Triple Viz:
A tool to explore document content from a graphical representation of subject-verb-object triples

Jahnavi Dhananjaya

A Thesis
in
The Department
of
Computer Science

Presented in Partial Fulfilment of the Requirements for
the Degree of Master of Computer Science at
Concordia University
Montréal, Québec, Canada

August 2016

CONCORDIA UNIVERSITY

Division of Graduate Studies

This is to certify that the thesis prepared

By : **Jahnavi Dhananjaya**

Entitled : **Triple Viz:**
A tool to explore document content from a graphical representation of subject-verb-object triples

and submitted in partial fulfilment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee :

_____ Chair
Dr. V. Haarslev

_____ Examiner
Dr. M. Kersten-Oertel

_____ Examiner
Dr. Y. Yan

_____ Supervisor
Dr. S. Bergler

Approved by _____
Dr. V. Haarslev
Graduate Program Director

_____ 2016.

_____
Dr. A. Asif
Dean of Faculty
(Engineering and Computer Science)

ABSTRACT

Triple Viz:
A tool to explore document content from a graphical representation of subject-verb-object triples

Jahnavi Dhananjaya

Most of the data available is unstructured. Text mining is the process of automatically extracting information from text. This thesis combines text mining with visualization to develop TripleViz, a lightweight, web-based tool used to process and analyze documents extracting subject-verb-object (SVO) triples, and visualize them as graphs. The SVO triples extracted from documents are visualized using the open-source visualization tools Turtled and Gephi. TripleViz extracts noun phrases and visualizes them in either full or head format to avoid overcrowding on the screen. For the same reason, TripleViz provides an option to select only triples that contain words of interest as provided by the user in the form of a word list. Within TripleViz, the user can also view color-coded output text highlighting words from a word list. This thesis presents an experiment in classifying newspaper articles and blogs into either "specific event" or "generic", which shows a moderate improvement over a strong baseline.

Acknowledgments

This dissertation represents three years of my work at Concordia University's Computational Linguistics at Concordia Lab. The journey of my thesis has been one of professional and personal evolution, made only possible by remarkable people who have guided and supported me. I take great privilege in expressing my gratitude to them.

First and foremost, I would like to express my sincere gratitude to my mentor and supervisor Dr. Sabine Bergler, for welcoming me as a member of her lab and for all the support, opportunities and challenges provided to me during my Master's.

I would like to thank all members of the CLaC lab especially Abtin, Michelle and Canberk and the McGill epidemiologists Kate and Guido for making this experience a memorable one.

I would also want to thank my partner Aditya and his parents Jagannath and Anitha for the support and love. My friends, Asif-Al-Wafiq, Aruna Sudarshan, Anupriya Kulkarni, Mihir Rajurkar, Mark Monaghan and Devika Satyanarayan for being there for me in all thick and thins. You all are very special to me.

To my loving grandparents Maruthi and Kanakalakshmi (Tata and Ajji), my fantastic parents B.V.Dhananjaya and Sudha Dhananjaya, my uncle and aunt B.V Hemaraj and Renuka Hemaraj and my sister Neha for your love and support. Thank you for instilling in me that education is the greatest investment of all, because it is something that can never be taken away from you. You have been my pillar of encouragement and support.

Thank You All..!!

Contents

List of Figures

Chapter 1

Introduction

There is an enormous volume of text available in the digital world today. Text expresses a vast and rich range of information, but encodes this information in a form that is difficult to analyze automatically [Hearst, 1999]. Text mining or text analysis is a variation on a field called data mining, that tries to find interesting patterns from large databases by automatically extracting information from different written resources. Key in text mining is linking the extracted information to find new facts or hypotheses that can be further explored by experiments [Hearst, 2003]. One strategy used in text mining is identifying important entities within the text and attempting to show connections among those entities through visualization. For example, mining a database for the adverse effects of smoking and displaying all these effects using a connected graph (similar to Figure 2.3a). In addition, visualization is a promising tool for the analysis and understanding of text collections [Hearst, 2009]. Section 1.1 discusses the importance of visualization for reviewing a large amount of data.

1.1 Importance of Visualization

There are hundreds of visualization tools available and every tool is different in its own way. Information visualization is increasingly applied as a critical component in research, digital libraries, data mining, financial data analysis, market studies, manufacturing production control and drug discovery [Bederson and Shneiderman, 2003]. Data Visualization helps people understand the significance of data by displaying it in its visual context. Data visualization excels in capturing a viewer's attention by addressing complex problems that could be easily overlooked while looking at the plain text. It is possible to find trends, detect exceptions and outliers, and find emerging patterns by using data visualization. There are different types of data such as textual data, numerical data, digital data, to name a few. Our focus in this thesis is on text visualization.

Real-world textual data such as newspaper articles or social media articles can be of high importance to, for example, epidemiologists and public health-oriented researchers to study and analyze patterns, causes, and effects of health and disease conditions demographically. Healthmap [Brownstein *et al.*, 2008] is an online tool that monitors disease outbreaks and real-time surveillance of emerging public health threats containing articles and blogs annotated for Adverse Events Following Immunization (AEFI). Our goal was to abstract the AEFI content from the Healthmap articles. To do this, we first analyzed how Healthmap had classified their documents into AEFI positive and AEFI negative. 36,000 Healthmap articles on Measles-Mumps-Rubella (MMR), annotated for AEFI for the time period June 2012 to Oct 2014 were requested from Healthmap and were analyzed by epidemiologists,

who disagreed with some of the Healthmap classifications of AEFI positive and AEFI negative annotations given to the articles and hence began the evolution of TripleViz. A detailed discussion of the annotation of adverse events following immunization from this corpus is presented in [Powell *et al.*, 2016].

1.2 Evolution of TripleViz

As a first step, Elasticsearch and Kibana were used to identify the presence of Adverse Drug Reaction (ADR) terms [Nikfarjam *et al.*, 2015] in the articles. Elasticsearch provides a full-text search engine with an HTTP web interface and Kibana is an open-source data visualization plugin for Elasticsearch. Elasticsearch[1] (refer to Appendix B) was used to index the documents (a process of tagging information with a file so it can later be used for searching) and Kibana[2] was used to provide a user interface to display the articles containing the ADR terms (see Appendix C). For example, a search for the term *smoking* in Kibana displays all the Healthmap articles that had the word *smoking* present in their text. Tools such as Elasticsearch and Kibana provide strong support for real-time analytics. In fact, the presence or absence of ADR terms can be marked by a simple string matching technique. Kibana also provided a numerical analysis (term frequency of a term, etc.) on the Healthmap articles by using the string match technique, but fails to provide insight on the documents. For example, consider a document that talks about a new drug in the market. We cannot decide if the document pro-

[1] http://www.elasticsearch.org
[2] http://www.elasticsearch.org/overview/kibana

vides positive or negative feedback on the drug by looking at the frequency of words. Extraction of more context from the document (positive or negative, what the adverse effects are, etc.,) requires the knowledge of Natural Language Processing (NLP).

Isolated terms with numerical count as displayed by Kibana are of limited use to domain experts. In order to approximate the textual context, we extract subject-verb-object (SVO) triples from the sentences as an exploratory approach in TripleViz. Consider Example 1,

(1) "*Smoking causes cancer*"

where the grammatical subject `Smoking` is the topic of the sentence and the grammatical object `cancer` is the result of the activity of smoking, as indicated by the verb `causes`. Therefore, the resulting SVO triple for the sentence is `causes ( Smoking , Cancer )` . Consider another example and its SVO triple,

(2) "*Ebola has a nasty reputation for the way it damages the body.*"
`has ( Ebola , reputation )`

While analyzed in isolation, the triple above does not convey what the sentence has to say. Hence, we consider entire noun phrases for the subject and the object. The triple in Example 3 shows the triple with complete noun phrases,

(3) `has ( Ebola , a nasty reputation )`

A long article can have a large number of triples, which can be

confusing, making it difficult to link the entities. Therefore, we use a visualization tool to display them. We display the subject-verb-object triples in two different formats, one being the simple text format and the other being a semantic web friendly format used to connect data on the Internet to form the Linked Open Data known as Resource Description Frameworks (RDF) [Klyne and Carroll, 2006]. Gephi [Bastian *et al.*, 2009] is used to visualize simple text triples and Turtled[3] is used to visualize the RDF triples. In addition, the user is also provided with an option to filter the triples against a word list. We integrate all these features into a web-based application named TripleViz.

TripleViz, a light-weight web-based, text extraction tool, is an extension of B2G [Bergler and Dhananjaya, 2015]. It is largely composed of open source software which is easy to configure as well as replace on demand. The user has an option to choose the data required to visualize and generate the type of visualization preferred for the SVO triples. TripleViz is not fine-tuned and definitely not domain-specific, making it open to a larger audience. It is built in a modular fashion and hence, can be customized easily. The tool provides three different views to visualize a text (see Figure 1.1). The first view in Figure 1.1 displays the RDF triples in a more readable format called Turtle triples [Prud'hommeaux *et al.*, 2014] visualized using Turtled. The second view displays the triples as a simple graph using Gephi [Bastian *et al.*, 2009] and the third view highlights key terms present in the text using different colors. While visualizing a large document, with many triples, the resulting graph can be too dense for the user. As different users have different needs from the same piece of text or may interpret the same

[3]https://github.com/mhausenblas/turtled

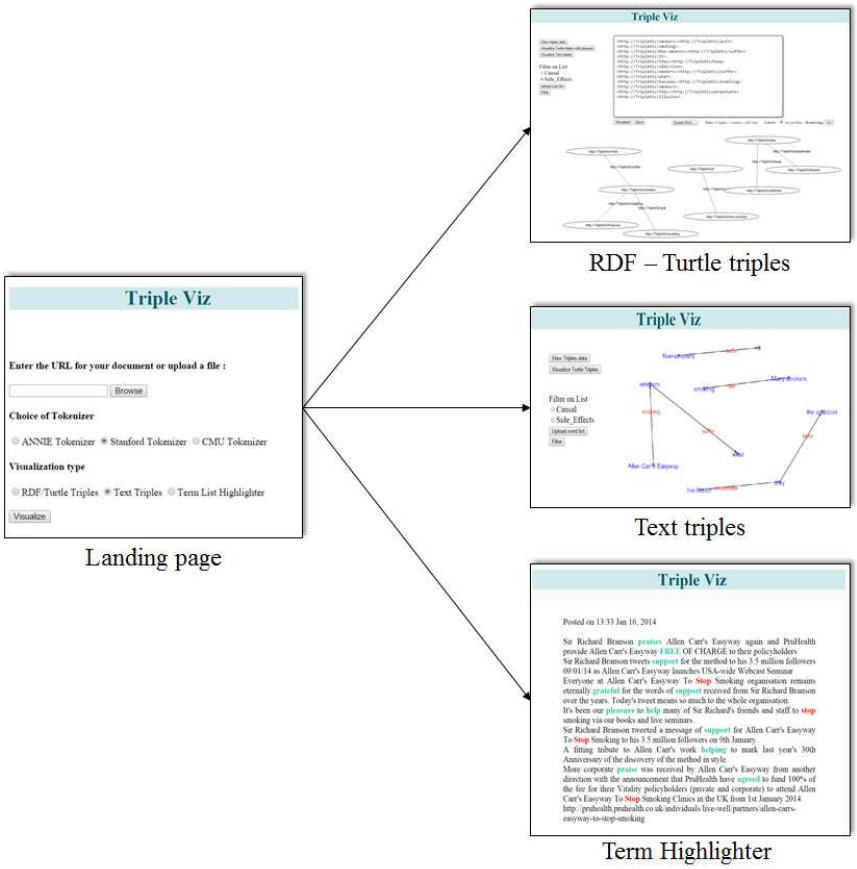


Figure 1.1: TripleViz Overview

piece of text in different ways, we provide the user with an option to select only those triples that contain terms that the user specified in certain word lists.

The thesis mainly expresses the work done to help experts who lack NLP knowledge and coding skills to obtain the context of a large database in a user-friendly environment by generating SVO triples. Two different visuals for the different format of SVO triples are provided which can be replaced by the user as per requirement. Chapter 2 discusses and critiquing some of the existing tools. Data preprocessing and implementation of components

used in TripleViz is explained in Chapter 3. The usefulness of the tool is demonstrated in Chapter 4 by describing an experiment. Chapter 5 explains the integration of the different tools that were used to build TripleViz.

Chapter 2

Related Work

There is an ongoing controversy in the information visualization community over how to determine the impact of visualisation in data understanding [Borkin *et al.*, 2013]. There are hundreds of tools available in today's world for visualization. Some of them may be open-source (such as Gephi [Bastian *et al.*, 2009]) while others are commercial (e.g. Tableau[1]). Some may be for analytics (such as Google Analytics[2]) while some are for data analysis (LODVisualization [Brunetti *et al.*, 2012]). In this chapter, we discuss some of the advantages and disadvantages of these tools by categorizing them into 3 groups; commercial tools available on the market, visualization tools that exist for RDF or linked data, and finally graph visualization tools.

In their work, Borkin et al., suggest that the key feature of visualization is to aid in memorizing/recollecting data [Borkin *et al.*, 2013]. They performe experiments to prove this by collecting visualization data from dif-

[1]http://www.tableau.com/
[2]https://www.google.ca/analytics/

ferent sources such as news media and government reports that were presented to subjects through Amazon Mechanical Turk [Paolacci *et al.*, 2010] in a sequence of images. The Subjects were asked to press a key if they saw an image for the second time in the sequence of data presented. Their experiments illustrate that adding attributes such as color or recognizable objects to the visualization makes them more memorable when compared to data in charts and graphs, hence providing evidence that human cognition, understanding and memory are intertwined [Borkin *et al.*, 2013]. With regard to this, we represent the nodes and edges of the triples in different colors making it easier for the user to understand.

Data Driven Documents (D3) [Bostock *et al.*, 2011] is a library of visualizations built on the JavaScript framework that provides efficient manipulation of documents based on data. It provides a number of JavaScript files for different visualizations that can be integrated easily with the web to create visualization. Many visualization tools including Gephi [Bastian *et al.*, 2009] use D3 to create their visualizations. The user can replace or add one of these libraries into TripleViz with minimal effort.

2.1 Commercial tools for Data Visualization

A commercial tool is computer software that is produced for sale or that serves a commercial purpose. In this section, we will be discussing some of the popular data visualization tools that were created for commercial purposes.

Tableau[3] is a web-based business intelligence visualization tool that

[3]http://www.tableau.com/

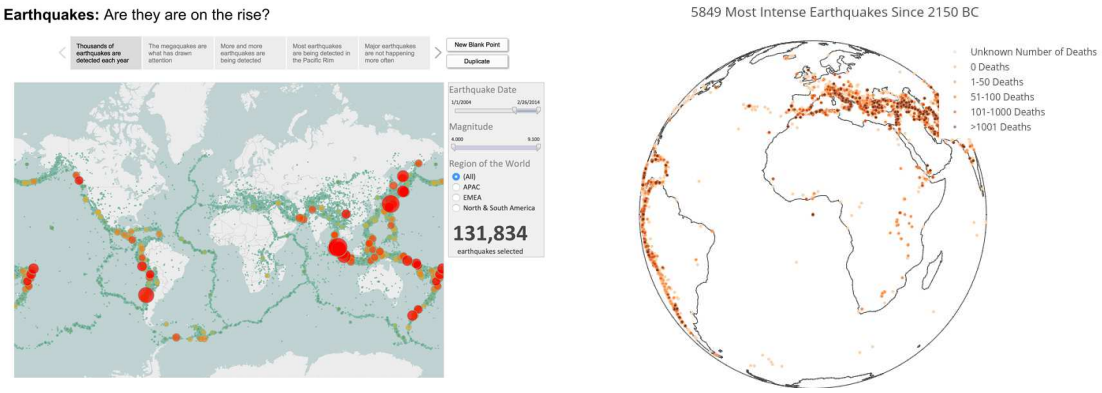
provides fast data analysis and interactive visualizations of large volumes of data. Figure 2.1a[4] shows a visualization obtained from Tableau indicating the regions where an earthquake has occurred over a period of time. Although the users need very little technical knowledge to analyze the output data provided by Tableau, the initial data preparation requires strong technical skills usually done by IT or a consulting organization.

Google Trends [Choi and Varian, 2012], is also a web-based analytics tool useful to monitor patterns and trends to show how often a particular search-term is entered relative to the total search-volume across various regions of the world, and in various languages. The tool uses information retrieval techniques to get the count on the number of hits for the term or phrase but does not go in depth to provide the context of the search terms.

Plot.ly[5] is a commercial visualization tool for companies in aerospace, materials science, energy prospecting, journalism, etc., used mainly for reporting. Plot.ly produces line graphs, bar charts, heatmaps, histograms and many other types of visualizations taken from data files, Dropbox, Google Drive, MS Excel, etc. Figure 2.1b[6] demonstrates regions affected by earthquakes differently from Tableau.

Healthmap [Brownstein *et al.*, 2008] explores large volumes of data obtained from various sources (newspapers, blogs, etc.) in multiple languages to provide a visualization demonstrating the ongoing global disease activity through an online portal. Figure 2.2a[7] shows locations of specific Ebola

[4] http://www.tableau.com/products/desktop
[5] https://plot.ly/
[6] https://plot.ly/~chris/15262/_5849-most-intense-earthquakes-since-2150-bc/
[7] https://publichealthwatch.wordpress.com/2014/08/10/how-a-computer-algorithm-predicted-west-africas-ebola-outbreak-before-it-was-announced/



(a) Tableau

(b) Plot.ly

Figure 2.1: Visualizations from different commercial tools

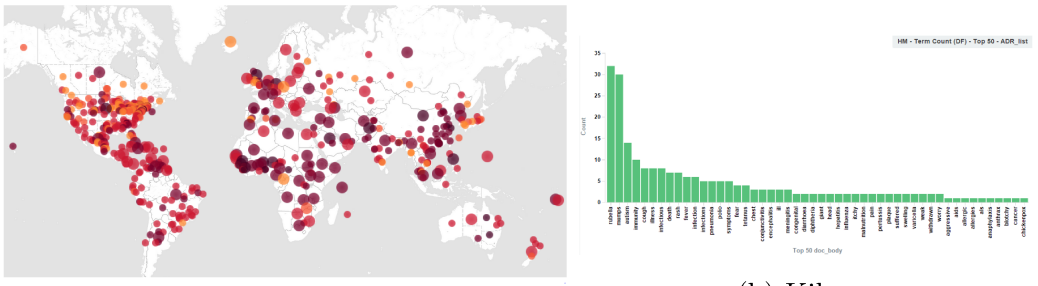
outbreaks detected by Healthmap.

Patterson [Patterson, 2016] recently developed an interactive web-based tool to visualize World Health Organization (WHO) data. The WHO data was in different forms - raw statistical data (numbers and tables), 2D map-based chart data, and the 3DImpact interactive 3D globe. The paper illustrates that the 3D nature of information representation offers the potential to enhance the user's interaction with the data.

