Index of Concepts and Sub-Concepts

(Phase 2)

PROJECT REPORT

PREPARED FOR QUILLSOFT

05/2021

TABLE OF CONTENTS

1. INTRODUCTION

Located in Toronto, Quillsoft Ltd. is a leader in assistive technology software that helps people read and write. Founded by Dr. Fraser Shein in 2000, in partnership with Holland Bloorview Kids Rehabilitation Hospital, Quillsoft's products, including WordQ, are used by over 2.5 million end-users, primarily in the education field. While WordQ, and other competing solutions primarily focus on writing, the solution will enable Quillsoft to increase and diversify its revenue stream and increase market penetration.

The Project Index of concepts and sub-concepts is funded by NSERC Engage grant. Seneca research team was led by the Principal Investigator, Dr. Asma Paracha, a full-time faculty member in Seneca's School of Software Design and Data Science (SDDS). Two student Research Assistants who were hired as research assistants to execute the project. Below is the table listing the role and responsibilities of all team members:

NAME	ROLE	RESPONSIBILITIES
Asma Paracha	Principle Investigator	Leading the project and giving the research direction
Anton Biriukov	Research Assistant	Responsible for the development, testing and implementation of Web UI.
Aayushi Baral	Research Assistant	Responsible for the development, testing and implementation of Web UI.
Sergio Hernandez	Research Assistant	Responsible for the development, testing and implementation of NLP Engine
Ralph Lisak	Project manager	Responsible for managing time, budget and other resources along with maintaining the communication between the team and the client.

2. PROBLEM

While reading e-text, readers are unable to handle larger texts sizes due to smaller screen size and are not able to conceive the relationship between interrelated concepts. This impacts their capability to retain the flow of information and they lose their connection to the text.

Quillsoft consistently hears from educators that students struggle with text comprehension. One of the most common complaints, from writing centres at colleges and universities, is that students cannot properly understand even relatively simple assignment questions. Teachers have consistently requested a true comprehension solution at the discourse level (e.g., understanding cause-and-effect, argumentation, etc.).

Quillsoft sees a business opportunity in developing text comprehension technology and the company is planning to include such technology in the next generation of its products, scheduled for market release in 2022. Quillsoft is partnering with Seneca to build a proof-of-concept for a visualization tool that will provide readers with insight into multi-level relationships between concepts contained in long texts.

Main purpose of the tool is to facilitate comprehension of such long texts. The tool will look for the key concepts in the text and how they are related to each other and to other sub-concepts and would provide a comprehensive view.

Project Objectives:

This project is the continuation of the work done in the last phase where the team worked on text extraction, key phrase extraction, and summarization of a set of scientific articles. As the scope of the problem is too big to be addressed in a short span of time, the team explored the preliminary work in phase one with lots of improvements to be tackled by later phases. In this phase we had focused on the following two objectives: Clustering and UI.

The main objective is to build clusters based on the key phrases generated in phase one. These clusters will allow users to navigate and understand the context of the reading by extracting their relations, distribution, and frequencies.

The second main objective is to deliver a user friendly and interactive web-based interface that provides two or three types of visualizations for users to evaluate the effectiveness of the algorithms developed.

Using Python as a language for the backend code for a production environment would not give the best performance, another objective is to evaluate and guarantee the possibility to migrate the text processor engine of phase one to another language programing.

Final Deliverables:

- Final report including the software architecture, discussion of the results (utility, performance, viability, …) and the suggestions for further research.
- Software code
- Software documentation (user and installation manuals)
- Test results
- UI implementation

3. SOLUTION

Based on the pipeline that was developed in phase one of the project, the current solution goes deep into some of the impediments and covers most of the given suggestions.

To start with, the team decided to work on enhancements to the text extraction and its evaluation. By updating libraries and components of the text-processor engine, splitting up heavy functions and processes to reuse code, and adding the capability to reconfigure the default path of the pipeline, it will be now easier to migrate parts or all the source code to a language other than Python.

After the evaluation of some grouping and clustering methods, we decided to implement the Word Embeddings approach to create an initial version of clusters. We also decided to apply this step at the document Level given that at this level a reader would need more than a list of concepts to understand the context of an article.

Finally, the solution provides a user-friendly web application with two views of clusters along with their key words: *TreeMa*p and *x-ray (Scrollspy)*.

3.1 USE OF UPDATED LIBRARIES AND CODE REFACTORING

DESCRIPTION

In stage one of this project, the team decided to use of Python as the coding language considering the multiple NLP libraries that it offers and thus speed up the development. However, for a production environment, the final product would be coded in a language that offers higher performance and scalability to the solution.
For this reason, the team decided to refactor those components that were heavy to read and maintain, so that, it will be easier to migrate.
Since some of the libraries employed are for the exclusive use of Python (i.e., Spacy) it could be hard to find an alternative in another language that gives the same results. With this in mind, we tried to make the components independent of each over to get the flexibility of conversion to another platform without affecting the flow of the solution.

PROCESS

Keeping the modular approach in mind, the project is developed in the following components:

- GROBID Extraction: GROBID is a machine learning library we used to extract the headings and full text from a given PDF file. Even though, it could be very accurate for well-structured documents, for other certain articles the result could be not the best. Scientific documents contain a lot of formulas and formats that can mess up the GROBID engine interpretation, whereby there is an optional step to preprocess these kinds of articles so GROBID can produce better results.

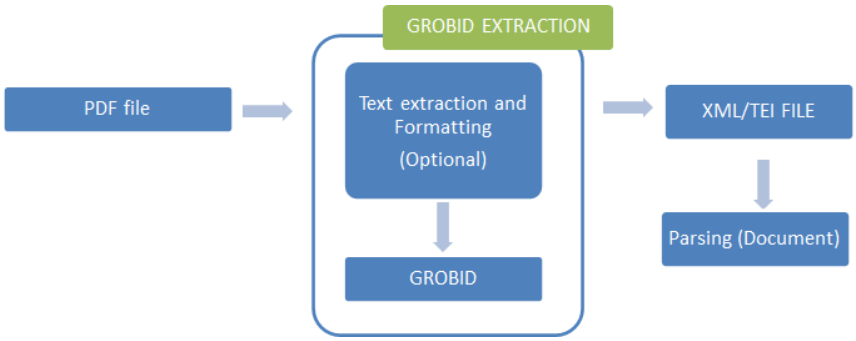

Figure 1. GROBID Extraction.

- Summarization: It uses multiple functions from Gensim and Spacy libraries along with some other generic NLP processes. This component could be the first candidate to keep and use in future versions of the solution.

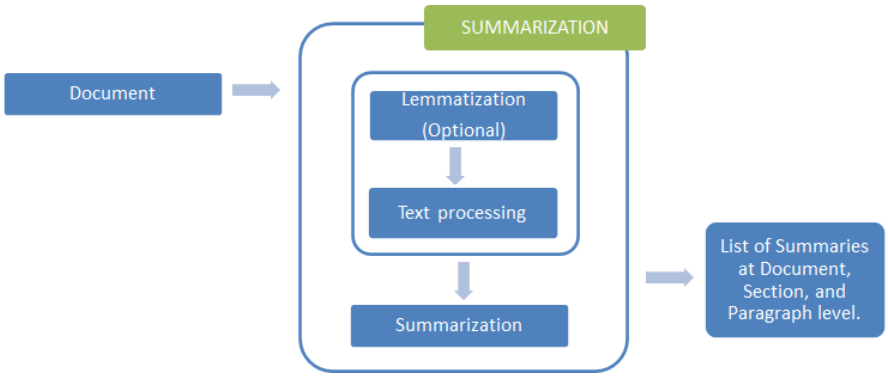

Figure 2. Summarization

- Phrase Extraction: It uses functions from Textacy and Spacy libraries along with other generic NLP processes. It could be another component to keep due to its dependency on python libraries.

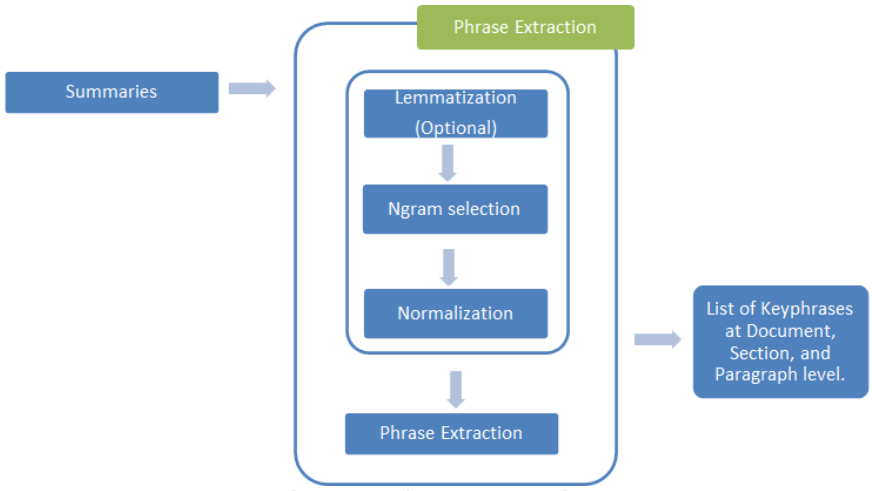

Figure 3. Phrase Extraction

LIBRARIES

Below, the list of components and libraries updated and a small description for those:

- GROBID: Current implementation support versions 0.6.0, 0.6.1, and 0.6.2. The 0.6.1 release significantly improved the header extraction model, for which this version is the minimum recommended. The latest versions of GROBID does not support Windows. Either way, it is a Java-based application for which its execution on Linux will give the best performance.

