IIR Final Project Report

Jin Guo

Introduction

The state-of-art search engines have made great effort on supporting user to form their queries, such as query auto-completion, spell correction, disambiguation, etc. Because certain gap exists between user's actual information need and the queries they form in the first time, their queries might evolve as they receive more information about the topic they concern. The techniques I mentioned above, however, might not be very helpful for supporting user's query evolution. The main goal of this project is to support user during their query evolution, indicating them other closely connect concepts and the concrete corresponding relationship between these concepts the query they typed in.

The key idea in this project is to select candidate concepts utilizing knowledge base from Wikipedia. DBpedia is such a project which extracts various kinds of structured information from Wikipedia and combines them into a huge, cross-domain knowledge base. This information includes info box templates, categorization information, image, geo-coordinates, links to external web pages, etc. However, because each entity in DBpedia normally contains a great number of related information, simple providing everything to user without selection would only be confusing and causing them more time to select the ones with the closest relationship. Moreover, the "closest" relationship is different for topics from different domain, only extract some specific relationship might be flawed as well. The info box structure from Wikipedia provides small size of important properties, but this structure is not available for every entry.

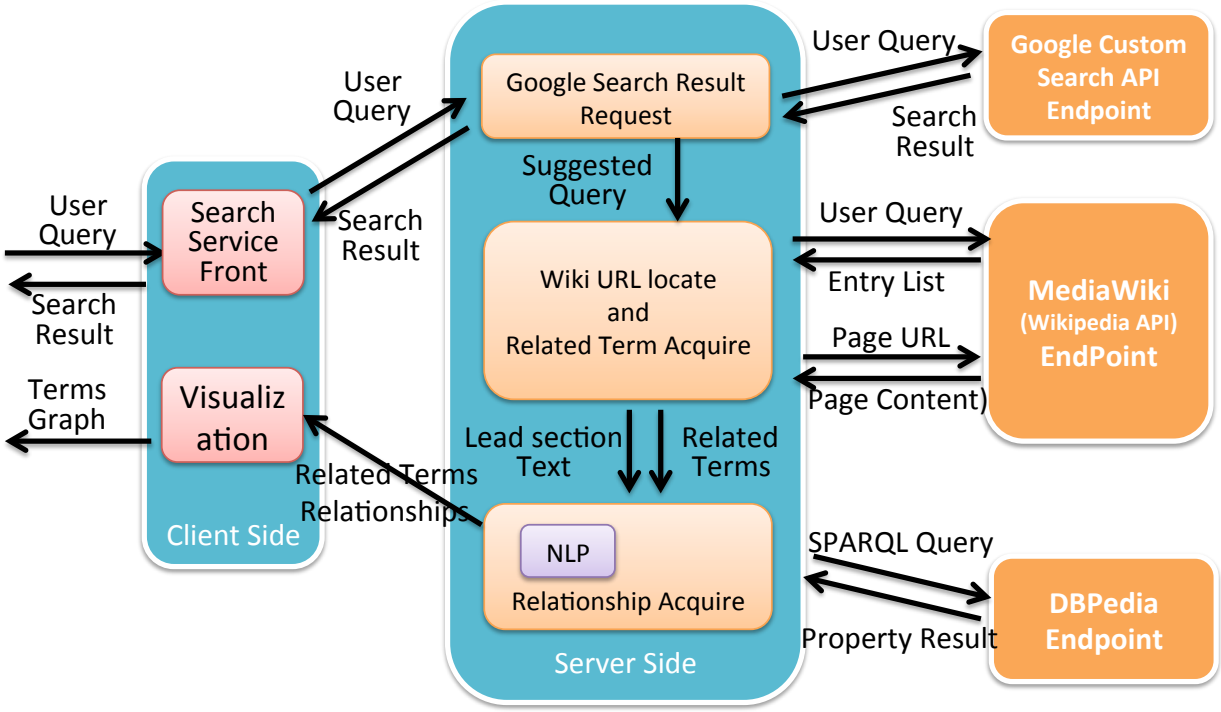
In this project, I combine the usage of Wikipedia and DBpedia, the extracted knowledge base from Wikipedia. Concretely, I use Wikipedia content layout to locate the closest concepts first, and then use DBpedia to extract their specific relationships. The "lead" section of each Wikipedia entry is extracted because: 1. it normally contains important related concepts; (based on Wikipedia style manual, lead section "should define the topic, establish context, explain why the subject is interesting or notable, and summarize the most important points—including any notable controversies.") 2. it's available for most entries (the entries without lead section normally contain link to redirect to pages with one). In the lead section, words/phrases with hyperlink to another web page are selected because they are normally the key related concepts and it's easy to pinpoint. Then the each hyperlinks are searched through the DBpedia dataset to extract the defined relationship.

