XLike
FP7-ICT-2011.4.2
Contract no.: 288342
www.xlike.org

Deliverable D6.4.2

Final Prototype Front-End

Editor:	Yixin Cao, THU
Author(s):	Yixin Cao, THU; Peng Zhang, THU; Zhixing Li, THU.
Deliverable Nature:	Report
Dissemination Level: (Confidentiality)	Public (PU)
Contractual Delivery Date:	M33
Actual Delivery Date:	M33
Suggested Readers:	Developers creating software components to be integrated, developers creating case study prototypes, software developers.
Version:	1.0
Keywords:	toolkit; visualization; components;

Disclaimer

Full Project Title:	Cross-lingual Knowledge Extraction
Short Project Title:	XLike
Number and Title of Work package:	WP6 – Integration and Toolkit
Document Title:	D6.4.2 – Final Prototype Front-End
Editor (Name, Affiliation)	
Work package Leader (Name, affiliation)	Esteban García-Cuesta, iSOCO
Estimation of PM spent on the deliverable:	9PM

Executive Summary

This deliverable presents the final prototype front-end which designs and implements a UI toolkit to enable a fast and flexible prototyping of rich end-user applications. Six components, independent of each other, are integrated in the toolkit. By providing a formalized interface, diversified data can be displayed conveniently to fulfil the requirements of the use cases. The goal of toolkit is to help people understand abstract statistical data more efficient in a visualized way.

According to the Bloomberg and STA use cases, seven components are well designed to display and lay stress on different aspects of diversified data. For entity tracking, Entity graph, one of the toolkit components, displays entities and the complex relations among them. Whereas, Radar chart focuses on different features of one entity. For article tracking, Timeline is provided which is a widely-used visualization component to organize news articles chronologically, and Time-Geo component aims to showing the development of an event dynamically in time and location. For relation tracking, Chord diagram and Radial tree are designed for network and hierarchical relation respectively. For topic tracking, Stream chart visualizes the changing of data stream.

All the components are programmed in JavaScript, which is most commonly used as part of web browsers, and this allows for high flexibility and scalability. The users rarely need to deploy a complex development environment, which is apparently against our goal of fast prototyping of rich desktop clients and Web front-ends.

Table of Contents

List of Figures

List of Tables

Abbreviations

API	Application Programming Interface
D	Deliverable
SOA	Service Oriented Architecture
T	Task
URL	Uniform Resource Locator
WP	Work Package

Definitions

Pipeline	Refers to the flux of different processes which are applied to a set of raw data in order to analyse it and interpret it. In XLike project. It covers the following phases: gathering data, pre-processing data, application of Natural Language Processing Tools, semantic interpretation, visualization, and finally domain interpretation

1 Introduction

This deliverable presents the final prototype front-end which designs and implements a UI toolkit to enable a fast and flexible prototyping of rich end-user applications. Seven components, independent of each other, are integrated in the toolkit. Each component provides a formalized interface to structured data, and displays diversified data conveniently to fulfil the requirements of the use cases. This may reveal the internal connection of entities included in news articles or other texts online, which is difficult to identify from data itself. The goal of toolkit is to help people understand abstract statistical data more efficient in a visualized way.