Elasticsearch[8] is a real-time search and analytics tool which is built on top of the Apache Lucene[9] search engine library used mainly for indexing data (index to a document acts like a tag through which the information content of the document in question may be identified [Maron and Kuhns, 1960]). Elasticsearch results can be visualized with an open-source tool called Kibana[10]. We used Elasticsearch and Kibana as our first tool to detect the presence of vaccine hesitancy in blogs and news articles. A detailed

[8]https://www.elastic.co/products/elasticsearch
[9]https://lucene.apache.org/
[10]https://www.elastic.co/products/kibana

explanation about the set up of Elasticsearch and Kibana is explained under Appendix B and Appendix C respectively. Figure 2.2b displays a bar chart generated using Kibana showing the document frequency of terms present in the data.



(a) Healthmap

(b) Kibana

Figure 2.2: Visualizations from different commercial tools

The above mentioned tools do not deal with text at all. For example, Tableau and Plot.ly provide visualizations of the regions affected by earthquake over a period of time by identifying mention of places in articles about earthquakes (see Figure 2.1a and Figure 2.1b). Healthmap however, does not specify or display any information regarding the number of people affected by the earthquake, the magnitude of the earthquake, etc but lists the articles from which the information was retrieved. Extracting information of this granularity requires further processing of data using Natural Language Processing tools. One thing common in the above mentioned tools is that they are all web-based tools inspiring TripleViz to be web oriented.

Resource Description Frameworks (RDF) [Klyne and Carroll, 2006] contribute to Linked Data [Bizer *et al.*, 2009a], a method of publishing structured data so it can be interlinked to become more useful through queries. Some of the visualization tools for RDF are explained below.

2.2 Visualization tools for RDF

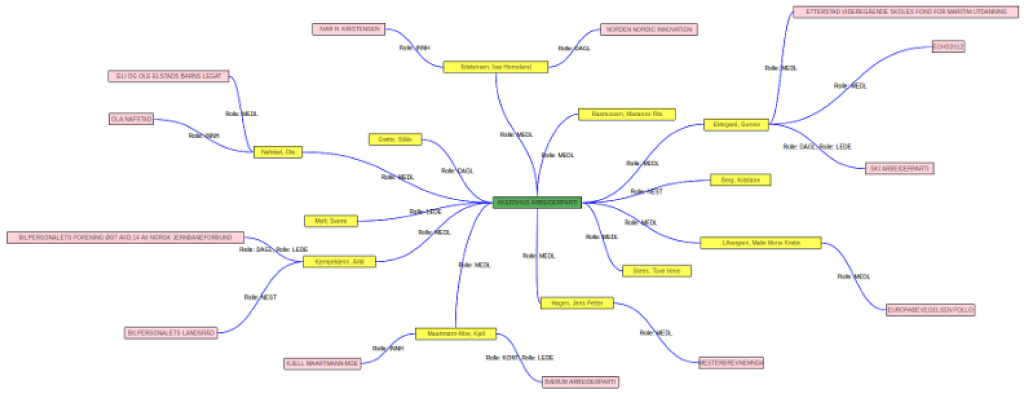
There are many libraries available to generate custom visualizations. LODVisualization [Brunetti *et al.*, 2012] is a linked data visualization tool which allows users to dynamically connect to different data sources such as DBpedia [Bizer *et al.*, 2009b], LinkedMDB [Hassanzadeh and Consens, 2009] and Wine ontology [Noy *et al.*, 2001]. This permits the visualization to display the hierarchy of classes and properties requested by a SPARQL query [Prud'Hommeaux *et al.*, 2008], which is an RDF query language, used to retrieve and manipulate data stored in RDF format. LODVisualization can produce visualizations in forms of treemaps [Johnson and Shneiderman, 1991], tables, bar charts, etc.

Sgvizler [Skjæveland, 2012] is web-based tool built on JavaScript to visualize SPARQL queries. Sgvizler uses HTML5 (a markup language used for structuring and presenting content on the web [Hoy, 2011]) to provide an interactive user interface where the users can query using SPARQL and obtain visualizations. It provides Cross-Origin Resource Sharing (CORS)[11] to support SPARQL [Prud'Hommeaux *et al.*, 2008] to query from external domains. However, Sgvizler requires the user to have previous knowledge of SPARQL making it a tool for experts only. Figure 2.3a[12] shows a graph generated by Sgvizler using the Dracula library[13] displaying a small set of RDF data.

[11] `https://www.w3.org/TR/`

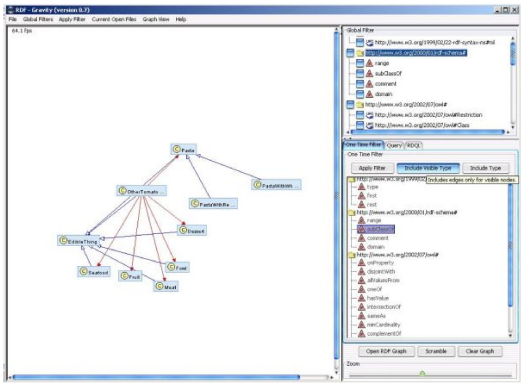
[12] `http://dev.data2000.no/sgvizler/wiki/Sgvizler/Chart/SgvizlerVisualizationDraculaGraph`

[13] https://www.graphdracula.net/



(a) Sgvizler

(b) W3Schools



(c) RDF Gravity

Figure 2.3: Visualizations from different Linked Data Visualization Tools

W3C[14] provides a online tool to validate the RDF syntax as well as visualize it. There is no API provided for this tool so it can't be integrated (see Figure 2.4b). RDF-Gravity[15] is another tool to visualize RDFs. It provides features to filter the graph, query on the RDF and also visualizes multiple RDF files. Figure 2.3c[16] represents the *Include Visible Type* option used for including selected edges to be displayed in the graph. After selecting the nodes to be displayed, the visualization shows only selected edges where both nodes were previously selected.

[14]https://www.w3.org/RDF/Validator/
[15]http://semweb.salzburgresearch.at/apps/rdf-gravity/index.html
[16]`http://semweb.salzburgresearch.at/apps/rdf-gravity/user_doc.html`

RelFinder [Lohmann *et al.*, 2010] extracts relationships between objects in RDF data and provides an interactive view. In particular, it aims to help the separation of relevant relationships from irrelevant ones by offering visual features. It has an easy-to-use interface even for non-experts and is very suitable to get a quick overview of the relationships between objects of interest. LinkedLifeData[17] is an online tool which has access to 25 public biomedical databases containing around 1,553,620,639 entities. It provides services to write complex analytical data queries answering questions such as "give me all human genes located on the Y-chromosome with known molecular interactions". Further, it exports entries with "all approved drugs and their brand name" and displays them using a graph to represent the findings on the query.

The above mentioned tools require that data be processed and stored as an RDF in a database before they can be visualized. They do not provide functionality to generate the database which can cause inconvenience in using the tool. TripleViz on the other hand processes plain textual data to produce RDFs as well as provides a visualization for it. These tools can be plugged into TripleViz to provide visuals for the RDF triples which can be generated using the existing components of TripleViz.

RDF syntax is difficult to read (see Figure 2.4a) which motivated the evolution of N-Triples [Beckett and Barstow, 2001], a plain text format for encoding an RDF graph (see Figure 2.4b). N-Triples are terms representing the subject, predicate and object of an RDF Triple bearing minimal syntax, making them easy to read. The triples need not always be SVO. Triples such

[17]http://linkedlifedata.com/

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:j.0="http://www.tripleviz.org/" >
   <rdf:Description rdf:about="http://www.tripleviz.org/virus">
      <j.0:produces rdf:resource="http://www.tripleviz.org/symptoms"/>
   </rdf:Description>
</rdf:RDF>
```

(a) RDF Representation

```
<http://www.tripleviz.org/virus>
<http://www.tripleviz.org/produces>
<http://www.tripleviz.org/symptoms>
```

(b) N-Triple representation

Figure 2.4: RDF Syntax Representation

as date_of_birth (Justin Trudeau, 1971) indicating Justin Trudeau's date of birth is also a triple. Turtled[18] is a web-based tool built on JavaScript to render N-Triples and is used to display our N-triples in TripleViz (see Section 3.2).

2.3 Graph visualization tools

The tools mentioned in Section 2.2 do not analyze textual data but require structured data. They either have their own database of RDF triples/ontol-

[18]https://github.com/mhausenblas/turtled

ogy or expect the user to provide them with a database. They focus solely on providing visualizations for Linked Data only. Prior to creating an ontology or database we need to process and analyze data. As explained in Section 1.1, visualization is important in understanding and analyzing data. We visualize the SVO triples extracted from our data and use basic graphing tools to visualize them. Some of the visualization tools that provide graphs are discussed under this section.

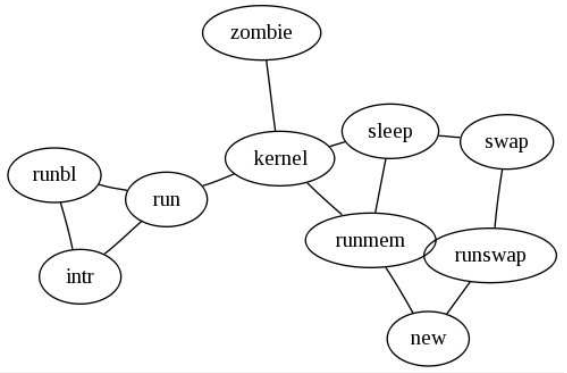
A popular graph visualization tools is Gephi [Bastian *et al.*, 2009]. It is an open-source graph visualization platform which caters to creating any type of graph (see Figure 2.5b[19]). Gephi is one of the tools used in TripleViz to visualize our SVO triples (see Section 3.3). Figure 2.5b shows a network graph for a database with curved edges.

Guess [Adar, 2006] is an exploratory data analysis and visualization tool for graphs and networks from Hewlett Packard. It is built on Jython, a Python programming language designed to run on the Java platform [Pedroni and Rappin, 2002], which binds the text typed and its corresponding visualization for interactive integration.

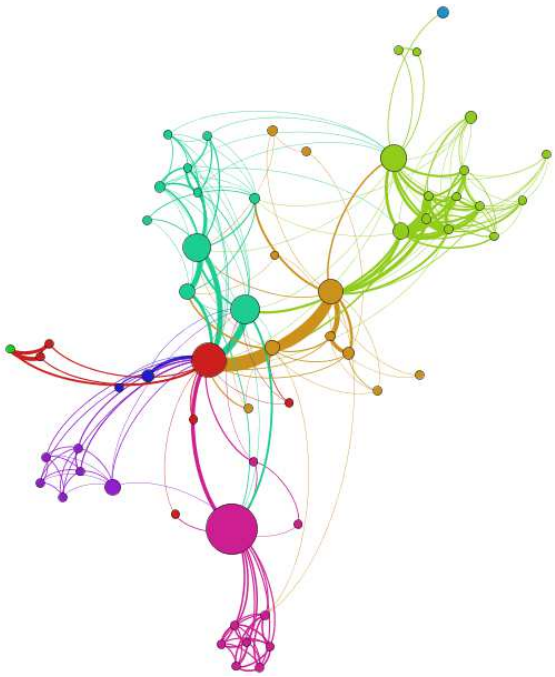
GraphViz [Ellson *et al.*, 2001] is an open-source graph visualization software used in networking, bioinformatics, software engineering, database and web design, machine learning and other technical domains to represent structural data such as text language inputs as graphs, networks or diagrams. It takes simple text language inputs and provides visualizations as images in different formats like PDF (graphics file format for the Web) [Roelofs and Koman, 1999], SVG (a language for describing graphics in XML) [Ferraiolo

[19] `https://gephi.wordpress.com/category/announcement/`

et al., 2003], etc. Graphviz provides the user with many useful features such as options for colors, fonts, tabular node layouts, line styles, hyperlinks, and custom shapes (see Figure 2.5a[20]).



(a) Graphviz



(b) Gephi

Figure 2.5: Visualizations from different Graph Visualization Tools

The tools mentioned above are mostly web-based making them operating system compatible. Tools such as Tableau, Plot.ly, Kibana, etc., provide numerical analysis on data by crunching numbers while other tools such as Sgvizler, RDF Gravity, etc., require pre-tailored data to plug in to their visualization. Buy considering the functionality from the above reviewed tools, we propose a tool named TripleViz, which is reconfigurable and performs primary functions to analyze the given data by providing visualizations for the extracted SVO triples. Chapter 3 explains the procedure adopted to clean or preprocess the input data and extract SVO triples.

[20] `http://www.graphviz.org/content/process`

Chapter 3

From text to triples to graphs

TripleViz is a lightweight visualization tool which takes text or web pages as input, processes it, generates triples and provides us with a visualization. This chapter explains in detail the steps taken to pre-process the text before analyzing it. It also motivates 2 different triple formats and highlights their advantages and shortcomings.

3.1 Pre-processing

Data pre-processing is an important step that needs to be performed on raw data to prepare it for further processing. TripleViz pre-processes texts using the GATE environment [Cunningham, 2002] and certain third-party Stanford tools [Manning *et al.*, 2014]. Stanford tools are regularly further developed, maintained and updated. The advantage of using third-party tools is to avoid the overhead of maintaining them.

Text obtained from the web may consist of advertisements, RSS feeds and images that are not of interest to us. We use Boilerplate [Christian Kohlschütter, 2010] to extract the textual data from the web pages. The textual data is then subjected to tokenization, sentence splitting and parsing to obtain a bunch of annotations of which only a few are considered to generate our Subject-Verb-Object triples. The different tools with their advantages are listed below. The procedure is outlined in Figure 3.1 and described below.

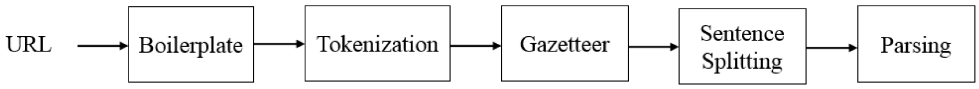
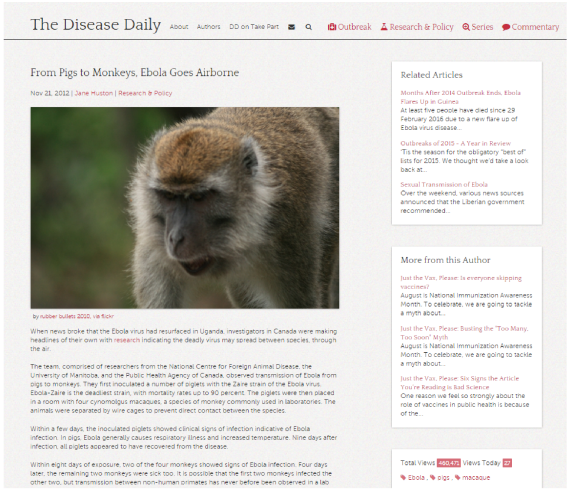


Figure 3.1: pre-processing pipeline for text

3.1.1 Boilerplate

We use Boilerplate [Christian Kohlschütter, 2010] to extract the textual data from web pages which may contain non-textual information like images and advertisements which is insignificant to us. Boilerplate [Christian Kohlschütter, 2010] comes built-in with GATE [Cunningham, 2002] and provides algorithms to detect and remove the unwanted "clutter" around the main textual content of a web page. We use the Article Extractor algorithm distributed with Boilerplate [Christian Kohlschütter, 2010]. Figure 3.2a shows a webpage with images, newsfeed and advertisements which are removed to get only the textual aspect of the page by Boileplate as shown in Figure 3.2b.



(a) Web page with advertisements and images

When news broke that the Ebola virus had resurfaced in Uganda, investigators in Canada were making headlines of their own with research indicating the deadly virus may spread between species, through the air.
The team, comprised of researchers from the National Centre for Foreign Animal Disease, the University of Manitoba, and the Public Health Agency of Canada, observed transmission of Ebola from pigs to monkeys. They first inoculated a number of piglets with the Zaire strain of the Ebola virus. Ebola-Zaire is the deadliest strain, with mortality rates up to 90 percent. The piglets were then placed in a room with four cynomolgus macaques, a species of monkey commonly used in laboratories. The animals were separated by wire cages to prevent direct contact between the species.
Within a few days, the inoculated piglets showed clinical signs of infection indicative of Ebola infection. In pigs, Ebola generally causes respiratory illness and increased temperature. Nine days after infection, all piglets appeared to have recovered from the disease.
Within eight days of exposure, two of the four monkeys showed signs of Ebola infection. Four days later, the remaining two monkeys were sick too. It is possible that the first two monkeys infected the other two, but transmission between non-human primates has never before been observed in a lab setting.
While the study provided evidence that transmission of Ebola between species is possible, researchers still cannot say for certain how that transmission actually occurred. There are three likely candidates for the route of transmission: airborne, droplet, or fomites.
Airborne and droplet transmission both technically travel through the air to infect others; the difference lies in the size of the infective particles. Smaller droplets persist in the air longer and are able to travel farther- these droplets are truly "airborne." Larger droplets can neither travel as far nor persist for very long.
Fomites are inanimate objects that can transmit disease if they are contaminated with infectious agents. In this study, a monkey's cage could have been contaminated when workers were cleaning a nearby pig cage. If the monkey touched the contaminated cage surface and then its mouth or eyes, it could have been infected.
Author Dr. Gary Kobinger suspects that the virus is transmitted through droplets, not fomites, because evidence of infection in the lungs of the monkeys indicated that the virus was inhaled.
What do these findings mean? First and foremost, Ebola is not suddenly an airborne disease. As expert commentators at ProMED stated, the experiments "demonstrate the susceptibility of pigs to Zaire Ebolavirus and that the virus from infected pigs can be transmitted to macaques under experimental conditions… they fall short of establishing that this is a normal route of transmission in the natural environment." Furthermore , because human Ebola outbreaks have historically been locally contained, it is unlikely that Ebola can spread between humans via airborne transmission.
However, the study does raise the possibility that pigs are a host for Ebola. If this proves to be true in the wild, there are direct ramifications for prevention and control measures. It is still unclear what role pigs play in the chain of transmission. To continue work on answering this question, the team plans to take samples from pigs in areas known to have recently experienced Ebola outbreaks.
The Disease Daily has previously reported on Dr. Kobinger's work on the Ebola vaccine .

(b) Text extracted after running through Boilerplate

Figure 3.2: Boilerplate extraction for web page

3.1.2 Tokenization

The text extracted with Boilerplate is subjected to Tokenization, a process of breaking the given text into units called tokens. These tokens may be words or numbers or punctuation marks[1]. There are many tokenizers available of which the ANNIE tokenizer[Cunningham, 2002], the Stanford tokenizer[Manning *et al.*] and the CMU tokenizer[Gimpel *et al.*, 2011] are integrated in TripleViz, giving the user an option to choose the tokenizer that works well with the user's text.

The **ANNIE tokenizer** [Cunningham, 2002] splits the text into tokens to provide features such as numbers, punctuations and different types of words. It uses grammar rules to improve the efficiency and flexibility of the tokenizer. One of the drawbacks of the ANNIE tokenizer [Cunningham, 2002] is that the grammar rules split the tokens on contractions such as It's,

[1]http://language.worldofcomputing.net/category/tokenization

don't, can't, etc instead of maintaining them as a single token. In Example 4, a sentence is tokenized using ANNIE and the tokens obtained are listed below.

(4) I won't go to the store tomorrow if it's raining.
Tokens:
I wo n't go to the store tomorrow if it 's raining .

The **PTBTokenizer** [Manning *et al.*] from Stanford commonly known as the Stanford Tokenizer, provides a resource suitable for tokenization of English text which imitates the Penn Treebank 3 (PTB) [Marcus *et al.*, 1993] part-of-speech (POS) tags, hence its name. Although the PTBtokenizer is efficient, fast and deterministic, it splits the tokens on contractions similar to the ANNIE tokenizer (shown in Example 5).

(5) *I won't go to the store tomorrow if it's raining.*
Tokens:
I wo n't go to the store tomorrow if it 's raining .

The **CMU tokenizer** [Gimpel *et al.*, 2011] is a Java-based tokenizer trained on tweets. Unlike the ANNIE and the PTBTokenizer, the CMU tokenizer does not split the tokens on contractions leaving *can't*, *isn't*, *couldn't*, etc., to remain as single tokens/words as shown in Example 6. However, when these tokens were subjected to the Stanford parser [Klein and Manning, 2003] (see Section 3.1.5), they were tagged with the wrong part of speech. For example, *couldn't* was tagged as a proper noun in the sentence *Kieran couldn't play the piano.*

(6) *I won't go to the store tomorrow if it's raining.*
Tokens:
I won't go to the store tomorrow if it's raining .

In TripleViz, the choice of tokenizer used for the pre-processing is given as an option to the user to choose based on their requirements. However, we use the Stanford tokenizer [Manning *et al.*] as a default when the user does not specify their choice of tokenization.

3.1.3 Gazetteers

Once the tokenization process is complete, the next step is to recognize entities by locating elements in the text and categorizing them using lists of words called gazetteer lists. The gazetteer processing resource annotates the document using a simple string matching technique where separate annotations are created for every gazetteer list which contains a match in the given text. The Gazetteer processing resource is independent of any other pre-existing annotations. Users can create their own gazetteer lists by following a standard template as stated below.

1. Every term/phrase must be placed on a new line
2. There should be no duplicates within a list
3. The document needs to have a .lst extension

We also provide the user with the following pre-defined lists containing single words or phrases:

- Adverse Drug reaction terms [Nikfarjam *et al.*, 2015] with 13699 entries
- FDA Drug Names[2] with 6746 entries

[2]http://www.fda.gov/Drugs/InformationOnDrugs/ucm079750.htm

- AFINN [Nielsen, 2011] with 2477 entries
- FDA Dosage Names[3] with 3586 entries
- Side Effects terms [Kuhn *et al.*, 2015] with 7665 entries

Figure 3.3 shows the annotations obtained after tokenizing and annotating for gazetteer lists.

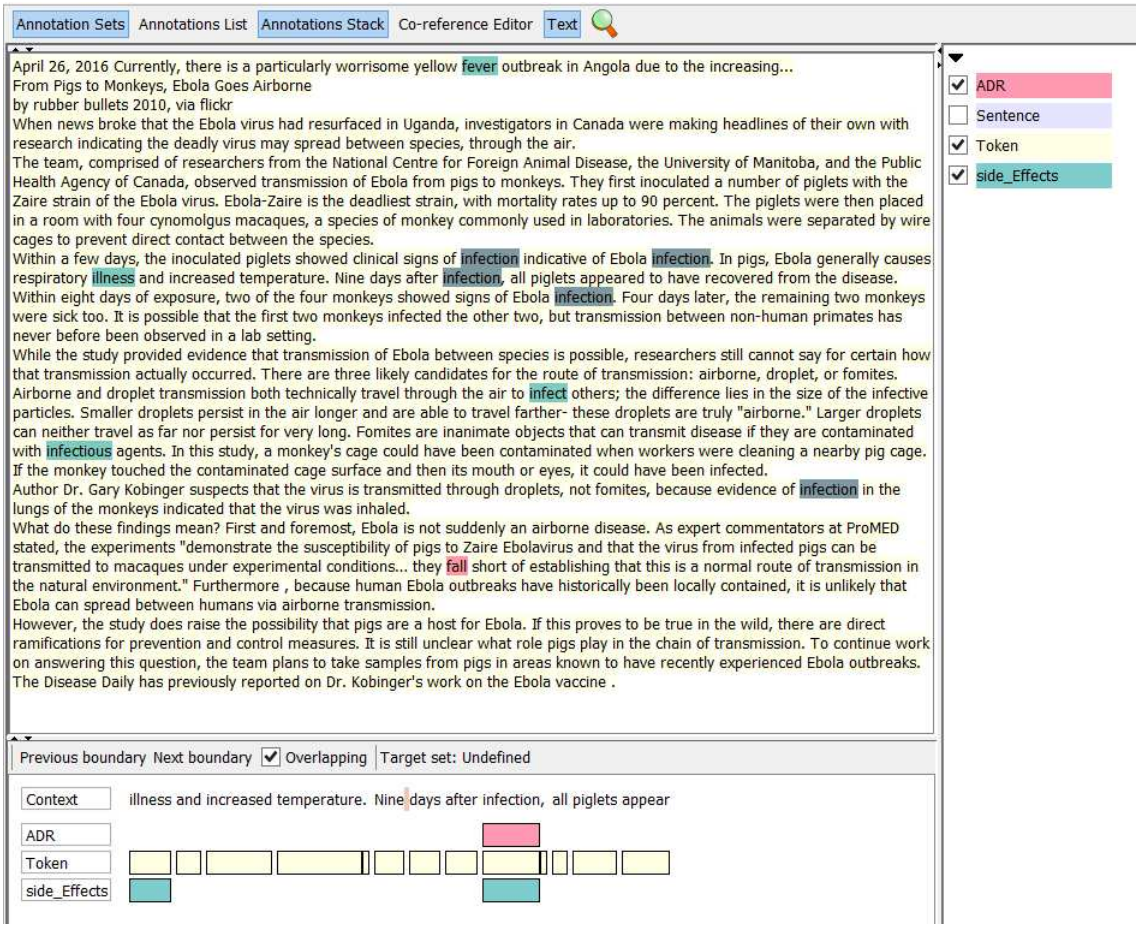


Figure 3.3: Token and gazetteer annotations in GATE

[3]http://www.fda.gov/Drugs/InformationOnDrugs/ucm079750.htm

3.1.4 Sentence Splitting

Sentence splitting combines tokens to form sentences. In English, the punctuation marks that appear at the end of a sentence need not really mark the end of the sentence making it difficult to correctly identify the complete sentence. The period acts as the end of a sentence in most cases but has exceptions as explained in Example 7. There are many sentence splitters such as the ANNIE sentence splitter and Stanford sentence splitter of which we use the Stanford sentence splitter. The sentence splitter module requires the tokens and is therefore executed after the tokenizer.

The **ANNIE sentence splitter** [Cunningham, 2002] is domain and application-independent. It comes built in with Gate and is a part of the default ANNIE pipeline. While testing our data with ANNIE sentence splitter, we notice that the sentences were incorrectly split for quotations. In Example 7, consider the two sentences below,

(7) "Parents don't realise that measles is not just a case of a few spots - it can be very serious illness."Symptoms include fever, cough, soreness of the eyes and a rash which spreads rapidly over the body.

Sentence 1
"Parents don't realise that measles is not just a case of a few spots - it can be very serious illness.

Sentence 2
"Symptoms include fever, cough, soreness of the eyes and a rash which spreads rapidly over the body.

The **Stanford sentence splitter** [Manning *et al.*, 2014] produces

sentences from either plain text or an XML document. The Stanford sentence splitter split quotations appropriately hence, we use the Stanford Sentence Splitter. For the same sentence in Example 8, shows the sentence splitting by Stanford sentence splitter.

(8) "Parents don't realise that measles is not just a case of a few spots - it can be very serious illness."Symptoms include fever, cough, soreness of the eyes and a rash which spreads rapidly over the body.

Sentence 1
"Parents don't realise that measles is not just a case of a few spots - it can be very serious illness."

Sentence 2
Symptoms include fever, cough, soreness of the eyes and a rash which spreads rapidly over the body.

3.1.5 Stanford Parser

The Stanford parser [Klein and Manning, 2003] requires the document to be pre-annotated with tokens, sentences and part-of-speech tags, therefore, we run it at the end of the pipeline. The parser determines the grammatical structure of a sentence by deciding which group of tokens go together to form a phrase and which tokens would be the subject or the object of a verb. The Stanford Parser is a lexicalized probabilistic parser and provides a parse tree (see Figure 3.4) and derives universal dependencies. It also adds part-of-speech (POS) tags such as noun, adjective, preposition, verb, modals, etc., to the tokens. The parser uses Penn Tree bank [Marcus *et al.*, 1993] part of speech annotation, that provides sophisticated tags such as JJR, which indicates an adjective in comparative form and JJS which indicates

an adjective in its superlative form giving a clear separation between the two types of adjectives. The Stanford dependencies depict the grammatical relations between tokens/words within a sentence [de Marneffe and Manning, 2008]. Example 9 shows the dependencies and the parse trees obtained from the Stanford parser.

(9) *In most people the Zika virus produces no symptoms at all.*[4]

Dependencies :

