PathVis: An Online Software for Visualising Locations and Character Movements in Literature

Undergraduate Honours Thesis
Faculty of Science (Computing Science)
University of Ontario Institute of Technology

Ammar Sidhu

Supervisor:
Dr. Christopher Collins

April 24, 2015

Abstract

The objective of this thesis was to design a software system that helps users visualise changes in locations and character movements in literature. Creating visualisations of data can provide many benefits, and when visualisations are developed for literature they can provide users with a new perspective on the works. Literature pieces can encompass many different settings, and as the story progresses characters can move to different cities and locations all over the world. Mapping these changes can provide users a higher understanding of the scope of the stories they are reading, and provide a more relatable visualisation of the literature. PathVis was developed to provide an online visualisation that dynamically maps out literature locations from user uploaded text. By combining a map visualisation with a text explorer, users can gain a higher understanding of the distances involved in their favourite pieces of literature and explore the context and situations that are associated with locations in a story. By providing links to the underlying text as well as points of interest on the map, the PathVis visualisation provides some much needed interactivity that some text based mediums can lack.

Contents

1. Introduction 1
1.1 Problem Statement and Motivation 1
1.2 Goals 2
1.3 Thesis Overview 2
2. Related Work 3
2.1 Interactive Game of Thrones Map with Spoilers Control 3
2.2 Interactive Map of Middle Earth – LoTR Projec 4
2.3 Summary 5
3. Design Approach 6
4. Implementation 8
4.1 Server 8
4.2 Text Parser and Stanford Named Entity Recogniser 10
4.3 GeoNames Database 10
4.4 HTML, Google Maps API, Google Geocode API, and Text Explorer 11
4.4.1 Google Maps API 12
4.4.2 Google Geocode API 12
4.4.3 PathVis Visualisation 13
5. Summary and Conclusion 20
5.1 Summary 20
5.2 Conclusion 20
6. Future Work 21
6.1 Biasing 21
6.2 Future Applications 22
7. References 24

List of Figures

Figure 1: Game of Thrones Interactive Map [1] 3
Figure 2: Interactive Map of Middle Earth [2] 4
Figure 3: Design Appoach Diagram 7
Figure 4: PHP Upload Interface 8
Figure 5: PHP Upload Interface File Explorer 9
Figure 6: An Example of Colour Coded Markers and Path Lines 14
Figure 7: An Example of the Info Window and Text Explorer Link 15
Figure 8: The Path Line Controls 16
Figure 9: An Example of the Show Subset Control in Action 17
Figure 10: The Text Explorer with Highlighted Sentence and Link to Map Location 18
Figure 11: The Completed PathVis Visualisation 19
Figure 12: An Example of how a Biasing Interface Might Look Like 22

1. Introduction

The objective of this thesis was to design a software system that helps users visualise changes in locations and character movements in literature. Creating visualisations of data can provide many benefits, and when visualisations are developed for literature they can provide users with a new perspective on the works. Literature pieces can encompass many different settings, and as the story progresses characters can move to different cities and locations all over the world. Mapping these changes can provide users a higher understanding of the scope of the stories they are reading, and provide a more relatable visualisation of the literature.

1.1 Problem Statement and Motivation

There are many different types of visualisations that you can develop using literature as your topic of study. Some visualisations have previously been made to explore the interactions between characters in the literature. Many others have a much smaller scope, such as studying and displaying the frequency of words or the associations between words. Maps provide an easy to understand visualisation of a novel that is very relatable. They can provide a high level understanding of the path a piece of literature takes, and they allow the user to grasp the distances involved in a story in a much easier way. Unfortunately location mappings of literature and novels are uncommon, and the process in developing a map can be tedious and time consuming. They are usually plotted out entirely by hand, with the creators scouring the text for locations and place names manually. Most importantly they are almost always only applicable to one piece of literature or novel, with no way of reusing their efforts or maps on other works.

1.2 Goals

The goal of this thesis is to provide easy to use prototype software that allows the user to create maps of their favourite pieces of literature quickly. The software is to be online based to allow a wide range of people access. It should allow the user to upload their own content, and then build a map of that content dynamically. In addition to the map the visualisation should allow the user to view and explore the uploaded text as well, as this can provide much needed context when exploring the literature locations on the map.

1.3 Thesis Overview

In Section 2 I will investigate related works of novel map visualisation in order to gain an understanding of what is involved in creating the maps and any shortcomings that I would like to solve with my implementation. Then in Section 3 I will discuss my design approach when developing PathVis. Afterwards in Section 4 I will explain the implementation of Pathvis, any justifications in the software used for the implementation, software design, and any problems faced in the implementation. In Section 5 I will provide a summary of the report and my conclusions for the project. Finally in Section 6 I will discuss any future work to be done as well as other future applications for PathVis.

2. Related Work

In this section I will briefly discuss a few examples of mapping applications that have already been developed. I will discuss some of their design choices and how these applications are related to the development of my own application PathVis.

2.1 Interactive Game of Thrones Map with Spoilers Control

This website provides a map of character movement for the novel series Game of Thrones, by George R. R. Martin. It features a map of the fictional world with a number of controls on the right hand side. These controls allow you to choose the characters whose progress through the story you want to track, which it then draws as a path line overlay on the map. One of the main things I noticed about this visualisation was that there is no way to view the actual text that the visualisation is based on. It would have been interesting to know how the place names are determined and the context that they were cited in.