- Spacy: Spacy was upgraded to version 3.0.3. It requires updating its dependencies: language packages and TextRank. This upgrade is mandatory since the syntax for pipeline integration with external algorithms had changed.

- PyMuPDF, Contractions: Two new versions of these libraries have been released recently. For future implementations, it is recommended to update these packages. However, it will not make a significant impact on the result of the current pipeline.

IMPEDIMENTS

- Gensim is the widely used library in this project, a few weeks ago a major upgrade was released. It is highly recommended to upgrade this package since it will offer higher flexibility and accuracy of the solution components. Though it could require major changes to the syntax of the current solution.

- README file of the back-end project was updated with the latest version of the text-processor engine that is in use. It is not possible to specify the version of libraries in the requirements.txt file to automate their installation. Whereby the libraries with a specified version on the README file must be installed or downgraded manually.

  pip install --upgrade <<library name>>==<<version>>

NOTES

- It is worth mentioning that the component architecture was thought to make it easy to replace them. For example, the first output required for the pipeline is a "Document" object which is filled based on a TEI file; We can replace the GROBID component with another library and fill the result in the same "Document" object, so, the rest of the pipeline will not be affected.

3.2 CLUSTERING

DESCRIPTION

At this point, the team decided to focus the clustering based on the key phrases at the document level. Listing these phrases is not enough for a user to understand the context and the content of a given article.
The technique we decided to go with is Word Embeddings. "Word embedding is one of the most popular representation of document vocabulary. It is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc." [1].

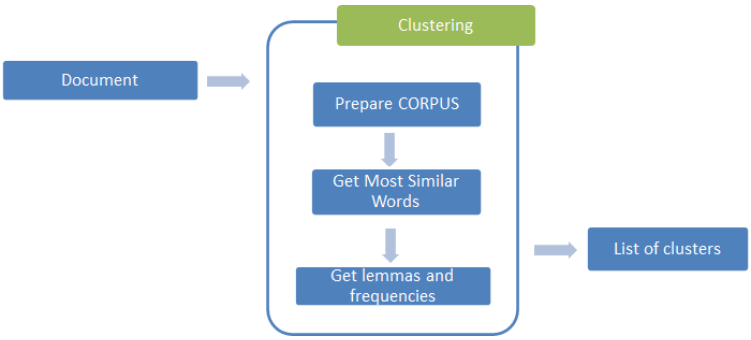

Figure 4. Clustering

3.2.1. Word2Vec with small Corpus: Figure 4 shows the general flow we designed in order to get the most similar words of a given cluster heading. As the first approach, we decided to use the document itself as the corpus:

- From the Document object, we retrieve the list of all the sentences that are part of the article.
- After preprocessing the text (POS tag filtering, stop word removal, and punctuation removal), the Word2Vec model was fed with all the unigrams of the clean text.
- Cosine distance is used to retrieve the closest words within the model and a provided word that should be part of the Corpus and should be a unigram.
- Since most of the key phrases have multiple words, what we did was to get the clusters (closest words) of each single word part of the phrase, and then we merge the result of the resulting groups. Appendix A shows an example of the initial clusters with the single words of each phrase.
- The resulting groups show a redundancy in their keywords; therefore, we got the lemma of each word to achieve a cleaner and smaller list of keywords for the final user. Appendix B shows an example of the final clusters for each phrase.
- On the front-end side, we are working with the XML/TEI file for text visualization and considering the rendering of the XML is not perfect into an HTML container, we calculated the frequency of the word concurrency in this step for visualization purposes.

3.2.2. Word2Vec with a Domain-Specific Corpus: For the second approach, we used a bigger corpus, which includes a set of documents related to the one we want to evaluate. The steps are the same as the first approach, the only difference is that the Corpus for feeding our model not only includes the article to evaluate but a set of articles all related to the same topic.
Appendix C shows the result of this approach. Overall, we can see how the distribution of the words is completely different. Even though, after evaluating them, this distribution seems to be more accurate and useful to understand the context of the cluster.
Either way, the complexity of this approach is higher, considering the need for a specific corpus depending on the document to evaluate. Also, the performance of this approach is considerably low since the resulting model is bigger.

3.2.3. Word2Vec with KMeans: Going forward with clustering and having a vectors model (Word2Vec) already in hand, we decided to explore the KMeans algorithm.

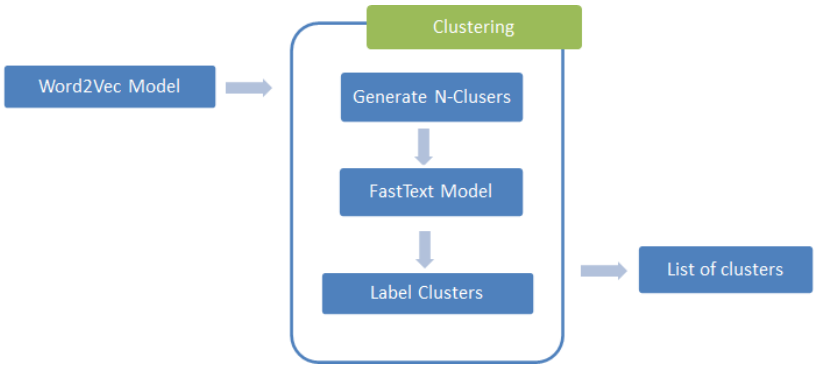

Figure 4. K-means clustering

- Word2Vec model is the same one we use for the two previous approaches.
- With the standard function "KMeansClusterer()" from NLTK, we generate as many clusters as key phrases we got from the text-processor engine at the document level. KMeans has the capability to choose between Euclidean and Cosine distance "to find homogeneous subgroups within the data such that data points in each cluster." [2], so, we tried both. Appendix D shows unlabeled set of clusters.
- The final step was to label the resulting clusters. For that purpose, we used the FastText approach, which is useful to find a relation for rare words within a given corpus. In this case, we can directly compare a compound phrase to the Word2Vec model even if its corpus doesn't include the phrase in it. Appendix E and F shows respectively the output of this approach with Cosine and Euclidean Distance. Clusters could be accurate, but the labeling phase does not show a good distribution of the key phrases.

Although the approach with a domain-specific corpus gives a good-looking distribution of words within the clusters, the complexity of its implementation impacts both the performance and the architecture of the engine. Cluster distribution with the third approach reduces the scope of key phrases with both Cosine and Euclidean distance. Hence, we integrated the first approach into the engine.

LIBRARIES USED

- Gensim
- Spacy
- NLTK
- Sklearn

IMPEDIMENTS

- As it was mentioned, a domain-specific corpus shows a better distribution of words in the clusters, the issue is that the resulting vector model occupies huge amount of space on the disc. For example, the model for the 12papers.pdf file is 900Mb in size, which means when loading the model, it will make use of 900Mb RAM. So bigger is the corpus, lower is the performance.

NOTES

- PoC_KMeans.py: this file is located at the root of the solution. It contains the code with which the third approach of clustering we tried. It uses a static list of key-phrases and loads on the memory Word2Vec and FastText models.
- PoC_Word2VecClusters.py: Even though the code of the second approach with a corpus of multiple documents is pretty like the approach that was implemented into the engine, this file contains the source code to test the clustering with a static list of phrases.

3.3 KNOWLEDGE GRAPH AND SEMANTIC RELATIONS

DESCRIPTION

The clustering phase gives a group of words that could help the user to understand their distribution on the document and focus on a certain section. The next step we went through was to look for semantic relations between keywords.

PROCESS

3.3.1. Semantic Relations - Word2Vec: Taking advantage of the Word2Vec technique implementation in the previous phase, we attempted to group closely related phrases as they represent their semantic relation. "Word2Vec ... It is used for semantic grouping which will group things of similar characteristic and dissimilar far away" [3]. In this approach, we trained a new Word2Vec model; this time it included not only unigrams but also bigrams and trigrams since we wanted to evaluate each phrase as one vector (until now, phrases could have up to three words).

Appendix G shows a 4-level phrase grouping that was obtained by calculating the closest (cosine distance) phrases at the document level. This process simply compares a phrase with the other ones within the context of the Word2Vec model (cosine distance) and keeps the ones whose distance is higher than the average. The same process is repeated as many times as required before it becomes a circular reference, for testing purposes we went through four iterations.

3.3.2. Knowledge Graph - Hearst Patterns: There is a lot of documentation and examples on the internet for building knowledge graphs in order to understand semantic relations from a given text. There is a specific article that implements the Hearst Patterns which are widely used to extract semantic relations (hyponyms): “Python Knowledge Graph: Understanding Semantic Relationships” [4].