```
amod (people , most)        prep_in (produces , people)
det (virus , the)           nn (virus , Zika)
nsubj (produces , virus)    dobj (produces , symptoms)
advmod (produces , at)      neg(symptoms , no)
pobj (at , all)
```

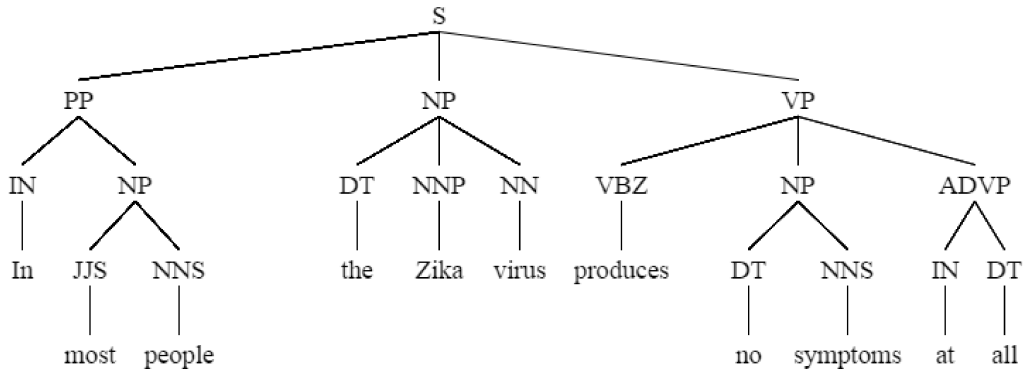


Figure 3.4: Stanford dependencies and constituent parse tree

Among the dependencies listed in Example 9, the nsubj and the dobj dependencies provide the context of the sentence and we select only these.

The Stanford Parser provides different types of dependencies of which we choose the Typed Collapsed Dependencies [de Marneffe and Man-

[4] http://www.cjad.com/cjad-news-quebec-beyond/2016/01/29/first-case-of-zika-virus-detected-in-quebec

ning, 2008] as they can be easily understood by people without linguistic expertise. The difference between universal dependencies and TypedCollapsed dependencies is shown in Appendix A.

The Typed Collapsed dependencies [de Marneffe and Manning, 2008] are also extracted as triples indicating the relation between a pair of words. Dependencies are created from parse trees created by Stanford parser [Klein and Manning, 2003]. The parse tree provides the grammatical information about the sentence such as Noun Phrases, Verb Phrases, SBAR, etc. An example of the Syntax Tree is shown in Figure 3.4.

3.2 RDF Triples

The Semantic Web, is an effort to annotate the web to interlink data and relationships among data in RDF format. RDF is a domain-neutral framework to describe any Internet resource. After pre-processing the data in TripleViz, the resulting subject-verb-object triples are transformed into Resource Description Framework (RDF) [Klyne and Carroll, 2006] using Uniform Resource Identifiers (URI) [Masinter *et al.*, 2005] to designate the ends and the relationship between the ends in a triple. An easy-to-understand visual representation of the RDFs can be through a directed graph, where the edges represent the link between two resources. Figure 3.5, shows SVO triples from a sentence represented as RDF.

(10) *In most people the Zika virus produces no symptoms at all.*

The RDF syntax in Figure 3.5 is not very easy to understand.

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:j.0="http://www.tripleviz.org/" >
   <rdf:Description rdf:about="http://www.tripleviz.org/virus">
      <j.0:produces rdf:resource="http://www.tripleviz.org/symptoms"/>
   </rdf:Description>
</rdf:RDF>
```

Figure 3.5: RDF Syntax

```
<http://www.tripleviz.org/virus>
<http://www.tripleviz.org/produces>
<http://www.tripleviz.org/symptoms>
```

Figure 3.6: An example of N-Triples

Hence, we represent our triples in N-Triple format [Beckett and Barstow, 2001], a textual format of RDF which is easily understandable by non-domain experts (shown in Figure 3.6).

For data analysis and visualization we generate the SVO triples in N-triples format (also referred to as Turtle triples) and move on to visualizing them using a tool called Turtled[5]. Example 11 shows the process undergone by the text from the pre-processing stage to the visualization process.

[5]https://github.com/mhausenblas/turtled

(11) *The Ebola virus is transmitted by mosquitoes in hot and humid climates and symptoms include joint pain, rash and fever but are usually mild, and those infected often are not aware they have the virus.*

Dependencies:

det (virus , the)	nn (virus , Ebola)
nsubjpass (transmitted , virus)	**agent (transmitted , mosquitoes)**
conj_and (transmitted , aware)	auxpass (transmitted , is)
conj_and (transmitted , include)	prep_in (mosquitoes , climates)
conj_and (hot , humid)	amod (climates , hot)
nsubj (include , symptoms)	**dobj (include , pain)**
conj_but (include , mild)	amod (pain , joint)
conj_and (pain , rash)	conj_and (pain , fever)
cop (mild , are)	advmod (mild , usually)
det (infected , those)	nsubj (aware , infected)
advmod (aware , often)	cop (aware , are)
neg (aware , not)	ccomp (aware , have)
nsubj (have , they)	**dobj (have , virus)**
det (virus , the)	

N-Triples :

<http://tripleviz/**mosquitoes**>

<http://tripleviz/**transmitted**>

<http://tripleviz/**virus**>.

<http://tripleviz/**symptoms**>

<http://tripleviz/**included**>

<http://tripleviz/**pain**>.

<http://tripleviz/**they**>

<http://tripleviz/**have**>

<http://tripleviz/**virus**>.

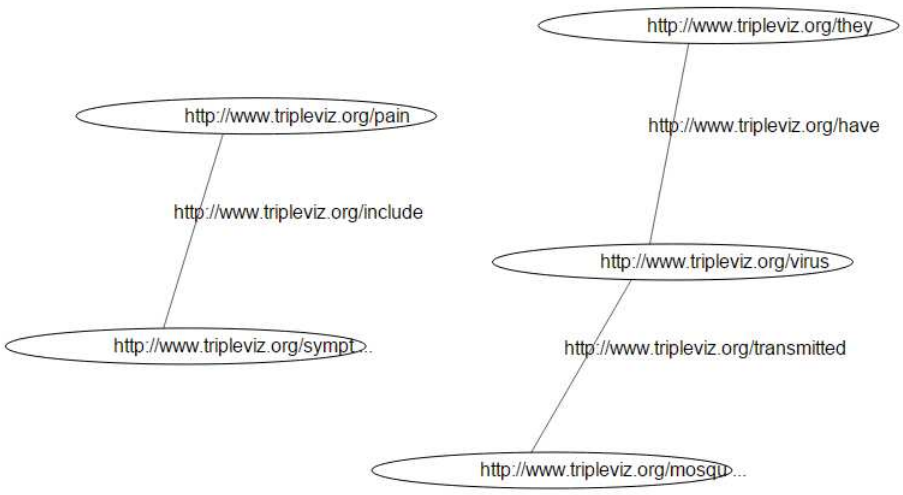


Figure 3.7: N-Triples visualized in Turtled

From the mentioned dependencies, we choose the 6 triples in bold, that are the SVO triples. These triples are then transformed to N-triple format and visualized using Turtled (see Figure 3.7). Turtled allows us to save and export the graph as SVG with a lot of potential for expansion. The integration of Turtled with TripleViz is explained in Chapter 5.

Figure 3.7, shows certain nodes where the text is blocked by the ellipse surrounding it. The URI preceding each triple takes more space than the data, making it illegible. For example, in the triple `include (pain , symptoms)`, the ellipse around *symptoms* is cutting off the text from being displayed entirely. The triples need to be represented as URIs, and as a result we cannot get rid of the http://www.tripleviz.org/ that precedes the token *symptoms*. This is one of the reasons we provide the user another option of visualizing the triples as plain textual triples. We use the http://www.tripleviz.org/ preceding the triple entities as a placeholder that can be easily replaced with a domain name connecting a database (such as

dbpedia). N-Triples are represented using directed graphs, where the arrow runs from the subject to point to the object. Turtled does not support directed graphs.

We replicate the process of extracting SVO triples from a full fledged document[6] and transform the SVO triples to N-Triples format. We visualize the same text as shown in Figure 3.8 only to obtain a graph that is dense and overwhelming and very difficult to read. The document containes 163 sentences that generated 618 dependency triples of which only 95 triples were of SVO type.

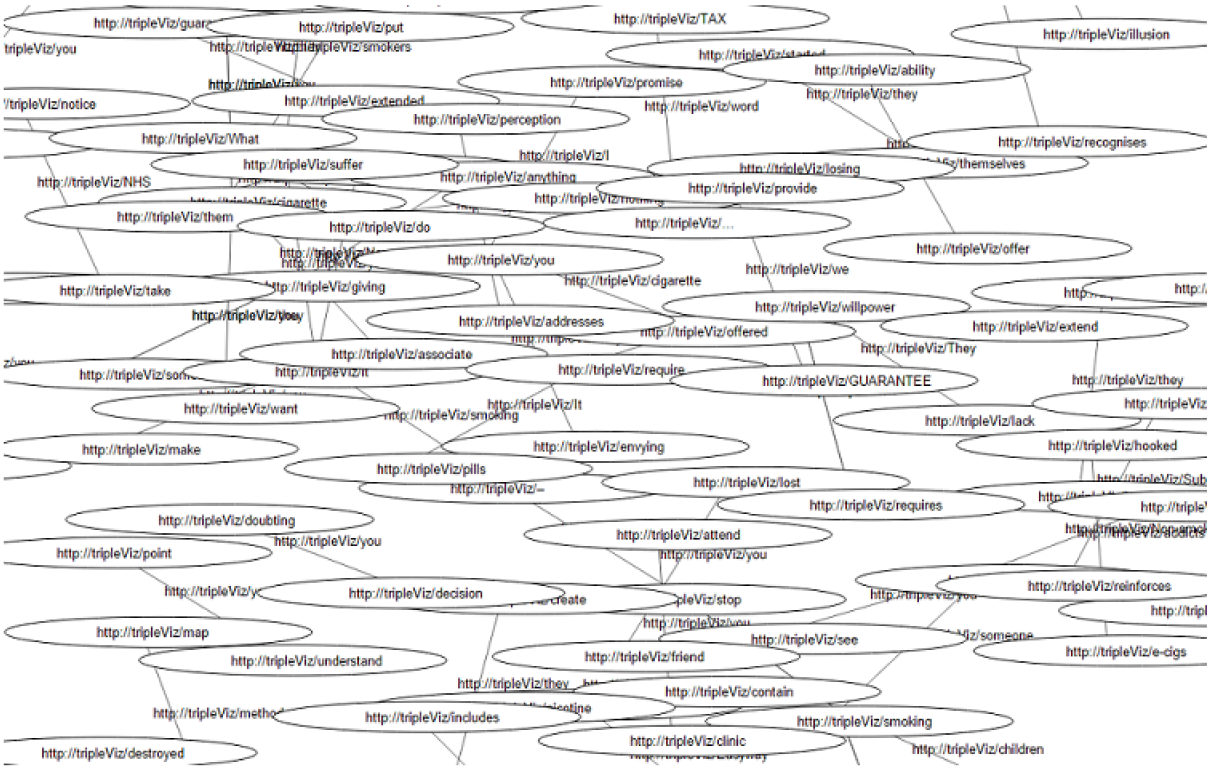


Figure 3.8: Turtle triples for http://www.allencarr.com/category/blog/

The excess clutter around the N-triple and the inability to generate

[6]http://www.allencarr.com/category/blog/

directed graphs made us look at the triples in a different way. We use a different visualization tool named Gephi [Bastian *et al.*, 2009] into TripleViz and in addition represent only textual triples without the URI as explained under Section 3.3.