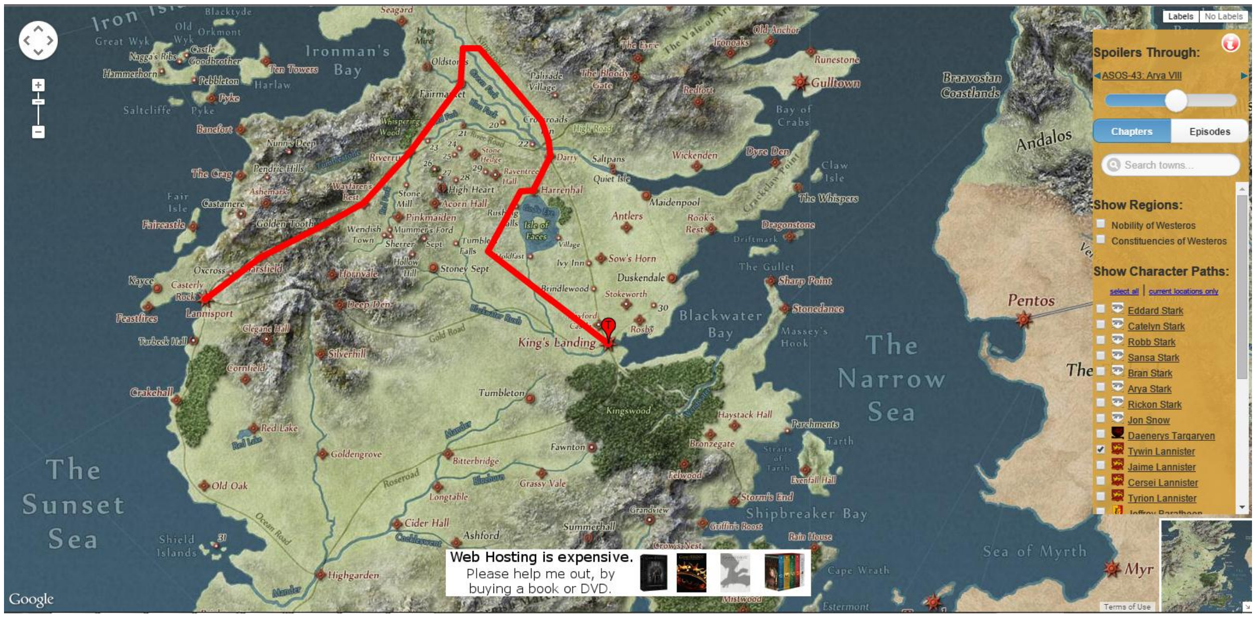
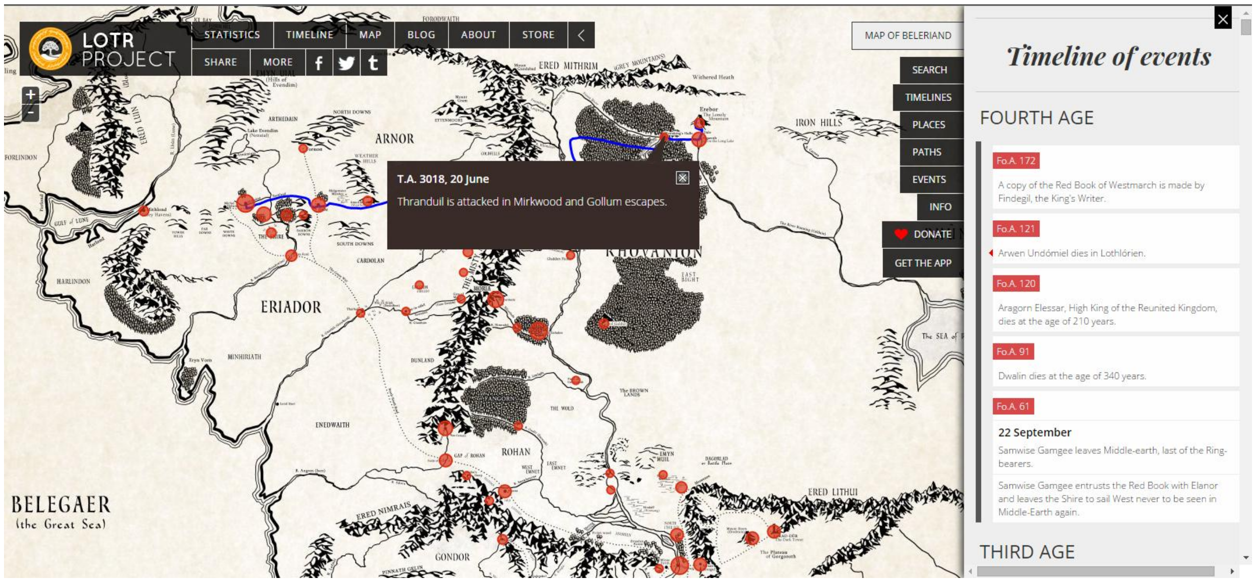


Figure 1: Game of Thrones Interactive Map [1]

Image obtained from: http://quartermaester.info/

2.2 Interactive Map of Middle Earth – LoTR Projec

This website provides an interactive map of the character movement for the novel series The Lord of the Rings, as well as the novel The Hobbit. Again it allows you to pick which characters that you want to track. It also draws a path line overlay onto the map. One advantage that I noticed was that it has a time line button which brings up the date in the series major events take place on. This provides a bit more context from the text than the Game of Thrones website. It also allows you to click on locations on the map, which then opens up a pop up tool tip that has brief descriptions of the location's relevancy to the novel and what events took place there. This pop up tooltip is definitely useful in providing much needed context from the literature that it is based on.



T.A. 3018, 20 June

Thranduil is attacked in Mirkwood and Gollum escapes.

Timeline of events

FOURTH AGE

Fo.A. 172

A copy of the Red Book of Westmarch is made by Findegil, the King's Writer.

Fo.A. 121

Arwen Undómiel dies in Lothlórien.

Fo.A. 120

Aragorn Elessar, High King of the Reunited Kingdom, dies at the age of 210 years.

Fo.A. 91

Dwalin dies at the age of 340 years.

Fo.A. 61

22 September

Samwise Gamgee leaves Middle-earth, last of the Ring-bearers.

Samwise Gamgee entrusts the Red Book with Elanor and leaves the Shire to sail West never to be seen in Middle-Earth again.

THIRD AGE

Figure 2: Interactive Map of Middle Earth [2]

Image obtained from:

http://lotrproject.com/map/#zoom=3&lat=-1315.5&lon=1500&layers=BTTTTTTTTT

2.3 Summary

It is readily apparent that both of these websites are hand developed, with someone parsing the text by hand and creating the visualisations manually. Another point to be made is that they are only based on one book series, and there is no way for the user to apply these map making applications to other pieces of literature. The pop up menu that the Lord of the Rings website had is very useful in providing context from the books, and is something that I decided that I wanted to include in my own project. By their very nature these websites do not allow for their methods to by applied to other pieces of literature dynamically, and both did not provide access to the underlying text for the user to explore.

3. Design Approach

The approach to designing my project began with the requirement that this software be web based. By making this software available online, the number of users and the audience that can use it increases dramatically. The second and most important requirement for this project was for the ability for the user to upload their own content to the system. This will allows the system to be dynamic and provides a mechanism for creating maps based on any piece of literature or text the user provides. This removes the single novel or series constraints that were applied to the websites explored in the Related Works section of this document and increases the usefulness of the overall project. There was also the requirement for the user to be able to explore the text that they have uploaded in real time. By displaying the text along with a map of the locations the user can gain some much needed context about how the locations changed and the relevancy of the location to the literature. Finally the application needed to be easy to use as well as quick. As this project was going to be web based, the speed in which the program processes the uploaded documents was important. Most users expect website load times to be relatively short, which meant that the back end of the software needed to be optimised in order to reduce wait times. This meant that I needed a way to cache data in order to offload as much processing from the user's web browser as possible.

