.value = Array$  then
5:     for  $r.value$  do
6:        $extractResources(r)$  // extracting list of resources, triples, or descriptor
7:     end for
8:      $outermostEntity := assignEntity(r, parent = 'outermost')$  // returning an entity
       from resources
9:   else if  $r.value = Object$  then
10:     $extractResources(r)$ 
11:     $outermostEntity := assignEntity(r, parent = 'outermost')$ 
12:   else
13:     $outermostEntity := assignEntity(r, parent = 'outermost')$ 
14:   end if
15:   if  $outermostEntity$  then
16:     $newNode := outermostEntity$ 
17:   end if
18:    $filterNode := filterNodeById(newNode)$  // filtering duplicate nodes in a resource
19:    $addNode(filterNode)$  // adding nodes to graph canvas
20:    $addEdge(newEdge)$  // adding edges to graph canvas
21: end if
```

Once the service resource has object data such as *rid*, *value*, *metadata*, this resource data will automatically be an outermost service compound node based on the resource identifier. Then, it checks the resource value whether it is an array, an object, or a descriptor. As long as a value of resources is an array, it will be extracted in a recursive function namely *extractResources()* for each value in the array. Once the value is a resource triples (*subject*, *predicate*, *object*) and each triple has a descriptor, it will be extracted directly to be a source node, an edge, and a target node. If it is a descriptor in the outermost compound, it will be extracted as a single node. It is a rare case but it still possibly occurs.

In the *extractResources()*, new nodes and new edges will be assigned after the triple extraction by *extractTriples()* function. After the new nodes and edges are assigned, the nodes will be filtered via *filterNodeById()* by filtering the duplicate resource identifiers inside one compound resource. There is a possibility that same nodes or same resource identifier could appear inside one compound or in the same level of a hierarchy. These nodes are basically one node and no need to be visualized more than one node. Finally, the filtered nodes and the new edges are visualized in the graph canvas by the *addNode()* and *addEdge()* functions provided by Cytoscape library.

The following algorithm is a function that recursively extracts resources that have a deeper value of array. A resource that has an array value will be assigned as a compound, and this compound will be passed as a parent in the next recursive extraction of each compound values. A resource that has triples object in its value will be extracted via *extractTriples()* function. This function will categorize a subject as a source node, an object as a target node, and a

predicate together with the source node and target node as a relation. A resource that has a value as descriptor will be assigned directly as a single node.

Algorithm 2: $\text{extractResources}(t := \text{Triples}, p := \text{Parent})$

```

1: if  $t.value = \text{Array}$  then
2:   for  $t.value$  do
3:      $compound := \text{assignEntity}(t, p)$ 
4:      $newNode := compound$ 
5:      $\text{extractResources}(compound.value, compound)$ 
6:   end for
7: else if  $t.value = \text{Object}$  then
8:   return  $\text{extractTriples}(t, p)$ 
9: else
10:  return  $newNode := \text{assignEntity}(t)$ 
11: end if

```

Triples are extracted via $\text{extractTriples}()$ function. First, this function accesses the value of the subject, object, and predicate from the objects of JSON. Then it checks whether the value of either subject, predicate, or object has another array value. If the value is a descriptor, this resource will be assigned as a child node via $\text{assignEntity}()$. Otherwise, it will be assigned again as a compound node that will contain other children nodes. The predicate is assigned to new edge via $\text{assignRelation}()$ function, which involves the value of subject, predicate, and object as parameters.

Once the subject, object, and predicate are assigned as entities and relations. These variables are ready to be attributed to the global variables, which are $newNode$ and $newEdge$, which later on these global variables will be executed as nodes and edges in the visual information management. After assigning triples to global variables, the triples will again face the value checking. Different to the array checking at the beginning of algorithm where the process of array checking here is only for assigning the subject or object to be a compound or a single node. This array checking will lead the triples to recursive extraction if the triples are still in the compound form to extract the children nodes of the triples compounds.

The $\text{assignEntity}()$ function or the $\text{assignRelation}()$ function is basically a simple assignment process for the properties of the resources and it returns a variable of object as a result, which is ready to be shown in the graph canvas via $\text{addNode}()$ or $\text{addEdge}()$ function. In the $\text{assignEntity}()$, these properties are the id of the nodes, resource id, the name for the node label, metadata, path, and provenances. Before assigning the properties, it will first check whether the resource is an outermost resource or not. Outermost resource/compound will not have a parent property and compound identifier property since it is in the root hierarchy, but a compound or a node definitely has a parent property. Similar to $\text{assignEntity}()$, the properties that are in the $\text{assignRelation}()$ are same as the properties in $\text{assignEntity}()$ with additional properties such as source and target properties. The source property for the edges takes the source node identifier, and the target property takes the target node identifier to draw a relation.

Algorithm 3: $\text{extractTriples}(r := \text{Resources}, p := \text{Parent})$

```
1:  $s := r.value.subject$ 
2:  $o := r.value.object$ 
3:  $p := r.value.predicate$ 
4: if  $s.value = \text{Array}$  then
5:    $subject := \text{assignEntity}(s, p)$ 
6: else
7:    $subject := \text{assignEntity}(s, p, child = true)$ 
8: end if
9: if  $o.value = \text{Array}$  then
10:   $object := \text{assignEntity}(o, p)$ 
11: else
12:   $object := \text{assignEntity}(o, p, child = true)$ 
13: end if
14:  $predicate := \text{assignRelation}(p, s, o)$ 
15:  $newEdge := predicate$ 
16:  $newNode := subject, object$ 
17: if  $s.value = \text{Array}$  then
18:   for  $s.value$  do
19:      $\text{return } \text{extractResources}(s)$ 
20:   end for
21: end if
22: if  $o.value = \text{Array}$  then
23:   for  $o.value$  do
24:      $\text{return } \text{extractResources}(o)$ 
25:   end for
26: end if
```

5.3 Compound Graphs Motivation

A typical ontological app as it is already discussed in Chapter 2 is usually a flat graph connected with common nodes and edges. We argue that it is not enough actually and we need some kind of better representation, a notion of generic hierarchies. We can actually have graphs recursively within nodes, and this concept is called compound graphs.

The following figures illustrate why flat graphs representation is not sufficient for representing concepts which have complex hierarchies. Graphs in the left of Figure 5.5 are flat. To represent a 'rabbit' or a 'turtle' is an 'animal' or to represent a 'race car' or a 'bus' is a 'vehicle', it requires three nodes and two edges. In compound graph implementation, these edges can be removed and as a replacement, two nodes will have a bounding box with a label 'vehicle' and 'animal'. Just imagine when we have more complex hierarchies such as the more general hierarchies and more specific hierarchies in the graphs.

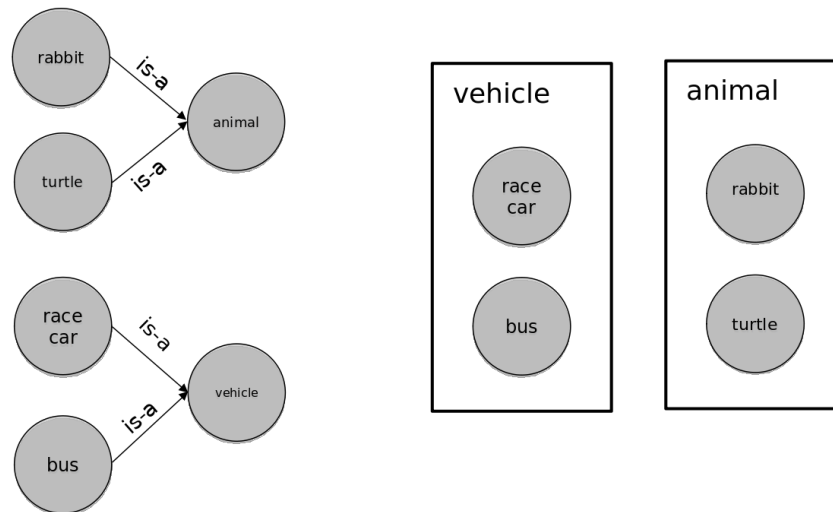


Figure 5.5: Hierarchies from flat graph

Figure 5.6 shows the more complex hierarchies in the compound graphs. A 'mammal' as an example is the more general concept than a 'rabbit', and a 'vertebrate' is the more general concept than a 'turtle'. In the compound graph implementation, we can quickly notice that the 'rabbit' is also a 'mammal', a 'vertebrate', and definitely also an 'animal'. In contrast, when we take a look at the flat graphs, we need to investigate first whether a 'rabbit' node has a relation to an 'animal' node or not. This flat graphs representation decelerates our cognitive process to understand the concept.

The more complex hierarchy in the concept, the more difficult is the flat graphs representation to comprehend. We can discuss another more complex example by adding an 'organism' entity as a general concept of an 'animal' and 'Belgian hare' and 'wild rabbit' as the specific concepts of rabbit. Different from compound graphs representation where we can switch the 'rabbit' node to a compound node to accommodate the 'Belgian hare' node and 'Wild rabbit' node and we can also add an additional layer of bounding box 'organism' compound to the 'animal' graphs for the more general concept, In the flat graphs representation, it highly depends on the edges as relations. We need to draw a relation from the 'animal' node to the 'organism' node and to draw relations from the 'Belgian hare' node and the 'wild rabbit' node to the 'rabbit' node.

With this representation in the flat graphs, it is even harder for users to notice that the 'rabbit' and 'turtle' are also a part of an 'organism' and whether the 'Belgian hare' and 'wild rabbit' have a relation to 'animal' and 'organism' or not. In the flat graphs, there is no guarantee that between nodes, which are separated by other nodes have a direct connection. To confirm between these nodes have any meaningful relation, the user needs to do further observation towards the flat graphs and it makes the cognitive process of the user even longer.

An option to make these kind of nodes to have a more obvious connection is that we can draw more edges to help users to notice that there are relations among these nodes. We can draw a connection directly from the 'Belgian hare' node to the 'animal' node or even 'organism' node or we can draw a relation also from the 'turtle' node to the 'vertebrate' node. But again this option is not a good solution when the concept has a deeper level of a hierarchy, and it only makes the graph noisier and messier to be visualized. From this example, we can

conclude that the representation of compound graphs improve the cognitive process of the user so that the user can quickly understand about the concepts compared to the representation of traditional graphs.