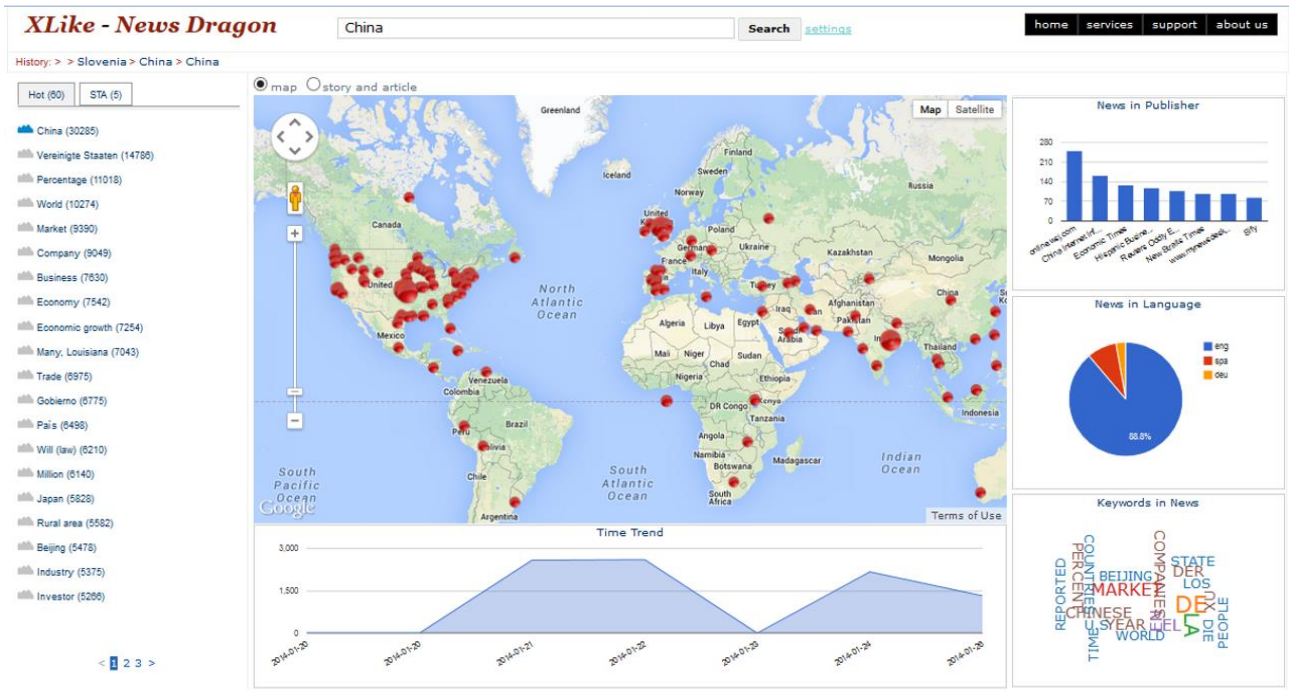

Figure 1 XLike News Dragon Demo

Figure 1 shows an example of visualizing the abstract data to fulfil the requirements of STA use case by combining several different visualization toolkits. For instance, a Time-Geo component listed in the central of this web page shows the geographic distribution of the news articles, which belongs to the same/nearby place are clustered together and shown in the map as a red pie. The size of the pie indicates the number of articles. In this way, users will find what news are reported in which place.

1.1 Final prototype front-end architecture

The final prototype front-end aims at integrating such a list of independent visualization components into a UI toolkit, so that developers can display their diversified data flexibly within a lightweight framework.

According to the previous work of T5.2, we summarize the use cases of the Bloomberg and STA that there are some common required functionalities, such as tracking articles by entities, which implies how to display entity is significant. Similarly, article/event, relations of both entities and articles, and topic are also the basic elements to be displayed.

Therefore, seven components are well designed to display and lay stress on different aspects of diversified data. For entity tracking, Entity graph, one of the toolkit components, displays entities and the complex relations among them. Whereas, Radar chart focuses on different features of one entity. For article tracking, Timeline is provided which is a widely-used visualization component to organize news articles chronologically, and Time-Geo component aims to showing the development of an event dynamically in

time and location. For relation tracking, Chord diagram and Radial tree are designed for network and hierarchical relation respectively. For topic tracking, Stream chart visualizes the changing of data stream, especially using the layers to represent the number of topics.

Figure 2 shows the toolkit architecture, which is easy to understand due to its independence of components. We use structured data as input, and the data format will be detailed in section 4.1. Seven visualization components, each of which can be invoked independently, are integrated into the UI toolkit. The thumbnails of them are listed at the right area.