3.3 Textual Triples

The textual triples are simply SVO triples. Unlike the N-Triples, we do not have to format the textual triples to adhere to a different syntax. To overcome the inability of Turtled to produce directed graphs, we use a different visualization tool named Gephi [Bastian *et al.*, 2009]. Gephi is an interactive visualization tool that facilitates visualization of all kinds of networks along with graphs that may be directed, undirected, weighted, un-weighted, labeled, un-labeled, etc. Some of the reasons why we choose Gephi are listed below.

- Gephi is compatible with various operating systems such as, Windows, Linux and Mac OS X.
- Gephi supports various types of input (directed, undirected, mixed graphs, etc.)
- Gephi supports many network visualization algorithms
- Gephi provides dynamic filtering
- Gephi exports to many different formats (PDF, SVG, PNG, etc.)

Since Gehpi is a desktop application, we use the API provided to integrate it with our tool to make it web compatible. To visualize the graph in Gephi, we export our triples to GEXF [Heymann *et al.*, 2009] format, which is similar to XML. We visualize the same triples from Figure 3.8 using Gephi (see Figure 3.9) but without the URI and with added arrows. Figure 3.9 already seems to look legible and the triples can now mostly be distinguished from one another. To arrange the triples in a circular form we use Fruchterman Reingold's [Fruchterman and Reingold, 1991] network visualization algorithm.

We integrate a module to generate and display the textual triples in TripleViz as one of the visualizations.

Consider Example 12,

(12) *In most people the Zika virus produces no symptoms at all.*
`produces (virus , symptoms)`

where the triple does not clarify details such as the type of virus or what symptoms providing an incomplete meaning of the sentence. Replacing these tokens with their respective noun phrases adds meaning to the triple providing a better understanding. The tokens that are currently present in the triple are the heads of the noun phrases. For ease of use, we call the triples with the head noun as Head triples and the triples with noun phrases as NP triples.

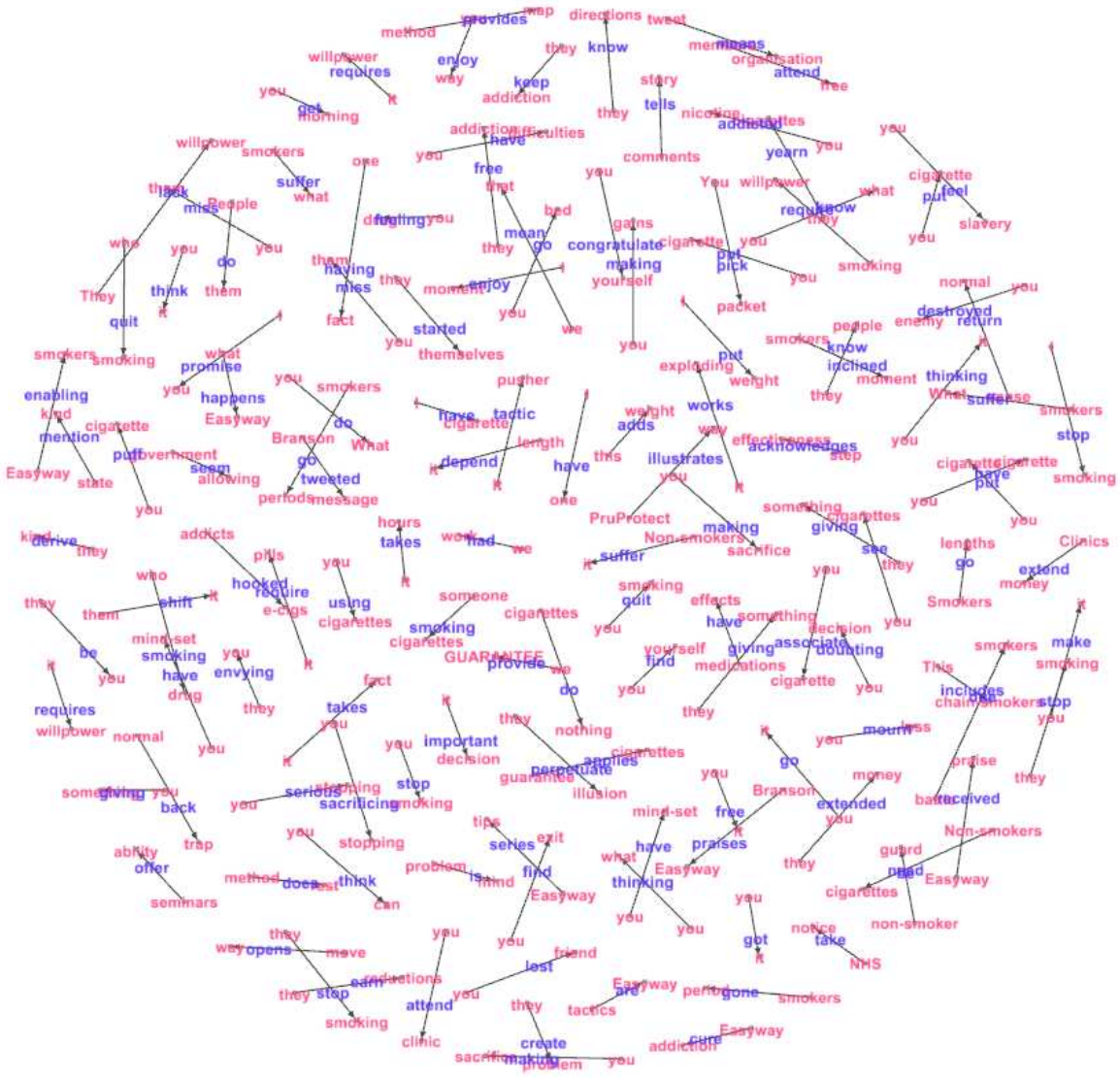


Figure 3.9: Textual Triples visualized using Gephi

In Example 13, *virus* is the head of the noun phrase *the Zika virus* and *symptoms* is the head of *no symptoms* giving us the following triples.

(13) *In most people the Zika virus produces no symptoms at all.*
Head triples : `produces (virus , symptoms)`
NP triples :`produces (the Zika virus , no symptoms).`

The noun phrases are obtained from Stanford parser [Klein and Manning, 2003] and we extract only the first level of noun phrases to form the NP triples. In a parse tree, the NP that is closer to the leaves is the first level of NP. In Figure 3.10, the first level noun phrases are marked in red with 1 and the second level noun phrase is marked in red with the number 2. First level NPs would be *The boy, the red shorts* and *the ball.* First level NP (The boy) and the prepositional phrase (with the red shorts) combine to form second level NP (The boy with the red shorts). For the NP, *The boy*, *boy* is the head noun.

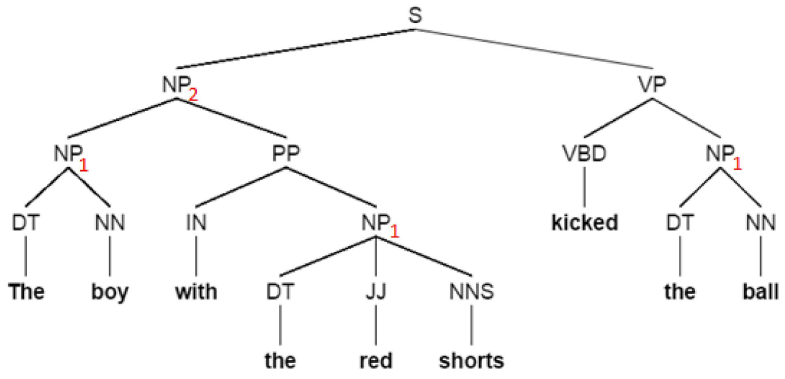


Figure 3.10: Levels of noun phrases in a sentence

The graph in Figure 3.11 contains the noun phrases extracted for all the 95 textual triples present in the document. The graph generated by Gephi is an SVG and does not permit user interaction. The graph with the addition of the NPs becomes ponderous and illegible. Trying to capture every piece of information in this view becomes counter-productive. Hence, we chose to filter the triples using lists of words of interest to the user. Section 3.4 explains in detail how we filter the triples and discuss the advantages and disadvantages of our method.

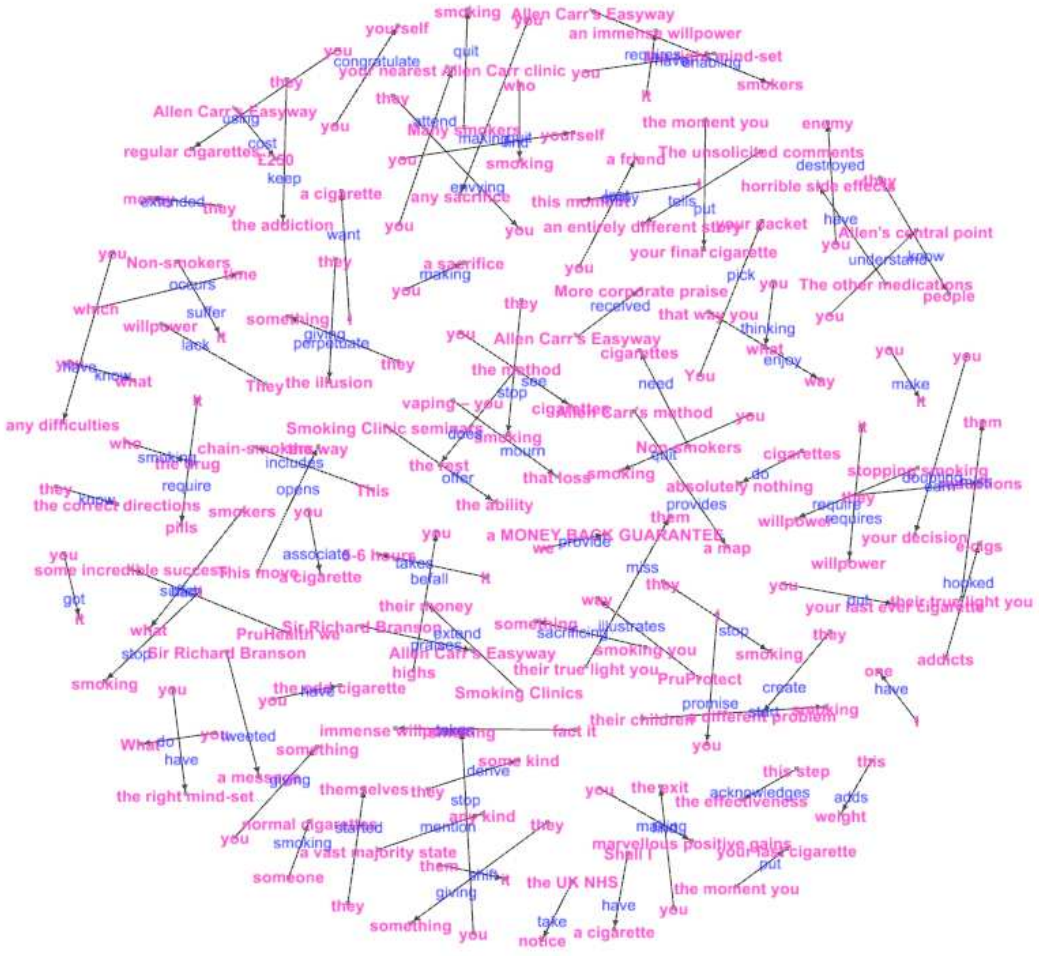


Figure 3.11: NP Triples with Noun Phrases visualized using Gephi

3.4 Filtering Triples

Different people may have different uses for the same data. So we provide the user with the option to give a list of words that are of interest using our web-interface on TripleViz. The list is then used to filter and display only those triples that have either vertex from the list. If two or more word lists are provided by the user, then only the triples that have at least one vertex from any of the lists are displayed. While mining for NPs, we expand only vertexes from the filter lists to NP format. Graph 3.11 obtained with textual NP triples was too dense and overwhelming. Adding the functionality of filtering through word lists to reduce the number of triples in the visuals provides the user with a better picture that contains only the triples of interest.

While looking into newspaper data for vaccine hesitancy, one factor that we felt could be interesting are the side effects that people may have mentioned. TripleViz includes a list of terms for common side effects [Kuhn *et al.*, 2015] such as headaches, drowsiness, etc. The side effects list produced by SIDER [Kuhn *et al.*, 2015] has 7665 terms and phrases. Our document[7] contains 62 instances from the side effects list and of the original 95 SVO triples from Figure 3.11, only 6 triples remain after filtering against the side effects list. Figure 3.12 shows the filtered NP triples visualized using Gephi and Figure 3.13 shows filtered NP triples visualized using Turtled.

The graph in Figure 3.12 shows only 6 triples for 163 sentences, there is thus a likelihood of missing some crucial information. To overcome this, we provide the user an option of selecting more than one list to filter

[7]http://www.allencarr.com/category/blog/

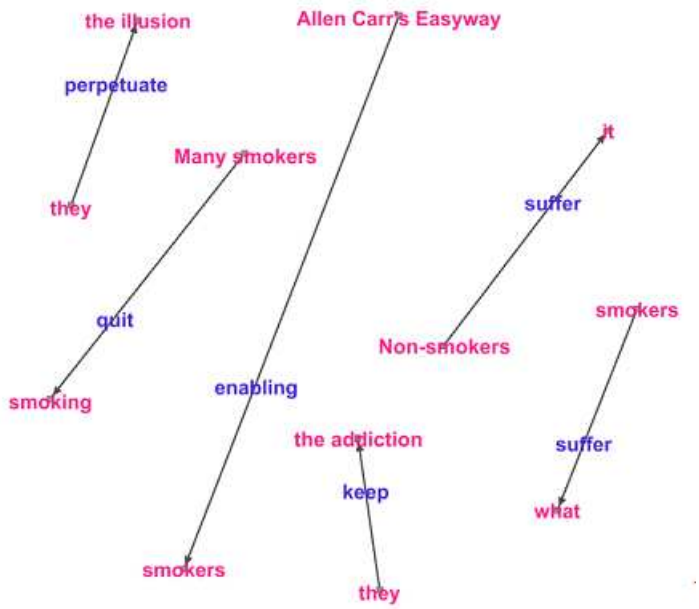


Figure 3.12: NP Textual Triples filtered by term list visualized by Gephi

the triples. TripleViz provides the user with a visualization that depicts the head triples extracted from the document in the form of N-triples and (see Figure 3.13) in text format. It also provides the user with an option to view the NP triples in N-triple as well as textual format.

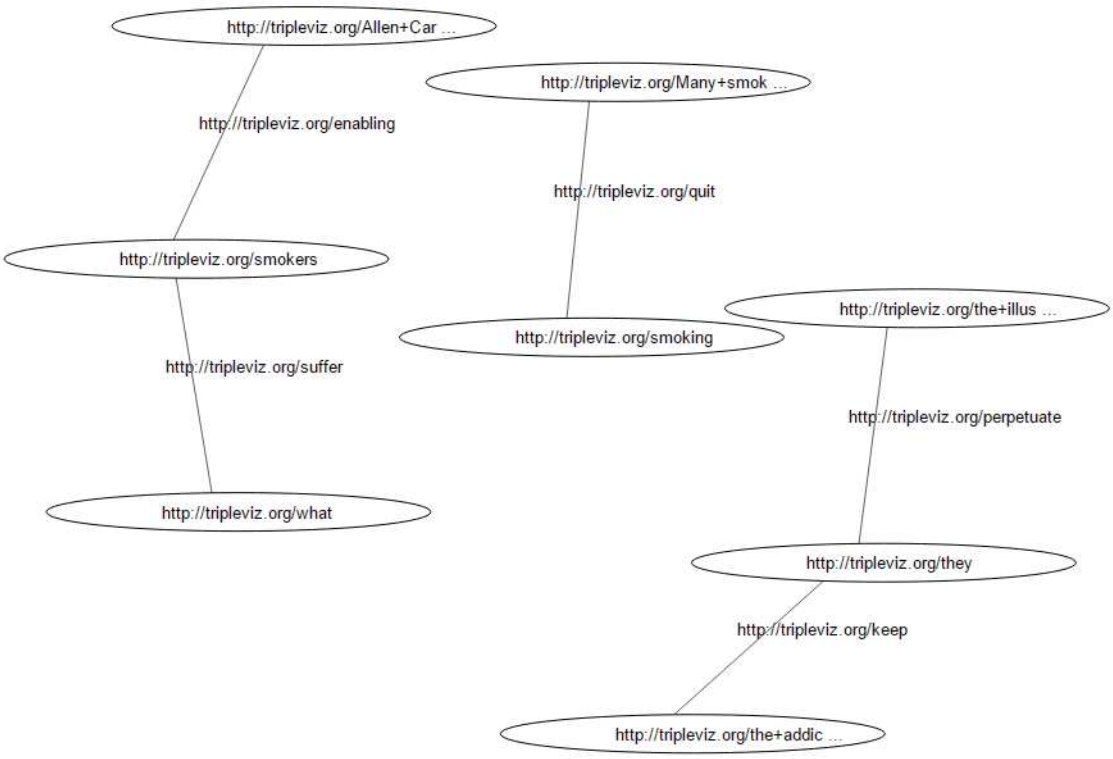


Figure 3.13: NP Textual Triples filtered by term list visualized by Turtled

TripleViz also provides the user with a visualization displaying the text of the document, where the words from the user list are highlighted in different colors. The text Example 25 is highlighted with ADR (blue) and Side effects terms (red). If a word is a part of both the lists, then we highlight it by merging the colors (purple). Section 5.5 addresses the implementation of this visualization.