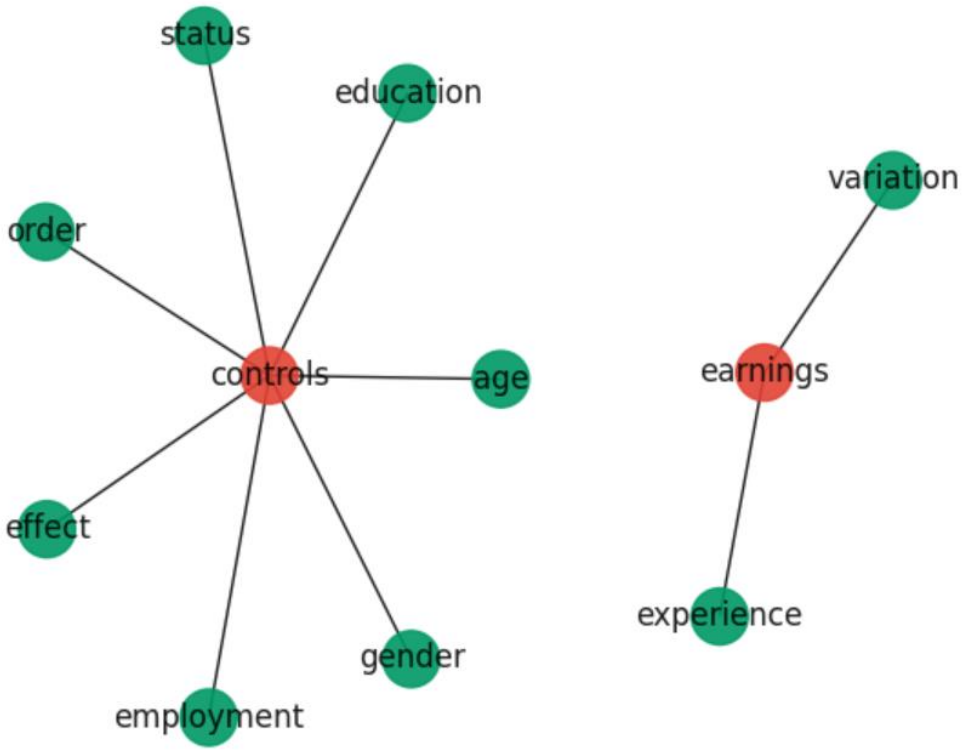

Figure 5. Knowledge Graph - Hearst Patterns (Amit95.pdf)

Figure 5 shows the resulting knowledge graph after implementing the code suggested in the article, with some modifications. The article explains in detail the purpose and implementation of the code; overall, the idea is to build a set of patterns on python to represent the Hearst Patterns on a given text, so that, the semantic relation of “hyponyms and hypernyms” [5] are exposed.

The first finding is that this implementation does not work with scientific articles; We tried random articles from 12papers and 24papers files without getting any kind of relation. We tried with the second set of documents provided "ENT527 Journal Articles" which are related to economics topics, and this time we got a few relations.
The reason could be that the structure of the sentences of a scientific article is more complex since those contain formulas, statistics, and rare words for which is more complicated to identify and extract the entities of the text.

3.3.3. Knowledge Graph – Triples: In order to find a way to extract semantic relations of any kind of text by relaxing the rules of the previous implementation, we found another well-explained article: "Knowledge Graph – A Powerful Data Science Technique to Mine Information from Text (with Python code)" [6].
What this approach does is to evaluate each sentence as a triple: the subject, the root word, and the object. It considers the fact that the subject/object could be a compound word, and that the root of the sentence should be in most cases the main verb.

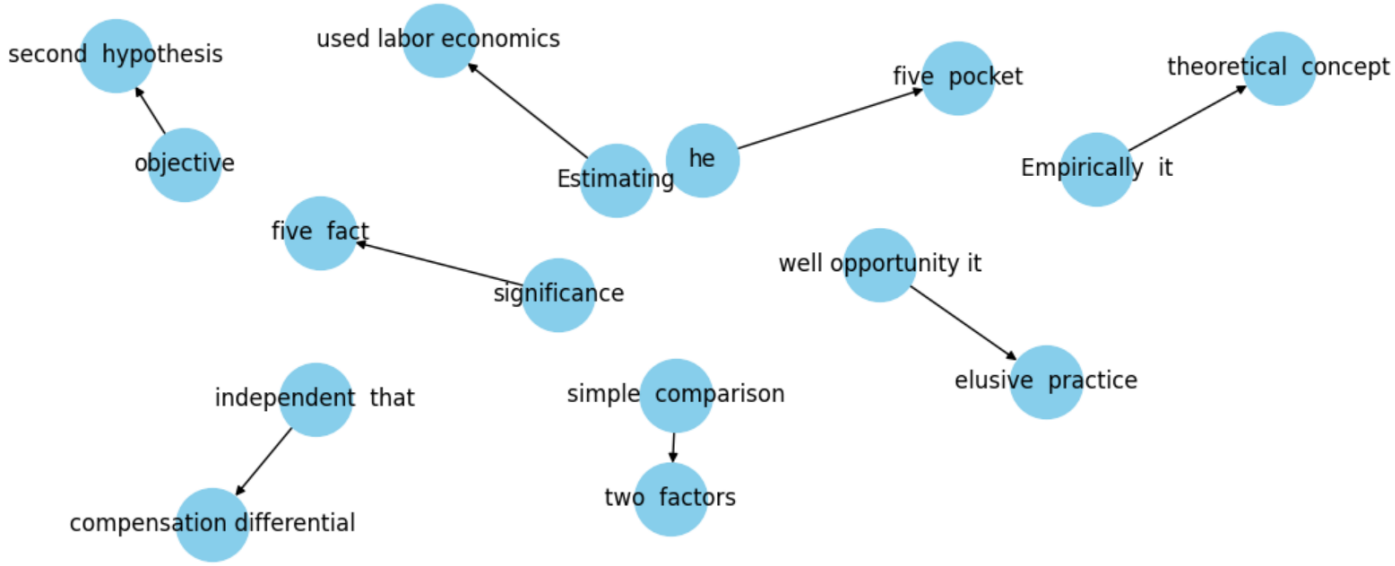

Figure 6. Knowledge Graph – “Is” relations (Amit95.pdf)

In Figure 6, we can see the triplets extracted that have the verb "Is" as the root word and their possible relations. Appendix H lists some more triplets, for a total of 168 possible relations. This for the same document from which we got very few relations with the Hearst Patterns. Even though, is evident not all the relations are precisely or makes sense.

LIBRARIES USED

- Spacy
- NLTK
- Pandas

IMPEDIMENTS

- Grouping phrases based on the Word2Vec semantic relations is not recommended since a single article of 5 pages generates a model (of n-grams) of almost 3GB.
- A knowledge graph could be a great and wise form to present to users the context and contains a given article. One of the drawbacks we have is that these graphs are completely disconnected and independent of the previous phases of the text-processor engine. Some of the branches or nodes of the graphs could match with the set of clusters we generated, but that is not the common case.

NOTES

It was developed a simple console application to allow the testing of Knowledge Graphs generation. The launcher of the application is located in the file init.py.

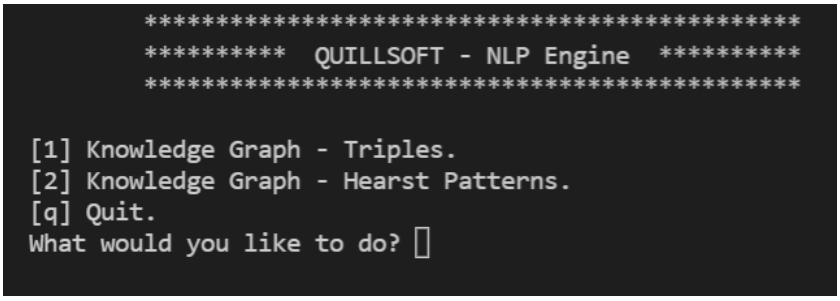

Figure 7 – Terminal Application for Knowledge Graphs.

This application does nothing but calling the following files:

- PoC_Knowledge_Graph.py: It contains the source code to test a Knowledge Graph based on triplets. It requires 2 variables:
  - file_path: physical path where the PDF file to test is located.
  - relation: kind of relation to visualizing in the graph.
  The terminal console will print out the list of relations available.

- PoC_Semantic_Relations.py: It contains the launcher for generating a Knowledge Graph (Hearst Patterns); all the source code of this implementation is in the folder "relationships" of the solution. It requires to fill the variable "file_path" to the physical path where the PDF file is located.

3.4 WEB CLIENT

DESCRIPTION

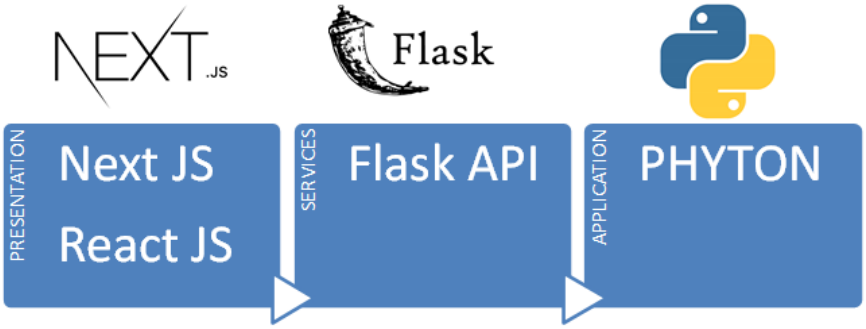

Figure 8 – Application Architecture

Our final solution is a three-layer tool for users to upload PDF files, process them and visualize some of the components integrated into the text-processor engine pipeline:

- Application Layer: Consist of the text-processor engine developed on phase one of this project. It is a Python-based application that extracts, processes, and structures a set of resources (summaries and key-phrases) from a PDF file. In addition to this, a clustering process was added that would help final users to understand and go deeper into the distribution of concepts over a given article.

- Services Layer: It helps the engine to expose the result of the extraction and analysis of the text to the presentation layer. It exposes a REST service endpoint for the presentation layer.

- Presentation Layer: It is a Progressive Web Application (PWA) developed on NextJS, a React Framework [7]. In this section, we are going to focus on this layer.