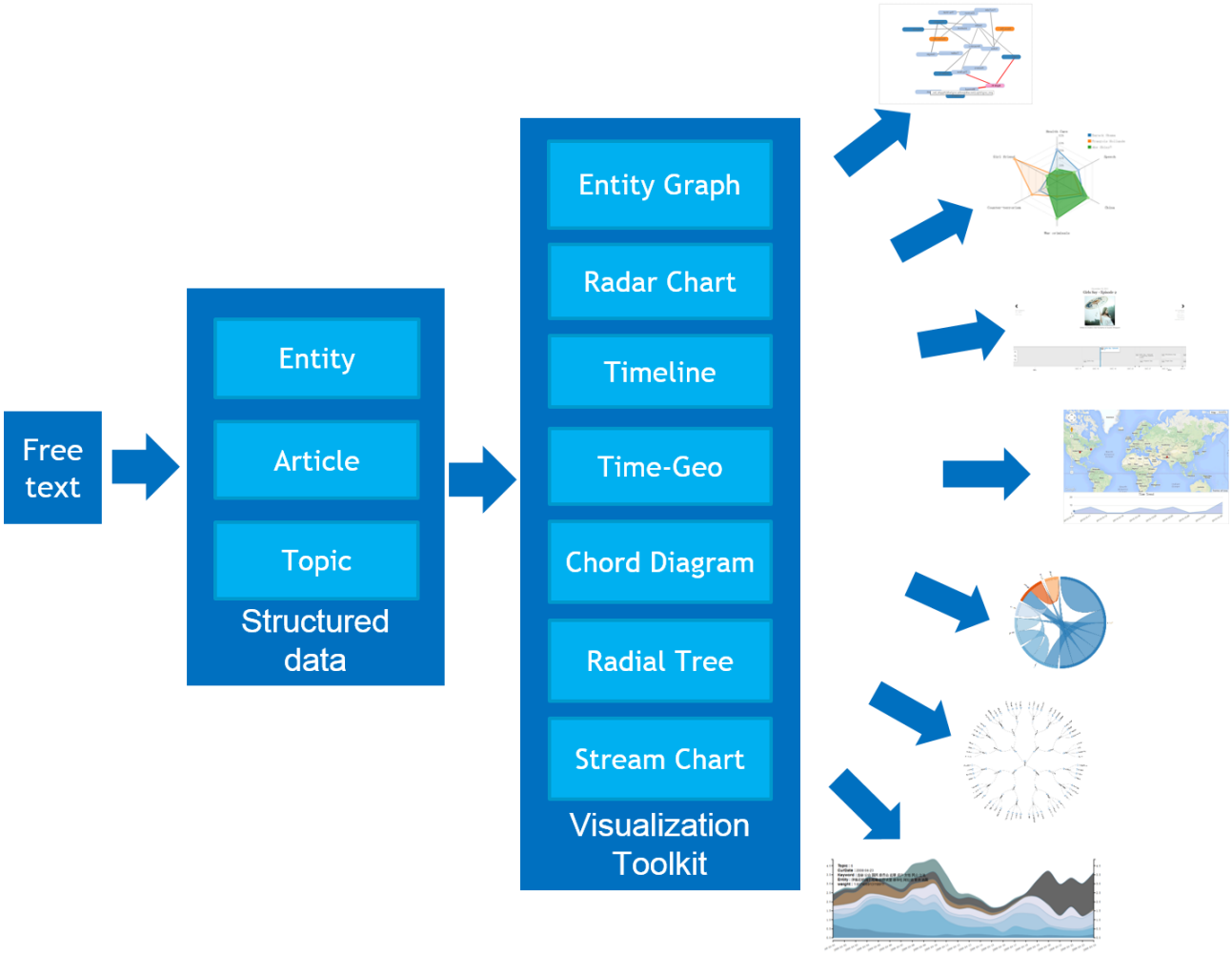

Figure 2 Final prototype front-end architecture

All the components are programmed in JavaScript, which is most commonly used as part of web browsers, and this allows for high flexibility and scalability. The users rarely need to deploy a complex development environment, which is apparently against our goal of fast prototyping of rich desktop clients and Web front-ends. As long as having a web browser, the users can invoke proper visualization component to display their data.

It should be noticed that the toolkit is based on several popular JavaScript libraries or framework. Using D3 JavaScript library, our JavaScript based components are easily developed, which provided a stable flexible framework. In addition, Google chart are also powerful tools. Simple to use, cross-browser compatibility and cross-platform portability allows for high flexibility and scalability. Based on these tools, all of the components of the UI toolkit are well encapsulated. For the convenience of easy integration of these controls in target systems, formalized interfaces are defined in all the controls.

This deliverable is structured as follows. Section 2 introduced the two use cases briefly. Section 3 details these components with the aspects of function and interactions. The predefined data format and depended libraries or services such as D3.js, Google chart and Google map would be introduced in Section 4. In Section 5, a conclusion will be made. Some references can be found in References.

2 Use case scenarios

This section introduces the two use case scenarios on which the final prototype front-end has been focus. As mentioned in our previous work of T5.2, three requirements are summarized, i.e., entity tracking, article tracking and topic tracking, which we have provided corresponding visualization components described in D6.4.1. This deliverable as an update to D6.4.1 added two new components tracking relations, another basic element for use cases.

First, we review the requirements of two use cases and then detailed all the visualization components in the UI toolkit according to their functionalities in the following section.

2.1 STA Use Case

- Article Tracking

Given an article, users may want to find more articles which is related or similar to this article in different languages. In monolingual environment, article tracking requires the ability of finding duplicates of the given article. In cross-lingual cases, this function requires that the prototype has the ability of understanding the given article to some extent and then find similar articles in different languages. Besides, these two function, article tracking also requires the ability of finding articles which are a part of the given article or contain the given article.