(14) TripleViz Presentation:

Recent headlines from across North America make it clear that some vaccine-preventable illnesses are making a comeback. First there was an outbreak of mumps in professional hockey teams, then over 100 cases of measles in 14 US states linked to the California Disneyland theme park, and now unrelated cases in Toronto, southern Ontario and Quebec. The majority of those infected had not been vaccinated against the disease. There was also a report of a vaccine-free day care as well as an Ottawa family of 7 (unvaccinated) children, all of whom contracted whooping cough!

Chapter 4 discusses the usefulness of SVO triples by a classification task and provides a brief analysis of the results.

Chapter 4

Is this useful?

Most tools are evaluated by comparing their results with related work in the field [Gediga *et al.*, 2002]. These comparisons can be made to test the performance of the system and the quality of the result by using standard techniques. Web-based tools are evaluated for their design, readability, and user-friendliness by using testing softwares like Selenium[1], which helps testers write test cases [Bruns *et al.*, 2009]. Integrating two or more tools into a web-based application requires the system to be tested more on functionality than on performance. The advantage of using third-party tools is that it reduces the overhead of maintaining them. Since all modules (Tokenizer, Sentence Splitter, Parser) used by our pre-processing system are third party tools, we conduct a small experiment to assess the usefulness of the combination of the various components used under TripleViz.

[1] http://www.seleniumhq.org/

4.1 Experiment : Specific-Event task

300 Healthmap articles were manually read and annotated for specific events by two epidemiologists, G and K (our domain experts) by looking at the entire document. A specific event for example could be "boy in Iowa dies after flu shot", rather than just discussion about beliefs on flu shots. The triples in Example 16 represent specific events that talk about a specific person.

(15) *Triples potentially indicating specific events:*
"My daughter developed epilepsy since being vaccinated, and when I share her experience with people, most doubt the connection," reader Nina Kenney wrote.
`developed(My daughter,epilepsy)`

The infant developed a fever after being vaccinated in the afternoon and died in the night.
`developed(The infant,a fever)`

Ms Stephen was deafened by a vaccine carrying the botched Urabe strain of mumps, which was later withdrawn.
`deafened(a vaccine,Ms Stephen)`

(16) *Triples that do not indicate specific events:*
In the recent anti-polio drive in the country, about one million children missed their inoculation.
`missed(about one million children,their inoculation)`

The school believes that children should get their health check up done from their family doctors.
`get(children,their health check)`

These 300 articles were treated as our gold standard. We tested the usefulness of SVO triples generated by TripleViz to approximate the context of the document for the task of classifying whether the document

concerns a *specific event*. To extract the text from these 300 Healthmap URLs, we used Boilerplate [Christian Kohlschütter, 2010]. We were able to retrieve text for only 187 articles due to firewall obstructions, expired links, etc., of which 30 were marked for specific events in our gold standard. For readability, we filtered the triples from the 187 documents for Person Names, Date or presence of family relationships (see Appendix E), which would be the baseline idea for simple Information Retrieval. 5 annotators (including G and K who authored the gold standard) performed the "specific event classification task". 93 of the 187 documents did not pass through the filters (i.e. did not contain Name, Date or Family Relationship terms in their triples), thus not generating any output for the document. Out of the 93 documents that did not generate any output, 13 were marked for specific events in our gold standards, lowering our potential performance. We have discussed the effect of the results in Section 4.1.2. We tabulate the results in two parts, the first part of the table contains results for all the documents (even those with no triples) and the second part contains results for only those 94 documents that generated an output. Both tables denote standard performance measures [Christopher D. Manning and Schütze, 2008] represented as follows:

- **Precision** : how many of the returned documents are correct
- **Recall** : how many of the correct documents does the model return
- **F-measure** : harmonic average of precision and recall

Section 4.1.1 defines the experimental setup whose results are further analysed under Section 4.1.2.

4.1.1 Experimental setup

1. **Data** : 300 Healthmap articles.

2. **Experiment** : Annotate 187 accessible Healthmap articles as concerning a specific event or as being generic by looking at SVO triples (see Example 16) filtered on specific lists.

3. **Gold Standard** : 300 Healthmap articles annotated by G and K for specific events looking at the entire document.

4. **Flow of Data** : The data runs through Boilerpipe (see Section 3.1.1), Tokenizer (see Section 3.1.2), Sentence Splitter (see Section 3.1.4), Parser (see Section 3.1.5) and a module that generates SVO triples (see Section 5.2.8).

5. **Filters applied** : Person name, Family Relationship and Date (from ANNIE plugin [Cunningham, 2002]).

6. **Input** : URL of 300 Healthmap articles.

7. **Output** : SVO triples filtered by word lists for 94 articles plus 93 empty output cells.

8. Four annotators plus the authors of the gold standard (G and K) categorize the articles into "specific event" or "generic" based purely on the extracted SVO triples.

   Table 4.1 shows the results in terms of Precision, Recall and F-measure.

	All documents			Non-empty output		
Name	Precision	Recall	F-measure	Precision	Recall	F-measure
G	0.35	0.46	0.39	0.35	0.82	0.49
K	0.24	0.17	0.0.2	0.24	0.30	0.27
B	0.31	0.50	0.38	0.31	0.88	0.45
A	0.29	0.43	0.35	0.29	0.76	0.42
N	0.27	0.37	0.31	0.27	0.65	0.38
H	0.19	0.23	0.21	0.19	0.41	0.26

Table 4.1: Performance Measure for experiment defining specific event

4.1.2 Analysis

Table 4.1 shows the performance measures for the task of identifying documents with specific events considering only the SVO triples. Figure 4.1 represents Table 4.1 in a graphical format and plots the performance of all 6 subjects on first, all the documents of the gold standard and second, the documents with nonempty output. We observe that the precision remains the same but, because 13 documents that were annotated for specific events in the gold standard did not output any results, recall improved by an average of 28%.

Recall rates are acceptably high for a high-level, vaguely defined task at ca 64%. Note that the gold standard author, G only achieves 82% recall, which is a decent upper bound for the informativeness of the SVO triple representation of documents. Precision is much lower and shows that the loss of contextual information in the highly abstracted, lossy triple view is serious: the gold standard author, G achieves precision of 35% only and K achieves precision of 24% only. This suggests that the vague task of identifying specific events is a reasonable approximation of exploratory browsing of a corpus in general and that the triple view and strong filters suggest a

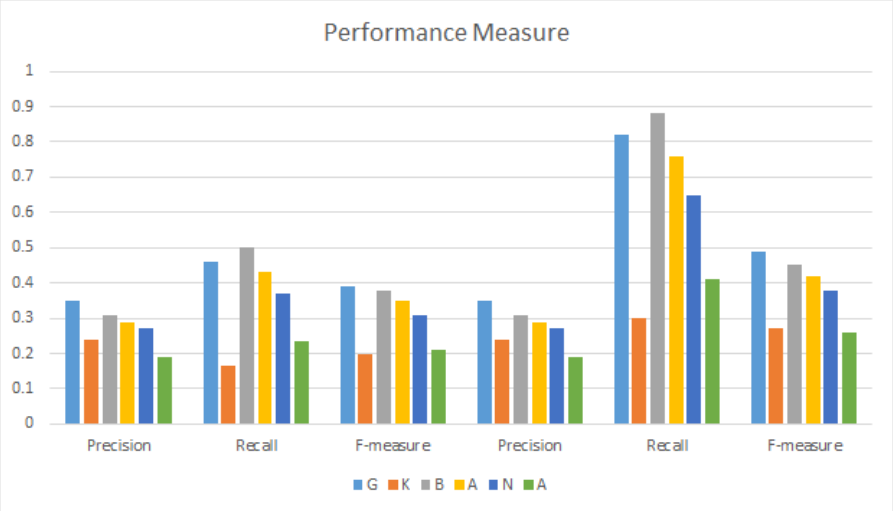


Figure 4.1: Precision, Recall and F-measure

heuristic approach of low accuracy. But note that the imagined goal of this tool is in selecting a seed list of documents for more careful inspection which will lead to the identification of better filter terms and more refined heuristics or even a broader basic information retrieval step. Such mixed strategy approaches are well known to benefit this type of unfocused browsing, see for instance scatter-gather [Christopher D. Manning and Schütze, 2008].

As mentioned above, the data was provided to us in the form of URLs where each URL represented a document. From the 300 documents in our gold standard, only 187 documents (62%) could be retrieved due to invalid links and firewall obstruction. Storing data in the form of URLs can be volatile, this calls for better methods to store or pass data. We considered 187 documents that had a valid URL to be our entire dataset. 30 documents from the entire dataset (16%) were marked for specific events in the gold standard. The entire dataset generated 3007 SVO triples. After passing the 3007 SVO triples from 187 documents through the Person, Date

and Family relationship filters we obtain 265 triples. 93 documents from the entire dataset (i.e; 49%) contained triples that did not pass through the filters of which 13 documents were marked for specific events in the gold standard. Note that for 50% reduction in size, TripleViz incurred only a 29% reduction in recall, which attests to the high predictive value of the filter lists employed. Below we discuss in detail the effect of passing the triples through a filter.

Filtering triples

Table 4.3 represents the number of triples obtained for the entire dataset (187 documents) as well as for documents marked for specific events (30 documents) and non-specific (157 documents) events when passed through each filter. Our baseline approach involved calculating standard performance measure for filter lists applied on triples (shown in Table 4.2). We discuss the impact of each of these lists below.

Filter list	Precision	Recall	F-measure
Person name	0.27	0.33	0.30
Date	0	0	0
Family relationship	0.19	0.46	0.27

Table 4.2: Baseline: Performance measure for filter lists

Document type	All documents	Specific event	Non-specific event
Number of documents	187	30	157
Number of triples	3007	307	2700
Triples filtered by Person Names	74	17	57
Triples filtered by Date	4	0	4
Triples filtered by Family Relationship	187	34	153
Triples filtered by all three filter lists	265	51	214

Table 4.3: Triple distribution over filter lists

Family relationship

From Table 4.3 we observe that 34 of 51 triples (67%) generated for 30 specific event documents are contributed by filtering through the family relationship list and 153 of 214 triples (71%) for 157 non-specific documents. The contribution of family relationship varies by 4% from specific and non-specific documents. Table 4.2 shows that 46% of the documents that contain triples contributed by family relationship are identified correctly. The low precision of 19% for family relationship in Table 4.2 indicates that most of the documents marked for specific event in the gold standard were classified incorrectly.

Person Name

From Table 4.3 we inspect that 17 of 51 triples (33%) generated for the 30 specific event documents are contributed by filtering through person name list and 57 of 214 triples (27%) for 157 non-specific documents. The contribution of person name varies by 6% from specific and non-specific documents. Table 4.2 shows that only 33% of the documents that contain triples contributed by person names are identified correctly. The precision for person name in Table 4.2 is higher than the precision produced for family relationship indicating that the documents marked for specific event containing triples with person names were classified more accurately(27%) than compared to family relationship (19%).

Since, the contribution of the person list varies by only 6% from specific and general documents and the contribution of family relationship varies by only 4%, it is difficult to classify the documents based on just the Person and the Family relationship list.

Date

From Table 4.3, we learn that 4 out of 3007 triples passed through the date list filter. None of the 4 triples belonged to the a document containing specific event. 3 of the 4 triples were due to the word "yesterday" as shown in Example 17 and one triple because of a year as shown in Example 18. Example 17 and 18 display the document link and the sentences for the triples. Since, none of these triples are from the documents marked for specific events it produces an f-measure of 0% as shown in Table 4.2. Thereby making the Date filter not a very good list to mark specific events. Since TripleViz generates output by filtering through the list, the user has to take measures to make sure that the list provided is effective to mark specific events.

(17) *Ms Wilyman yesterday expressed sympathy for the McCaffery family.*
`expressed(yesterday,sympathy)`
URL:http://www.illawarramercury.com.au/story/270790/vaccination-a-human-rights-issue-judy-wilyman/

My daughter had the vaccine yesterday (Polio), it was at about 2pm yesterday.
`had(My daughter,the vaccine yesterday)`
URL: http://www.mothering.com/forum/373-selective-delayed-vaccination/1357159-my-daughter-having-bad-normal-reaction-polio-vaccine.html

Justice Kerian Nwankpa of the Abia State High Court, Umuahia, yesterday restrained the Chief Judge of the state and the President, Customary Court of Appeal, Abia State.
`restrained(yesterday,the Chief Judge)`
URL: http://guardian.ng/?option=com_content&view=articleid=94686:why-polio-still-finds-sanctuary-to-hide-&catid=72:focus&Itemid=598

(18) *The WHO has also recommended that the 2012 to 2013 flu vaccines include protection against the H1N1 strain.*
`include(the 2012,protection)`

URL: http://www.nhs.uk/news/2012/07July/Pages/Swine-flu-deadly-condition-claim.aspx

In Example 18, the year *2012* has a POS tag *CD* given to it by the Stanford parser indicating a number. The Stanford parser uses a rule $NP \rightarrow$DT CD which basically says a phrase that contains a determiner (DT) followed by a number (CD) is considered as a Noun Phrase. In Example 18 the parser does not handle conjunction correctly and applies this rule, when in fact the NP should have been *the 2012 to 2013 flu vaccines* with head noun *vaccines*. Hence, we conclude that the date filter is ineffective in the subject-verb-object triples, because occurrences of the date components in the subject or object position is mostly in error. Below we provide a brief description of the results of our experiment by dividing them into groups for better understanding.

Documents in gold standard identified correctly by all annotators

2 documents were correctly judged by all 6 annotators as denoting specific events. The triples from these documents are listed in Example 19.

(19) *Kaitlyn had a pre-existing metal allergy.*
`had(Kaitlyn,a pre-existing metal allergy)`
Kaitlyn by this point has quit all her activities as she is in too much pain.
`quit(Kaitlyn,her activities)`
URL: http://www.nhs.uk/news/2012/07July/Pages/Swine-flu-deadly-condition-claim.aspx

Injection: Katie had the measles, mumps and rubella jab when she was just 15-months-old
`had(Katie,the measles, mumps and rubella jab)`
Katie Stephen was left deaf by an MMR jab
`left(an MMR jab,Katie Stephen)`
A woman has won her fight to prove she was left deaf by the MMR jab – only the second time it has been linked to disability.

`won(A woman,her fight)`
But a medical assessment panel ruled Katie Stephen, 21, will not receive compensation because she is not considered disabled enough.
`receive(Katie Stephen,compensation)`
URL: http://www.dailymail.co.uk/health/article-2190391/Panel-rules-MMR-jab-girl-deaf–payout.html?ito=feeds-newsxml

Both these documents contained at least one triple with Person name. 30 documents of the entire dataset had triples with Person names of which 10 documents were marked to define a specific event in our gold standard and 2 documents were marked correctly by all 6 annotators.

Documents in gold standard identified incorrectly by all annotators 3 documents were falsely labelled by all 6 annotators. 14 triples were generated by these 3 documents. 9 of 14 triples contained person names and 5 of 14 triples contained terms from family relationship list.

The triples for 3 out of the 6 documents do not contain bio-medical terms. This concludes that Guido only ever looked for specific events with respect to bio-medical terms.

(20) *Congressman Burton used this hearing to rehash a series of some of the most thoroughly discredited anti-vaccine positions of the past decade.*
`used(Congressman Burton,this hearing)`
His organization urges parents not to vaccinate their children, and giving him such a prominent platform only serves to spread misinformation among parents of young children.
`urges(His organization,parents)`
And Burton went off the deep end with this.
`went(Burton,the deep end)`
It wasn't so bad when a child gets one or two or three vaccines.
`gets(a child,one)`
URL: http://genome.fieldofscience.com/2012/12/congress-holds-anti-vaccine-hearing.html

Our concerns that Claire had been damaged by the vaccination and her hospital stay were dismissed by doctor after doctor albeit one doctor suggested that it was possible that Claire had suffered a psychological trauma in hospital.
`damaged(the vaccination,Claire)`
`suffered(Claire,a psychological trauma)`
Claire had a frightening response to the jab and our General Practitioner sent Claire to hospital where she stayed for a week.
`sent(our General Practitioner,Claire)`
My son twenty months earlier had experienced the very same reaction although not sent to hospital.
`experienced(son,the very same reaction)`
Claire had a frightening response to the jab and our General Practitioner sent Claire to hospital where she stayed for a week.
`had(Claire,a frightening response)`
In April I learned that one of the top UK Vaccine Advisors had informed a mother of a patient that Thiomersal was going to be ended.
`informed(one,a mother)`
My own daughter has poor central vision and watches TV through peripheral glances at the screen.
`has(My own daughter,poor central vision)`
URL: http://www.ageofautism.com/2012/11/mercury-vaccines-and-autism.html

"We strongly condemn all acts, methods and practices of terrorism in all their forms and manifestations," the leaders, which included Obama and Chinese president Xi Jinping, said in an end-of-summit declaration.
`included(which,Obama)`
The Islamic State group was also a top concern when Obama met newly-elected Canadian counterpart Justin Trudeau on the sidelines of the summit.
`met(Obama,newly-elected Canadian counterpart Justin Trudeau)`
Obama, meanwhile, reiterated his demand that Syria's civil war would only end if Russia-backed Bashar al-Assad left power.
`reiterated(Obama,his demand)`
URL: http://www.brecorder.com/general-news/172/1248488/

Specific documents classified incorrectly by G

Three documents were classified incorrectly to be non-specific by G while they were classified as specific in the gold standard. The documents and

their triples are iterated below. All 6 triples that were obtained as output for the three documents contain terms from the Family relationship list. These triples produce more generalized context than specific.

Document 1 provided by the URL (as stated in Example 21) produces a single triple when filtered through the lists. The sentence and the triple are as shown in Example 21.

(21) *The children had symptoms including swollen lymph nodes and abscesses, and 50 of the 115 children have been hospitalized since March, the Stockholm-based ECDC said, citing Romanian media reports.*
`had(The children,symptoms)`
URL: http://actmedia.eu/daily/bloomberg-news-danish-made-tb-vaccine-sickens-115-children-in-romania/43298

Document 2 provided by the URL (as stated in Example 22) produces a single triple as well when filtered through the lists. The sentence and the triple are as shown in Example 22.

(22) *Director of Immunisation Professor David Salisbury said: "It is important that parents get their child vaccinated against measles, mumps and rubella - all of which are highly infectious.*
`get(parents,their child)`
URL: http://www.telegraph.co.uk/news/health/news/9521728/Rogue-strain-of-MMR-vaccine-caused-deafness.html

Document 3 provided by the URL (as stated in Example 23) produces four triples when filtered through the lists. The sentence and the triple are as shown in Example 23.

(23) *Since this report, the true extent of this tragedy is coming to light, as parents of these vaccinated children have reported yet more injuries.*

`reported(parents,yet more injuries)`

"We wish that our children would get their health back," shared the parent of a sick child.
`get(our children,their health)`

The children were not seen by the only doctor in the region until a full week after their injuries!
`seen(the only doctor,The children)`

Those children also suffered hallucinations and convulsions.
`suffered(Those children,hallucinations)`
URL: http://www.prisonplanet.com/minimum-of-40-children-paralyzed-after-new-meningitis-vaccine.html

Specific documents classified incorrectly by K

25 documents were classified incorrectly to be non-specific by K while they were classified as specific in the gold standard. The documents and their triples in Appendix F. 13 of 25 documents did not pass through the filters generating no output. 14 documents contained at least one triple with terms from family relationship and 9 documents contained at least one triple from the person name list. 7 of the 25 documents contained triples with terms from both family relationship and person name list.