There's another reason that we want to also use Wikipedia. The DBpedia knowledge base is created through processing Wikipedia pages, and represented by RDF, a directed, labeled graph data format for representing information in the Web. We can use SPARQL query language to access it, but like conventional Database query language, SPARQL cannot find and rank the "relevant" document. Since in most cases, the user query would not be exactly the same with the entity names in DBpedia, we need to use Wikipedia API to search the title of the most related page first, and find the correspondent DBpedia resource based on the title.

DBpedia is not a complete knowledge base like any other one. As attempting, for the concept without relationship defined in DBpedia, some NLP techniques are used to extract the relationship. Because of short developing time for this project and the requirement of fast processing speed for real time user interaction, the accuracy for the relationship extraction using NLP is currently far from satisfying. Other limitations and expected improvement of this project will be discussed in the later section.

System Component

The structure of the system is shown in the following figure:



Main components are described in detail as follows:

Search Internet with user query

User query is sent directly to Google Custom Search (GCS) API, and corresponding search result is returned in format JSON. The result includes meta-information about this search and the returned web sites. The title, URL, and snippet of the top returned web sites are displayed to user.

Acquire lead section from related Wikipedia page and extract term/phrases with link

Acquire the most related Wikipedia page to user query is achieved through MediaWiki API. For example, when request the search with query “chicago subway”, the list contains pages of “Chicago Transit Authority", “Chicago 'L'”, “Blue Line (CTA)”, etc. will be returned. Because no spell correction is processed during the search, the function of the search suggestion from to Google Custom Search is

used. When suspected misspell happens, GCS search result will include the query suggestion as metadata about this search. Therefore, instead the original query user type in, this suggested query returned by GCS in the last step would be used to search Wikipedia.

The Wikipedia page ranking on top will be used ("Chicago Transit Authority" in the last example), and the lead section in that page will be extracted. Then process the lead section to achieve the links that appear in it, as well as the plain text (for later NLP).

Extract relationship from DBpedia and NLP

Form SPARQL queries to search through the resource in DBpedia knowledge base that is corresponding to Wikipedia page from the last step; find if the links are values of any properties. The titles of the links that appears as value of properties are considered as closely related to the original query while the properties' names are their relationship. Other links are considered as less important.

For the less important links, their position in the original text from the lead section is located. Because all Wikipedia entries are "things", the relationship between two entries in Wikipedia is normally the verb except the case of a single *be* verb. For instance, in the Wikipedia page of "Google", the there's a link in the sentence "In 2006, the company moved to its headquarters in Mountain View, California". We can consider the verb "move to" as the relationship between "Google" and "Mountain View, California". In the case of the *be* verb, relationship should be the noun phrase appears as subject when the link appears in the object. For instance, in the sentence "company's unofficial slogan is 'Don't be evil'", the relationship between Google and "Don't be evil" is expected to be "company's unofficial slogan" (omitting *is*). Using the above analysis, NLP is done by Gate framework. After a series of process of Tokenizing, Sentence Splitting, and POS tagging, the words representing relationship are extracted.

Display to user with interaction

The server side will return a list of related terms/phrases and available relationship extracted from DBpedia and NLP, and the client side will show the result to user as a graph. The key concept from the user query is shown as the center node in the graph while other related terms/phrases is shown as nodes connected to the center one. Their name and relationship is shown on the node. Because the terms/phrases with relationship extracted from DBpedia are more important, and the relationships are more reliable comparing to NLP, they are shown bigger with apparent color. The nodes with no extracted relationship are displayed the smallest.

User can drag each node for clearer view, and also click the node to add the corresponding terms/phrases to their query.

Deployment

To ensure the availability, I use Google App Engine service for deployment. The project is deployed to Google cloud, and can be accessed through the link: http://equery-web.appspot.com/ without limitation of time and IP address. But there comes a problem with this deployment. The NLP framework Gate contains the usage of restricted class for Google App Engine; therefore the NLP module in this project

cannot be used. Because other deployment is not available in short time (the internet service I am using (AT&T U-verse) doesn't allow server hosting), the version without NLP is used for user evaluation. In the example scenario section, the local version with NLP is used.