PROCESS

3.4.1. Mockups

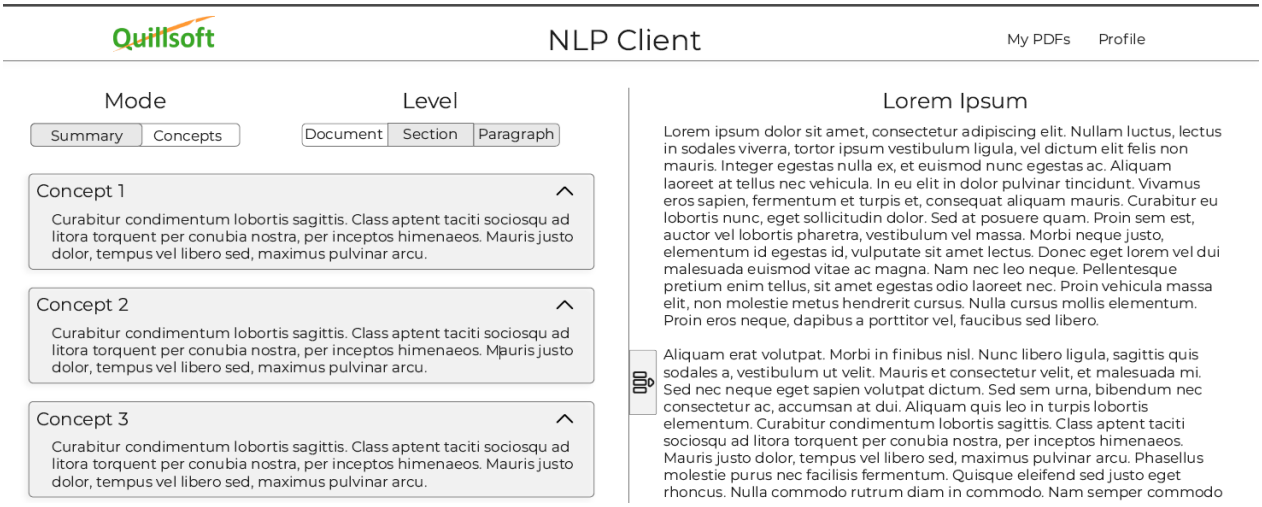

Figure 9. Chunk of UI Mockups V1.

The first step was to develop a mockup that covered a friendly interface of the HTML Viewer developed on phase one of the project. It includes a screen to visualize summaries and phases at the document, section, and paragraph level. In addition, the options for the user to upload new files and to change some available settings at the backend side. This version is detailed in Appendix I.

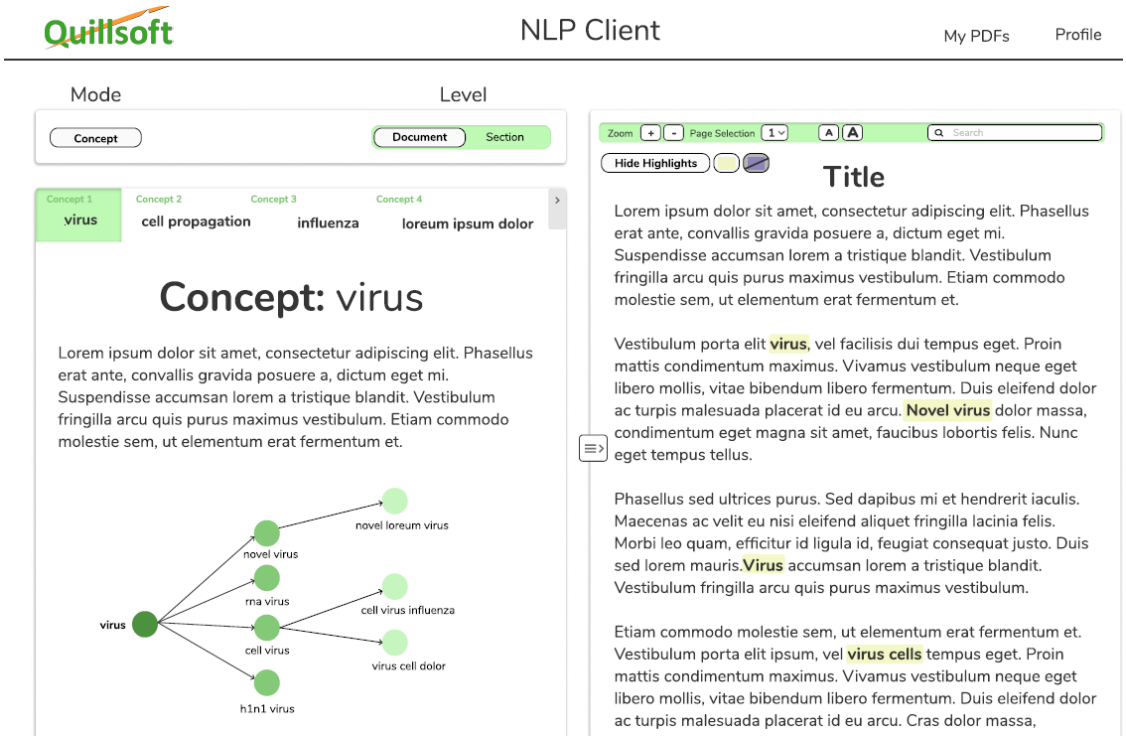

Figure 10. Chunk of UI Mockups V2.

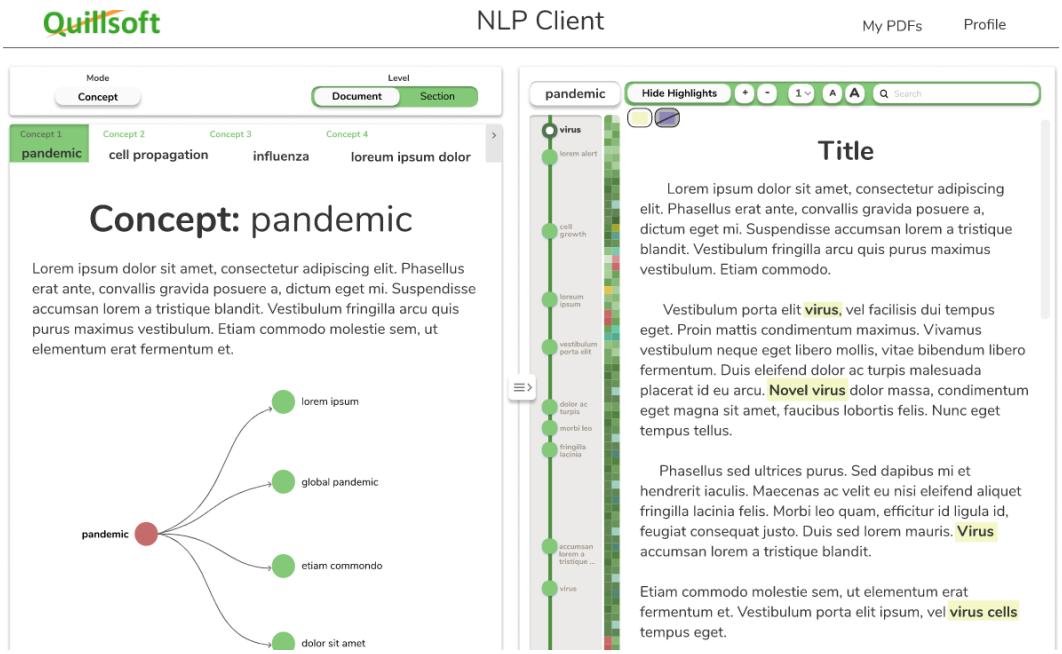

Figure 11. Chunk of UI Mockups V3.

As mentioned in the clustering phase, it was decided to start working on the set of phrases extracted at the document level and showing their distribution into the original article text. So, it was designed a new version of the Mockups to visualize the list of phrases and their possible relations in a more dynamic way more than a list (Appendix J, K).

3.4.2. Modules

- Upload files: this initial step allows a user to upload PDF files for them to be evaluated for the text processor. So, the first screen gives to the user the capability to select a PDF file from the local storage or drag and drop it to a container on the screen.

  For testing purposes, the upload and processing methods allow files with the extension '.tel.xml' so that the engine does not need to call the GROBID service. As long as the files follow the "Text Encoding Initiative (TEI)" [8] guidelines, the engine should be able to process them. Appendix L and M are examples of TEI/XML files from articles 1 and 3 of *12papers.pdf* file.

- Settings Modal: This is an alternative step that allows a user to change default configurations of the text-processor engine for it to apply or not certain rules that could enhance the result of the extraction process depending on the type of article to evaluate. These settings are nothing but the ones that are included in the configuration file of the engine at the back-end side.

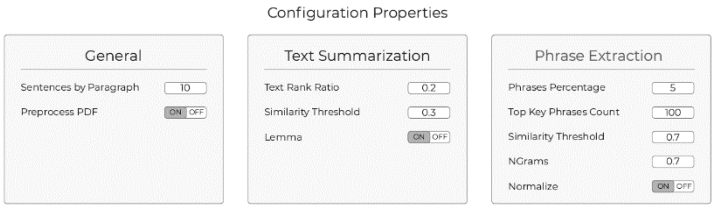

Figure 12. Settings Modal.

- Processing Modal: It shows a loading bar with a set of steps that are being processed at the backend side. These steps do not necessarily match with the processes listed, taking into count that the text-processor engine developed on phase one, is a pipeline that is called by a single method.

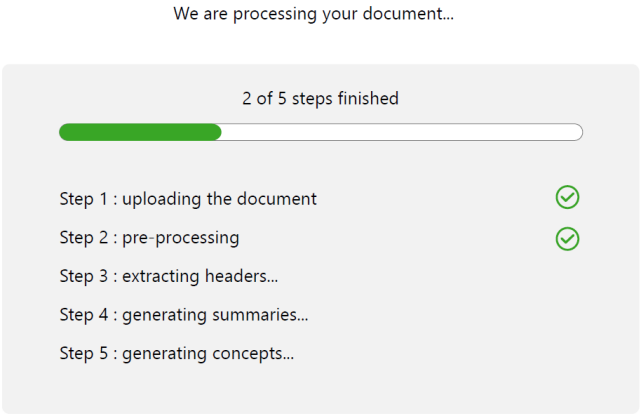

Figure 13. Processing modal

- TreeMap view: As it was represented on the second version of the mockups, the idea was to build a dendrogram to show the distribution and relations of clusters. After some trials, we did not find a way to distribute the clusters into a dendrogram for it to make sense. The team came out with the idea of reducing the complexity of the visualization by implementing a treemap component with the key phrases, their clusters, and their frequencies.