- Entity Tracking

Entities are the important and meaningful instances or objects. Entity tracking aims to tracking the articles which contain some specific entities. In previous modules, we are able to recognize the entities (Names, Locations, Organizations and so on) from plain text. Both these entities and the custom entities defined by STA are included in Entity tracking.

- Topic Tracking

Topic tracking refers to finding the related articles given a topic. STA has categorised their custom entities into different groups. However, these groups are too general as topics and in practice each of these group usually matches large amount of articles. A more practice way is clustering articles into different topic collections and each of these collections stands for a topic. In this way, it is more practical to tracking articles belong to the same topic.

2.2 Bloomberg Use Case

- Entity tracking

The goal of entity tracking in the Bloomberg use case is to maintain an up-to-date list of relevant company news, preferably from their home markets. The task can be roughly defined in two steps: (1) detect mentions of the entity in the multi-lingual news stream, and (2) determine which four are most suitable to be displayed on the company news profile page. The first step requires entity extraction from multi-lingual stream, while the second step requires integration and summarization across all languages with relevant articles.

- Related or Relevant Articles tracking

The related or relevant articles in Bloomberg use case is assembling an article list custom fitted for the specific user, based on his local markets and his/her history. Such list of articles can include external mainstream source from his/her local language.

3 Visualization Toolkit

This toolkit contains seven independent components developed with Google V8 JavaScript engine [1]. This allows for easy entry level from programming point of view, high flexibility (since it's a scripting language) and scalability. Moreover, Node.js [2], a software platform that allows for running a web server without the use of external software, is used when develop this prototype. Therefore, all these controls could be deployed simply through a JavaScript API with any popular browser.

In this section, each component will be introduced in detail with aspects of functionalities and interactions according to its functionality. The data interface will be introduced in the following section with the relevant libraries and tools together.

3.1 Entity-oriented components

3.1.1 Entity Graph

Functionality

Entity graph is designed for displaying instances and the complex relationships among them. Particularly, nodes and edges of Entity graph denote instances and their relationships respectively. In addition, the length and thickness of edges could be justified automatically with respect to its strength of relationship based on Potential Field Method. That is to say, by setting appropriate weights scale, instances with strong connections will cluster together, those have weak or no connections will depart away from each other, which is helpful to understand the internal relation of instances directly.

There are four major funcitonalities that should be noticed as follows.

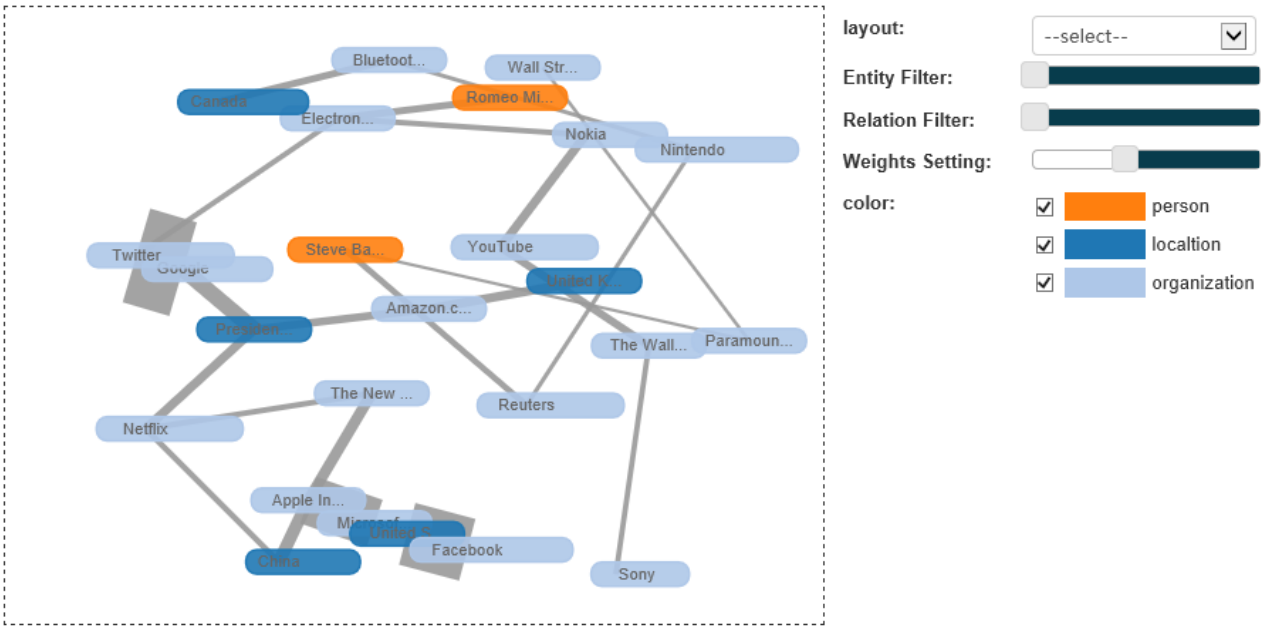

Figure 3 A demo of Entity Graph

- Classification of entities

As is shown in Figure 3. A Demo, each node of Entity graph is filled with one kind of color. In this case, both nodes with label of Google and Twitter are filled with color of light blue, which means that they belong to the same category in this sample. On the contrary, the China node has the different color of dark blue, for it's not a company but a location.