The comparison between returned graph with and without NLP is shown below (with query "Sutton Foster"):

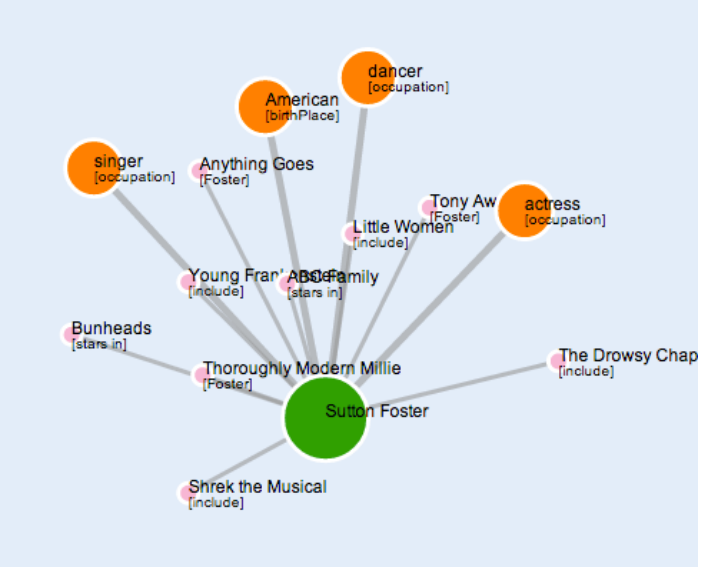


Figure: the returned graph with NLP module

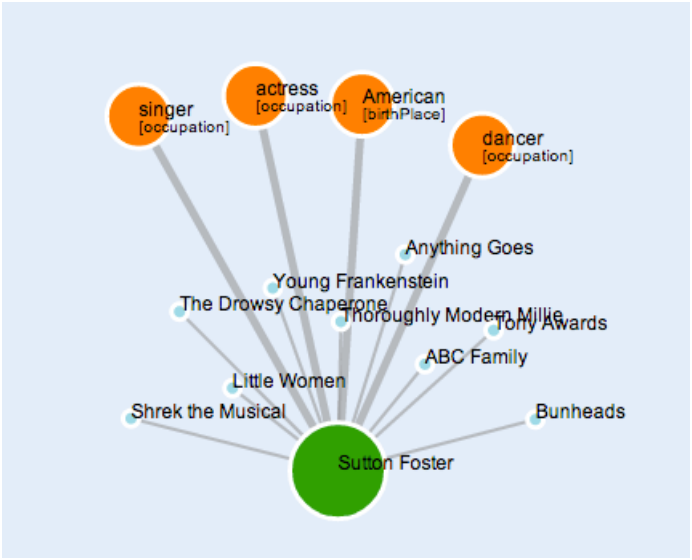
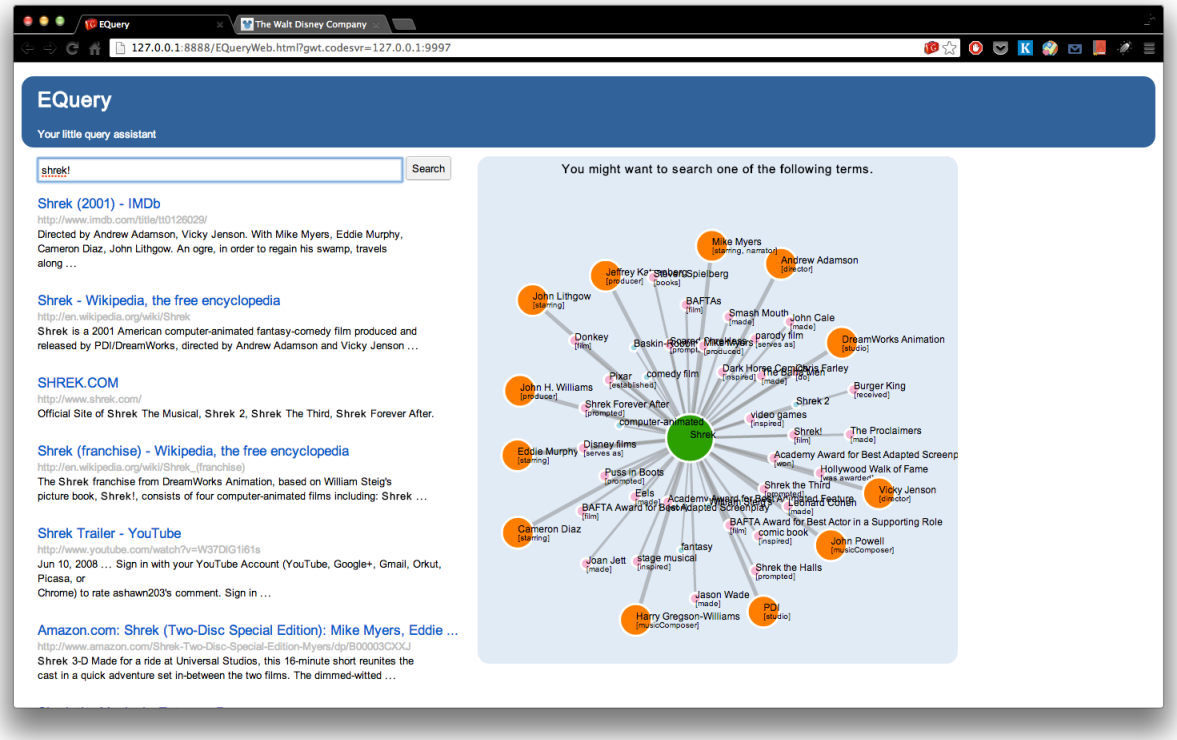