From the analysis above we observe that the data provided in the form of URLs is elusive and leads to loss of data. This may be due to various reasons such as broken or dead links and/or unauthorized links that require permission, etc. For instance in example 24, the URL was labelled to contain a specific event but the document could not be retrieved because of a broken link.

(24) http://www.hindustantimes.com/Punjab/Chandigarh/ Two-month-old-boy-dies-after-vaccination/SP-Article1-

Hence we conclude that URLs are not the best way to store data and other methods such as computer files or databases need to be adapted for data storage. Also, the baseline approach to identify the context of the documents in a corpus as large as 187 documents proved to be effective yielding a success rate of 49% f-measure.

Chapter 5 introduces the architecture of TripleViz and also the different procedures followed to integrate the visualizations tools such as Turtled and Gephi.

Chapter 5

Development of TripleViz

TripleViz is a light-weight tool that mainly uses open-source third-party applications to provide the user with an extensible interface to interact with textual data. Tools like Gate [Cunningham, 2002] or even visualization tools such as Turtled[1] and Gephi [Bastian *et al.*, 2009] can be technically challenging and may require in depth knowledge just for installation. Hence, for ease of use, TripleViz is implemented as a web application with a user friendly interface. The implementation of these tools is hidden from the user, displaying only the resulting visualizations through different web-pages.

5.1 Web Architecture

A Web application can be accessed by the users through a Web browser from many devices, even those of reduced functionality such as ipads or

[1] https://github.com/mhausenblas/turtled

smart phones. HTML and CSS (Cascade Styling Sheets that are used to add presentation such as color, font size, etc., to the HTML pages) are used to build the interface of TripleViz. The Web server returns HTML pages depending on the request made by the user and displays them on the web.

TripleViz contains three layers in its architecture, namely presentation layer, business layer, and data layer. The presentation layer is used to accept user data (such as the document to be analyzed), which is passed on to the business layer where all the SVO triples are generated and are stored as XML in the data layer. The presentation layer uses the XML stored in the data layer to display the visualization to the user. A detailed explanation of the layers and their functionality in TripleViz are explained below.

5.2 Presentation Layer

The Presentation layer provides user interaction with the system bridging into the core business logic encapsulated in the business layer. A presentation layer may include HTML [Hoy, 2011], CSS [Lie *et al.*, 2005], Servlets (a Java programming language class used to extend the capabilities of applications hosted by the web server by means of a request-response programming model) [Moss, 1999], JavaScript (a dynamic programming language) [Ford, 1998], etc. A presentation layer is used to provide an input/output interface to the user. TripleViz uses its presentation layer to accept a user document and display the SVO triples for these documents in N-triple and textual form.

5.2.1 Landing page

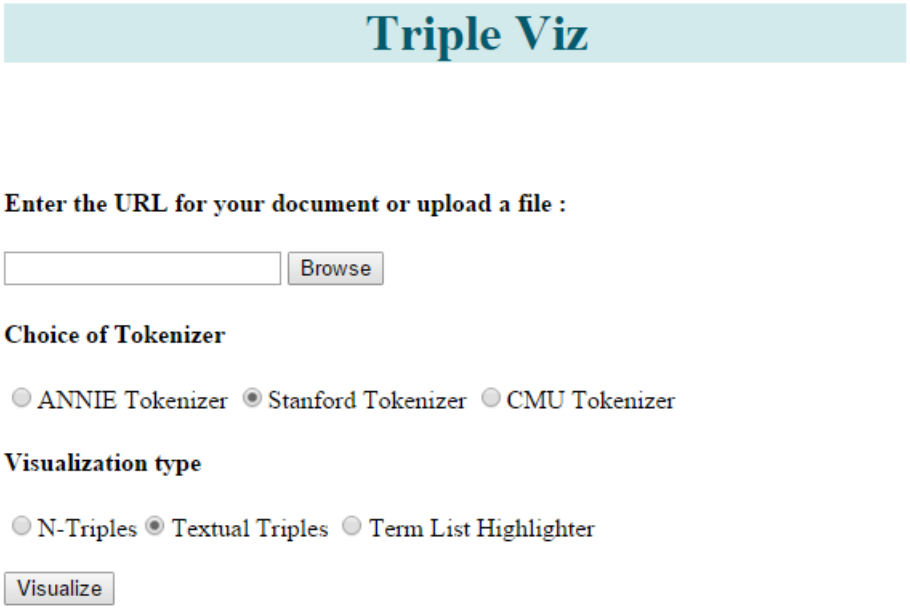


Figure 5.1: Landing page

Figure 5.1 is an HTML page, styled using CSS which provides the user with the interface to TripleViz. The landing page is the only static page in TripleViz. All the other pages are loaded dynamically using servlets. It is here that the user specifies the document to be analysed and chooses the tokenizer and the type of visualization. After providing the document and choosing the tokenizer and the visualization, the user clicks on the *Visualize* button. The Web server takes control of this request and directs the request to the MainServlet (refer to Section 5.2.2).

The document can be a simple text document or an XML file and could be present locally on the users system or a web-page that the user may be interested in. In the former case, the user can use the *Browse* button

to select the files on his system to upload. For the latter, the user simply enters the URL of the web-page in the text area provided. As explained in Section 3.1.2, we provide the user with an option to choose the tokenizer. By default, we use the Stanford tokenizer. Chapter 3 outlined the different types of visualization provided by TripleViz. The *N-triples* use Turtled[2] to generate a graph while the *Textual Triples* use Gephi [Bastian *et al.*, 2009]. The *Term list highlighter* which is independent of all other modules, displays the text of the document and highlights different word lists in different colors. The landing page makes calls to the MainServlet which is explained in detail under Section 5.2.2.

5.2.2 Main Servlet

The MainServlet controls the navigation of the application by making calls to other servlets to render the required HTML pages. Different servlets are called for different visualizations, as the requirements to render these visualizations differ largely from each other. For example, Turtled uses JavaScripts while Gephi has a Java API. Figure 5.2 shows the workflow of MainServlet. The MainServlet calls the NTriple servlet, TextualTriple servlet or the TermHighlighter servlet depending on the visualization type selected. Before these servlets are called, two Java classes from the business layer, GateInit and OutputGenerator (explained under Section 5.2.7 and Section 5.2.8) are invoked. Every time a request is made from the landing page, the MainServlet is called. Different servlets and their functionality are explained in detail below.

[2]https://github.com/mhausenblas/turtled

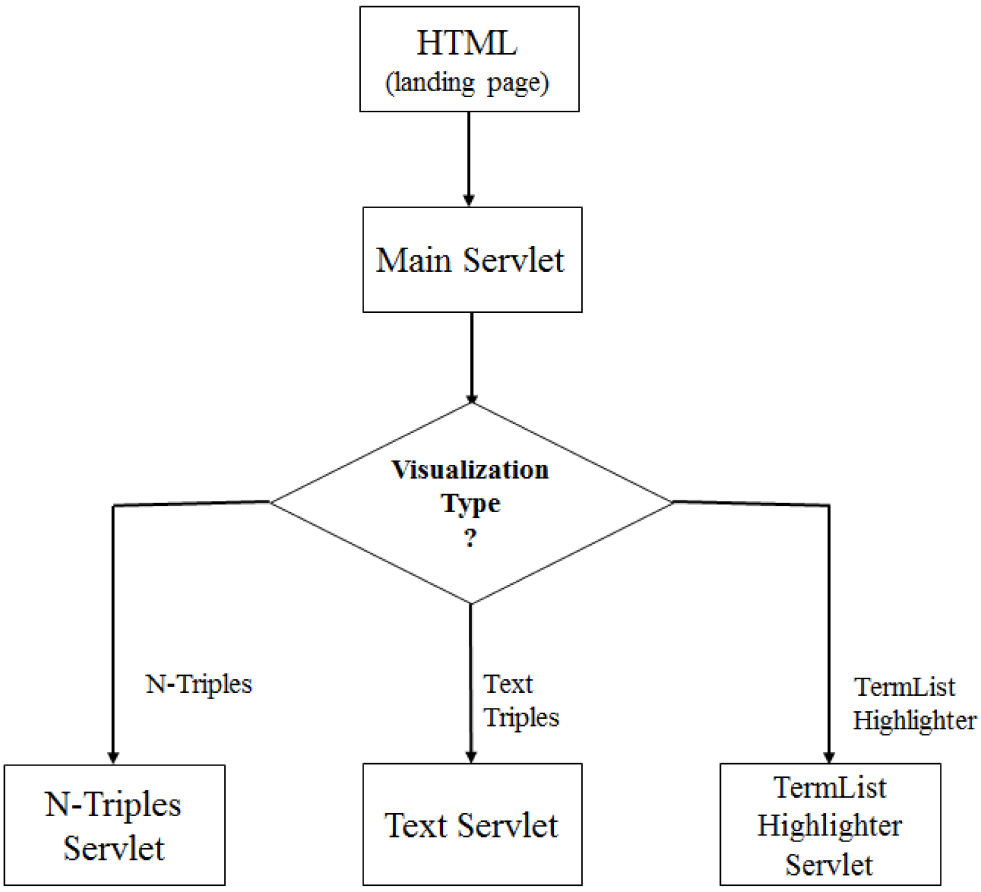


Figure 5.2: MainServlet Workflow

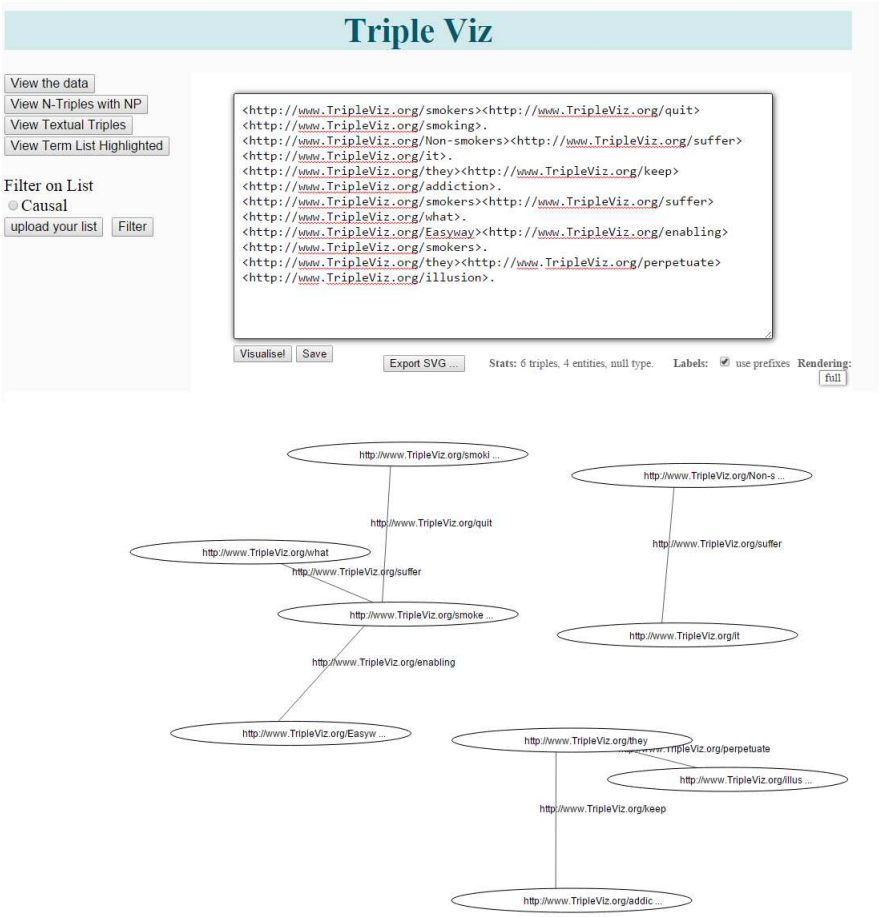
5.2.3 Servlets to render N-Triples

The NTriple servlet takes care of producing a visualization only for the RDF style N-Triples. Turtled uses a number of JavaScript files to produce the visualization added to the application. The NTriple servlet links these JavaScript files to the HTML page rendering the N-Triples. Since, we embed Turtled into TripleViz, we have to take care of making all the function calls on be-

half of Turtled. Turtled has a text area where the N-Triples are displayed. It also has a button that triggers the visualization of these N-triples. The trick was to insert the N-Triples that are generated by the OutputGenerator (explained in Section 5.2.8) into the text area. Since this needs to be done dynamically, we use servlets and not HTML.

As explained under Section 3.4, TripleViz provides an option to filter the triples based on a word list. We choose to display the triples that are obtained from the document before we provide the filtering option. When the user selects an option to filter these triples the OutputGenerator method from the business layer is called to generate an XML file that has only the filtered triples. The NTriple servlet then goes and reads this file and refreshes the page to display a new visualization. TripleViz also comes preloaded with 5 word lists mentioned earlier in Section 3.1.3.

An option to switch from head triples to NP triples is also provided. When the user switches to visualize the NP triples, a different servlet named NTriple_NP is invoked, to keep from swamping the NTriple servlet. The NTriple_NP follows the same workflow as the Ntriple servlet with a slight difference being the output file read (explained under Section 5.2.8) to populate the text area in Turtled. There is an option to switch from NTriples to Textual triples or view the highlighted terms. There is a button on the navigation panel, *View the data*, which shows all the triples along with their sentences at the bottom of the page. Figure 5.3 shows the view generated for N-Triples in TripleViz. The next section talks about the integration of textual triples into TripleViz.



View Triples data

Many smokers quit smoking still believing that they derive some kind of pleasure or benefit from cigarettes.
quit(smokers,smoking)

Non-smokers do not suffer it.
suffer(Non-smokers,it)

Substitutes that contain nicotine, i.e. so-called Nicotine Replacement Therapy patches, gums, nasal sprays and inhalators are particularly unhelpful as they simply keep the addiction to nicotine alive.
keep(they,addiction)

What's more, it's what smokers suffer all their smoking lives.
suffer(smokers,what)

Allen Carr's Easyway has been enabling smokers to get free since 1983.
enabling(Easyway,smokers)

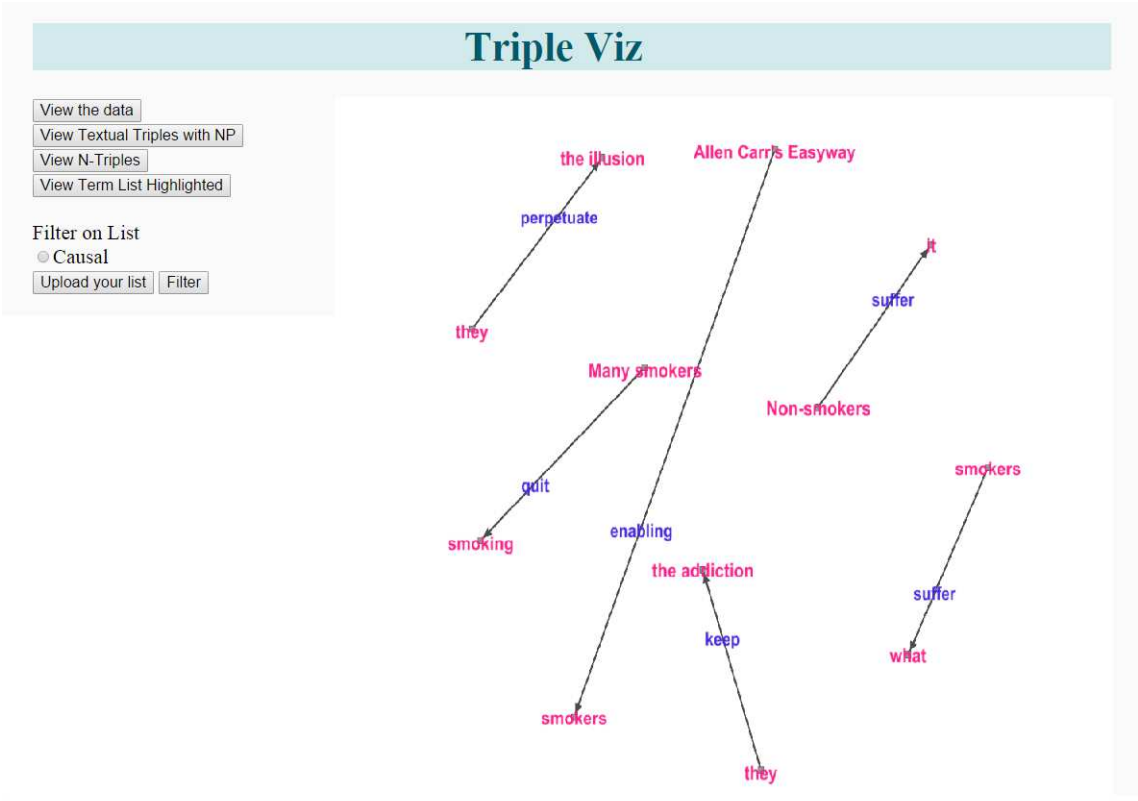
They all make it more difficult to stop because they perpetuate the illusion that you're making a sacrifice.
perpetuate(they,illusion)

Figure 5.3: NTriples in TripleViz

5.2.4 Servlets to render Textual Triples

The TextualTriple servlet is invoked when the user chooses to view the Textual triples either from the landing page or from the NTriple page. The TextualTriple servlet performs the same navigation functions as the NTriple servlet but the visualization is conceived in a very different way. The NTriple servlet uses Turtled to generate a graph using JavaScripts that is loaded by the NTriple servlet. On the contrary, the TextualTriple servlet uses Gephi [Bastian *et al.*, 2009] to visualize the triples. Since, Gephi has an API, we create a class named GephiGraphGenerator (see Section 5.2.9) in the business layer, to make calls to this Gephi API. The GephiGenerator is responsible for generating the visualization of the textual triples in the form of SVG (XML based images) which, the TextualTriple servlet displays on the web-page.

Similar to NTriples, we use a separate servlet named TextualTriple_NP to display the texual triples with Noun Phrases. The TextualTriple_NP servlet uses a different SVG file generated by GephiGraphGenerator that holds NP triples. Figure 5.4 shows the page rendered by the TextualTriple servlet for NP triples. The following section explains how the words from the term list are highlighted.