The design of the project would have to be broken up into different areas. The first major component of the system would be a server on which most of the processing would take place. This would provide a back end for which to run programs and hold any data that would benefit from being cached. This data was to be stored in a database for an easy implementation and querying. The second major component would be the text processing. The application would have to be able to parse the text and extract the locations and place names, again to be implemented offline in order to reduce the load on the user's web browser. The final major component would be the web interface and map. The application would have to make use of some mapping software in order to show the points of interest and the different locations of the literature as the story progressed. The design would also have to provide a way to link the literature text to the map in a fluid and easy to use way.

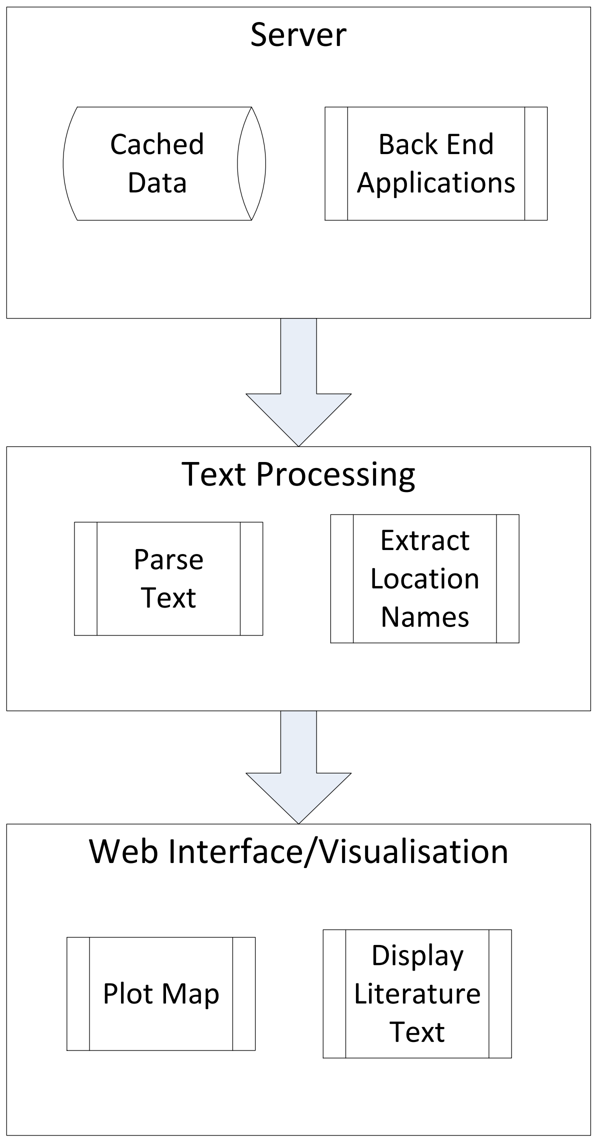


Figure 3: Design Appoach Diagram

4. Implementation

4.1 Server

In order to run any applications and store any cached data I set up a server for the back end of the application. For my project I set up a LAMP server, which stands for Linux Apache MySQL PHP. LAMP servers are some of the most widely used open source servers today. They are easy to set up and applicable to almost any sitution. In my implementation I made use of the PHP server and the MySQL server. My thesis project was one of my first experiences with PHP coding.

The first component of the PathVis system is the user uploadable content. To accomplish this I created a PHP form that allows the user to select a file using a file explorer and then upload the file to the server. The file is then stored on the server and the file name is then sent as a variable to the execute PHP program. The execute PHP program then runs the Parser program and includes the location of the uploaded text file as a runtime argument. During this time the program displays an informative message indicating that the file is being processed.

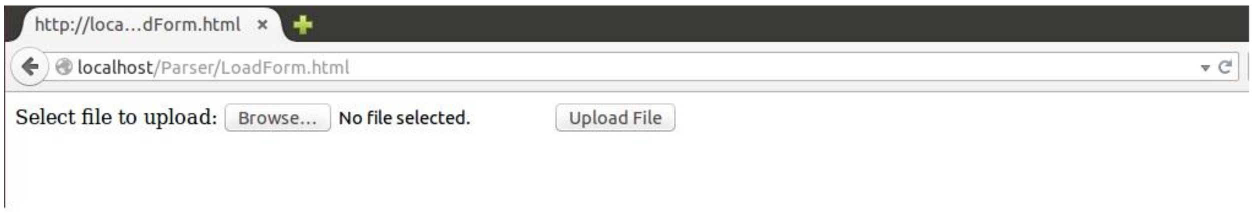
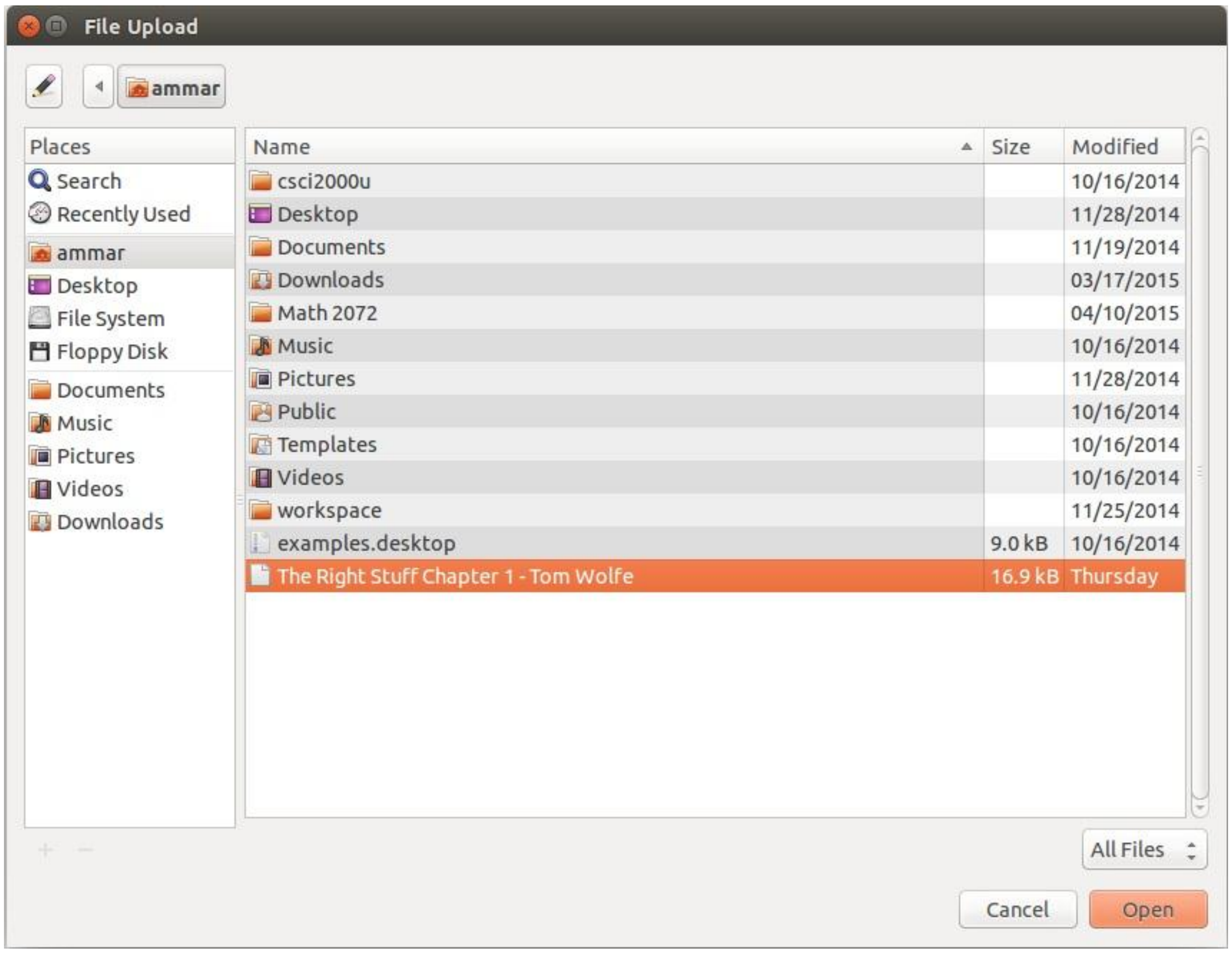