Figure: the returned graph without NLP module

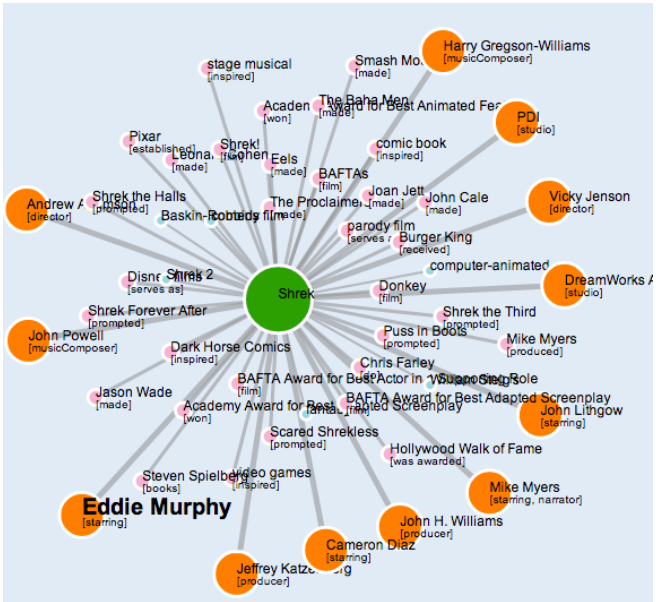
They both show the terms/phrases with relationship from DBpedia in a big orange dot, and the terms/phrases without any relationship in small blue dot. With NLP, the system will show the terms/phrases with relationship extracted by NLP in pink dot.