View the data

Many smokers quit smoking still believe that they derive some kind of pleasure or benefit from cigarettes.
quit(Many smokers,smoking)

Non-smokers do not suffer it.
suffer(Non-smokers,it)

Substitutes that contain nicotine, i.e. so-called Nicotine Replacement Therapy – patches, gums, nasal sprays and inhalators – are particularly unhelpful as they simply keep the addiction to nicotine alive.
keep(they,the addiction)

What's more, it's what smokers suffer all their smoking lives.
suffer(smokers,what)

Allen Carr's Easyway has been enabling smokers to get free since 1983.
enabling(Allen Carr's Easyway, smokers)

They all make it more difficult to stop because they perpetuate the illusion that you're making a sacrifice.
perpetuate(they,the illusion)

Figure 5.4: Textual Triples in TripleViz

5.2.5 TermListHighlighter

The TermListHighlighter servlet is invoked when the user chooses to view the Term Lists Highlighted either from the landing page or from the NTriple page. The TermListHighlighter servlet is used to provide the user with the text extracted from the document after preprocessing. It adds a different color to words that belong to the word list either uploaded or selected to filter the triples. The TermListHighlighter servlet reads a file that is specifically generated for this purpose by the OutputGenerator and creates and generates the HTML page. Figure 5.5 shows words from ADR [Nikfarjam *et al.*, 2015] and Side effects [Kuhn *et al.*, 2015] being highlighted with blue and green respectively. For terms that occur in both lists, the colors assigned to the lists are merged to give the term a new color. In Example 25, the term cough belongs to two different term lists, ADR (assigned with color blue) and Side effects (assigned with color red). Therefore, the term cough takes a color purple (merging red and blue).

(25) There was also a report of a vaccine-free day care as well as an Ottawa family of 7 (unvaccinated) children, all of whom contracted whooping cough!

The presentation layer uses the output generated by the business layer and displays them on the HTML pages. It does not perform any functionality to process or analyze the data. The processing and the analysis of the data is performed by the business layer. Section 5.2.6 explains in detail the functionality of the business layer in general and then narrows down to explain the logic behind TripleViz.

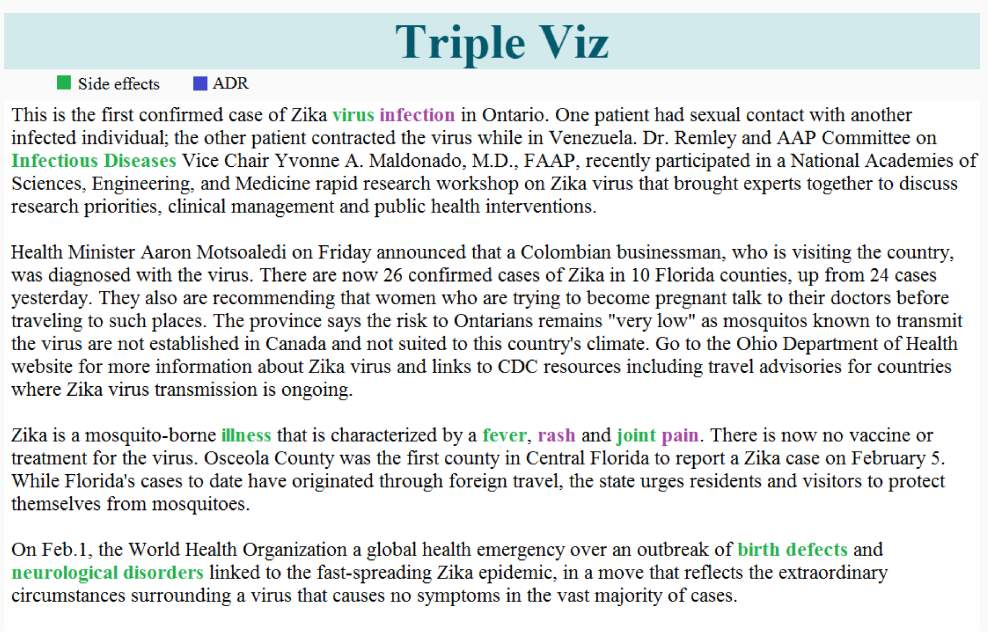


Figure 5.5: Term Lists Highlighted in TripleViz

5.2.6 Business Layer

The Business layer encapsulates the business logic of the system and implements the core functionality of the application. It is concerned with ensuring data consistency and validity and defines any application logic handling retrieval, processing, transformation, and management of application data. After the presentation layer collects the input from the user and passes it to the business layer, the application uses the input to perform one or many functions. The Business layer in TripleViz, built on the Java platform, provides the main functionality to the tool. To keep things modular, we have many functions performing small tasks that can be customized easily. Some of these classes are discussed in detail below.

5.2.7 GateInit

The GateInit class sets the classpath for GATE to function correctly. It mainly initializes the gate parameters such as *gate.home* and *gate.plugins* by setting their path to point to the folder where GATE is installed. Since we use GATE [Cunningham, 2002] to pre-process our documents, all the preprocessing resources need to be registered on GATE [Cunningham, 2002] and added to a pipeline. For example, the Tokenizers, the Sentence Splitters, the Parser, etc., need to be registered before they can be used by GATE. Each of these resources have a run-time parameter that may have to be set. For example, the PTB tokenizer provides options to rename the tokens for annotation. We can set this parameter while registering the PTB tokenizer.

A post processing resource called TripleOutput, which is a TripleViz resource, is added to the end of the pipeline. It analyzes the document for Dependency and SyntaxTreeNode annotations created by the Stanford parser and generates SVO triples in the form of XML. Two separate files are generated, one for SVO triples with head triples and the other for SVO triples with Noun Phrases. The MainServlet passes the document provided by the user as a parameter to the GateInit function which then runs the resources added to the pipeline on this document and returns the control to the MainServlet. The MainServlet now calls the OutputGenerator to use the XML files produced by TripleOutput and generates output files in a format that the Servlets can use.

5.2.8 Output Generator

The Output Generator is the only Java class that converts system friendly output to user friendly format. The output of this resource is accessed by the NTriple servlet, TextualTriple servlet and the TermListHighlighter servlet to render the visualizations via the HTML page. The Output Generator reads through the XML file generated by the TripleOutput, and calls the methods below to generate different outputs.

NTriple Output

As mentioned earlier, Turtled requires the triples to be separated by new lines and also enforces them to be a URI. Hence, The NTriple Output method reads through every triple in the XML and adds a URI to all the tokens in a triple. It writes these triples to a new file with a period and a newline at the end of every triple. This file is read by the NTriple Servlet to generate the visualization. An example of a triple is shown below.

<http://www.tripleviz.org/virus>
<http://www.tripleviz.org/produces>
<http://www.tripleviz.org/symptoms>.

The method generates a separate file for triples with noun phrases. While adding the URI for the noun phrases, one has to take care of the space between the phrases as URI's cannot have spaces and can be from different domains. We fill this space with a "+". The example below shows a triple with different URIs (tripleviz1, tripleviz2, tripleviz3).

<http://www.tripleviz1.org/virus>

<http://www.tripleviz2.org/produces>

<http://www.tripleviz3.org/no+symptoms>.

View Data

The View Data method is used to produce an overview of the triples. It reads through the XML generated by TripleOutput and writes to a file the sentence and its triples. This file is used to display the sentence and the triple when the View the Data button is clicked on TripleViz. Figure 5.3 shows the triples and their sentences displayed to the user. A separate file is written in the same way for NP triples.

The Gephi API uses a file format called gexf to generate graphs. We define a method with the Output Generator to convert an XML file to gexf format. The gexf file is further read by another method within the same class to generate the visualization.

gexfGenerator

This method is used to generate a .gexf file that is used as an input by a method called GephiGraphGenerator to generate the SVG file for the graph by integrating Gephi [Bastian *et al.*, 2009]. The integration of Gephi with TripleViz is explained in Appendix D. Figure 5.6 shows a representation of gexf.

After generating all these outputs, the OutputGenerator calls the

```
<?xml version="1.0" encoding="UTF-8"?>
<gexf xmlns="http://www.gexf.net/1.2draft" version="1.2">
  <graph mode="static" defaultedgetype="directed">
      <nodes>
          <node id="1423" label="you"/>
          <node id="1433" label="sacrifice"/>
      </nodes>
      <edges>
          <edge source="1423" target="1433" id="1429" label="making"/>
      </edges>
    </graph>
</gexf>
```

Figure 5.6: GEXF representation of triple

GephiGraphGenerator.

5.2.9 GephiGraphGenerator

The GephiGraphGenerator generates an SVG image of the graph with all the nodes and edges. Gephi lets us customize the size, color, shape, etc of the nodes and edges. Gephi provides us APIs to set the layout algorithm of our choice to customize the shape of the entire graph (explained under Appendix D).

5.2.10 Data Layer

A Data layer encapsulates data retrieval and storage logic and is only concerned with the management of the data and data sources of the system such as database, XML, web services, flat file etc. In TripleViz, we use Java classes defined in the business layer such as Output Generator, GephiGraph-Generator, etc., to access and modify the XML file generated by GATE [Cunningham, 2002].

TripleViz uses interactive web pages to accept input and display the outputs by making web service calls to business layer functions. The business layer functions not only provides the functionality to support the creation of different visualizations but also allows creation, editing and maintenance of the output files present in the data layer. The function services and their related outputs are achieved though a combination of servlets and API calls which process the various user inputs against the appropriate method, as determined by TripleViz. TripleViz provides the user with view options to switch between Turtled triples and textual triples as well as between head triples and NP triples. It uses a modular approach where any segment can be replaced or removed with little effort.

Chapter 6

Conclusion

TripleViz is a web-based modular visualization tool which takes text or web pages as input and uses third party resources to extract and pre-process the textual data present in the input document to generate subject-verb-object (SVO) triples. Since it consists of third party tools it is easily adaptable and expandable even by non-programmers. SVO triples proved to be effective in reducing the size of the text and giving high level view of the document. These SVO triples are represented in N-triple and plain text format using third party visualization tools for each of these representations and can be filtered using word lists provided by the user. The user can easily add, replace or delete the resources integrated by TripleViz with minimum effort. As well as the visualization for triples, TripleViz also displays the text of the input document, highlighting words from the user-provided list in different colors to help the user interpret the document.

To test the functionality of TripleViz we performed an experiment in

which text for 300 healthmap articles were manually read and classified by our domain expert based on mentions of specific events. This classification was considered to be our gold standard. These healthmap articles were then pre-processed using TripleViz to generate SVO triples. The triples were filtered using three word lists containing person names, family relationship and date which was considered as the baseline of our experiment (show in Table 4.2). The baseline yielded an average of 19% f-measure. The filtered SVO triples were also given to 6 annotators including the authors of the gold standard to classify the documents into specific and non-specific by considering only the SVO triples provided. The experiment yielded an average of 33% f-measure (shown in Table 4.1) which beats our baseline. The experiment proved to convey rudimentary content information.

6.1 Limitations of TripleViz

There are plenty of visualization tools available today which provide visuals with existing functionality. Having said that, it is important to pick the right visualization tool for the task in hand. TripleViz uses Turtled[1] to display N-triples and Gephi to display textual triples. Although Turtled has some useful features such as saving the file and exporting the graph, it does not provide a good visualization for large amounts of triples (see Figure 3.8). Gephi [Bastian *et al.*, 2009] also performs in an unsatisfactory manner while analyzing large data (see Figure 6.1). Therefore, graphical visualization may not be the best way to represent large amount of textual data. One way to reduce space is by integrating co-reference.

[1]https://github.com/mhausenblas/turtled

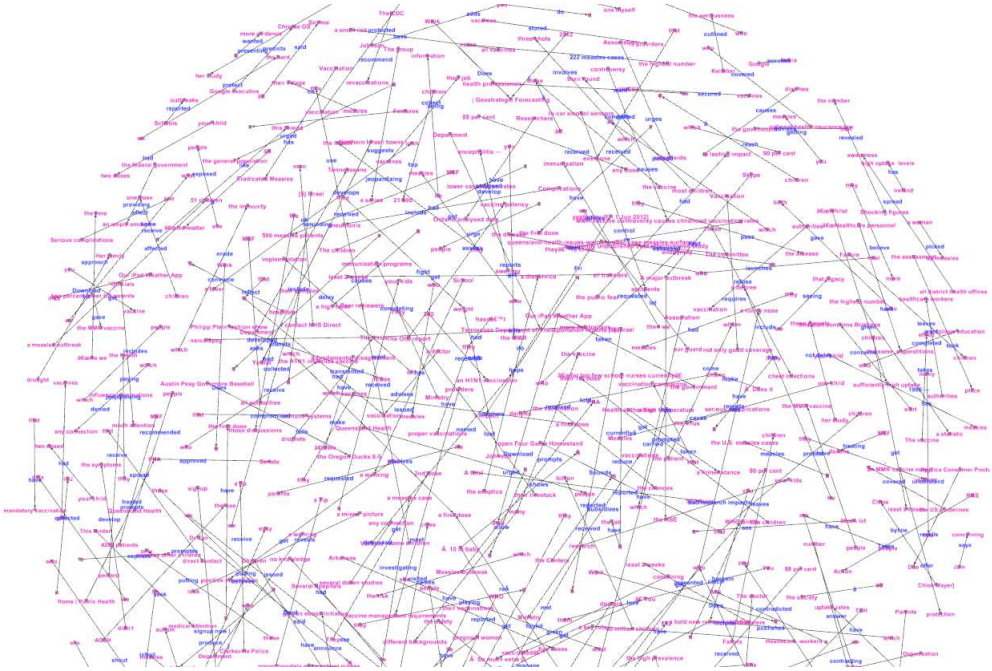


Figure 6.1: 340 NP triples generated for 20 Healthmap articles visualized using Gephi

Many triples loose context without coreference. Examples 26 and 27 show lack of clarity in triples without coreference.

(26) *Parents have this myth in their mind that it can cause autism or some other complications, he said, but added that its just not true.*
Head triples : `cause(it,autism)`
NP triples : `cause(it,autism)`

(27) *Historically, it is clear that if you are not vaccinated there is almost a 100 per cent chance of you getting measles if you come in contact with someone who has got measles, so it is highly contagious.*
Head triples : `got(who,measles)` ; `getting(you,measles)`
NP triples : `got(who,measles)` ; `getting(you,measles)`

The section below provides avenues of future work to expand TripleViz.

6.2 Future Work

The tools can be extended to capture many aspects of NLP. Some of which that might be intersting are listed below.

Coreference

Coreferencing determines whether two phrases refer to the same object. It is considered important in various applications focused on document summarization, question answering, information extraction and many more. Example 28 shows a good example for coreference where *Adults born before 1957* and *These kind of people* refer to the same entity.

(28) *Adults born before 1957* are immune to measles. *These kind of people* do not require vaccination.

Negation

Negation in linguistics is a grammatical constituent that contradicts (or negates) all or part of the meaning of a sentence. It is considered important in applications focused on sentiment analysis and content extraction. Example29 shows a sentence and its negated form.

(29) He **will** go.
He **won't** go.

Integrating these linguistic modules may generate output that might need to be visualized in a form when compared to the SVO triples which may require integration of different visualization tool.

The proposition behind this work was to provide a domain expert with a tool to browse large number of documents with configurable resources

that do not require coding skills or knowledge in natural language annotations. The document content was reduced for visualization purpose by using SVO triples and user provided word lists as filters. This often reduced the documents into a single triple. From our experiment, we further concluded that this was insufficient for browsing but produced nearly 45% F-measure for a specific, baseline classification task. The experiment also shows that the approach was an effective first step in understanding the context of a corpus and would lead to better results with more iterations by a domain expert. Ultimately, proof of usefulness can only be made by expert users using TripleViz to identify some incredibly important information from a large collection that had gone unnoticed before. I feel that researchers in this area can draw inspiration from our experiments and include the useful aspects on which TripleViz is based into their designs and further improve on their use.

Bibliography

Eytan Adar. Guess: a language and interface for graph exploration. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 791–800. ACM, 2006.

Mathieu Bastian, Sebastien Heymann, Mathieu Jacomy, et al. Gephi: an open source software for exploring and manipulating networks. *ICWSM*, pages 361–362, 2009.

Dave Beckett and Art Barstow. N-triples. *W3C RDF Core WG Internal Working Draft*, 2001.

Benjamin B Bederson and Ben Shneiderman. *The craft of information visualization: readings and reflections*. Morgan Kaufmann, 2003.

Sabine Bergler and Jahnavi Dhananjaya. Graphical view of blog content using B2G. In *W3PHI Workshop at the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data-the story so far. *Semantic Services, Interoperability and Web Applications: Emerging Concepts*, pages 205–227, 2009.

Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia-a crystallization point for the web of data. *Web Semantics: science, services and agents on the world wide web*, 7(3):154–165, 2009.

Michelle A Borkin, Azalea A Vo, Zoya Bylinskii, Phillip Isola, Shashank Sunkavalli, Alfonso Oliva, and Hanspeter Pfister. What makes a visualization memorable? *IEEE Transactions on Visualization and Computer Graphics*, 19(12), 2013.

Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D^3 data-driven documents. *IEEE transactions on visualization and computer graphics*, 17(12):2301–2309, 2011.

John S Brownstein, Clark C Freifeld, Ben Y Reis, and Kenneth D Mandl. Surveillance sans frontieres: Internet-based emerging infectious disease intelligence and the healthmap project. *PLoS Med*, 5(7):e151, 2008.

Josep Maria Brunetti, Sören Auer, and Roberto García. The linked data visualization model. In *International Semantic Web Conference (Posters & Demos)*, 2012.

Andreas Bruns, Andreas Kornstadt, and Dennis Wichmann. Web application tests with selenium. *IEEE software*, 26(5):88–91, 2009.

Hyunyoung Choi and Hal Varian. Predicting the present with Google Trends. *Economic Record*, 88(1), 2012.

Wolfgang Nejdl Christian Kohlschütter, Peter Fankhauser. Boilerplate detection using shallow text features. *WSDM, The Third ACM International*

Conference on Web Search and Data Mining New York City, NY USA., 2010.

Prabhakar Raghavan Christopher D. Manning and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

Hamish Cunningham. Gate, a general architecture for text engineering. *Computers and the Humanities*, 36(2):223–254, 2002.

Marie-Catherine de Marneffe and Christopher D Manning. The Stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8, 2008.

John Ellson, Emden Gansner, Lefteris Koutsofios, Stephen C North, and Gordon Woodhull. Graphviz—open source graph drawing tools. In *Graph Drawing*, pages 483–484. Springer, 2001.

Jon Ferraiolo, Fujisawa Jun, and Dean Jackson. Scalable vector graphics (svg) 1.1 specification. *World Wide Web Consortium (W3C). URL http://www. w3. org/TR/SVG11*, 2003.

Nigel Ford. *Web developer. com guide to building intelligent Web sites with JavaScript*. John Wiley & Sons, Inc., 1998.

Thomas MJ Fruchterman and Edward M Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991.

Günther Gediga, Kai-Christoph Hamborg, and Ivo Düntsch. Evaluation

of software systems. *Encyclopedia of computer science and technology*, 45(supplement 30):127–53, 2002.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. Part-of-speech tagging for Twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 42–47, 2011.

Oktie Hassanzadeh and Mariano P Consens. Linked movie data base. In *Linked Data On Web*, 2009.

Marti A Hearst. Untangling text data mining. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 3–10, 1999.

Marti Hearst. What is text mining. *School of Information Meetings, UC Berkeley*, 2003.

Marti Hearst. *Search user interfaces*. Cambridge University Press, 2009.

S Heymann, M Bastian, M Jacomy, C Maussang, A Rohmer, J Bilcke, and A Jacomy. Gexf file format. *GEXF Working Group*, 2009.

Matthew B Hoy. HTML5: a new standard for the Web. *Medical reference services quarterly*, 2011.