Figure 4: PHP Upload Interface



Name	Size	Modified
csci2000u		10/16/2014
Desktop		11/28/2014
Documents		11/19/2014
Downloads		03/17/2015
Math 2072		04/10/2015
Music		10/16/2014
Pictures		11/28/2014
Public		10/16/2014
Templates		10/16/2014
Videos		10/16/2014
workspace		11/25/2014
examples.desktop	9.0 kB	10/16/2014
The Right Stuff Chapter 1 - Tom Wolfe	16.9 kB	Thursday

Figure 5: PHP Upload Interface File Explorer

4.2 Text Parser and Stanford Named Entity Recogniser

One of the major hurdles in implementing a dynamic literature mapper is extracting the different locations from the text. In order to do this I researched a number of different parsers and language processors. For my implementation I decided to use the Stanford Natural Language Processor [3]. The Stanford NLP is a popular Java based suite of tools that takes raw text, parses it and assigns tags for a variety of different applications including parts of speech, dates and Entities. For the PathVis application I made use of their Named Entity Recogniser Tool [4]. This tool parses raw text sentence by sentence and assigns tags based on three different classes: Names, Locations, and Entities. In this application I used the Location tag in order to extract any literature locations and settings.

There were a couple of modifications that I had to implement in order for the NER parser to produce useable data. One problem I faced was concerning the data structures that the NER parser produced. When the parser processes a sentence it first puts each word into a string array. This was a problem because I needed to use the raw text sentences later on in the PathVis program. My solution was to combine the word array back into a sentence with some regular expressions in order to keep the original sentence structure and formatting. The NER parser also treats each word as a separate entity, for example the location “New York” would be flagged as two separate locations, “New” and “York”. My solution was to check consecutive words for a location tag and combine the two separate strings into one final location. Once the location names have been extracted the geotagging process can begin.

4.3 GeoNames Database

In order to reduce the load and processing time of the PathVis application I needed a way to offload as much processing from the user’s browser as possible. One way to accomplish this task is to cache information on the server in the form of a database. During my research I found the GeoNames Database Project [5]. GeoNames is a free to use repository of over 2.5 million populated places. The GeoNames project provides a number of different scopes and data levels depending on the level of detail required for your application. They provide geographic location data for separate countries, for the whole world or for cities over a certain size. In order to keep memory space down and limit the amount of the tuples that any queries had to search through, I decided to use the database that includes all the geo location data for cities with a population size of 15,000 or more. I felt that this was a good trade-off between the amount of data that could be cached and the likely hood that a place name would be found. I reasoned that most locations in literature take place in more populated areas and cities, and any edge cases could be solved with the Google Geocode API.

The GeoNames table provides a number of different attributes from which to query on. For my implementation I focused on the location name, alternative names or aliases, the latitude and longitude, and the population size. The PathVis application would take the place names extracted with the NER parser and search for them in the database. If it found a match it would take the corresponding latitude and longitude coordinates and attach them to the city name. If there were any ambiguous cases it chose the one with the larger population size. This process of eliminating ambiguity is further discussed in the biasing section. The location name, the sentence that the location name was in, and any matching coordinates are then packaged in a JSON file that is passed on to the HTML and Javascript code.

4.4 HTML, Google Maps API, Google Geocode API, and Text Explorer

The majority of the PathVis visualisation is produced using HTML and Javascript functions. There are a few third party APIs that I made use of in the implementation of my project and I will discuss them briefly before continuing on with the overall implementation.

4.4.1 Google Maps API

This project revolves almost completely around mapping and plotting locations, therefore I researched some ways to display this data effectively and efficiently. Many people have used Google Maps, and I felt that users would feel the most comfortable if my map used their API. The Google Maps API [6] has a wide range of built-in tools and functions that lent themselves well to the application that I was developing. As well as displaying a map that is both scrollable and zoomable, the Google Maps API includes built in functions that allow for markers to be put on the map as well as coloured overlays that can applied to the maps. These two functions were extremely helpful in building a path line for the uploaded literature. The markers also had a tooltip type menu that could be displayed when they were clicked, which was an idea that I wanted to implement after I reviewed the Lord of the Rings Interactive Map in the Related Works section of this document.

4.4.2 Google Geocode API

Another third party API that I made use of was the Google Geocode API [7]. This API accepts location names and text data, and then returns the corresponding geographical coordinates as part of a geocode object. It also has a number of options and flags that you can set in order to narrow down the amount of data it returns and to try to eliminate some of the ambiguity that can arise when searching for city names or places. One shortcoming of the Google Geocode API were the limits that were applied to the number of geocode requests that you could run. It was limited to a total of 2500 requests per day and a maximum speed of 5 requests per second. These limits highlighted the need and value of having a backend database that cached coordinate data, as the Geocode requests were only needed to find coordinates for place names that weren't found in the database. I also had to implement some methods that slowed down the number of requests so that the query limit would not be reached. This ultimately made my application slower but it was necessary in order to provide a seamless and accurate mapping of the literature.