Example Scenario

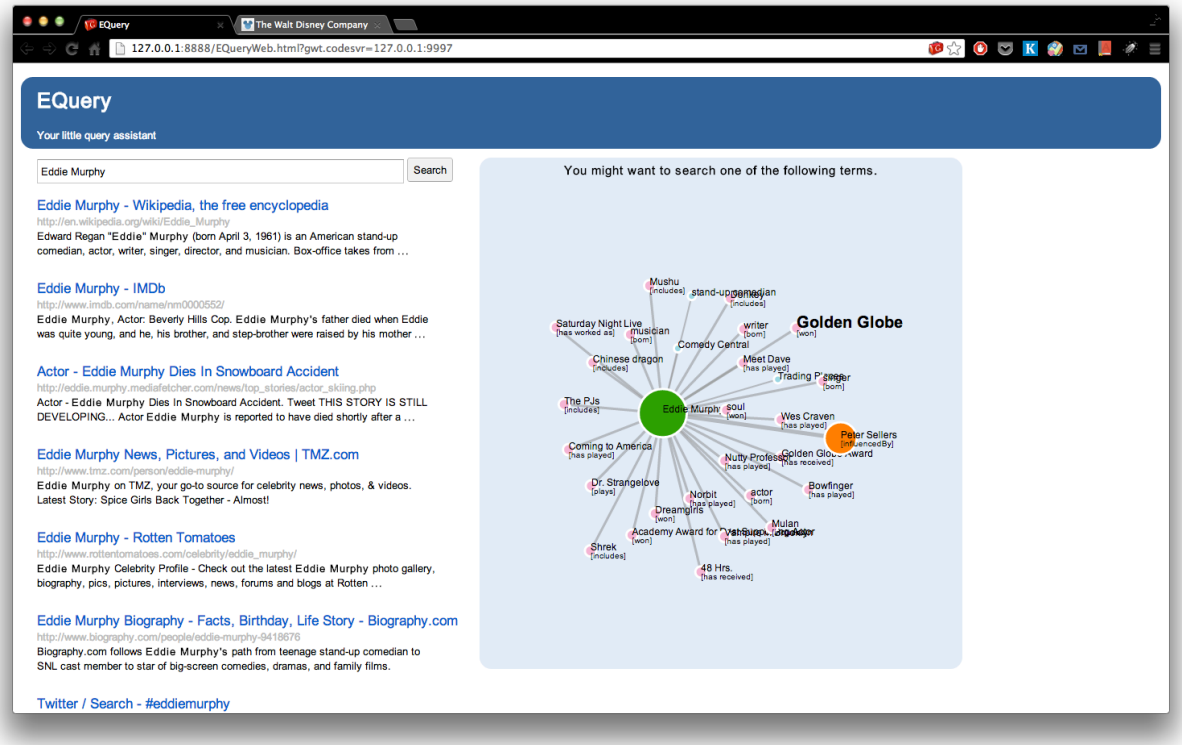
I've heard one movie named "Shrek!" from my friend, and I want to know more about it. So I typed the name "Shrek!" in the search box.



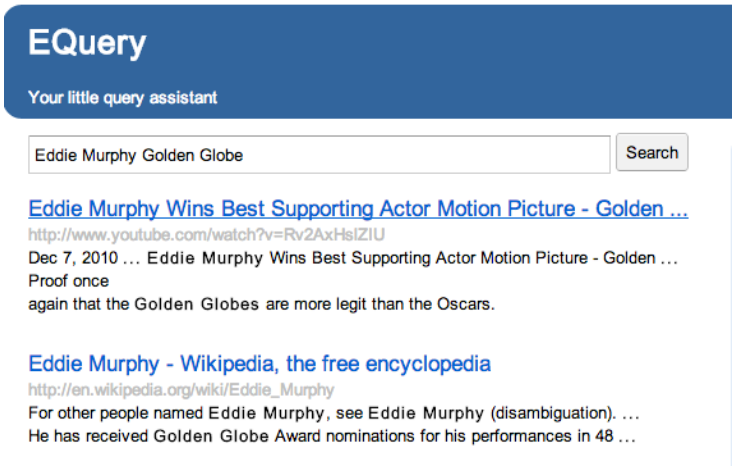
After glancing the returned graph, I find that the actor Eddy Murphy stars in this movie. I like some of his other works, so I'd like to know more about him. So I click his name from the graph to add it to the search box, and delete the original query "shrek!".



The graph with related terms about Eddie Murphy is returned along with the search result. I find that he's won Golden Globe before. I'd like to know more about this. So I click "Golden Globe" to add it to my query.



After searching with query "Eddie Murphy Golden Globe", the video of this event is shown as the first result. I click this link and complete my search for now.



Evaluation

Participant:

4 Ph.D. students participated in the evaluation of this system including two in Computer Science and the other two in Electrical Power System.

Method:

The evaluation method includes user observation and survey. Each user will be asked to search anything they are interested in. They can interact with the returned graph representing the related terms/phrase if necessary. They can reform their query continuing their search until they find the web site they want to click. The observation process ends when they click any external web site.

After the observation, each user will be asked to complete a survey as follows:

1. The displayed terms/phrases are related and important to my interest.
2. The indicated relationships along with the term/phrases are informative.
3. Graph is an effective way to represent related terms/phrases.
4. Overall, this query evolution tool is helpful during my searching.
5. Do you have any suggestions?