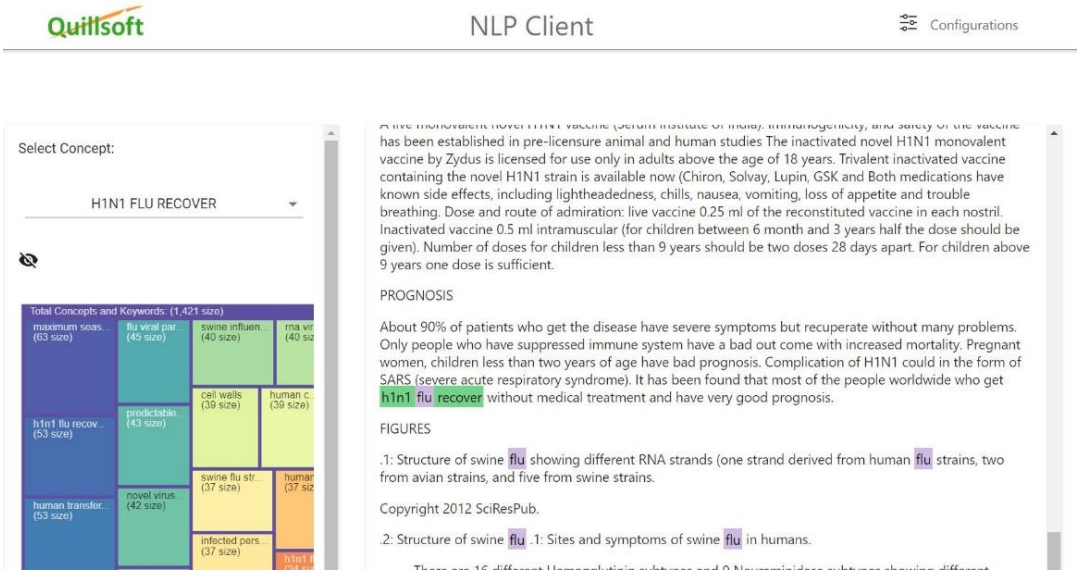

Figure 14. TreeMap view.

The left side has two components: the Treemap, with the key phrases and their frequencies, and a list view with the keywords within their respective cluster (phrase).
The right-hand side shows the extracted text from the original PDF and highlights the key phrase and its keywords selected on the list view on the left side.

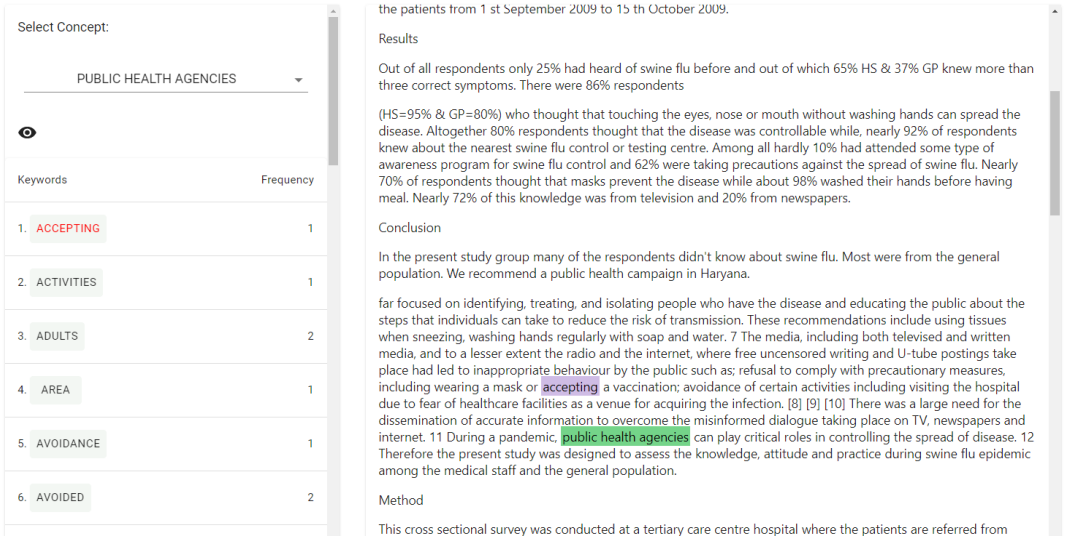

Figure 15: Keywords table on treemap view.

The treemap in Figure 13 is the main view that lists the cluster headings (key phrases). Figure 14 shows a detailed view where the user can select a cluster to be able to visualize keywords within it along with their frequencies.

- Scrollspy view: “Scrollspy is a bootstrap plugin that automatically updates links in a navigation list based on the current scroll position.” [9]. We implemented a non-bootstrap component that changes the style of the phrases (on the right-hand side) based on the scrolling of the text (on the left-hand side). Whatever phrase that is part of the text is being displayed, will be highlighted on both the left-side and right-hand side.

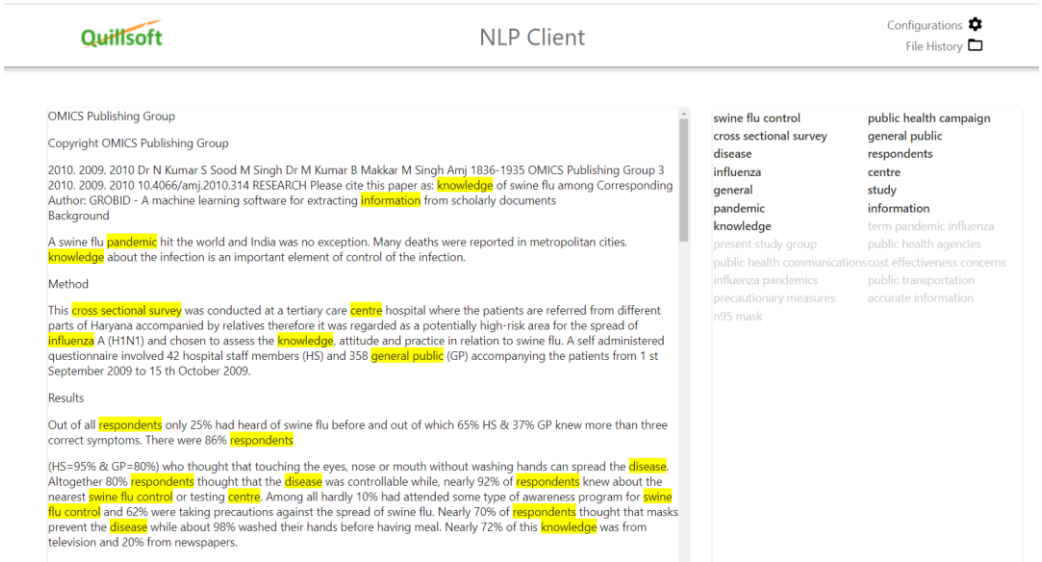

Figure 16. ScrollSpy view.

- History view: Each time a file is uploaded from the front-end running the engine, it is stored in local storage which is part of every JavaScript-based web application. To retrieve the history of files, navigating to the “File History” button will open a new tab with up to 10 files in history. If a file is selected, it will be the only one shown (i.e. If Document 1 is selected, cannot switch to Document 3 directly). To see another file from history, open another new history tab and select a different file.

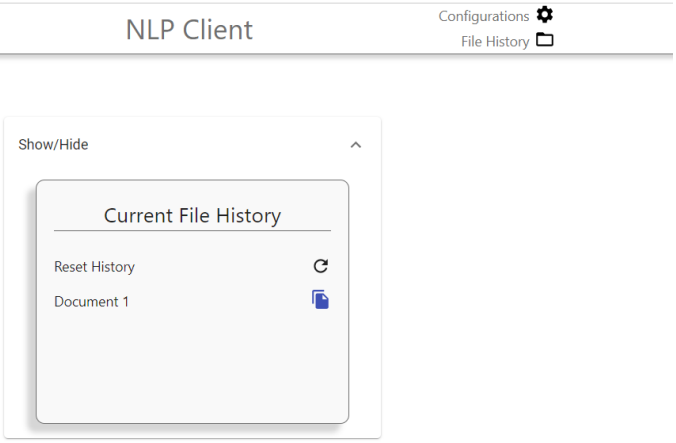

Figure 17: History view (with one file in local storage)

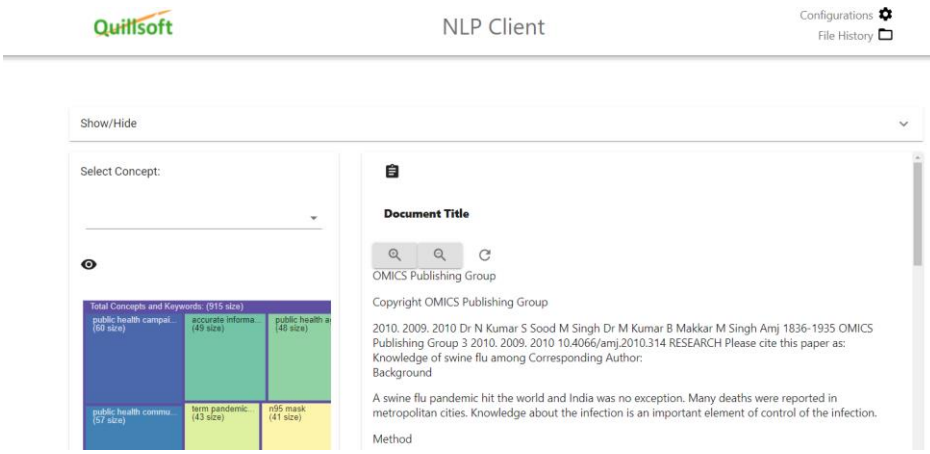

Figure 18: History view with document selected.