4.4.3 PathVis Visualisation

The implementation of the PathVis visualisation started with the JSON file that the parser created. This JSON file contained an array of location names, the sentence from the literature that contained the mentioned place name, the coordinates of the place name if they were found in the GeoNames database, and a flag indicating that coordinates were found. The first step in the implementation was loading the JSON file into the Javascript program in order to use its contents. This was not as simple as I thought it would be as the asynchronous nature of the Javascript calls meant that the order in which some actions were processed differed from the order they were written in the code. In this instance my code was trying to run functions on the JSON data before it was finished loading. This caused a number of out of bound exceptions and similar problems. My solution to this problem was to switch the async flag for my JQuery calls in order to force the rest of the program to wait until JSON data was completely finished loading.

The next step in my implementation was using the Google Geocode API in order to request coordinates for place names that were not found in the GeoNames database. In order to do this I traversed the JSON array and checked the geocode flag to see if the coordinates were missing. If they were a request was sent out to the geocode service to search for relevant coordinates using the place name. The result of the request would be in the form of 3 different status, OK, OVER QUERY LIMIT or FAILED. If the status was OK, the coordinates were copied into the JSON array. If the stats was for an over query limit, the set timeout function would be called to pause the program for a short while and then run the request again. If the status was FAILED then the geocoder could not find any relevant coordinates and the place name was discarded with an error message. Again as the geocode requests are asynchronous calls, I ran into problems with the ordering of my data. My solution to this problem was to use JQuery deferrals. Deferrals are a form of promises that set flags when the functions that they are called on complete. I created a new deferral for each asynchronous geocode call, and then put them in a stack. Once and only once all the deferrals in the stack were completed and popped, the program could continue.

After all of the coordinate data was created the mapping process could begin. For each location I created a marker object. Each marker has a number of attributes including the colour of the marker, its location, and an info window that opens up when the marker is clicked. The colour of the marker was coded so that it changed from red to green when you clicked it, indicating that it was the active marker.

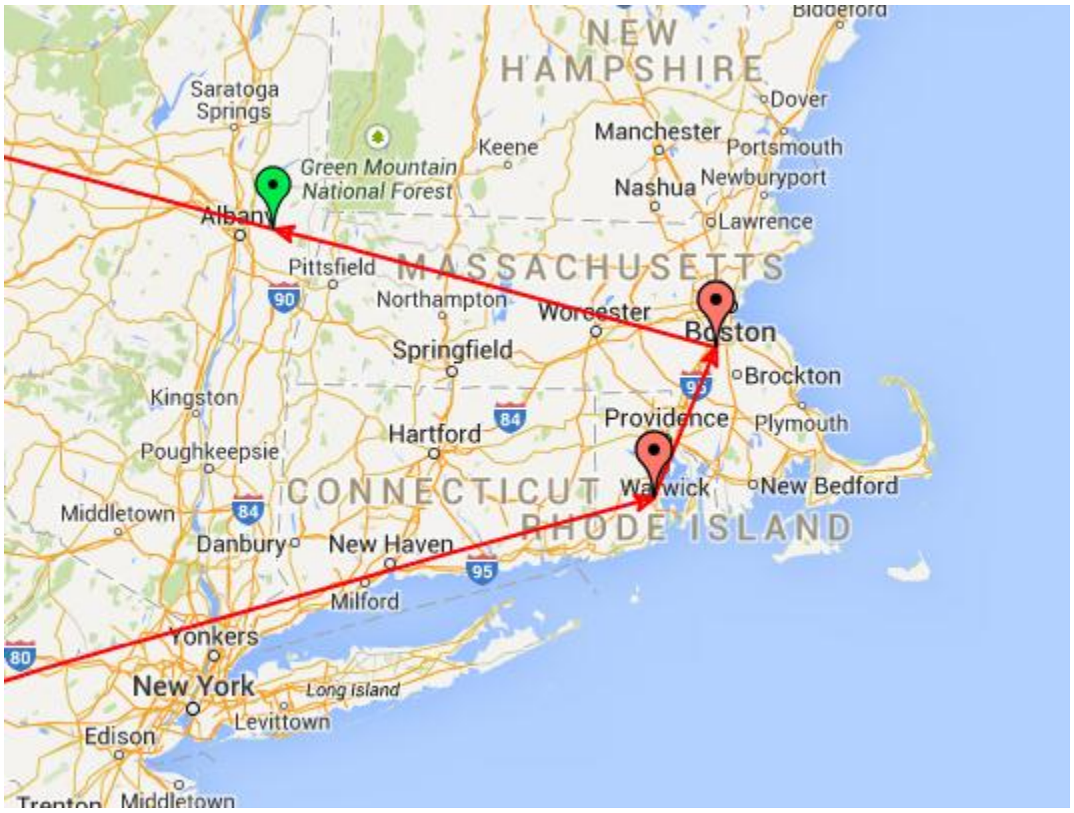
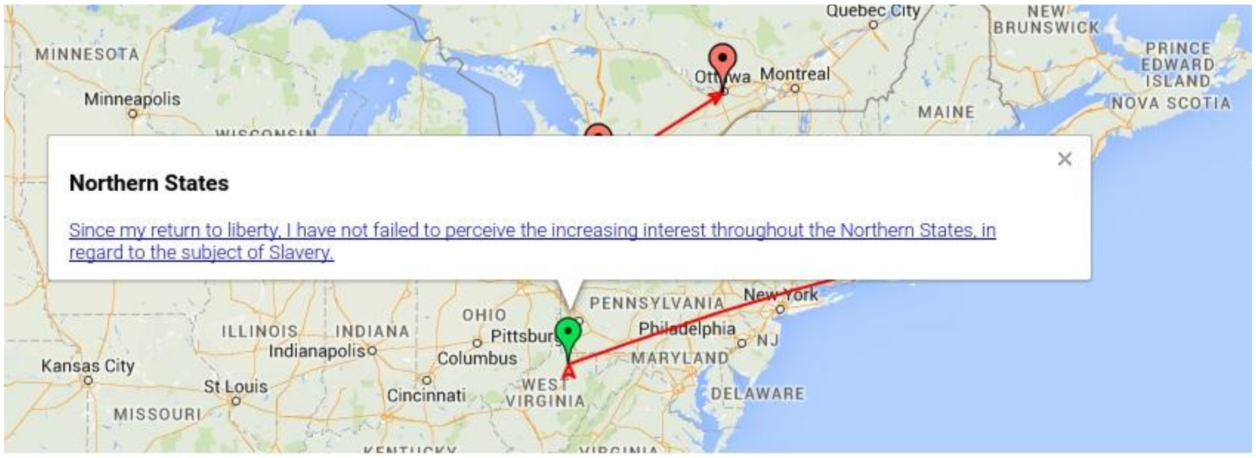


Figure 6: An Example of Colour Coded Markers and Path Lines

One of the most important aspects of the visualisation concerned the info window. An event listener was attached to each marker, and an info window was opened when the markers were clicked. In this window I included information about the location as well as the sentences from the literature that referenced it. This window was able to be formatted with HTML code. The info window was crucial in creating an interactive visualisation as it provided the context in which the place names were mentioned from the literature. The ability for the info window contents to be formatted with HTML tags allowed me to create links to other parts of the page. I used this ability to create links to the text explorer in order to jump to the exact part of the literature that the locations were mentioned. These markers were then placed in an array, and the rest of the visualisation was built around this array structure. The corresponding sentence strings were also put into their own array to be used for linking and formatting of the text explorer.