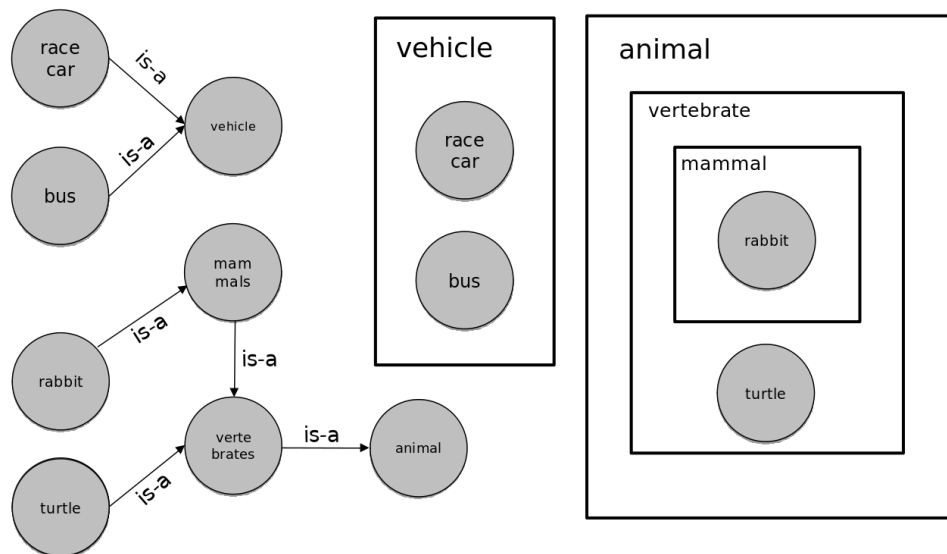


Figure 5.6: More complex hierarchies for compound graphs

Different from flat graphs, compound graphs tackle the problems from a different perspective that happened in the flat graphs and facilitate the concepts with a more presentable look. We can have another layer abstraction or bounding box representing organism around the animal compound. The rabbit node can be switched to a compound that contains Belgian hare and wild rabbit nodes. With this implementation, it is noticeable for users to see and to understand the concepts faster without complex and crowded graph drawing.

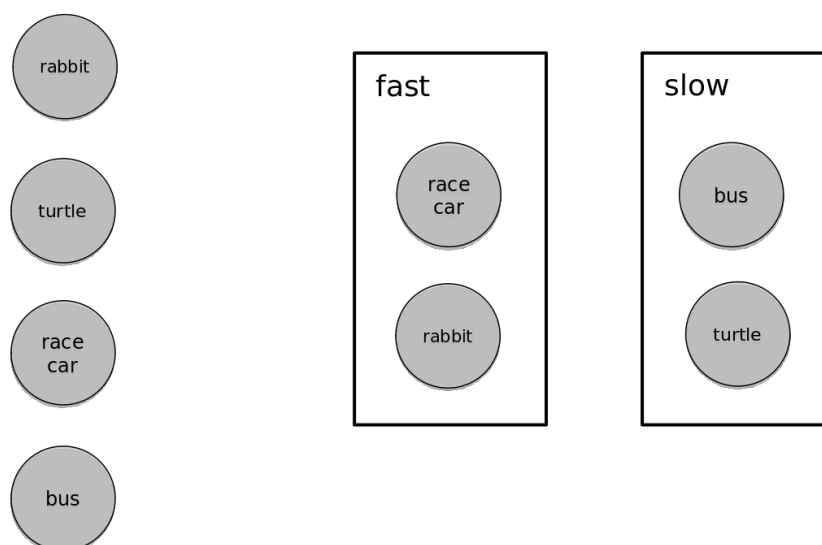


Figure 5.7: Hierarchies based on properties

Another advantage of compound graph implementation is we can specify the hierarchy level based on subjective preferences. Figure 5.7 shows that when there is no animal entity

or vehicle entity for more general hierarchy, flat graphs cannot categorize the rabbit and turtle with their properties since the node representation should be an entity for the sake of consistency. In compound graphs, we can classify the nodes inside a bounding box with a label based on their identical property.

In *autolinks*, compound graphs implementation has a feature to expand and collapse the compound graph itself and this feature will not be possible to be implemented in the normal graphs. This feature aims to help users to have a better management of complexity in the knowledge graphs. When there are too many knowledge graphs in the graph view, we can collapse some compound graphs to get some extra space and to improve the readability of the knowledge graphs. If we want to revert back the compound graphs to the regular form to see the details, we can expand them by clicking a plus button in the upper left of the compound node. With these benefits of compound graphs, we aim to extract and visualize the data by using compound graphs for a better representation of the concepts and to improve the cognitive process of the user.

5.4 A Layout Algorithm for Compound Graphs

To manage knowledge graphs with compound nodes, we need to make sure to use a layout that is suitable both for this kind of graph model and for Cytoscape.js network visualization library [Franz et al., 2016]. Even though it is computationally more expensive, CoSE (Compound Spring Embedder) - Bilkent layout [Dogrusoz et al., 2009] was chosen since this layout produces near-perfect end results for the compound graphs. This algorithm supports compound graphs handling with non-uniform sizes of the node, inter-graph edges, and clustered and compound graph structures [Dogrusoz et al., 2009].

The algorithm behind the CoSE Bilkent layout has three significant phases with an initialization phase:

1. **Initialization:** This initialization phase is a phase to calculate the size of initial nodes and the values of threshold in order to determine convergence (based on nodes number) and to perform randomly the nodes initial positioning. As reasons for the quality of layout and also for the efficiency, some parts of the graphs, which are trees are removed temporarily. It means that leaf nodes of root graphs are removed iteratively until there is no such node is left. The remaining part of the graph then constructs the "skeleton" of the graph.
2. **Phase 1:** This first phase is where the skeleton graph is constructed using the spring embedder model, but at this phase the gravitational forces and application constraints are disabled.
3. **Phase 2:** The leaf nodes where are removed earlier in the initialization phase are introduced back in this phase as level by level. This phase also enables the gravitational forces and application constraint.
4. **Phase 3:** The last phase is the stabilization phase, where the layout is polished.

The simplified algorithm is presented as follows:

Algorithm 4: CompoundGraphsLayout($GraphMgrM = (S, I, F)$)

```

1: Initialization()
2: set phase to 1
3: if layout type equals incremental then
4:   set increment phase to 3
5: end if
6: while phase  $\leq 3$  do
7:    $state := maxIterCount[phase]$ 
8:   while (step > 0 or !allTressGrown) do
9:     calculateSpringForces(M)
10:    calculateRepulsionForces(M)
11:    if phase  $\neq 1$  then
12:      calculateGravitationForces(M)
13:      calculateRepulsionForces(M)
14:    end if
15:    calculateNodePositionsandSizes(M)
16:    if phase = 2 and !allTreesGrown and step%GrowingStep = 0 then
17:      growTressOneLevel(M)
18:    end if
19:    step + = 1
20:  end while
21:  phase - = 1
22: end while

```

Algorithm 4 is used by [Dogrusoz et al., 2009] to introduce a force directed layout algorithm for undirected compound graphs, which can handle multilevel nesting with edges among varied nesting levels and node sizes. The implementation of the algorithm is based on the algorithm by [Fruchterman and Reingold, 1991]. The authors define the *graph manager* term as $M = (S, I, F)$ with S is the *graph set* $S = G_1, G_2, \dots, G_l$, I as the set of inter graph edge, and for the rooted nesting tree denoted as $F = (V^F, E^F)$. The compound graph topology in this representation is divided into multiple graphs, which are nested within each other. The geometry of compound nodes is represented by a rectangle, and the geometry of the nodes could have varied shapes starting from circle, ellipse, star, etc. The size of a compound node is associated with its children nodes and as big as the boundaries of the children nodes' position. The graph or compound graph that locates at the root of the nesting tree is simply called as *root graph*.

The underlying simulated physical system in the layout algorithm was implemented to satisfy the conventions of general drawing in compound graphs. This physical system comprises "electrically charged" particles and connected via "springs" of a desired ideal length, which are associated with the nodes and the adjacency edges in the graph. The objects or nodes are considered to repel each other if only in the case where they are located in the same graph and if any pair of objects that are located "too close" to each other, minor repulsion forces are used in order to avoid overlaps between nodes. Additionally, a "gravitational force"

is introduced towards the center of each subgraph to keep graph components together. This force is also used to handle multiple node sizes and is defined to be independent for the node size or its center to the graph center in order to avoid overlaps with neighboring nodes. The ideal length of the edges is computed according to the edge parts in between the end-nodes borders.

In Algorithm 4, it has the calculation functions of spring, repulsion, and gravitation forces. The spring force $e = (u, v)$ is calculated as follows:

$$F_S = \frac{(\lambda - \|P_u - P_v\|)^2}{\eta} \overrightarrow{P_u P_v} \quad (5.1)$$

In Equation 5.1, the lambda λ is the ideal edge length, mu η is the elasticity constant of the edge and the positions of the nodes are u as P_u and v as P_v . The following algorithm is implemented to compute the spring forces acting on each end of edges:

Algorithm 5: CalculateSpringForces(*GraphMgrM*)

```

1: for  $e = (u, v) \in E^M$  do
2:    $idealLength := \lambda$ 
3:   if  $e$  is an inter-graph edge then
4:      $idealLength* = (u.depth + v.depth) * NESTING\_FACTOR$ 
5:   end if
6:    $c_u := LineSegment(u.center, v.center) \cap u.boundRect$ 
7:    $c_v := LineSegment(u.center, v.center) \cap v.boundRect$ 
8:    $F_s := (idealLength - \|c_u - c_v\|)^2 / \eta \cdot \overrightarrow{C_u C_v}$ 
9:    $F_s(u) += F_s$ 
10:   $F_s(v) -= F_s$ 
11: end for

```

The repulsion force is defined as follows:

$$F_S = \frac{\alpha}{\|P_u - P_v\|^2} \overrightarrow{P_u P_v} \quad (5.2)$$

In the equation 5.2 where α is the constant of repulsion, the sizes, and positions of nodes are calculated bottom up, the compound nodes propagate back to their children to update bounds. The authors mentioned that when the nesting depth increases, the performance of the algorithm decreases dramatically.

Algorithm 6: CalculateRepulsionForces(*GraphMgrM*)

```

1:  $S := \{\}$ 
2: for  $u \in V^M$  do
3:    $S := S \cup \{u\}$ 
4:   for  $v \in V^M - S$  do
5:      $c_u := \text{LineSegment}(u.\text{center}, v.\text{center}) \cap u.\text{boundRect}$ 
6:      $c_v := \text{LineSegment}(u.\text{center}, v.\text{center}) \cap v.\text{boundRect}$ 
7:     if  $u \& v$  in same graph and  $\|c_u - c_v\| < \text{REPULSION\_RANGE}$  then
8:        $F_r := \alpha / \|c_u - c_v\|^2$ 
9:        $F_r(u) + = F_s$ 
10:       $F_r(v) - = F_s$ 
11:     end if
12:   end for
13: end for

```

A gravitation force, which has a fixed magnitude pushes nodes towards the center of the bounding rectangle of a compound graph to keep children components closer together.