3.4.3 Alternate Views

There are several experimental views created for the project that did not get implemented in the final version. As previously mentioned, the key challenges were finding ways to display the concepts and the keywords in a logical yet user-friendly manner.

- Tree Node Graph (Dendrogram) Relational View: During mockup version 2, a decision was made to allow users to visualize the relationships each concept had with one another. In Figure 10, the mockup shows that when a concept is selected, a graph with other concepts that are related to the selected concept is visible as its child nodes, each with its own branches.

  This view includes relationships between concepts represented by nodes and their branches. Upon selecting a concept, a tree-graph is generated with other related concepts that have the same root words. The graph can be moved using the cursor and dragging it inside the grid card. This view also includes a subway-tile representation of the keywords, but highlighting is not functional here.

  Given an array of clusters which includes its concepts and its keywords, the concept strings (i.e. “Care Centre Hospital”) were broken down into single words. In Figure 11, the single words are shown to be “care”, “centre” and “hospital”. Next, each concept that included that specific single word were added as its child nodes.

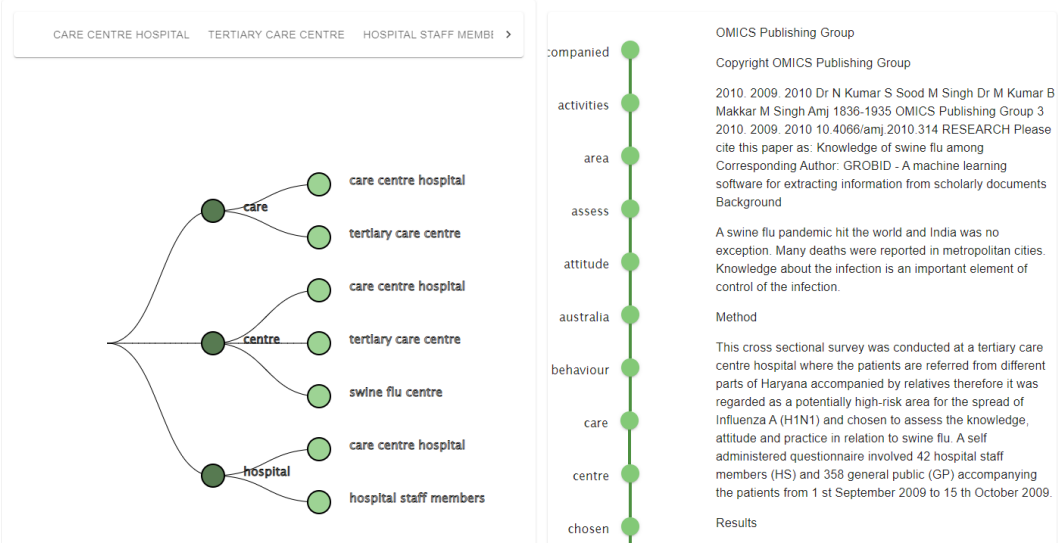

Figure 19. UI representation of Mockup V2

It was concluded that this representation of the relationships between the concepts was rather pointless. There were two major flaws:

- There exists redundancy of concepts in the branch nodes. In Figure 11, it is noticeable that concepts like “care centre hospital” and “tertiary care centre” are repeated more than once.
- Though the diagram shows relations between concepts, these relationships are lacking meaning. The selected concept might have better relationships with other concepts and do not contain matching substrings. Section 3.3 of the current report aims to solve this issue.

- Subway Tile Keywords View: As the dendrogram view was swapped for a tree-map representation of the concepts, the debate on how the keywords should be displayed was still in the works. As the mockup V2 and Figure 11 shows, a subway tile view was integrated to show the keywords of the selected concept. The functionality to highlight keyword(s) on the document was crucial.

  For the selected concept, its keywords are displayed as individual dots on the track. A single keyword can be highlighted at a time by clicking on the circle representing that keyword. Upon clicking, two things will happen:

  - The keyword will be highlighted in purple throughout the document.
  - The currently highlighted keyword and its occurrences are represented on the top right grid below the document title.

  For this alternate view, only one keyword can be highlighted at a time.

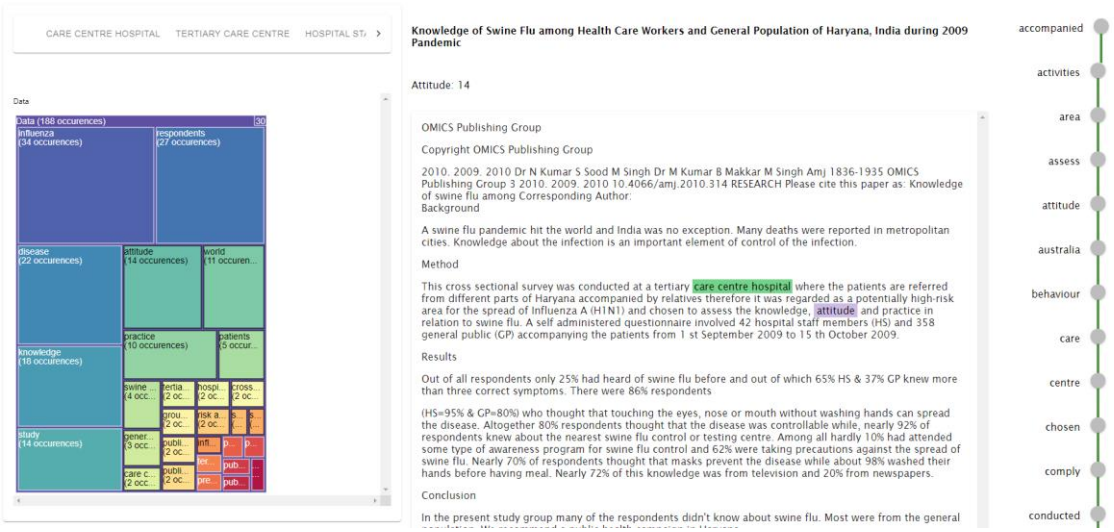

Figure 20: UI representation of concept “care centre hospital” and keyword selected “attitude”

There is nothing fundamentally problematic with this view, as it serves the purpose of highlighting keywords on the document. The major concern was the inconsistency of the tracks when the screen size of the UI was either smaller or larger than expected.

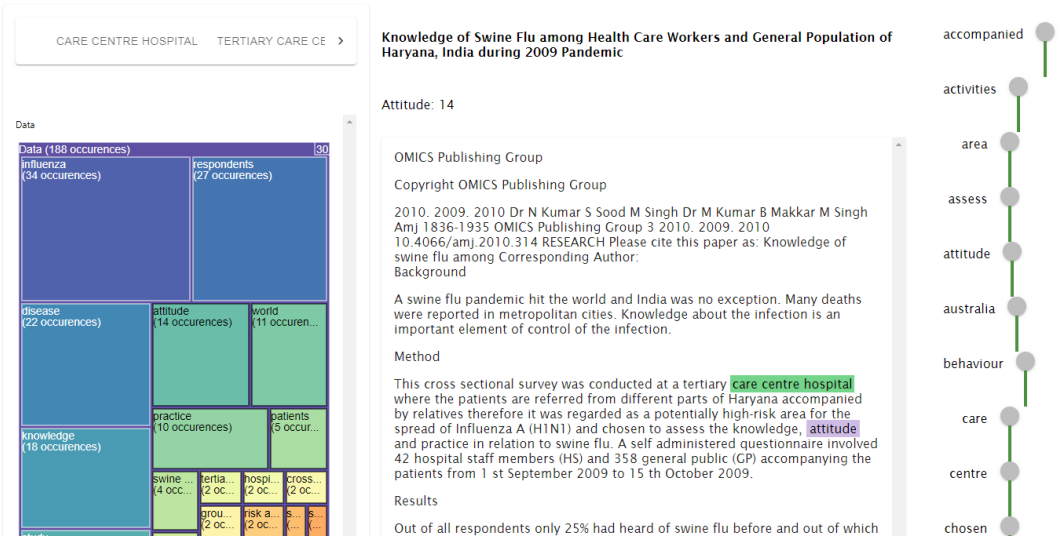

Figure 21: Misalignment of subway track in Tree Map view.

LIBRARIES USED

- material-ui
- react-d3-treemap
- drag-scroll-provider
- react-heatmap-grid
- @material-ui/lab

NOTES

- Live Demo of the Tree Node Graph view can be found at https://kx4o0.csb.app.
- Live Demo of the Subway Tile Keywords View can be found at https://l06q0.csb.app.

IMPEDIMENTS

- The lack of knowledge of standards for text visualization tools in the industry could have been a stumble along the way to came up faster with a final solution.

4. PROJECT METHODOLOGY

- An agile process methodology was used for this project
- In addition to the two PDFs provided on phase one (12 papers and 28 papers), there tested the solution with a set of 21 economic articles.
- The team decided to split the efforts from the beginning to work on both the back end and front-end by parallel, so that, we can build the base of a PWA useful for further phases.
- Weekly meetings were held with the client during the development of key parts of the project. Besides that, the *principal investigator* and *research assistants* had very frequent formal and informal meetings to discuss the technical aspects
- Clients were encouraged to ask questions or provide feedback regarding weekly presented results or weekly demonstrations for an improved iteration.