Northern States

Since my return to liberty, I have not failed to perceive the increasing interest throughout the Northern States, in regard to the subject of Slavery.

Figure 7: An Example of the Info Window and Text Explorer Link

I created a series of maps, as in the data structure, that determined the marker index of a certain location. These maps allowed me to reference a certain marker when creating other structures. An example of one map was for the sentence string to the array index of the marker.

This was important in order to create the links in the info window so they could reference places in the literature and the text explorer.

In order to show the progress of the literature through time a path line was needed. The Google Maps API allows for overlays to be placed on top of the map. The lines were created by using the coordinates of the markers, with each marker representing one end of a line. The program would then loop through all the markers, creating arrows from one marker to the next in order to indicate the path the literature took as the story progressed. Each of these path lines was added to an array, which was used as a way to control their visibility later on.

As the number of markers increased the resulting path lines started to become hard to follow. They would crisscross the map and started to look very unorganised. I needed a way to make the visualisation more legible, and this meant that I had to implement a way to filter the path lines. I created a series of controls in the form of clickable buttons at the top of the map. There were 3 different controls: remove line, restore line, and show subset.

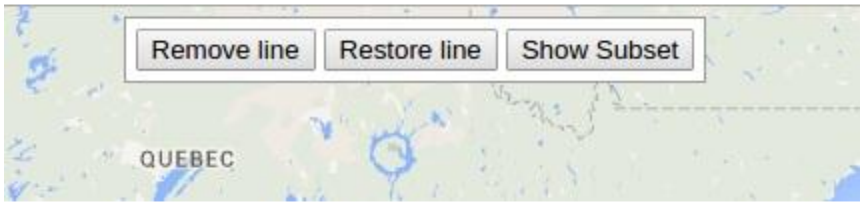


Figure 8: The Path Line Controls

These three controls acted on the path line array mentioned earlier. In order to remove the line, or to restore them, a function was created that would loop through all the individual path lines and change an attribute that determined if they were visible or not. The Show Subset control was a little more complicated. Its purpose was to show only a few path lines before and after the active marker. When you clicked a marker its index number was sent to a function that determined the two path lines before and after the active marker, and it set those attributes to visible while disabling the rest of them. This cleaned up the visualisation as well as made it easier for a user to follow the path line one step at a time, as clicking one marker would point the user in the direction of the next marker in line.

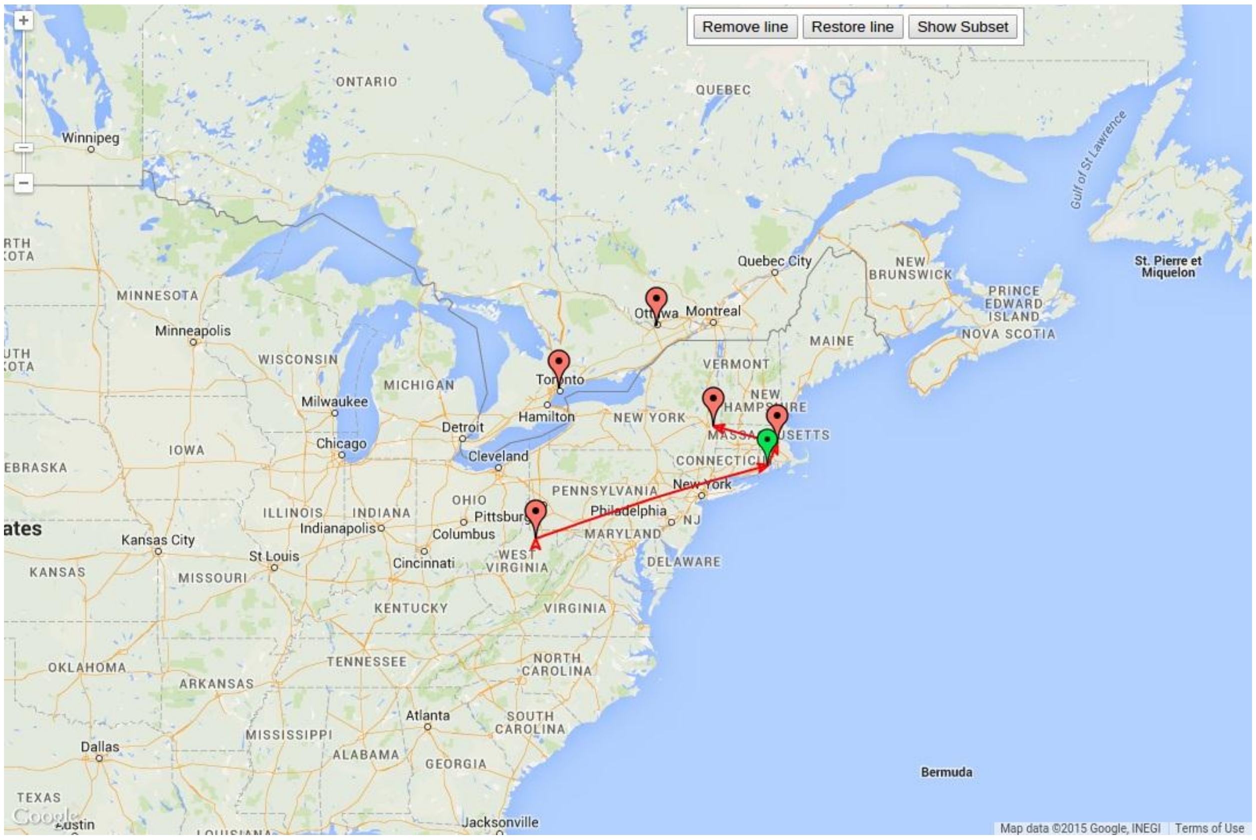
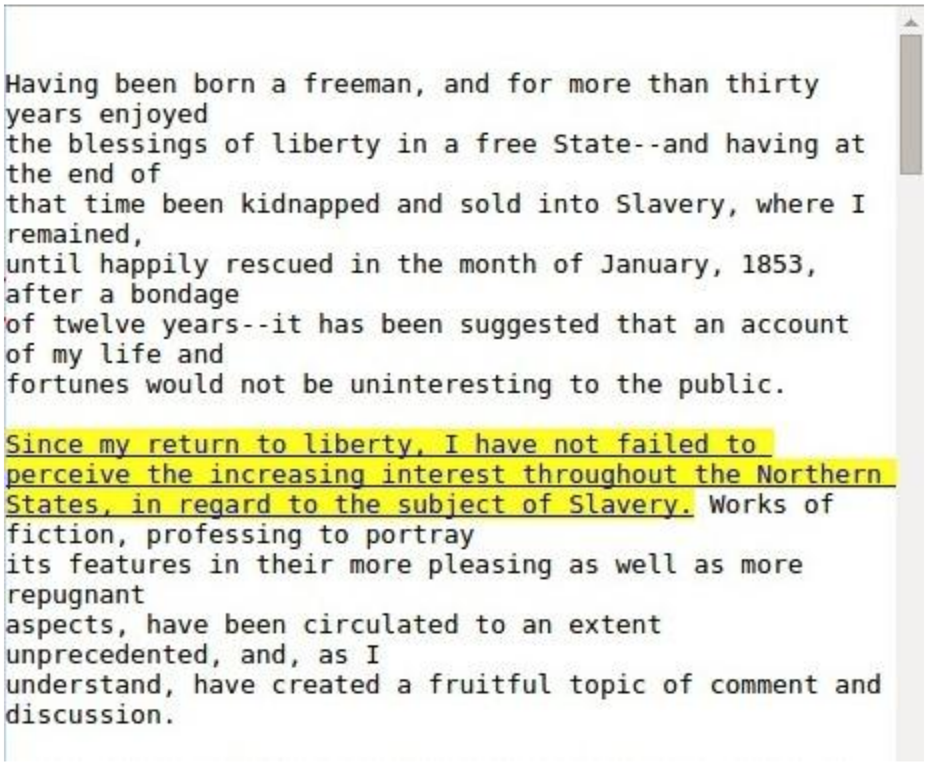