Algorithm 7: CalculateGravitationForces(*GraphMgrM*)

```

1: for  $u \in V^M$  do
2:    $\text{center} := u.\text{ownerGraph}.\text{boundRect}$ 
3:   calculate gravitation force  $F_g$  towards center
4:    $F_g(u) + = F_g$ 
5: end for

```

From the algorithms above we can conclude that it has three significant phases wherein each phase has a number of iterations. In the first phase, it constructs the skeleton graph with no subgraphs, which are trees and in this phase relativity, and gravitational forces are deactivated. The next stage is that the trees are introduced to the graph constructed level by level, and the full force is starting to be applied, then in the third phase, the layout is improved by more iterations, and all forces computations are enabled.

5.5 Further Main Features in *autolinks*

Besides compound graphs as the core feature in *autolinks*, other main features that have already been mentioned in the introductory chapter are semantic services, query exploration, document exploration, annotation exploration, and discontinuous annotation. In this section, we discuss those features in more details.

5.5.1 Semantic Services

There are some reasons why *autolinks* is built with service-oriented architecture. One of the reasons is the implementation of multiple semantic services. This architecture enables the principles of independent of vendors, products, and technologies so that this tool doesn't depend on the specific service. Any service can be added and removed as we want according to the needs.

We want to design future *autolinks* not only working in the biomedical domain by using medical entity recognition but also for other fields such as technology, sciences, or more general entity recognition. Semantic services are the core feature in *autolinks* to get data, and multiple semantic services enable users to get information from different sources.

Sparql	Wiki	Demo
/wikidata	/findarticles	/bar
/wikidataLimit		/baz/username
/wikidataObjects		/demo
/wikidataObjectsLimit10		/foo
/wikidataPlainExact		
/wikidataPlainExactLimit20		
/wikidataSubjects		
/wikidataSubjectsLimit10		

Table 5.2: Endpoints lists of semantic services

Table 5.2 displays the available semantic services with their endpoints. These semantic services are *autolinks* SPARQL service, *autolinks* wiki service, and *autolinks* demo service. Sparql service is based on an RDF query language that supports querying over diverse data sources such as wiki data and DBpedia. Wiki service provides access to Wikimedia indices such as wiki data, Wiktionary, and Wikipedia. The demo service is intended for demonstration and research purpose. These semantic services generate diverse results for the knowledge graphs.

5.5.2 Query Exploration

The feature of query exploration is the feature that uses the semantic services to generate knowledge graphs based on queries via the search bar. What makes this feature unique compared to other traditional search bar functionalities is that this feature facilitates generating multiple queries in one click, so that users can explore queries such as 'B-cell', 'T-cell', and others in a single operation.

To get the desired results, we need to enable at least one of semantic services or to enable the annotation search checkbox. When the annotation search is enabled, it will generate an annotation resource, which is related to the query. From this annotation resource, we can access other knowledge graphs or documents that are again related to the query with the provenances feature, which provides traceable sources for the query.

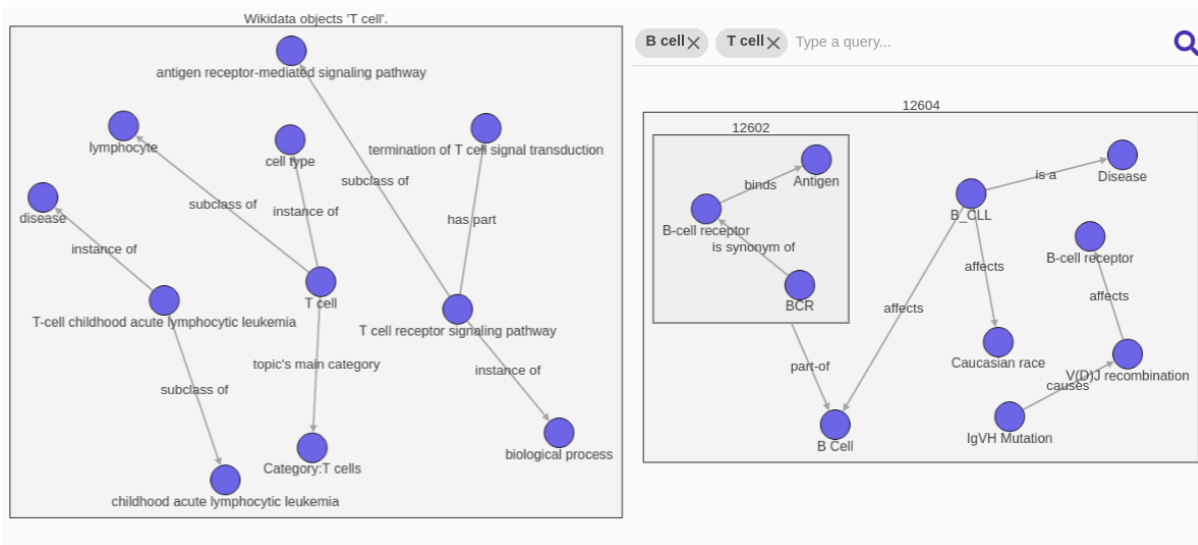


Figure 5.8: Two knowledge graphs generated by two queries in the search bar

Figure 5.8 displays two knowledge graphs, which are generated by two queries, 'B cell' and 'T cell', in the search bar. From the Figure, we can see that in *autolinks*, it is possible to generate multiple knowledge graphs from multiple queries by multiple semantic services. With this feature, we provide users with rich and diverse informations from different sources so that users can compare the results of knowledge graphs and decide which source of information that fits them the best.

5.5.3 Document Exploration

Document exploration helps users to explore entities in the document text. With this feature, the uploaded documents are analyzed to extract the information within the text to get the entities with the annotation types. The documents then will be interpreted by `/nlp/interpret` endpoint to generate annotation resources from each entity in the document text.

In the visual information management, documents are processed by sentence splitter provided by medical entity recognition, so that the documents are shown in sentencewise inside the document lens. *autolinks* provides sentence navigation in the document lens for users in order to enable users navigating to the previous or to the next sentence. A slide number is also provided to help users having direct navigation to the desired sentence by a number. In the main navigation, document list components provide a collection of uploaded documents and once a user clicks the document, it will show all annotation types related to the chosen document in the annotation types component. Some of available annotation types generated by MER in *autolinks* are shown in the Table 5.3.

The sentence without annotation types shown in Table 5.3 is in the normal mode. This normal mode is set once a user chooses a document. In the example given in the table, we use a document, which explains about b-cells. As a user who is probably a medical student or a doctor, we want to investigate the content of the document that we upload, and we want to understand what kind of entities that are provided in the document. With this feature, we can quickly recognize that annotation types available in the document are anatomical site mention,

Annotation Types	Sentence
-	B cells, unlike the other two classes of lymphocytes, T cells and natural killer cells, express B cell receptors (BCRs) on their cell membrane.[1]
AnatomicalSite Mention	<u>B cells</u> , unlike the <u>other</u> two classes of <u>lymphocytes</u> , <u>T cells</u> and natural <u>killer cells</u> , express <u>B cell</u> receptors (BCRs) on their <u>cell membrane</u> . <u>[1]</u>
Medication Mention	B cells, unlike the other two classes of lymphocytes, T cells and natural killer cells, express <u>B cell receptors</u> (BCRs) on their cell membrane. <u>[1]</u>
Procedure Mention	B cells, unlike the other two classes of lymphocytes, T cells and natural killer cells, express B cell receptors (<u>BCRs</u>) on their cell membrane. <u>[1]</u>
PunctuationToken	B cells, unlike the other two classes of lymphocytes, T cells and natural killer cells, express B cell receptors (BCRs) on their cell membrane. <u>[1]</u>
NumToken	B cells, unlike the other two classes of lymphocytes, T cells and natural killer cells, express B cell receptors (BCRs) on their cell membrane. <u>[1]</u>
WordToken	<u>B cells</u> , <u>unlike</u> <u>the</u> <u>other</u> <u>two</u> <u>classes</u> <u>of</u> <u>lymphocytes</u> , <u>T</u> <u>cells</u> <u>and</u> <u>natural</u> <u>killer</u> <u>cells</u> , <u>express</u> <u>B</u> <u>cell</u> <u>receptors</u> (<u>BCRs</u>) <u>on</u> <u>their</u> <u>cell</u> <u>membrane</u> . <u>[1]</u>

Table 5.3: Sentences based on the annotation types

medication mention, procedure mention, and other tokenizations such as punctuation token, numeric token, and word token.

We can enable one or more annotation types for the document, and from this example, we can quickly understand about entities that are related to the chosen annotation type. For anatomical site mention, we can find B cells, T cells, and killer cells; for medication mention, we can find B cell receptors; and for procedure mention, we have BCRs in the document text. For the tokenizations, we can also easily and quickly notice about the punctuations, numeric, and word in the documents.

Since we can enable more than one annotation type, the combination of multiple annotation types is possible in the sentence. As an example, we can enable medication mention and procedure mention and then 'B cell receptors' and 'BCRs' will be annotated in the sentence. Moreover, we can see a difference when we enable word token and medication mention. As we know, word token will annotate every word in the sentence. And for 'B cell receptors', word token will annotate three words 'B', 'cell', and 'receptors' to be three annotations. When we enable medication mention, 'B cell receptors' will be fully annotated as one annotation.

5.5.4 Annotation Exploration

This feature assists users to generate annotation resources from a single annotation in the document text and explore the traceable sources related to the annotation via annotation resources. In the graph view, these annotations resources are visualized as annotation nodes. From a single annotation in the document text, it can interpret more than one annotation resource with the different types of annotation. The example 'B cell receptors' interprets four annotation resources, which are 'B cell receptors', 'B cell', 'cell', and 'receptors'. Then from these four annotation resources, we can identify two different annotation types, which are medication mention for 'B cell receptors' and 'receptors', and anatomical site mention

for 'B cell' and 'cell'. We can have an additional three annotation resources if we include tokenization type such as word token, we will have 'B', 'cell', and 'receptors' additional annotation resources.