5. FUTURE DIRECTIONS + IMPROVEMENTS

- Knowledge graphs and semantic relations (Section 3.3) provide a meaningful view and distribution of interconnected entities of a given text. A future improvement should be to look up a way to connect these entities to the clusters.
- Throughout the project, the team discussed and evaluated multiple ways to present the information to the final user and implemented the ones that were viable given the short period. In addition to evaluating the mockups the team came up with, future phases should research into text visualization and interpretation industry standards.
- Looking up to getting a high-performance, scalable, and stable back-end solution, components like the service layer, PDF text extraction, and generic utilities, should be migrated to a more dynamic and faster programing language.

6. CONFIGURATION FILE + SETUP ENVIRONMENT

- Configuration and Setup guidelines are the same as phase one; all the information and updates can be found on the Readme file of the solutions.
- As a reminder, for new installations of the python pipeline, it is necessary to install manually the libraries in which its description is detailed the version number. There is no way to include the version on the requeriment.txt file.

7. REFERENCES

1. Karani, D. (2018, September 01). Introduction to Word Embedding and Word2Vec. Retrieved from https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa
2. Dabbura, I. (2018, September 17). K-means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks. Retrieved from https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a
3. Borcan, M. (2020, August 08). Python Knowledge Graph: Understanding Semantic Relationships. Retrieved May 4, 2021, from https://programmerbackpack.com/python-knowledge-graph-understanding-semantic-relationships/
4. Word Embedding Tutorial: Word2vec with Gensim [EXAMPLE]. (n.d.). Retrieved from https://www.guru99.com/word-embedding-word2vec.html
5. Nordquist, R. (2019, July 03). Definition and Examples of Hypernyms in English. Retrieved from https://www.thoughtco.com/hypernym-words-term-1690943
6. Prateek JoshiData Scientist at Analytics Vidhya with multidisciplinary academic background. Experienced in machine learning. (2019, October 14). What Is Knowledge Graph: Building Knowledge Graph From Text. Retrieved from https://www.analyticsvidhya.com/blog/2019/10/how-to-build-knowledge-graph-text-using-spacy/
7. Next.js by Vercel - The React Framework. (n.d.). Retrieved from https://nextjs.org/
8. P5: Guidelines for Electronic Text Encoding and Interchange. (2021, April 09). Retrieved from https://tei-c.org/release/doc/tei-p5-doc/en/html/index.html
9. What is a scrollspy? (n.d.). Retrieved from https://www.educative.io/edpresso/what-is-a-scrollspy

APPENDICES

APPENDIX	ITEM	LOCATION
A	Word2Vec clusters of single words	../Appendices/Spread_Sheets.xlsx (Appendix A)
B	Word2Vec clusters of phrases	../Appendices/Spread_Sheets.xlsx (Appendix B)
C	Word2Vec clusters of phrases	../Appendices/Spread_Sheets.xlsx (Appendix C)
D	KMeans clusters	../Appendices/Spread_Sheets.xlsx (Appendix D)
E	Labels of KMeans – Cosine Distance	../Appendices/Spread_Sheets.xlsx (Appendix E)
F	Labels of KMeans – Euclidean Distance	../Appendices/Spread_Sheets.xlsx (Appendix F)
G	Semantic Relations – Word2Vec	../Appendices/Spread_Sheets.xlsx (Appendix G)
H	Semantic Relations – Knowledge Graph	../Appendices/Spread_Sheets.xlsx (Appendix H)
I	UI Mockups Version 1	../Appendices/Quillsoft NLP Client_V1.pdf
J	UI Mockups Version 2	../Appendices/Quillsoft NLP Client_V2.pdf
K	UI Mockup Version 3	../Appendices/Quillsoft NLP Client_V3.pdf
L	TEI FILE example 1	../Appendices/article1.tei.xml
M	TEI FILE example 2	../Appendices/article2.tei.xml

Appendix A

Word2Vec clusters of single words

Cluster									
cell	virus	flu	human	vaccine	swine	h1n1	mask	wall	…
flu	virus	people	swine	human	vaccine	h1n1	cell	shot	…
rna	flu	virus	cell	human	swine	mask	h1n1	antigen	…
segment	flu	cell	virus	antigen	swine	h1n1	human	cause	…
shot	flu	vaccine	virus	human	swine	low	swell	people	…
strain	swine	flu	virus	surface	people	human	touch	cough	…
h1n1	virus	flu	swine	vaccine	human	people	cell	antigen	…
recover	researcher	age	effect	droplet	flu	slight	testing	predictable	…
virus	flu	human	h1n1	swine	vaccine	cell	people	mask	…
human	virus	flu	swine	h1n1	people	cell	avert	use	…
infected	virus	h1n1	influenza	flu	vaccine	cause	hand	human	…
person	flu	people	h1n1	swine	virus	nasal	human	particle	…
influenza	flu	virus	human	vaccine	h1n1	swine	people	touch	…
live	people	flu	low	h1n1	infection	human	vaccine	cell	…
main	virus	swine	h1n1	flu	cause	call	low	vaccine	…
surface	swine	virus	flu	mask	people	human	h1n1	cell	…
antigen	virus	h1n1	swine	flu	human	cell	mask	cause	…
novel	virus	flu	human	vaccine	swine	h1n1	surface	cell	…
swine	flu	virus	h1n1	human	people	vaccine	cell	touch	…
particle	virus	flu	human	wall	h1n1	cell	vaccine	mask	…
people	flu	swine	human	h1n1	virus	mask	touch	surface	…
pig	virus	flu	disease	cause	vaccine	human	influenza	h1n1	…
predictable	human	flu	virus	swine	h1n1	people	vaccine	show	…
respiratory	virus	human	mask	exposure	wall	spray	cell	h1n1	…
type	virus	swine	human	h1n1	flu	influenza	cell	shot	…

Appendix B

Word2Vec clusters of phrases

Cluster									
main surface antigen	antigen	cause	cell	flu	h1n1	human	mask	people	…
infected person	avert	cause	cell	cough	disease	droplet	flu	human	…
predictable flu strain	avert	cause	cell	flu	h1n1	human	mask	people	…
h1n1 flu recover	call	cause	cell	change	flu	h1n1	human	individual	…
respiratory	call	cause	flu	h1n1	human	influenza	surface	swine	…
flu rna segment	cause	cell	flu	glycoprotein	h1n1	hand	human	mask	…
rna virus	cause	cell	flu	h1n1	hand	human	mask	occur	…
rna	cause	cell	flu	h1n1	hand	human	occur	people	…
h1n1 virus	cause	cell	flu	h1n1	human	influenza	mask	particle	…
flu strain	cause	cell	flu	h1n1	human	mask	people	shot	…
swine flu strain	cause	cell	flu	h1n1	human	mask	people	shot	…
human	cause	cell	flu	h1n1	mask	people	shot	swine	…
h1n1	cause	cell	flu	human	influenza	mask	people	swine	…
live virus	cell	come	flu	h1n1	human	mask	particle	people	…
influenza virus	cell	conduct	flu	h1n1	human	individual	mask	occur	…
swine influenza virus	cell	conduct	flu	h1n1	human	individual	mask	occur	…
novel virus particle	cell	flu	h1n1	human	mask	occur	particle	people	…
novel swine flu	cell	flu	h1n1	human	mask	occur	people	shot	…
swine flu virus	cell	flu	h1n1	human	mask	particle	people	shot	…
virus particle	cell	flu	h1n1	human	mask	particle	people	shot	…
people	cell	flu	h1n1	human	mask	particle	shot	surface	…
flu shot	cell	flu	h1n1	human	mask	people	shot	spray	…
type	cell	flu	h1n1	human	people	shot	surface	swine	…
novel	cell	flu	human	mask	occur	people	swine	use	…
pig	disease	flu	grade	h1n1	human	influenza	surface	swine	…
cell	flu	human	mask	particle	people	swine	use	vaccine	…

Appendix C

Word2Vec clusters of phrases

Cluster								
h1n1 flu recover	cause	diagnose	emphasize	follow	human	influenza	kill	…
h1n1	diagnose	follow	influenza	lab	novel	positive	rise	…
swine influenza virus	ability	antigen	avian	bud	cause	derive	detect	…
swine flu virus	ability	antigen	avian	bud	cause	derive	flu	…
rna virus	ability	antigen	avian	bud	derive	hemagglutinin	mingle	…
novel virus particle	ability	avian	aware	bud	cause	clothe	comprise	…
virus particle	ability	avian	aware	bud	clothe	comprise	contain	…
live virus	ability	avian	aware	bud	clothe	comprise	cultivate	…
influenza virus	ability	avian	bud	cause	derive	detect	diagnose	…
h1n1 virus	ability	avian	bud	derive	diagnose	follow	hemagglutinin	…
flu rna segment	act	antigen	call	cause	derive	hemagglutinin	human	…
infected person	affected	clothe	comprise	contact	doorknob	eat	hamper	…
main surface antigen	aim	amass	antigen	attach	belong	blend	call	…
predictable flu strain	antigen	article	attach	avian	being	categorize	cause	…
swine flu strain	antigen	article	avian	cause	derive	flu	human	…
human	antigen	cause	derive	envelop	infested	mexico	mingle	…
novel swine flu	antigen	cause	diagnose	flu	h1n1	human	identification	…
rna	antigen	derive	mingle	molecular	mutate	protein	segment	…
flu strain	article	avian	cause	derive	human	kill	lifecycle	…
type	article	call	derive	immunological	mutate	rna	segment	…
cell	breach	break	culture	glycolipid	glycoprotein	infest	lung	…
flu shot	cause	clothe	comprise	doorknob	grade	hamper	human	…
novel	cause	diagnose	h1n1	influenza	lab	mexico	origin	…
pig	cause	eat	flu	human	ingestion	mexico	mingle	…
people	contact	cook	cough	crowd	droplet	eat	ingestion	…
respiratory	fever	headache	sore	throat	tract	weakness		