Figure 9: An Example of the Show Subset Control in Action

The final piece of the visualisation to be implemented was the text explorer. By having the complete text on hand to explore while the map was displayed, the user would be able to read the context in which the place names are mentioned and determine how settings changed as well as the situations in which the character's location changed. The literature was first imported into the JavaScript program as raw text. Then for each entry in the text array, which held a copy of the sentence strings, a function would search the raw text and surround them with anchor tags. Each anchor tag was given a unique ID, and these IDs were referenced in the marker info windows, allowing the user to jump to the exact point in the literature by clicking a point of interest on the map. There was one issue with the sentence strings though. In the process of extracting the location names, the parser stripped the sentences of their original structure. This made searching for the strings in the raw text unreliable, as if the sentence had any new line characters or formatting issues they would not match up with the raw literature text. My solution to this problem was to use regular expressions in the search to ignore any whitespace characters when matching the sentence strings.

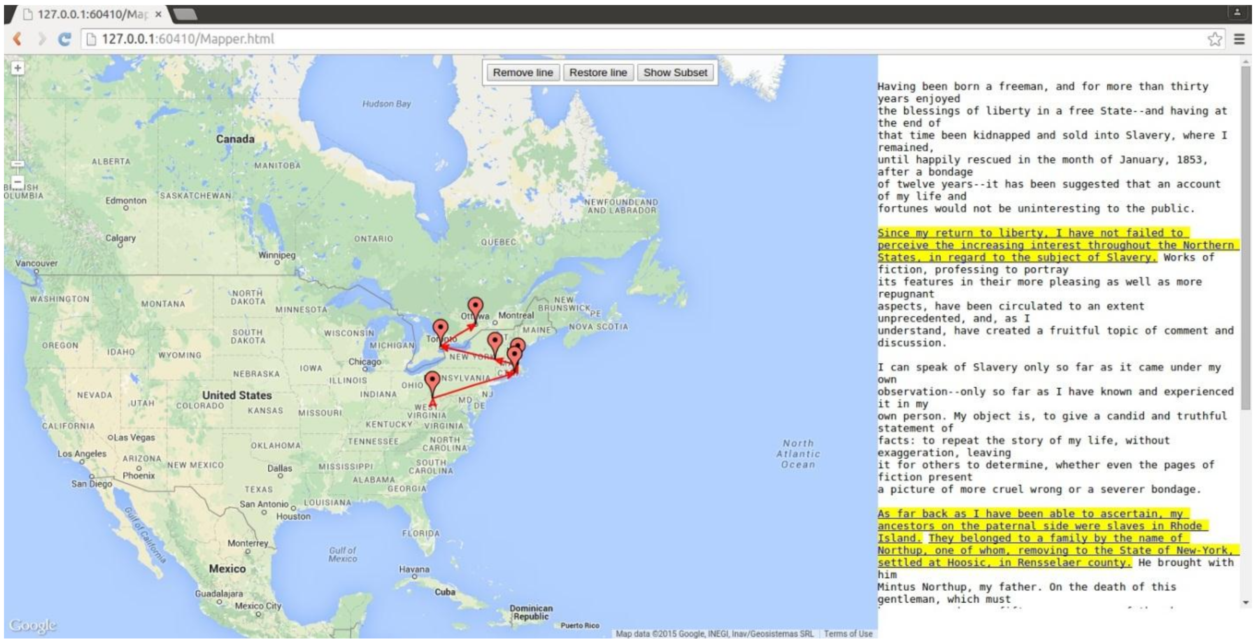


```
Having been born a freeman, and for more than thirty
years enjoyed
the blessings of liberty in a free State--and having at
the end of
that time been kidnapped and sold into Slavery, where I
remained,
until happily rescued in the month of January, 1853,
after a bondage
of twelve years--it has been suggested that an account
of my life and
fortunes would not be uninteresting to the public.

Since my return to liberty, I have not failed to
perceive the increasing interest throughout the Northern
States, in regard to the subject of Slavery. Works of
fiction, professing to portray
its features in their more pleasing as well as more
repugnant
aspects, have been circulated to an extent
unprecedented, and, as I
understand, have created a fruitful topic of comment and
discussion.
```