The simpler example is shown in Figure 5.9. From 'B cell' entity by ignoring the word token type, it could interpret two annotation resources, which are 'B cells' and 'cells' with anatomical site mention as an annotation type. In that figure, we can notice that there is a compound node bounding the 'B cells' node and the 'cells' node. This bounding box is a special node, which contains all the annotation nodes namely annotation container. We designed this annotation container to organize the annotation nodes inside one container so that they do not spread in all over the graph canvas. We can not draw relations to this container or duplicate this container to the outermost service compound since this container cannot be stored to the database.

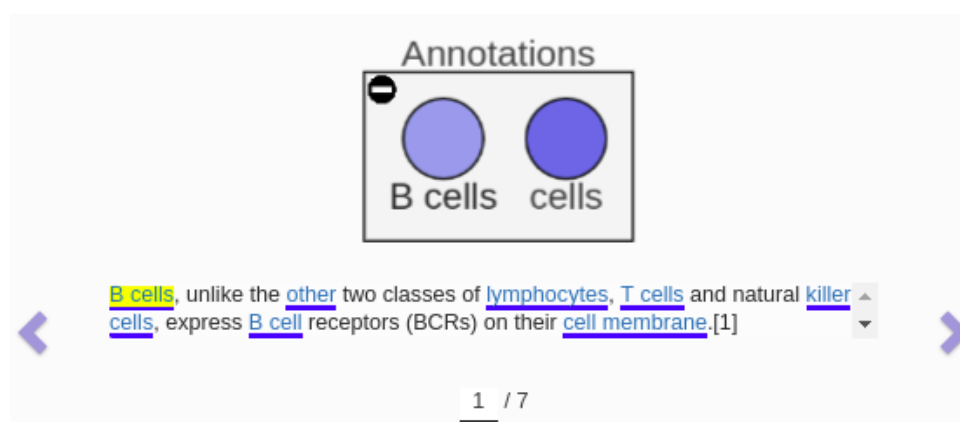


Figure 5.9: Annotation nodes

Similar to annotation container, annotation nodes are also designed as a special node. We can not draw relations from this node inside the annotation container since there is no triples structure designed for annotation resources, so that subject, predicate, object relations are not possible inside the annotation container. However, the nodes inside a service compound are still able to draw relations to this annotation node by duplicating it, turn it to a normal target node, and move the target node to the same hierarchy of the source node.

In Figure 5.9, we can also see that 'B cells' annotation is highlighted in the yellow color. It is a visual state of the text, which is activated once a user hovers an annotation resource, which has a relation with an annotation. In the same sentence, there is a possibility that the same texts appear more than once and in the resource container, same annotation nodes also could appear many times. This visual state helps the user to distinguish from which annotation the annotation node comes. In this case, the 'B cells' node comes from the 'B cells' annotation, which has an index of the sentence, 0 - 7.

In this subsection, we can conclude that the primary purpose of an annotation node is helping the user to explore the related sources of the desired annotation. With this function, the user can generate knowledge graphs and jump to a sentence in particular documents, which have a relation to the annotation. From the example 'B cells' above, the user can explore the knowledge graphs of 'B cells' and 'cells' that have already been previously generated by the

user. This provenance feature is intended to help users accessing their previous knowledge from the knowledge graphs and documents.

5.5.5 Discontinuous Annotation

In this last subsection of visual information management, we discuss about the discontinuous annotation feature. With this feature, users can add more entities in the document text so that the annotating process is not only done automatically by the medical entity recognition but also it can be created or changed manually by the users. Different from other traditional text annotation tools, discontinuous annotation feature enables users to annotate multiple entities from different offsets to be a single new annotation.

As an example, Figure 5.10 shows how 'B cells' and 'antigen' can be annotated as a single entity despite 'B cells' and 'antigen' have different offsets since it has an offset gap where the word 'present' exists between those entities. To manually annotate an entity, the user needs to press the alt key while highlighting the desired text. The annotation dialog is shown once the user releases the alt key and the user can provide the entity with an annotation type.

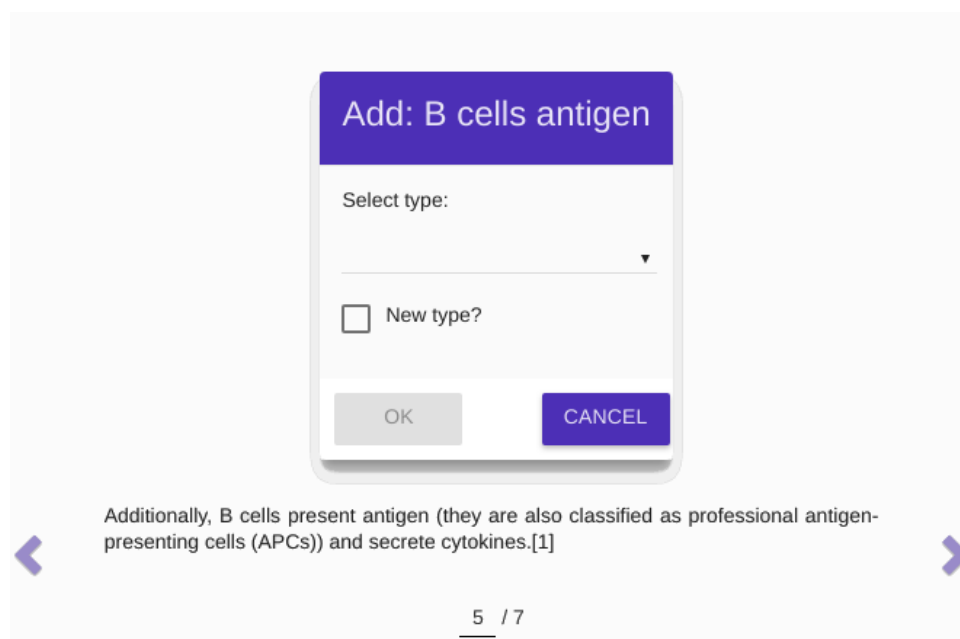


Figure 5.10: Discontinuous annotation 1

Besides the existing annotation types, *autolinks* provides a text input for a new annotation type. It could be shown by clicking checkbox 'New type?' and it will toggle an input that asks for a new type so that users can give the chosen entity with a new annotation type. To proceed with the new annotation type, the user needs to click 'OK' button and the new type will be stored in the database. After confirming the new annotation, the selected texts will be immediately annotated as the new annotation in the document text.

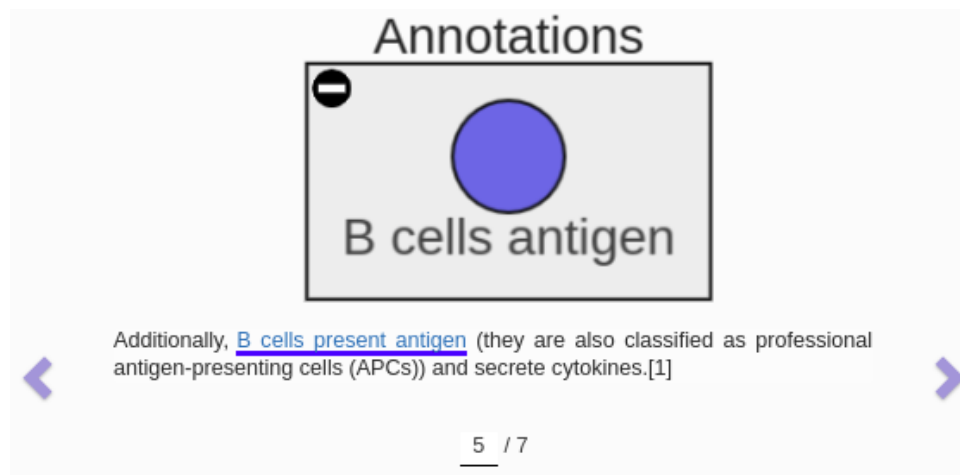


Figure 5.11: Discontinuous Annotation 2

Figure 5.11 shows an annotation resource 'B cells antigen', which is also shown immediately after the user confirming the new annotation. From the Figure, we can notice that 'B cells present antigen' text is fully annotated as a new annotation and on the contrary, the annotation resource has a label 'B cells antigen' without 'present' word. From that annotation, we can comprehend the aim of discontinuous annotation feature where we can annotate the words in which we only need to be a single annotation even though the chosen words have different spans.

6 Qualitative Evaluation

In this chapter, we present qualitative evaluation as an evaluation of whether our system has already met the expectations and satisfied the research questions. The evaluation is implemented as feedback questionnaires gathered from the users after conducting the usability test. In the research questions, we argue that *autolinks* helps users to access information faster with the integration of semantic services and the compound graph implementation helps users to have a better concept representation. Twenty-one people participate in this qualitative evaluation and they were given tasks to try features in *autolinks* starting from building a knowledge graph with compound nodes, query exploration, document exploration, annotation exploration, and discontinuous annotation.

6.1 Evaluation Setup

To evaluate the usability and the usefulness of *autolinks* for helping users to access information and to comprehend concepts faster with the help of compound graphs implementation. We conducted a usability test and we provide a user manual in our GitHub page²³ to show how to operate *autolinks* and perform the features available in *autolinks*. We ask users to do the following steps:

1. First, the user needs to create a new account in the registration form and login to the *autolinks* server.
2. To build a knowledge graph, it can be done by 'click and hold' in graph canvas. Once a context menu appears, the user can select an add icon to create a new node. Once a node is created, the user can draw a relation to another node by clicking the node to show a qTip and click a link icon to create an edge. Other buttons such as an arrows button and an edit button are respectively for merging nodes and for update the node details.
3. To do a query exploration, the user needs to enable at least one active semantic service in the main navigation and optionally can enable annotation search and case-insensitive option. In a single operation, the user can explore multiple queries by filling queries in the search bar and click the search icon. This action will generate knowledge graphs related to the given queries.
4. To do a document exploration, the user can click the circle navigation in the lower right side, and select the upload icon button. An upload modal will pop up and the user can select a document, enable the overwrite option, and click the upload text button. The document then will be uploaded, analyzed, and interpreted. A new document will be shown in the document lists; the annotation types of the document will be extracted; the document lens will be enabled. Once annotation types menu is ready in the main

23. <https://uhh-lt.github.io/autolinks>

navigation, the user can start to explore annotation types, which are available in the document.

5. To do an annotation exploration, the annotation resources based on the text spans can be retrieved by clicking the desired annotation in the text. The annotation resources are shown in the annotation nodes and will be placed inside an annotation container. The annotation resource node will highlight the annotation text span in the document lens once we hover it. The annotation resource helps the user to get provenances in the node. It will show all traceable sources which have relations from the node itself to the documents and knowledge graphs.