Appendix D

KMeans clusters

Cluster 1	term	conventional	epidemic	zoonotic	mrna	carry	trial	…
Cluster 2	pig	strain	person	measure	avian	glycoprotein	combination	…
Cluster 3	infect	sialic	nucleus	detect	month	diminish	crowd	…
Cluster 4	close	similar	future	cell	mushroom	touch	clinical	…
Cluster 5	isolated	people	proximity	flu	country	mortality	sar	…
Cluster 6	typical	important	rare	vessel	prevent	face	relenza	…
Cluster 7	world	call	old	avert	aware	distance	swell	…
Cluster 8	inhalation	cough	ensure	derive	neuraminidase	wall	shape	…
Cluster 9	see	usual	breach	linkage	subtype	penetrate	condition	…
Cluster 10	antigen	observe	respiratory	tract	research	unique	enzyme	…
Cluster 11	type	droplet	cook	protein	like	attach	make	…
Cluster 12	article	chance	cylindrical	initiate	sanitizer	wear	pregnant	…
Cluster 13	estimate	come	light	copyright	consist	hemagglutinin	identical	…
Cluster 14	influenza	main	eat	vaccine	less	spread	break	…
Cluster 15	h1n1	designate	affected	envelop	populace	seasonal	family	…
Cluster 16	surface	common	category	infection	mingle	hour	infected	…
Cluster 17	sialidase	glycolipid	doorknob	rapid	strata	good	severity	…
Cluster 18	novel	exhibit	report	transmit	pork	scirespub	show	…
Cluster 19	swine	virus	usa	identify	technique	sneeze	human	…
Cluster 20	notice	segment	fall	fix	membrane	release	replica	…
Cluster 21	cause	pandemic	complication	mutate	zoonosis	topical	support	…
Cluster 22	researcher	rate	lime	different	infest	migrate	being	…
Cluster 23	emphasize	mexico	work	maximum	animal	projection	molecule	…
Cluster 24	remain	spanish	immunological	note	nonvaccine	form	lifecycle	…
Cluster 25	discover	transferable	acid	bubble	reach	alcohol	antiviral	…
Cluster 26	low	ingestion	disease	major	structure	specialize	large	…

Appendix E

Labels of KMeans – Cosine Distance

cell	article	people	flu	cause	envelop	mutate	show	…
flu strain	article	people	flu	cause	envelop	mutate	show	…
human	article	people	flu	cause	envelop	mutate	show	…
infected person	article	people	flu	cause	envelop	mutate	show	…
novel virus particle	article	people	flu	cause	envelop	mutate	show	…
people	article	people	flu	cause	envelop	mutate	show	…
pig	article	people	flu	cause	envelop	mutate	show	…
swine flu strain	article	people	flu	cause	envelop	mutate	show	…
swine flu virus	article	people	flu	cause	envelop	mutate	show	…
type	article	people	flu	cause	envelop	mutate	show	…
virus particle	article	people	flu	cause	envelop	mutate	show	…
main surface antigen	identify	important	disease	lime	scirespub	human	bind	…
flu rna segment	influenza	strain	spread	respiratory	mingle	break	hemagglutinin	…
h1n1 virus	influenza	strain	spread	respiratory	mingle	break	hemagglutinin	…
h1n1	influenza	strain	spread	respiratory	mingle	break	hemagglutinin	…
influenza virus	influenza	strain	spread	respiratory	mingle	break	hemagglutinin	…
live virus	influenza	strain	spread	respiratory	mingle	break	hemagglutinin	…
predictable flu strain	influenza	strain	spread	respiratory	mingle	break	hemagglutinin	…
respiratory	influenza	strain	spread	respiratory	mingle	break	hemagglutinin	…
rna virus	influenza	strain	spread	respiratory	mingle	break	hemagglutinin	…
swine influenza virus	influenza	strain	spread	respiratory	mingle	break	hemagglutinin	…
flu shot	pandemic	usa	ingestion	cough	pork	work	measure	…
h1n1 flu recover	pandemic	usa	ingestion	cough	pork	work	measure	…
novel swine flu	pandemic	usa	ingestion	cough	pork	work	measure	…
novel	pandemic	usa	ingestion	cough	pork	work	measure	…
rna	pandemic	usa	ingestion	cough	pork	work	measure	…

Appendix F

Labels of KMeans – Euclidean Distance

cell	pig	term	novel	h1n1	infect	antigen	report	usual	typical	…
flu rna segment	pig	term	novel	h1n1	infect	antigen	report	usual	typical	…
flu strain	pig	term	novel	h1n1	infect	antigen	report	usual	typical	…
h1n1 virus	pig	term	novel	h1n1	infect	antigen	report	usual	typical	…
h1n1	pig	term	novel	h1n1	infect	antigen	report	usual	typical	…
human	pig	term	novel	h1n1	infect	antigen	report	usual	typical	…
infected person	pig	term	novel	h1n1	infect	antigen	report	usual	typical	…
influenza virus	pig	term	novel	h1n1	infect	antigen	report	usual	typical	…
live virus	pig	term	novel	h1n1	infect	antigen	report	usual	typical	…
main surface antigen	pig	term	novel	h1n1	infect	antigen	report	usual	typical	…
novel virus particle	pig	term	novel	h1n1	infect	antigen	report	usual	typical	…
novel	pig	term	novel	h1n1	infect	antigen	report	usual	typical	…
people	pig	term	novel	h1n1	infect	antigen	report	usual	typical	…
pig	pig	term	novel	h1n1	infect	antigen	report	usual	typical	…
predictable flu strain	pig	term	novel	h1n1	infect	antigen	report	usual	typical	…
respiratory	pig	term	novel	h1n1	infect	antigen	report	usual	typical	…
rna virus	pig	term	novel	h1n1	infect	antigen	report	usual	typical	…
rna	pig	term	novel	h1n1	infect	antigen	report	usual	typical	…
swine flu strain	pig	term	novel	h1n1	infect	antigen	report	usual	typical	…
swine flu virus	pig	term	novel	h1n1	infect	antigen	report	usual	typical	…
swine influenza virus	pig	term	novel	h1n1	infect	antigen	report	usual	typical	…
type	pig	term	novel	h1n1	infect	antigen	report	usual	typical	…
virus particle	pig	term	novel	h1n1	infect	antigen	report	usual	typical	…
flu shot	proximity	designate	transmit	person	chance	mutate	topical	molecule	combination	…
h1n1 flu recover	proximity	designate	transmit	person	chance	mutate	topical	molecule	combination	…
novel swine flu	proximity	designate	transmit	person	chance	mutate	topical	molecule	combination	…

Appendix G

Semantic Relations – Word2Vec

flu shot
h1n1 virus
conventional flu
flu rna segments
h1n1 flu recover
main surface antigens
predictable flu strain
virus particles
flu rna segments
conventional flu
flu strain
h1n1 flu recover
swine flu virus
virus particles
h1n1 flu recover
conventional flu
flu rna segments
main surface antigens
novel swine flu
swine flu strain
infected person
flu shot
h1n1
novel swine flu
novel virus particles
swine influenza virus
virus particles
novel swine flu
h1n1 flu recover
h1n1
infected person
novel virus particles
swine influenza virus
infected person
h1n1 virus
conventional flu
flu rna segments
flu shot
h1n1 flu recover
novel swine flu
…

infected person
flu shot
h1n1 virus
conventional flu
flu rna segments
h1n1 flu recover
infected person
novel swine flu
predictable flu strain
conventional flu
swine flu virus
flu rna segments
flu strain
main surface antigens
virus particles
h1n1 virus
conventional flu
flu rna segments
h1n1 flu recover
main surface antigens
predictable flu strain
virus particles
flu rna segments
conventional flu
flu strain
h1n1 flu recover
swine flu virus
virus particles
flu shot
infected person
predictable flu strain
swine flu virus
h1n1 flu recover
conventional flu
flu rna segments
main surface antigens
novel swine flu
swine flu strain
novel swine flu
h1n1 flu recover
…

Appendix H

Semantic Relations – Knowledge Graph

Subject	Relation	Object
We	provide empirical	empirical hypothesis
theoretical we	emerged from earlier	paid venture employees
We	use	1992 Canadian
who	find	paid survey period
% survey who	establish	employment
multivariate regression we	performed	marital employment country
new entrepreneurs	was	paid workers
who	paid	employees
Canada V6 I	Z2	British Columbia
authors	acknowledge	f'mancial Venture Capital
We	thank	excellent research assistance
partially author	carried	Distingmshed Visiting Faculty
who	entrepreneurs in	paid survey period
causality	established	result
current that	doing	entrepreneurial attributes
performance	affected by	coincidental factors
relatively some	considering	own business
behavioral dimensions	contributed to poor	poor job peers
independent that	is	compensation differential
then it	is likely	former
who	imply	employed individuals
This	explain	new failure ventures
Future work	directed at	causality
Such research	contribute to	established failure enterprises
Canadian writer	asks	short Stephen the
he	is	five pocket
he	heard	again dollars
He	tried	himself
he	borrowed	town
Stephen Leacock	known as	better economics
story	has more	economic thought
significance	is	five fact
he	has	five pocket
be millionaire	has	five cents
who	risking	much VISA gold more
person	is likely	VISA credit nickel
it	lies in	holder
opportunities	be of lower	expected returns
so that	has	stranger
above discussion	lead to	distinct hypotheses
…	…	…