Brian Johnson and Ben Shneiderman. Tree-maps: A space-filling approach to the visualization of hierarchical information structures. In *Visualization,*

1991. Visualization'91, Proceedings., IEEE Conference on, pages 284–291, 1991.

Dan Klein and Christopher D Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics, 2003.

Graham Klyne and Jeremy J Carroll. Resource description framework (RDF): Concepts and abstract syntax. 2006.

Michael Kuhn, Ivica Letunic, Lars Juhl Jensen, and Peer Bork. The sider database of drugs and side effects. *Nucleic acids research*, page 1075, 2015.

Håkon Wium Lie, Bert Bos, C Lilley, and I Jacobs. Cascading style sheets. *WWW Consortium,(September 1996)*, 2005.

Steffen Lohmann, Philipp Heim, Timo Stegemann, and Jürgen Ziegler. The relfinder user interface: Interactive exploration of relationships between objects of interest. In *Proceedings of the 14th International Conference on Intelligent User Interfaces (IUI 2010)*, pages 421–422, New York, NY, USA, 2010. ACM.

C Manning, T Grow, T Grenager, J Finkel, and J Bauer. Stanford tokenizer, 2010.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60, 2014.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330, 1993.

Melvin Earl Maron and John L Kuhns. On relevance, probabilistic indexing and information retrieval. *Journal of the ACM (JACM)*, 7(3):218–219, 1960.

Larry Masinter, Tim Berners-Lee, and Roy T Fielding. Uniform resource identifier (URI): Generic syntax. *The Internet Society*, 2005.

Karl Moss. *Java servlets*. McGraw-Hill, Inc., 1999.

Finn Arup Nielsen. AFINN. *Informatics and Mathematical Modelling, Technical University of Denmark*, mar 2011.

Azadeh Nikfarjam, Abeed Sarker, Karen O'Connor, Rachel Ginn, and Graciela Gonzalez. Pharmacovigilance from social media: mining adverse drug reaction mentions using sequence labeling with word embedding cluster features. *Journal of the American Medical Informatics Association*, 22(3):671–681, 2015.

Natalya F Noy, Deborah L McGuinness, et al. Ontology development 101: A guide to creating your first ontology, 2001.

Gabriele Paolacci, Jesse Chandler, and Panagiotis G Ipeirotis. Running experiments on amazon mechanical turk. *Judgment and Decision making*, 5(5), 2010.

Dale Patterson. Interactive 3d web applications for visualization of World

Health Organization data. In *Proceedings of the Australasian Computer Science Week Multiconference*, page 76. ACM, 2016.

Samuele Pedroni and Noel Rappin. *Jython essentials*. " O'Reilly Media, Inc.", 2002.

Guido Powell, Kate Zinszer, Jahnavi Dhananjay, Chi Bahk, Lawrence Madoff, John Brownstein, Sabine Bergler, and David Buckeridge. Monitoring discussion of vaccine adverse events in the media: Opportunities from the vaccine sentimeter. In *W3PHI Workshop at the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Eric Prud'Hommeaux, Andy Seaborne, et al. SPARQL query language for RDF. *W3C recommendation*, 15, 2008.

Eric Prud'hommeaux, Gavin Carothers, and Lex Machina. RDF 1.1 turtle. *W3C Recommendation*, 2014.

Greg Roelofs and Richard Koman. *PNG: the definitive guide*. O'Reilly & Associates, Inc., 1999.

Martin G Skjæveland. Sgvizler: A JavaScript wrapper for easy visualization of SPARQL result sets. In *The Semantic Web: ESWC 2012 Satellite Events*, pages 361–365. Springer, 2012.

Appendix A

Stanford Dependencies

Some differences between universal dependencies and TypedCollapsed dependencies [de Marneffe and Manning, 2008] is shown below. We display universal and TypedCollapsed dependencies for the sentence and highlight the dependencies that are not similar.

(30) *Sentence : Cancer is a complex group of diseases with many possible causes.*

Universal dependencies	TypedCollapsed dependencies
nsubj (group , cancer)	nsubj (group , cancer)
cop (group , is)	cop (group , is)
det (group , a)	det (group , a)
amod (group , complex)	amod (group , complex)
prep (group , of)	
pobj (of , disease)	**prep_of (group , disease)**
prep (disease , with)	
pobj (with , cause)	**prep_with (disease , cause)**
amod (causes , many)	amod (causes , many)
amod (causes , possible)	amod (causes , possible)

Appendix B

Indexing document using Elasticsearch API

Elasticsearch is a scalable open-source, search and analytics engine that runs on Lucene. It provides functions to store, search and analyze big volumes of data rapidly in real-time. Elasticsearch is a Java based tool with the ability to search all kinds of documents. Elasticsearch is distributed, where it's indices can be divided into shards and these shards can be combined individually with one another to compare text.

The installation comes with a server with pre-assigned port number to run them locally. Once the package for Elasticsearch is downloaded, a small script calls the Elasticsearch API to index our document (see below). Fields such as the doc body, file name, the index name and the index timestamp are specified.

```
from datetime import datetime
from elasticsearch import Elasticsearch
import glob
import re

# by default we connect to localhost:9200
es = Elasticsearch()

path = 'bp/*.txt'

# index
id = 1
for doc in glob.glob(path):
    f = open(doc, 'rb')
    content = f.read()
    fn = re.search('\d+.txt',doc).group(0)
    es.index(index='test-index', doc_type='test-type', id=id, body={'doc_body': content, 'file_name':
                fn, 'index_timestamp': datetime.now()})
    id += 1
```

Appendix C

Setting up Kibana

Kibana is an open-source visualization platform coupled with Elasticsearch. Kibana provides a browser-based interface that makes it easy to analyze large volumes of data. It uses the indices created by Elasticsearch to search, view and interact with data. Advanced data analysis can be performed using Kibana to generate visualizations of type chart, maps and tables.

How to setup Kibana

The Kibana package is downloaded from `https://www.elastic.co/downloads/kibana`. We then extract the files and make changes in the kibana.yml file to point to the port on which our Elasticserach server is running. Kibana is then run using the command prompt to open interface an that appears on the browser. Parameters for the index are set after which we need to visualize. A detailed explanation of the setup is shown below.

How to setup kibana

For detailed instructions on how to use Kibana, please refer to the official

documentation on `https://www.elastic.co/guide/en/kibana/current/index.html`. The default index pattern currently chosen for use with Kibana is set to the index which can be changed by setting another index as the default under settings. Note: When launching Kibana, you should always first expand your desired date range using the "Time Picker" in the top right hand corner of the screen.

The Discover page lets you interactively explore the content of an index and the Visualize page allows you setup new visualizations (such as data tables, bar/pie/line/area charts, etc.,). Pre-existing visualizations can also be accessed through this tab.

We generated visualizations to analyse the data by filtering them in different ways. Our main goal was to look at the pattern of tokens that occurred in the text. We filtered the tokens to visualize only some of the lists to see how the terms were ranking over a period of time. One of the visualizations is shown in figure C.1 where we look at the top 50 terms sorted in the descending order of their document frequency. The most common terms that popped up during these visualization experiments were the stopwords such as a, the, etc. We used the stopwords list to filter the terms so that they don't occur in the visualization.

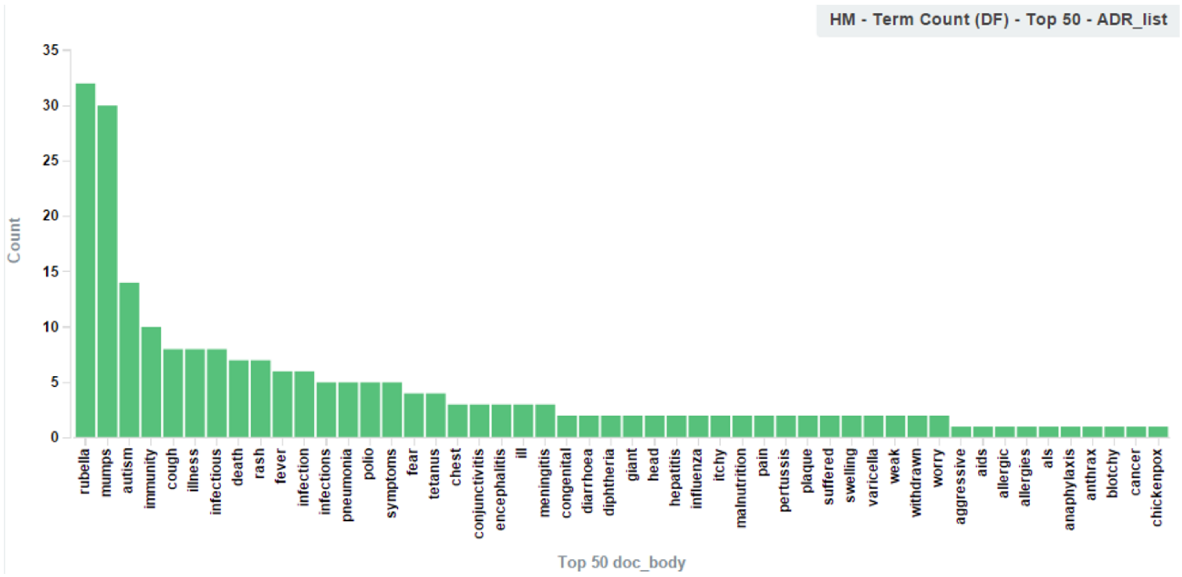


Figure C.1: Visualiztaion of document index in Kibana

Appendix D

Gephi Intergration

Gephi is an is open-source and freeware, leading in visualization for all kinds of graphs and networks. The Java API integration of Gephi is shown below.

```
/* Create a project * /
ProjectController pc = Lookup.getDefault().lookup(ProjectController.class);
pc.newProject();


/* Create a workspace within the project */
Workspace workspace = pc.getCurrentWorkspace();


/* Graph API calls to create and store a graph */
GraphModel graphModel = Lookup.getDefault().lookup
(GraphController.class).getGraphModel();
```

```
/* Create nodes */
Node n0 = graphModel.factory().newNode("n0");
n0.getNodeData().setLabel("Node 0");


Node n1 = graphModel.factory().newNode("n1");
n1.getNodeData().setLabel("Node 1");


/*Create a directed edge with weight 1 */
Edge e1 = graphModel.factory().newEdge(n1,n0,1,true);


/*Generate the graph */
DirectedGraph directedGraph = graphModel.getDirectedGraph();
directedGraph.addNode(n0);
directedGraph.addNode(n1);
directedGraph.addEdge(e1);


/* Initialize Frutcherman Reingold layout */
FruchtermanReingold layout = new FruchtermanReingold(null);
layout.setArea(1000.0f);
layout.setGravity(10.0);
layout.setSpeed(2.0);
layout.setGraphModel(graphModel);
layout.resetPropertiesValues();
layout.initAlgo();
```

```
layout.goAlgo();


/* Run the Frutcherman Reingold algorithm */
for (int i = 0; i ¡ 500000 && layout.canAlgo(); i++)

layout.goAlgo();

layout.endAlgo();


/* Export the graph as an SVG */
ExportController ec = Lookup.getDefault().lookup(ExportController.class);
ec.exportFile(new File("path/filename.svg"));
```

Appendix E

Filtering Lists applied for the Specific task experiment

The experiment generates Subject-Verb-Object triples filtered by three word lists and provides the results to 5 annotators to identify specific events in each of the documents. To obtain these lists, there are a number of gazetteer lists and JAPE rules(Regular expression combined with java) used. How we obtain the three word lists are listed below.

Person names

To get the list of person names, we add the Gazetteer lists provided by the ANNIE plug-in into our pipeline. The gazetteer lists used by ANNIE to identify person names are listed below.

person_ending.lst	person_female.lst	person_female_ambig.lst
person_female_cap.lst	person_female_lower.lst	person_full.lst
person_male.lst	person_male_ambig.lst	person_male_cap.lst
person_male_lower.lst	person_sci.lst	person_spur.lst
cabinet_ministers.lst	foreign_ministers.lst	

We also add the named entity recognizer by ANNIE into our pipeline which contains a number of JAPE rules to generate an annotation for Person using the gazetteer lists mentioned above.

Date

Similar to Person Names, ANNIE plug-in uses the following gazetteer lists to identify date, days, months, seasons, etc.

1. months.lst
2. ordinal.lst.lst
3. numbers.lst

The named entity recognizer from ANNIE then combines them into a single annotation using JAPE rules to produce a Date annotation.

The entries obtained for both Date and Person annotations are taken to filter our SVO triples.

Family Relationships

A list expressing family relationships was put together specifically for this purpose by us. A number of websites were referred to while creating this list. A complete list of 125 entries is given below.

adolescent
adopted child
adopted daughter
adopted son
adult
aunt
baby
boy
brother-in-law
brothers-in-law
child
childhood
children
close relatives
closest relatives
cousin
daughter-in-law
daughter
daughters-in-law
distant relatives
elder sister
family members
father-in-law
father-in-law
father
female child
first cousin
foster brother
foster child
foster daughter
foster family
foster father
foster home
foster mother
foster parents
foster sister
foster son
girl
grandchild
grandchildren
granddaughter
grandfather
grandmother
grandparents
grandson
great-grandfather
granny
great-grandchild
great-grandmother
grownup
half-brother
half-sister
head of the household
immediate family
in-laws
infant
kid brother
kid
lad
little boy
little girl
male child
man
middle-aged man
middle-aged woman
mother-in-law
mother
my family
my folks
my kin
my kinfolk
kinsfolk
my relatives
nearest relatives
nephew
next of kin
niece
offspring
offspring
old man
old woman
older sister
orphan
orphans
senior members of the family
parents
second cousin
sibling
siblings
sister-in-law
sister
sisters-in-law
son-in-law
son
sons-in-law
spouse
stepbrother
stepchild
stepfather
stepmother
stepsister
teenage boy
teenage girl
teenager
toddler
twin brother
twin brothers
twin sister
twin sisters
twins
uncle
woman
young boy
young girl
young man
young woman
younger brother
younger members of the family
youngster
youth
youths
members of the family
brother

Appendix F

List Of Triples analyzed

The triples listed in the below section are referenced from Chapter 4. The document name, the triple and its sentence are provided below for the incorrect annotation by our domain expert and gold standard author K.

Document 1:

http://actmedia.eu/daily/bloomberg-news-danish-made-tb-vaccine-sickens-115-children-in-romania/43298

The children had symptoms including swollen lymph nodes and abscesses, and 50 of the 115 children have been hospitalized since March, the Stockholm-based ECDC said, citing Romanian media reports.

`had(The children,symptoms)`

Document 2:

http://timesofindia.indiatimes.com/city/jaipur/Nurse-administered-vaccine-in-a-faulty-manner/articleshow/16907683.cms

A day after the incident, a medical team met all the children on Sunday to

find out their condition.

`met(a medical team,the children)`

Document 3:

http://www.naturalnews.com/037338_Gardasil_death_brain_tissue.html

"My daughter, in the middle of her series of injections of Gardasil, had a bout of Bell's palsy that paralyzed the right side of her face," wrote Della Smith.

`had(My daughter,a bout)`

"My daughter developed epilepsy since being vaccinated, and when I share her experience with people, most doubt the connection," reader Nina Kenney wrote.

`developed(My daughter,epilepsy)`

She said she and her daughter had dinner the night before.

`had(her daughter,dinner)`

In testimony days after Mrs. Renata spoke to authorities looking into her daughter's death, Shaw said he and U.S. pathologist Sin Hang Lee noted "heavy aluminum staining in Ms. Renata's brain tissue could have acted as a 'trojan horse,' bringing the human papillomavirus, or HPV, into her brain," said the Post.

`said(Shaw,he)`

"My 21-year-old daughter, Chris, got her third shot of Gardasil on June 3, 2008," Emily Tarsell wrote in an email to Kotz.

`got(My 21-year-old daughter,her third shot)`

She said it was vital to discuss any weaknesses in the research so parents and potential vaccine recipients had all the necessary information to make proper care decisions.

`had(parents,the necessary information)`

The parents of a New Zealand teenager say a drug aimed at preventing cervical cancer was instead responsible for her death, a claim that is being "rejected as convoluted pseudoscience" by a researcher at the University of Auckland.

say(The parents,a drug)

Document 4:

http://www.stuff.co.nz/national/health/7709965/Cervical-cancer-vaccine-link-to-death-disputed

It was important to discuss the weaknesses in the research so parents and possible vaccine recipients had all the information, she said.

`had(parents,the information)`

Document 5:

http://www.heraldscotland.com/news/13072060.No_payout_for_jab_damage/

Katie Stephen, 21, from Stonehaven in Aberdeenshire, lost the use of her left ear days after being inoculated as a toddler.

`lost(Katie Stephen,the use)`

Ms Stephen was deafened by a vaccine carrying the botched Urabe strain of mumps, which was later withdrawn.

`deafened(a vaccine,Stephen)`

Document 6:

http://www.telegraph.co.uk/health/healthnews/9521728/Rogue-strain-of-MMR-

vaccine-caused-deafness.html

Director of Immunisation Professor David Salisbury said: "It is important that parents get their child vaccinated against measles, mumps and rubella - all of which are highly infectious."

`get(parents,their child)`

Document 7:

http://crofsblogs.typepad.com/h5n1/2012/08/nepal-infant-death-raises-questions-about-dpt-vaccine.html

The infant developed a fever after being vaccinated in the afternoon and died in the night.

`developed(The infant,a fever)`

Infant death raises questions about DPT vaccine.

`raises(Infant death,question)`

With the parents of the dead infant indicating the vaccine as the cause of death, the quality of vaccines has again come into question.

`indicating(the parents,the vaccine)`

Document 8:

http://www.radionz.co.nz/news/national/112720/mother-convinced-vaccine-killed-her-daughter

The mother of an Upper Hutt teenager has told an inquest she blames the Gardasil vaccine for her daughter's death.

`told(The mother,inquest)`

A cardiologist has also given evidence saying Ms Renata could have had genetic heart problems.

`had(Renata,genetic heart problems)`

Document 9:

http://timesofindia.indiatimes.com/city/chennai/Vaccine-not-seen-as-cause-but-postmortem-report-awaited/articleshow/14858402.cms

"We have been administering the vaccine from December and not a single child has had such problems," said an official.

`had(a single child,such problems)`

While some children can develop a slight temperature after vaccination, Tanujashree had no such symptoms after her first dose - administered when she completed one month.

`develop(some children,a slight temperature)`

Document 10:

http://www.durhamregion.com/community/education/article/1379379–vaccine-a-sore-spot-for-whitby-teen

Kaitlyn's mother, Yvonne Armstrong, has been sharing her daughter's story at various school boards.

`sharing(Kaitlyn's mother,her daughter's story)`

"When it came to the allergies, I even told them," she says of the nurse on the day she received the vaccine, adding her mother had noted her allergy on the consent form.

`noted(her mother,her allergy)`

Document 11:

http://www.durhamregion.com/community/education/article/1379379–vaccine-a-sore-spot-for-whitby-teen

Since this report, the true extent of this tragedy is coming to light, as parents of these vaccinated children have reported yet more injuries.

`reported(parents,yet more injuries)`