6. To create a new annotation and to perform discontinuous annotation, the user can hold 'Alt Key' in the keyboard to temporarily hide all annotations and the user can start to select a certain text or texts in the different spans to be new annotations. Once the texts are selected, the user can release the 'Alt Key' to open an annotation modal. The selected texts are shown in the top of modal and the type of the new annotation can be chosen either from existing types or a new type.

6.2 Feedback Details

By conducting a usability test, we gathered feedback that allows us to conclude how *autolinks* performs. Specifically, we discuss that *autolinks* helps users to access information and compound graphs represent concept better than normal graphs.

We asked users for both quantitative feedback with Likert scales and qualitative feedback. The Likert scale is set with the scale one to five, with the scale value one, meaning that a user strongly disagrees until the scale value five, which means a user strongly agrees.

The first question is, if *autolinks* is able to help users to access information. The intention of this question is that users obtain the benefits of *autolinks* in general in which it could help them to access information faster with the integration of semantic services and the implementation of compound graphs. In Figure 6.1, we summarize our findings that 2 (9.5%) of the users answered "totally agree", 10 (47.6%) of the users answered "agree", 6 (28.6%) of the users answered "neutral", 2 (9.5%) of the users answered "disagree", and 1 (4.8%) of the users answered "strongly disagree".

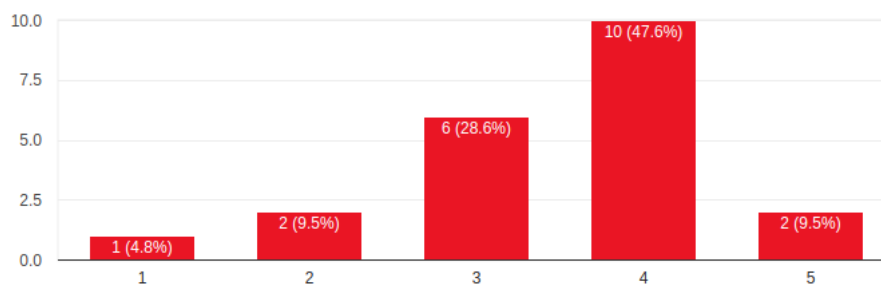


Figure 6.1: Does autolinks help you to access information?

The following question is, if compound graphs are better than the normal graphs. In this question, users should see the difference between compound graphs and normal graphs and understand why the compound graphs are better than the flat graphs in order to represent a concept. In Figure 6.2, we summarize that 7 (33.3%) of the users answered "totally agree", 10 (47.6%) of the users answered "agree", and 4 (19%) of the users answered "strongly disagree".

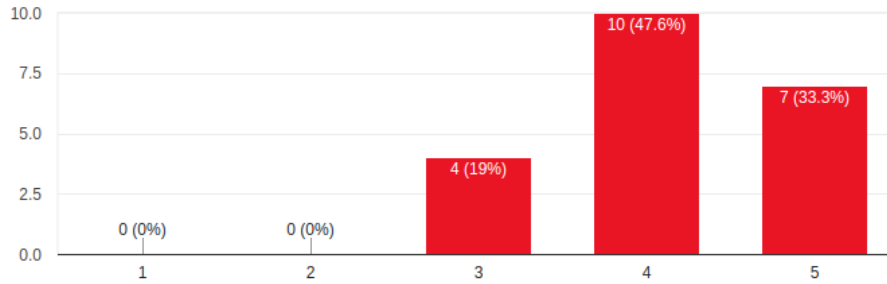


Figure 6.2: Do you think that compound graphs represent concepts better than normal graphs?

Figure 6.3 shows the result of the third question, which stated about the integration of semantic services helps research findings of the users. We argue that the integration of semantic services could help users to get more diverse results of knowledge graphs and help the process of research findings. From this question, We summarize that 2 (9.5%) of the users answered "totally agree", 11 (52.4%) of the users answered "agree", 6 (28.6%) of the users answered "neutral", 1 (4.8%) of the users answered "disagree", and 1 (4.8%) of the users answered "strongly disagree".

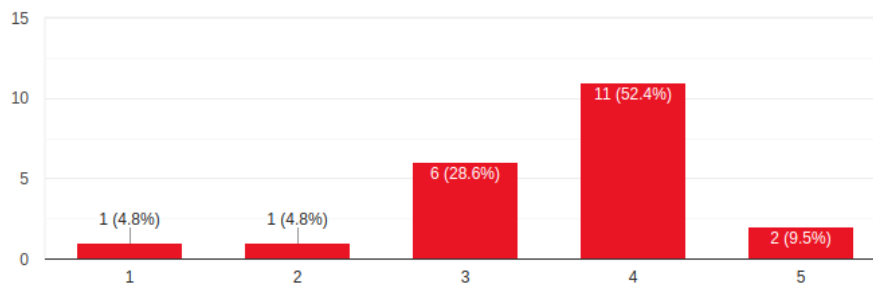


Figure 6.3: Does the integration of semantics services help your research findings?

The next question asked that autolinks hinders/blocks users to get information. This question is principally a control question, which refers to the first question. A control question is needed to make the answers from the subjects more consistent. This question intends to improve the validity of question regarding *autolinks* helps users to access the information. The result is shown in Figure 6.4 and we summarize that 2 (9.5%) of the users answered "totally disagree", 12 (57.1%) of the users answered "agree", 5 (23.8%) of the users answered

"neutral", 1 (4.8%) of the users answered "disagree", and 1 (4.8%) of the users answered "strongly disagree".

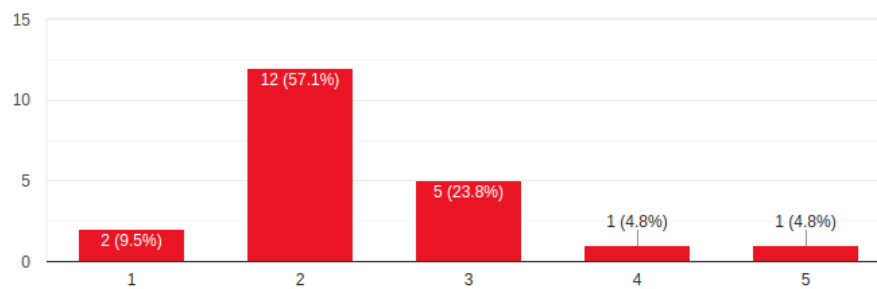


Figure 6.4: Do you think that autolinks hinders / blocks you to get information?

The last Likert-scale question stated about the compound graphs, which are ineffective compared to the normal graphs. This question is also a control question and mainly refer to the second question, which is to improve the validity of question regarding compound graphs. It can be seen in Figure 6.5 that 5 (23.3%) of the users answered "totally disagree", 12 (57.1%) of the users answered "disagree", and 4 (19%) of the users answered "strongly disagree".

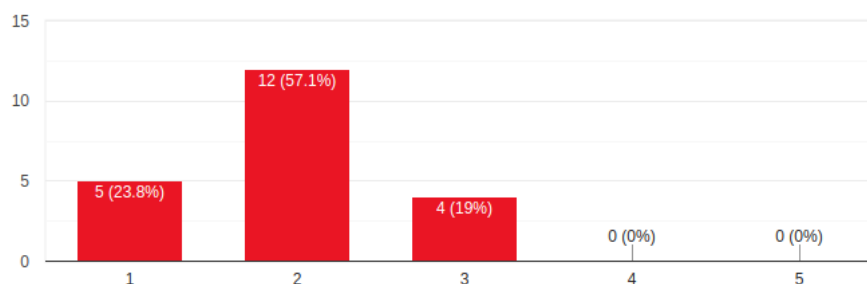


Figure 6.5: Compound graphs are ineffective compared to normal graphs

Not only gathered we quantitative feedback with Likert scales, but also we gathered qualitative feedback by asking the opinions from the users. In general, users agreed that *autolinks* is an information management tool that helps them to access information from multiple sources and visualize the information in compound graphs.

Some of the opinions of the users about *autolinks* are that it is helpful to find associated keywords or sources based on a search key; it helps to annotate medical terms via uploaded texts; it supports compound graphs implementation and supports users to have a better understanding of results given. However, *autolinks* in the current state needs to be more intuitive and for effective usage, it needs some improvements in the results given as well as more semantic services that help users to find more information. At the moment, Wikipedia service is the only service that provides information, which is not that productive, as it still provides way too much unnecessary information.

Those were about the opinions from users about *autolinks* in general. For the compound graphs implementation, we found that most of the users agreed that compound graphs are better in representing concepts than flat graphs, from both quantitative and qualitative feedback. The opinions of the users that we gathered are a compound graph helps clustering related information of a topic together to get a clearer picture or having a clear boundary; it reduces unnecessary nodes and edges and replace them with a single node to represent a generalized concept in ontology; a compound graph can be expanded and collapsed in which this feature is not found in the flat graphs so that users got a better understanding of the concept hierarchies.

After the control question of compound graphs, we asked users which features that they thought were the best. These features are displayed in Figure 6.6. We found that the compound graphs are still the most favorite feature in *autolinks* as 11 (52.4%) of the users voted for this feature. Query exploration becomes second as being voted by 9 (42.9%) of the users, followed by semantic services voted by 7 (33.3%) of the users and document exploration voted by 6 (28.6%) of the users. More details can be seen in Figure 6.6.

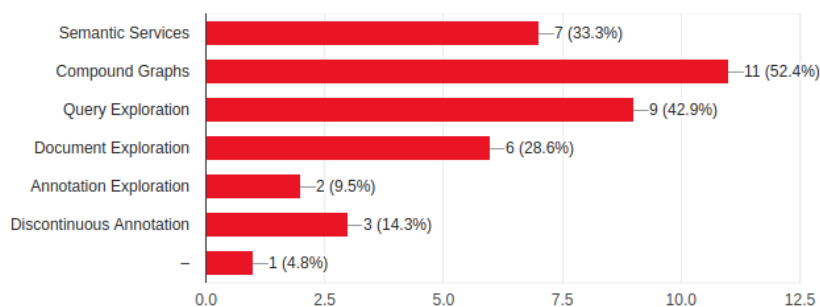


Figure 6.6: What are the best features shown in the autolinks?

The reason why compound graphs and query exploration features became the two most favorite features would be that compound graphs help users to have a faster comprehension towards the concept and query exploration helps users to access the informations not only from a single query but also multiple queries. Semantic services became the third favorite since this feature is also a vital feature to provide quick access to the source of information and gain new knowledge.

6.3 Feedback Summary

The goal of feedback from the usability test is that the majority of the users agree subjectively that our implementation is better than the traditional approach, from the compound graphs and the integration of semantic services. We argue that compound graphs represent concepts better than the normal graphs and the integration of multiple semantic services help users get richer knowledge than only one static service.