Figure 10: The Text Explorer with Highlighted Sentence and Link to Map Location

Once each sentence was surrounded with tags, the text was displayed in an HTML pre element in order to preserve the spaces and the new line formatting. The tags also allowed the strings to be highlighted, as well as turn them into clickable elements. When each highlighted entry was clicked, it sent the sentence string to a function that centred the map on the corresponding location, providing the user a way to explore the text of the literature and provide interactivity with the map. The final result was a web based visualisation of the locations in a piece of literature that provided a way to explore both the distances involved and the text in a fluid and easy to use manor.



```
Having been born a freeman, and for more than thirty
years enjoyed
the blessings of liberty in a free State--and having at
the end of
that time been kidnapped and sold into Slavery, where I
remained,
until happily rescued in the month of January, 1853,
after a bondage
of twelve years--it has been suggested that an account
of my life and
fortunes would not be uninteresting to the public.

Since my return to liberty, I have not failed to
perceive the increasing interest throughout the Northern
States, in regard to the subject of Slavery. Works of
fiction, professing to portray
its features in their more pleasing as well as more
repugnant
aspects, have been circulated to an extent
unprecedented, and, as I
understand, have created a fruitful topic of comment and
discussion.

I can speak of Slavery only so far as it came under my
own
observation--only so far as I have known and experienced
it in my
own person. My object is, to give a candid and truthful
statement of
facts: to repeat the story of my life, without
exaggeration, leaving
it for others to determine, whether even the pages of
fiction present
a picture of more cruel wrong or a severer bondage.

As far back as I have been able to ascertain, my
ancestors on the paternal side were slaves in Rhode
Island. They belonged to a family by the name of
Northup, one of whom, removing to the State of New-York,
settled at Hoosic, in Rensselaer county. He brought with
him
Mintus Northup, my father. On the death of this
gentleman, which must
```

Figure 11: The Completed PathVis Visualisation

5. Summary and Conclusion

5.1 Summary

After researching different types of literature visualisations, I determined that mapping out the scenes and locations was relatively uncommon, and that most of the map visualisations were parsed and composed by hand. This led to the development of a dynamically generated, online literature mapper for my thesis project. Using a LAMP server as the back end for the application, I developed an application that parsed and extracted locations in a piece of literature with the help of the Stanford Named Entity Recogniser. Using these location names, coordinates were then found by querying a GeoNames database as well as sending requests through the Google Geocode API. Finally a map visualisation was created that plotted the different locations in the uploaded literature as well as displaying the text in its entirety, allowing the user to visualise the distances involved in a story as well as well explore the text in order to gain relevant context.

5.2 Conclusion

While developing PathVis for the thesis project, I gained valuable experience in project planning and developed an understanding of the work involved in order to create a software application. By creating an online application I gained experience in software languages that I had never used before, as well as the server side of applications. Through many different stages I developed an online application that can aid readers and users in achieving a higher understanding of their favourite literature pieces, as well as helping users conceptualise the vast distances involved as their stories progress.

6. Future Work

6.1 Biasing

Although a working prototype of PathVis was created for my thesis project, there are still improvements that can be made in order to increase the accuracy of the map produced in its reflection of the underlying literature. One of the problems faced in my implementation was the issue of ambiguous place names. Some cities and locations do not have a unique names, and determining which location is the right one can have a vast impact on the overall path that a piece of literature takes. An example of this problem are the two cities: London, Ontario and London, England. One solution to this problem is to apply biases on the place names in order for preferences to be made about the ambiguous cases. By applying biases the visualisation can choose the most reasonable place name and increase the accuracy. One way of applying biases is to give preferences for certain countries or continents. If your results are biased towards Europe for example, you could discard London, Ontario and choose London, England. The GeoNames back end database currently holds country information. By giving the user an option to set the bias, from a drop down menu or some other interface, PathVis can apply that information in its queries in order to find the correct location. Currently due to limitations in the Google Geocode API, you can only bias the results on a single country, which precludes continent wide biases. In the future Google may make improvements that can allow multiple country biasing, or even continent wide biasing.

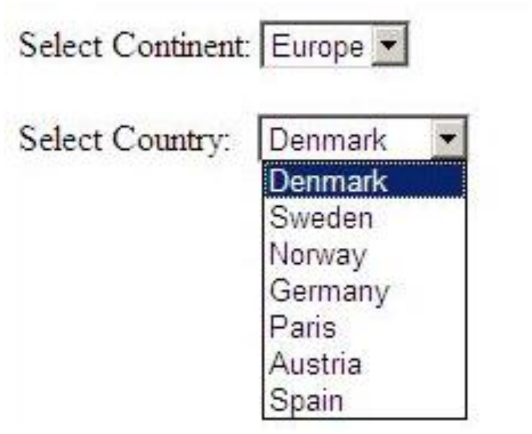


Figure 12: An Example of how a Biasing Interface Might Look Like

Image obtained from: http://rahulrajatsingh.com/wp-content/uploads/2014/06/page.jpg

Another way to reduce ambiguity is to apply biases based on an algorithm. One such algorithm could be based on the average of previous location coordinates that are explicit and have no ambiguity. By computing the centroid of a group of coordinates, and applying a radius or bounding box to this centroid, ambiguous cases could be eliminated. Again I will use the London, Ontario and London, England as an example. If the first 5 locations are unambiguous cities in North America and the ambiguous location name London appears next in line, then by computing the centroid of the 5 cities and applying an effective range from that centroid London England can be eliminated as an option. This is because it is too far in distance and out of the biasing range. Currently Google Geocode allows for biasing to be made according to a bounding box based on coordinate points. This could be applied with the centroid algorithm to increase the accuracy of the PathVis application.

6.2 Future Applications

Though the PathVis application was developed to visualise locations in literate and stories, its text based parsing can be applied to other applications as well. One such application could be travel itineraries. By uploading a travel itinerary the user can get an idea about how their trip will progress and can allow for changes or adjustments that can improve their overall enjoyment. An example of this could be someone visiting a city for the first time. If they upload their itinerary and find that their path criss-crosses the city many times, they may be able to change their schedule and group certain activities and locations based on their distances from each other, and spend less time travelling from location to location and more time enjoying their trip.

Another application that PathVis could be adapted to would be to plot someone's path based on an uploaded photo album. Currently a large majority of people take pictures with their cell phones. These pictures contain a set of attributes that are called EXIF data. EXIF data contains information about the date the picture was taken, as well as a set of GPS coordinates from where the picture was taken if the option is enabled on the phone. As PathVis's implimentatoin heavily involves mapping and coordinates, it could be adapted to create a map visualisation based on photo EXIF data rather than parsing locations from a text based source. This could increase the overall versatility of the PathVis application.

7. References

[1] *Interactive Game of Thrones Map with Spoilers Control.* [Online] Available: http://quartermaester.info/

[2] E. Johansson (Jan, 2012). *Interactive Map of Middle Earth – LotrProject*. [Online] Available: http://lotrproject.com/map/#zoom=3&lat=-1315.5&lon=1500&layers=B

[3] C. D. Manning et al. "The Stanford CoreNLP Natural Language Processing Toolkit". In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2014, pp. 55-60.

[4] J. R. Finkel et al. "Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling". *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics* (ACL 2005), 2005, pp. 363-370.

[5] Unxos GmbH. *GeoNames.* [Online] Available: http://www.geonames.org/

[6] Google. Google Maps JavaScript API v3. [Online] Available: https://developers.google.com/maps/documentation/javascript/reference

[7] Google. The Google Geocoding API. [Online] Available: https://developers.google.com/maps/documentation/geocoding/