For question 1-4, please give a number between 1 and 5 where 1 represents strong disagreement and 5 represents strong agreement.

Results:

Observation Result:

	User1		
	Query Evolving Sequence	Returned Graph	Modify Query Based on Graph
1	board game	11 Blue	Yes
2	Dungeon & Dragon	5 Orange 19 Blue	Yes
3	Gary Gygax	3 Orange 16 Bule	Yes
4	Castles & Crusades	10 Blue	-

	User2		
	Query Evolving Sequence	Returned Graph	Modify Query Based on Graph
1	medical insurance	3 Blue	Yes
2	health system	4 Blue	Yes
3	health system governments	4 Blue	-

	User3		
	Query Evolving Sequence	Returned Graph	Modify Query Based on Graph
1	android	2 orange	Yes
2	Android (operating system)	6 Orange 18 Blue	Yes
3	Android (operating system) Java	38 Blue	No
4	Android (operating system) Java sdk	3 Orange 19 Blue	-

	User4		
	Query	Returned Graph	Modify Query Based on Graph
1	ferris wheel	8 Blue	Yes
2	World's Columbian Exposition	21 Blue	-

Survey result:

Question	User1	User2	User3	User4
1	4	5	3	4
2	5	4	5	5
3	4	4	5	4
4	5	5	4	5
5	It should support search history navigation. Sometimes I'd like to jump back to the query in several searching actions ago.	The graph more complex. Give different term different weight I can have better clue about their importance.	The accuracy needs to be higher. The central topic sometimes is not what I want to search	The terms without any relationship would lose my attention. It's better for every related term to have relationship.

Discussion:

The first user represents the typical user with query migration behavior. They don't have a clear goal of their search in mind at the beginning; they want to know more about different aspects of a topic, and might want to go back and forth during searching. The third user also doesn't have a clear goal in the beginning when he typed in Android. When he found Java is related to Android, however, he became another type who has a clear search goal in mind - Java SDK for Android development. He complains about the returned graph is not so relevant to his thoughts, and formed the query by himself and found the link he wanted. Even so, the tool helped him clarify the search goal in the first place. Overall, they are all satisfied about this tool and think it's useful. Their opinion also exposes some important problems of the current system. Some is easy to respond such as adding query history navigation, while others are rather difficult and needs deeper investigation. This content will be discussed further in the next section.

Limitation and Expected Improvement

Firstly, more precise structure of the connections needs to be extracted. The related terms/phrases should form a complex net with more inter links, rather than a simple graph with all nodes connected to the center one. For instance, when searching “Google”, the extracted terms “Stanford University” should be linked to other extracted terms “Larry Page” and “Sergey Brin” instead of being linked directly to “Google”. The might be achieved by deeper mining of DBpedia knowledge base.

Secondly, the accuracy of NLP relationship extraction is relatively low. Currently, only simple processing has been done. More complex semantic analysis needs to be done if we want to achieve high precision. However, we also have to consider the trade-off precision and processing time.

Furthermore, user query preprocessing is necessary when it is complex and contains several key concepts. In this case, it might be better to identify each key concept and do the further processing individually.

Finally, user feedback analysis is lacking for this project. The importance of the related terms might be determined by user behavior analysis. We can take the nodes clicked more than others as important nodes, and display them in a more apparent way. When the related terms are too many to display in a clear way, the nodes never been clicked might be trimmed.

Outside Source

MediaWiki provides the Wikipedia API. In this project, search through Wikipedia content with user query and acquire the lead section is achieved by MediaWiki.

DBpedia is knowledge base with extracted structured information from Wikipedia. It allows people to make sophisticated queries against Wikipedia, and to link other data sets on the Web to Wikipedia data. The relation extraction in this project is done by access DBpedia.

Jena is a collection of Java libraries to develop semantic web and linked-data apps, tools and servers. In this project it is used for creating SPARQL query and accessing the DBpedia endpoint.

Gate is a framework for NLP. It is used in this project for extracting relationship for the terms/phrase without relationships found in DBpedia.

Google Web Toolkit (GWT) is a development toolkit for building browser-based applications. It handles the client-server communication, and allows writing the client side application in Java.

Google App (GAE) is a platform-as-a-service cloud computing platform for developing and hosting web applications in Google-managed data centers.

D3.js is a JavaScript library for manipulating documents based on data. In this project, the visualization of related terms/phrases is achieved by D3.