We can summarize that our implementation has already met the expectations from the very first result of the feedback. We found that 12 (57.1%) of the users "agree" and "strongly agree"

that *autolinks* help them to access information. This statement is strengthened by the fourth question, which showed that 14 (66.6%) of the users answered "strongly disagree" and "agree" for the question "Do you think that *autolinks* hinders/blocks you to get information?".

We also found a more confidence result from the question of "compound graphs represent concepts better than normal graphs?" as the majority or 17 (81%) of the users answered "agree" and "strongly agree" that compound graphs give a better representation for the concepts compared to the normal graphs. This statement is also strengthened by the fifth question where 17 (81%) of the users answered "strongly disagree" and "agree" towards the statement "Compound graphs are ineffective compared to normal graphs."

In the question of "Does the integration of semantic services help your research findings?", we found that 13 (61.9%) of the users answered "agree" and "strongly agree". So we can conclude that the majority of the users agreed that the integration of semantic service helps users to access information and to gain new knowledge.

We also can conclude that the results of the feedback of our usability test proved that the design of *autolinks*' user interface is not intrusive for users to access information. During the development, we devised *autolinks* with a simple user interface design approach with a focus on the extensive space for the knowledge graphs. That is why 95% of the screen in *autolinks* is a graph canvas.

Besides the graph canvas, we have a main navigation on the left side only shown once it is clicked to accommodate the functionalities of services, documents, and annotation types. A side navigation is used for the nodes/edges details including its traceable sources, and external features are placed in the bottom grid controller. These components are again only shown when they are desirable by the users. We designed these components as dynamic as possible to be a non-intrusive user interface with a toggling approach. This toggling approach enables components, which are mentioned above can be shown only when they are needed. With this approach, the user's screen is designed and focused mostly on the knowledge graphs without any static intrusive component, which is particularly not needed during the whole time of utilizing *autolinks*.

However, some of the users who have no idea about *autolinks* before found some difficulties when they tried to operate *autolinks* for the first time. We gathered those feedback as follows:

- The first entry page should not be empty, it is confusing for the new users and makes the purpose of the tool no clear.
- The wiki service was not enabled by default. At least one service should be enabled directly in order to let a user get the results of knowledge graphs in the first trial.
- Interaction should be more intuitive, add guidelines inside the app.
- Semantic services at the moment provide way too much unnecessary information and are not that productive.

From the feedback above, we realize that some improvements need to be done for *autolinks*. Onboarding feature will be a solution for the users who felt that the first entry page should have hints or guidelines about how to operate *autolinks*. The result of semantic service will be a concern for future works about how to produce a better semantic result, and this issue

does not become the concern for this thesis. Other recommendations from the feedback for *autolinks* will be discussed in the future works.

7 Conclusion and Future Work

In this last chapter, we would like to give conclusions and future works, which might be useful for the better *autolinks*. We consider that this application system still needs enhancements and more ideas to fulfill and to satisfy the user requirement. Some weaknesses still appear, but it could be improved and maintained. The recommendations in the future works are delivered in order to continue improving this application so that it will give more benefits to the user in the long term.

7.1 Conclusion

We presented *autolinks*, a bottom-up information management tool. With this tool, users can build their own ontology with a bottom-up approach where they can construct knowledge graphs from sources such as texts from uploaded documents or semantic services and starting from the most specific concept such as a rabbit or a mammal to the most generalized concept such as an animal. With this approach also, users can model and enhance the information from the mentioned sources, so that users can have better and personalized knowledge graphs.

We designed *autolinks* to help users access information and gain new knowledge faster with the implementation of compound graphs and integration of multiple semantic services. The user interfaces (UI) of *autolinks* are also devised as dynamic as possible without any static intrusive component. The components such as main navigation, side navigation, or bottom grid controller are modeled with a toggling approach where they are shown/hidden according to the user's needs.

With the implementation of compound graphs, *autolinks* has a unique visualization implementation where the information, which is represented by graphs can be put inside of a node. Compound graphs facilitate multi-level generalization of the concepts and reduce the number of edges and nodes so that the knowledge graphs are clearer to be seen and easier to comprehend.

The integration of multiple semantic services helps users in research findings. With this feature, users can quickly access information and gain new knowledge from multiple sources of information. Users might choose the service that they think useful and disable the other unneeded services. With service-driven architecture, users may have options to choose compared to static design. Moreover, new semantic services can be added to the broker according to the needs.

We conducted a usability test and gathered feedback from the users with the aim of testing the system and exploring the features of *autolinks*. We asked them about what they think about *autolinks* in general, how can compound graphs give a better representation for the concepts in their opinion, what are the best features shown in *autolinks*, and why those features are helpful. The results showed that our information management tool helps users to access information

and compound graphs visualization has been proven to have a better representation of the concepts than the normal graphs.

Compound graphs became the best features chosen by the users followed by query exploration and semantic services features. Compared to flat graphs, compound graphs feature is more interactive, and there is no need to draw relation/edge to describe parent and children relation. Since relations/edges in compound graphs can be reduced, the knowledge graphs become more organized. Compound graphs also give a more precise boundary of generalized hierarchy since inside a compound graph, children nodes or another knowledge graphs are grouped together according to their similarities.

Finally, the integration of semantic services also help users to gain new and various knowledge from multiple semantic services, and we can conclude that *autolinks* as a bottom-up information management tool helps users to access information from multiple sources of information and helps users to comprehend information given by the sources of information faster.

7.2 Future Works

Several future works are discussed in this section with expectations to make the system better and to improve the user experience of both existing features and new features. Recommendations in the future works are also gathered together with the feedback in the usability test.

First, a human-in-the-loop approach will definitely be a priority for the next version of *autolinks*. A human-in-the-loop approach enables human to evaluate the results given for future reference. The evaluation done by the human becomes a recommender for the system so that *autolinks* could receive the recommendation in order to learn and correct the mistakes previously done to improve future results. We can take an example from this human-in-the-loop approach in the *autolinks* implementation. *autolinks* currently is implementing the NLP in the biomedical domain so we can say this tool is biomedical information management. Knowledge graphs generated by semantic services do not always produce ideal results. With the involvement of biomedical researchers or even medical doctors, the bad results can be evaluated by them, and this evaluation will be a recommendation for *autolinks* for generating better results based on the feedback given.

Then, onboarding feature, a feature that gives hints to the users how to operate the application in the first entry page, is necessary to implement. Many users faced difficulties in operating the application since *autolinks* at the moment only shows empty canvas after login. By onboarding feature, step-by-step guide can be performed automatically after the new user entering the start page. This feature also could be used for introducing new features published in *autolinks*.

Some new features are also required to be available in the next version of *autolinks* such as upload documents with richer extensions (.pdf, .docx, .odt) and document viewer feature so that users can see the whole sentence in a read mode of the full document. With this feature, users can select or pick a particular sentence directly from that document instead of navigating with left and right navigation. Coloring based on annotation types is also a feature that the

users want. users had difficulties to recognize the type of the annotations since at the moment *autolinks* annotates entity in the document lens only with a blue color. Annotation types can actually be enabled or disabled for the purpose of knowing which annotation type the annotation is. However, annotation types based on color are felt as a better option for the users to recognize the type of annotation because colors are more perceptible by the human eyes than texts.

Information sharing feature is another feature that users also think would be beneficial in *autolinks*. With information sharing feature, multiple users can collaborate to build knowledge graphs. This feature facilitates a user to make corrections towards concepts done by other users and to annotate entities in the document texts collaboratively.

Finally, A/B testing to evaluate the results of semantic services is suggested to be performed for future works. This testing can split the evaluation to different groups of users such as a control group, a group that doesn't use the app and a test group, a group that utilizes the app. Then we can compare the results from both of the groups. This evaluation requires a better result of semantic services. At the moment the results of semantic services are not that productive and provide much unnecessary information.

Bibliography

- [Aven, 2013] Aven, T. (2013). A conceptual framework for linking risk and the elements of the data–information–knowledge–wisdom (dikw) hierarchy. *Reliability Engineering and System Safety*, 111:30–36.
- [Battista et al., 1998] Battista, G. D., Eades, P., Tamassia, R., and Tollis, I. G. (1998). *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition.
- [Benikova et al., 2014] Benikova, D., Fahrner, U., Gabriel, A., Kaufmann, M., Yimam, S. M., Landesberger, T., and Biemann, C. (2014). Network of the day: Aggregating and visualizing entity networks from online sources. In *Proceedings of the NLP4CMC Workshop at KONVENS*, Hildesheim, Germany.
- [Bertault and Miller, 1999] Bertault, F. and Miller, M. (1999). An algorithm for drawing compound graphs. In Kratochvíl, J., editor, *Graph Drawing*, pages 197–204, Berlin, Heidelberg. Springer Berlin Heidelberg.
- [Bontcheva and Cunningham, 2011] Bontcheva, K. and Cunningham, H. (2011). Semantic annotations and retrieval: Manual, semiautomatic, and automatic generation. In *Handbook of Semantic Web Technologies*, pages 78–113. Springer Publishing Company, Incorporated.
- [Bontcheva et al., 2013] Bontcheva, K., Cunningham, H., Roberts, I., Roberts, A., Tablan, V., Aswani, N., and Gorrell, G. (2013). Gate teamware: a web-based, collaborative text annotation framework. *Language Resources and Evaluation*, 47(4):1007–1029.
- [Breslin et al., 2011] Breslin, J., Passant, A., and Vrandečić, D. (2011). Social semantic web. In *Handbook of Semantic Web Technologies*, pages 467–506. Springer Publishing Company, Incorporated.
- [Buzan, 1974] Buzan, T. (1974). *Using Both Sides of the Brain*. Dutton: New York.
- [Buzan and Buzan, 1976] Buzan, T. and Buzan, B. (1976). *The Mind Map Book: How to Use Radiant Thinking to Maximize Your Brain’s Untapped Potential*.
- [Candan et al., 2001] Candan, K. S., Liu, H., and Suvarna, R. (2001). Resource description framework: Metadata and its applications. *Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD) Explorations Newsletter*, 3(1):6–19.
- [Cao et al., 2010] Cao, N., Sun, J., Lin, Y.-R., Gotz, D., Liu, S., and Qu, H. (2010). Facetatlas: Multifaceted visualization for rich text corpora. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1172–1181.
- [Card et al., 1999] Card, S. K., Mackinlay, J. D., and Shneiderman, B., editors (1999). *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

- [Cejuela et al., 2014] Cejuela, J. M., McQuilton, P., Ponting, L., Marygold, S. J., Stefancsik, R., Millburn, G. H., Rost, B., and the FlyBase Consortium (2014). tagtog: interactive and text-mining-assisted annotation of gene mentions in plos full-text articles. *Database*, 2014:1–8.
- [Chisholm and Warman, 2006] Chisholm, J. and Warman, G. (2006). *Experiential Learning in Change Management*. In Silberman, Melvin L. The Handbook of Experiential Learning. Jossey Bass.
- [Dogrusoz et al., 2005] Dogrusoz, U., Giral, E., Cetintas, A., Civril, A., and Demir, E. (2005). A compound graph layout algorithm for biological pathways. In Pach, J., editor, *Graph Drawing*, pages 442–447, Berlin, Heidelberg. Springer Berlin Heidelberg.
- [Dogrusoz et al., 2009] Dogrusoz, U., Giral, E., Cetintas, A., Civril, A., and Demir, E. (2009). A layout algorithm for undirected compound graphs. *Information Sciences*, 179(7):980 – 994.
- [Dogrusoz et al., 2018] Dogrusoz, U., Karacelik, A., Safarli, I., Balci, H., Dervishi, L., and Siper, M. C. (2018). Efficient methods and readily customizable libraries for managing complexity of large networks. *Public Library of Science (PLOS) ONE*, 13(5):e0197238.
- [Domingue et al., 2011] Domingue, J., Fensel, D., and Hendler, J. A. (2011). *Handbook of Semantic Web Technologies*. Springer Publishing Company, Incorporated, 1st edition.
- [Dörk et al., 2012] Dörk, M., Riche, N. H., Ramos, G., and Dumais, S. T. (2012). Pivotpaths: Strolling through faceted information spaces. *IEEE Transactions on Visualization and Computer Graphics*, 18:2709–2718.
- [Eppler, 2006] Eppler, M. J. (2006). A comparison between concept maps, mind maps, conceptual diagrams, and visual metaphors as complementary tools für knowledge construction and sharing. *Information Visualization*, 5(3):202–210.
- [Franz et al., 2016] Franz, M., Lopes, C. T., Huck, G., Dong, Y., Sumer, O., and Bader, G. D. (2016). Cytoscape.js: a graph theory library for visualisation and analysis. *Bioinformatics*, 32(2):309–311.
- [Fruchterman and Reingold, 1991] Fruchterman, T. M. J. and Reingold, E. (1991). Graph drawing by force-directed placement. In *Software Practice and Experience 21*, volume 21, pages 1129–1164.
- [Ghoniem et al., 2005] Ghoniem, M., Fekete, J., and Castagliola, P. (2005). On the readability of graphs using node-link and matrix-based representations: A controlled experiment and statistical analysis. *Information Visualization*, 4(2):114–135.
- [Gruber, 1993] Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220.
- [Guarino, 1995] Guarino, N. (1995). Formal ontology, conceptual analysis and knowledge representation. *Int. J. Hum.-Comput. Stud.*, 43(5-6):625–640.
- [Guarino and Giarretta, 1995] Guarino, N. and Giarretta, P. (1995). Ontologies and knowledge bases: Towards a terminological clarification. In *Towards very Large Knowledge bases: Knowledge Building and Knowledge sharing*, pages 25–32. IOS Press.

- [Harel, 1988] Harel, D. (1988). On visual formalisms. *Communications of the ACM*, 31(5):514–530.
- [Harth et al., 2011] Harth, A., Janik, M., and Staab, S. (2011). Semantic web architecture. In *Handbook of Semantic Web Technologies*.
- [Hlava, 2014] Hlava, M. M. K. (2014). *The TaxoBook: Principles and Practices of Building Taxonomies*. Morgan Claypool, San Rafael, CA.
- [Hoekstra, 2009] Hoekstra, R. (2009). Ontology representation design patterns and ontologies that make sense. In *Proceedings of the 2009 Conference on Ontology Representation: Design Patterns and Ontologies That Make Sense*, Amsterdam, The Netherlands. IOS Press.
- [Jurafsky and Martin, 2000] Jurafsky, D. and Martin, J. H. (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition.
- [Kaufmann, 2017] Kaufmann, M. (2017). *Storyfinder: Graphen zur Verbesserung des Textverständnisses auf Webseiten*. MA Thesis, TU Darmstadt, Germany.
- [Keim et al., 2008] Keim, D., Andrienko, G., Fekete, J.-D., Görg, C., Kohlhammer, J., and Melançon, G. (2008). Visual Analytics: Definition, Process and Challenges. In Kerren, A., Stasko, J. T., Fekete, J.-D., and North, C., editors, *Information Visualization - Human-Centered Issues and Perspectives*, number 4950 in Lecture Notes in Computer Science (LNCS), pages 154–175. Springer.
- [Keim et al., 2010] Keim, D., Kohlhammer, J., Ellis, G., and Mannsmann, F. E. (2010). *Mastering the Information Age. Solving Problems with Visual Analytics*. Eurographics Association, Goslar, Germany.
- [Knowles, 2005] Knowles, E. (2005). *Oxford Dictionary of Phrase and Fable*. Oxford reference online premium. Oxford University Press.
- [Kochtchi, 2013] Kochtchi, A. (2013). *Networks of Names: Obtaining Lombardi’s Narrative Structures by Combining Visual Analytics and Language Technology*. MA Thesis, TU Darmstadt, Germany.
- [Kochtchi et al., 2014] Kochtchi, A., Landesberger, T., and Biemann, C. (2014). Networks of names: Visual exploration and semi-automatic tagging of social networks from newspaper articles. *Computer Graphics Forum*, 33(3):211–220.
- [Kramer et al., 2015] Kramer, J., Roschuni, C., Zhang, Q., Zakskorn, L., and Agogino, A. (2015). Design talking: An ontology of design methods to support a common language of design. In *Proceedings of the International Conference on Engineering Design (ICED’15)*, volume 2, pages 285–294, Milan, Italy.
- [Lievesley, 2006] Lievesley, D. (2006). *Data information knowledge chain*. Health Informatics now. Swindon: The British Computer Society.
- [Manning and Schütze, 1999] Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA.

- [Nazemi, 2014] Nazemi, K. (2014). *Adaptive Semantics Visualization*. Springer International Publishing, Switzerland.
- [Novak, 1998] Novak, J. D. (1998). Learning, creating, and using knowledge : Concept maps as facilitative tools in schools and corporations / j.d. novak. *Journal of E-Learning and Knowledge Society*, 6.
- [Novak et al., 1984] Novak, J. D., Gowin, D. B., and Kahle, J. B. (1984). *Learning How to Learn*. Cambridge University Press.
- [Pedrinaci et al., 2011] Pedrinaci, C., Domingue, J., and Sheth, A. P. (2011). Semantic web services. In *Handbook of Semantic Web Technologies*. Springer Publishing Company, Incorporated.
- [Perer et al., 2011] Perer, A., Guy, I., Uziel, E., Ronen, I., and Jacovi, M. (2011). Visual social network analytics for relationship discovery in the enterprise. *2011 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 71–79.
- [Philip Ogren and Chute, 2008] Philip Ogren, G. S. and Chute, C. (2008). Constructing evaluation corpora for automated clinical named entity recognition. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco.
- [Remus et al., 2017] Remus, S., Kaufmann, M., Ballweg, K., Landesberger, T., and Biemann, C. (2017). Storyfinder: Personalized knowledge base construction and management by browsing the web. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM '17*, pages 2519–2522, Singapore, Singapore.
- [Rowley, 2007] Rowley, J. (2007). The wisdom hierarchy: representations of the dikw hierarchy. *Journal of Information Science*, 33(2):163–180.
- [Savova et al., 2010] Savova, G., Masanz, J., Ogren, P., Zheng, J., Sohn, S., Kipper-Schuler, K., and Chute, C. (2010). Mayo clinical text analysis and knowledge extraction system (ctakes): architecture, component evaluation and applications. *Journal of the American Medical Informatics Association*, 17(5):507–513.
- [Stein and Blaschke, 2009] Stein, K. and Blaschke, S. (2009). Corporate wikis: A comparative analysis of structures and dynamics. *Proceedings of 5th Conference on Professional Knowledge Management*, pages 77–86.
- [Stenetorp et al., 2012] Stenetorp, P., Pyysalo, S., Topić, G., Ohta, T., Ananiadou, S., and Tsujii, J. (2012). Brat: A web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics, EACL '12*, pages 102–107, Avignon, France.
- [Stenetorp et al., 2011] Stenetorp, P., Topić, G., Pyysalo, S., Ohta, T., Kim, J.-D., and Tsujii, J. (2011). Bionlp shared task 2011: Supporting resources. In *Proceedings of the BioNLP Shared Task 2011 Workshop, BioNLP Shared Task '11*, pages 112–120, Portland, Oregon.
- [Sugiyama and Misue, 1991] Sugiyama, K. and Misue, K. (1991). Visualization of structural information: automatic drawing of compound digraphs. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(4):876–892.

- [Terra and Angeloni, 2003] Terra, J. C. and Angeloni, T. (2003). Understanding the difference between information management and knowledge management, international conference on management of technology (iamot) conference. *TerraForum Consultores, Toronto, ON, Canada, M4L 3S5*.
- [Thomas and Cook, 2006] Thomas, J. J. and Cook, K. A. (2006). A visual analytics agenda. *IEEE Computer Graphics and Applications*, 26(1):10–13.
- [Trudeau, 1993] Trudeau, R. J. (1993). *Introduction to Graph Theory (Corrected, enlarged republication. ed.)*. New York: Dover Publications.
- [Wiedemann et al., 2018] Wiedemann, G., Muhie, S., and Biemann, C. (2018). New/s/leak 2.0 – multilingual information extraction and visualization for investigative journalism: 10th international conference, socinfo 2018, st. petersburg, russia, september 25-28, 2018, proceedings, part ii. pages 313–322.
- [Yimam et al., 2013] Yimam, S. M., Gurevych, I., Eckart de Castilho, R., and Biemann, C. (2013). Webanno: A flexible, web-based and visually supported system for distributed annotations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 1–6, Sofia, Bulgaria.
- [Yimam et al., 2016] Yimam, S. M., Ulrich, H., Landesberger, T., Rosenbach, M., Regneri, M., Panchenko, A., Lehmann, F., Fahrner, U., Biemann, C., and Ballweg, K. (2016). new/s/leak – information extraction and visualization for investigative data journalists. In *Proceedings of ACL-2016 System Demonstrations*, pages 163–168.
- [Zeleny, 2005] Zeleny, M. (2005). *Human Systems Management: Integrating Knowledge, Management and Systems*. World Scientific.

Appendices

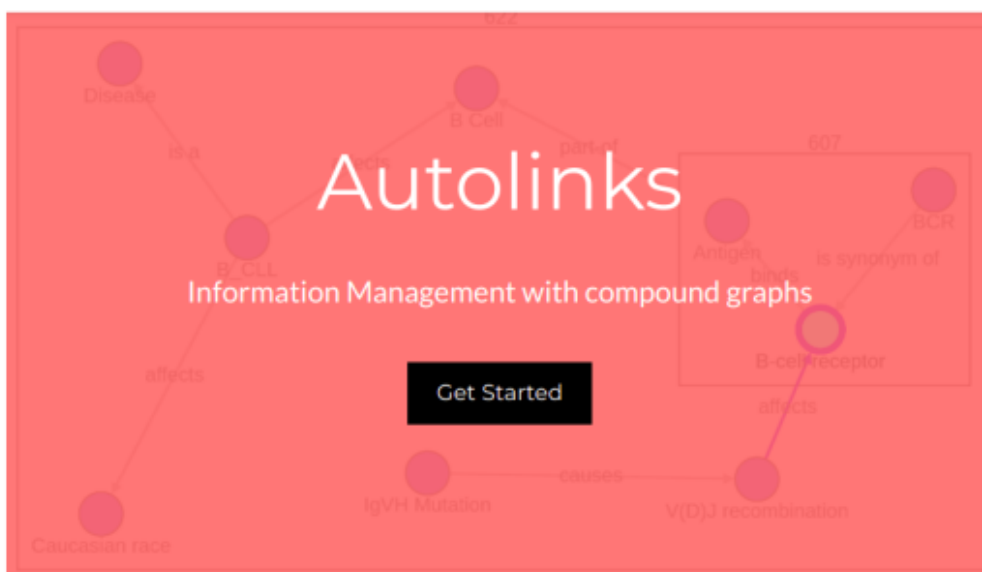
A Questionnaire

Usability Test: Feedback for autolinks

Thank you very much for your involvement in the usability test for autolinks. In this questionnaire, you are asked to give feedback for autolinks improvements. Since this prototype is still in the demo / experimental version, so the feedback focuses on the visualization part rather than the result given by the services.

Likert-Scale (1-5): (1) Strongly Disagree - (2) Disagree - (3) Neutral - (4) Agree - (5) Strongly Agree

* Required



Questionnaire (~ 5 - 15 minutes)

Does autolinks help you to access information? *

	1	2	3	4	5	
Strongly Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

Do you think that compound graphs represent concepts better than normal graphs? *

	1	2	3	4	5	
Strongly Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

Does the integration of semantics services help your research findings? *

	1	2	3	4	5	
Strongly Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

Do you think that autolinks hinders / blocks you to get information? *

	1	2	3	4	5	
Strongly Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

Compound graphs are ineffective compared to normal graphs *

	1	2	3	4	5	
Strongly Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

What do you think about autolinks in general? *

Your answer

Why do you think that compound graphs give a better representation for the concepts in your opinion? *

Your answer

What are the best features shown in the autolinks? *

☐ Semantic Services

☐ Compound Graphs

☐ Query Exploration

☐ Document Exploration

☐ Annotation Exploration

☐ Discontinuous Annotation

☐ Other:

Tell us why those features are helpful for you? *

Your answer

What kind of services would you like to see? e.g.: (SPARQL) Services for Linked Data, webqueries, or automatic inference. *

Your answer

Would you see information sharing feature should be implemented? *

Among a defined user group working on the same knowledge graphs

Your answer

Suggestions / Recommendations *

Your answer

B Feedback

What do you think about autolinks in general?

It is an app which helps to find associated keywords or sources based on a search key. It is my first time using it and I think it would be useful for finding related information regarding a topic.
Biomedical App that helps to annotate medical terms via uploaded texts.
Information Management that supports compound graphs implementation.
Help getting information.
This app is a biomedical information management that implements compound graph and it helps users to have a better understanding of the results given.
Doing annotations from the medical or biological related document.
I did not understand what it does. It does nothing but show me blue bubbles above some words or characters of my text.
Good concept of knowledge representation and exploration
Autolinks in it's current state is a nice demo application for showing compound graphs. However, for effective usage it needs some polishing as well as more semantic services that help the user to find more information. Right now the wikipedia service is the only service that is provided and using it to find information isn't that productive, as it provides way to much (unnecessary) information. The application in general feels very smooth. Dragging compound graphs, nodes and edges works very well. Also, the highlighting of the entities is great as well as creating new edges. However, I dont like that I can for example drag a compound graph above another compound graph. There should be some "collision" detection, so that this won't happen. If something like this is implemented, the merging procedure could also be reworked. If I drag for example one node above another node, they should be merged automatically by creating a new compound graph. Regarding the creation of new edges: After creating a new edge, the next thing a user will do is changing the label. Therefore, after creating a new edge something like a popup should show up, enabeling the user to give the edge a new name.
It helps users to get information from multiple sources and visualize the information with unusual graphs.
it needs to be more intuitive.
it is interesting, but some use-cases are needed.
information management tool that implements compound graphs
Helping users to build knowledge graphs from the semantic services.
Text annotation tool from the uploaded document
Compound graphs concept representation for the biology and medicine domain.
Just like what alvin said, this app implements network graph framework and NLP in the biomedical domain

Why do you think that compound graphs give a better representation for the concepts in your opinion?

If I understand it correctly, compound graphs are a bunch of graphs in brackets. It helps clustering related information of a topic together to get a clearer picture or having a clear boundary.
Compared to flat graphs, a compound / a parent node gives a clearer hierarchy.
Compound graphs reduce unnecessary nodes and edges and replace them with a single parent node to represent a higher level concept in ontology.
It encapsulates children graphs.
I like the feature of expanding and collapsing in the compound graphs and in this feature is not possible in the normal graphs.
Implementing compound graphs I think is also reducing the numbers of nodes and edges compared to the normal graphs.
I have no idea what compound graphs are and I did not see any graph in the program.
compound graphs make hierarchy level in the knowledge graphs become easily seen and noticeable.
they are able to visualize hierarchies while still being a graph.
Gives clearer level of parent-children relations.
I need to compare with flat graphs. But it looks easy understand concepts at the first glance.
help to more compactly represent the information.
A better concept representation than the normal graphs.
Flat graphs give more relations to the graphs in order to represent concepts.
compound graphs reduce the number of edges so that the concept in the knowledge graphs is easier to see and to comprehend.
Giving a rectangle shape for the parent graph.
More interactive compared to the normal graphs and no need to draw relation/edge to describe parent and children relation.

Tell us why those features are helpful for you?

The search bar with Wiki enabled helps to pull useful information with their relations.
I have a background in the medicine and these features help me to relate medical terms with the types faster.
It's very unusual to have and gives a faster comprehension towards the concept.
Help me to find any information
I have given an answer for compound graphs and for query exploration, it really helps the users for finding knowledge graphs that they want and document exploration also really helps especially for medical students to find biomedical terms.
I am interested about the machine learning implementation in the medical domain. How it annotates the text and how we can add another annotation by ourselves.
This feature help me to explore new knowledge given by the semantic services even though at the moment the results are incomprehensible.
I think semantic services are the most important feature of this tool. By using these services the user can quickly access and gain new knowledge. The compound graphs are also necessary to display the different and complex concepts. They make it easy to understand and visualize hierarchies.
Semantic services help me to combine the information from different sources.
easy to search for required information.
let represent complex graphs in a more compact way.
Integration of semantic services help me to have diverse knowledge graphs.
Search bar for query exploration helps me to get knowledge graphs easier.
Document exploration supports user to get medical annotations in the text.
Better representation. It has an expand and collapse feature.
This compound graphs is unique and rarely seen in the most graph-based app.

What kind of services would you like to see? e.g.: (SPARQL) Services for Linked Data, webqueries, or automatic inference.

I am not familiar with the terms. For information searching, I would suggest a larger repository of data and a more sophisticated coverage on relations.
I don't know about these things.
Probably services for linked data gives more data to process.
None.
Larger repository of data would be needed, maybe services that understand about medical world, so they can give accurate results of biomedical knowledge graphs.
automatic inference.
Have no idea about this.
(no idea, sorry, i think wikipedia might be the most important one).
Probably wikipedia is the good choice.
automatic inference.
N/A.
webqueries or google maybe.
I don't know about this.
No idea.
webqueries.
services for linked data

Would you see information sharing feature should be implemented?

Yes
I can see the benefit of information sharing on these graphs, especially for a team brainstorming session.
It's a nice to have when we can work together in the same graphs.
There are always advantages to have a collaboration.
I think, information sharing is needed to have not only personalized knowledge but also for a certain group of people.
Yes, I think by implemementing the information sharing feature, everyone involves in the same knowledge graphs could evaluate the works done by the others.
I think this is also a good feature to be implemented.
i am not so sure about this. defining your own knowledge graphs may have the problem, that these graphs are personalized and therefore maybe not understandable for other users.
I agree that information sharing feature could enable sharing knowledge.

Suggestions / Recommendations

The Wiki search was not enabled by default. I think it should be enabled in order to let user experience the search bar functionality. (The search bar is the first thing I saw and explored)
The app needs to give better results for the graph.
I think, it should not only in the biomedical domain, but also in other domains such as Engineering or Science.
no.
Results from the semantic services need to be improved.
I think that this app still has a lot to improve. The knowledge graphs are still far from perfect and in the document lense, there are still medical terms that are not yet annotated.
directly login after registration (not register and then login separately), call “overwrite files” “yes/no”, not “true/false”.
The visualization is still incomprehensible, but I think I have already got the idea to which direction this app will be
I'd like to be able to "close" a compound graph so it becomes a single node. Then I want to expand the node to the compound graph. In this way I can easily maintain clarity (Übersichtlichkeit) in my knowledge graph.
interaction should be more intuitive. add guidelines/helps link at least. Allow to annotate text.
add some start-up hints and samples and tutorials.
Onboarding feature in the start page would be helpful.
At least one service needs to be activated by default. It is better to have different colors for different annotation types in the document text.
Upload a larger file / pdf file should be possible.
Different color for different types.

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Masterstudiengang Intelligent Adaptive Systems selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Hamburg, den December 5, 2018

Alvin Rindra Fazrie

Erklärung zur Veröffentlichung

Ich stimme der Einstellung der Master Thesis in die Bibliothek des Fachbereichs Informatik zu.

Hamburg, den December 5, 2018

Alvin Rindra Fazrie