Also, user can increase/decrease the number of color for more/less categories of instances, and the color can be set as you wish.

➢ Weights of edges

To visualize the connections between entities, Entity graph use edges with weights to link relavant nodes. Particularly, short/long edges with more/less weights signifying strong/weak connections, which can be adjusted automatically according to the current weight scale based on Potential Field Method. Thus, nodes can be clustered into groups to imply their relationship.

➢ Filter

The Entity Graph also provides functions to filter entities/relations/weights to remove irrelavant node pairs or unimportant nodes.

There are three filters to simplify this graph.

- Entity filter: Each entity (node of Entity graph) has a score to imply its significance, which will be introduced in detail in 2.1.2 Data Format. A threshold can be set to filter out any entity with a score below it, and all the entities will be shown by default.
- Relation filter: Each relation (edge of Entity graph) has a score to imply its significance, which will be introduced in detail in 2.1.2 Data Format. A threshold can be set to filter out any relation with a score below it, and all the relations will be shown by default.
- Classification filter: Each instance (node of Entity graph) belongs to a classification in the initial setting, which will be introduced in detail in 2.1.2 Data Format, either. As states in the previous section, the colors of nodes denote different classifications of instances. By cancelling a color, the matching nodes will disappear correspondingly.

➢ Movability of nodes

User can change the position of node. However, it should be noticed that other nodes will be motivated to move simultaneously until the forces in virtual potential field converge back to balance. This might help user to analyze data as groups more conveniently.

Interactions

The main user interface is shown in Figure 3. The entire page is split into two area vertically. The left part shows the entity graph with the current configuration, and the right part is the functionality panel with filters and settings of necessary.

To satisfy the primary requirements of general users, there are several kinds of interactions designed in this prototype. In this subsection, they will be introduced by the method of giving examples.

➢ Layout

Layout item lists on the top of the functionality panel. When user clicks it (the blank with a text “select”), a drop-down menu pops up. For convenience, four common kinds of configuration for different browsing modes are provided here. Thus, user can select the best one.

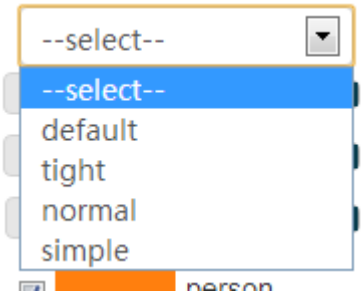

Figure 4 layout configuration

As shown in Figure 4, in the drop-down menu, user can choose:

- Select: displaying all the nodes and edges, which is the default option when initialize.

- Default: a recommended choice that balance the weights of both entities and relations.
- Tight: High relation entities will be pulled closely.
- Normal: Low rank edges and entities will be filtered
- Simple: Only the most important nodes and edges will be shown.

➢ Entity filter

This filter utilizes a slide to control the entity importance threshold which can be changed simply by dragging the button. At default, the minimum of this parameter is 0 (nothing will be filtered), and the maximum is set up automatically according to all the scores of entities when initialize.

➢ Relation filter

This filter utilizes a slide to control the threshold of edges. At default, the minimum of this parameter is 0 (nothing will be filtered), and the maximum is set up automatically according to all the scores of relations when initialize.

➢ Weights setting

The slide controls the influence of relations on the length/width of edges. With higher weights, the influence of relationship between nodes on the length/width will be multiplied. For example, by dragging the button, we increase the weight scale a little bit. Consequently, all the nodes cluster into two groups distinctly, and inside the upper group there are six subgroups that connect each other with different edges according to their strength of relations.

➢ Color (Entity Category)

In Entity graph, nodes are filled with colors to identify its Category. So, user can check/uncheck the corresponding color, at the bottom of the right-side panel, to decide which category will be displayed.

Figure 5 shows an example that the location entities with the color of dark blue are removed in the chart, and it should be noticed that the relevant edges disappear either.

The last interaction that facility user's operation should be mention: highlight.

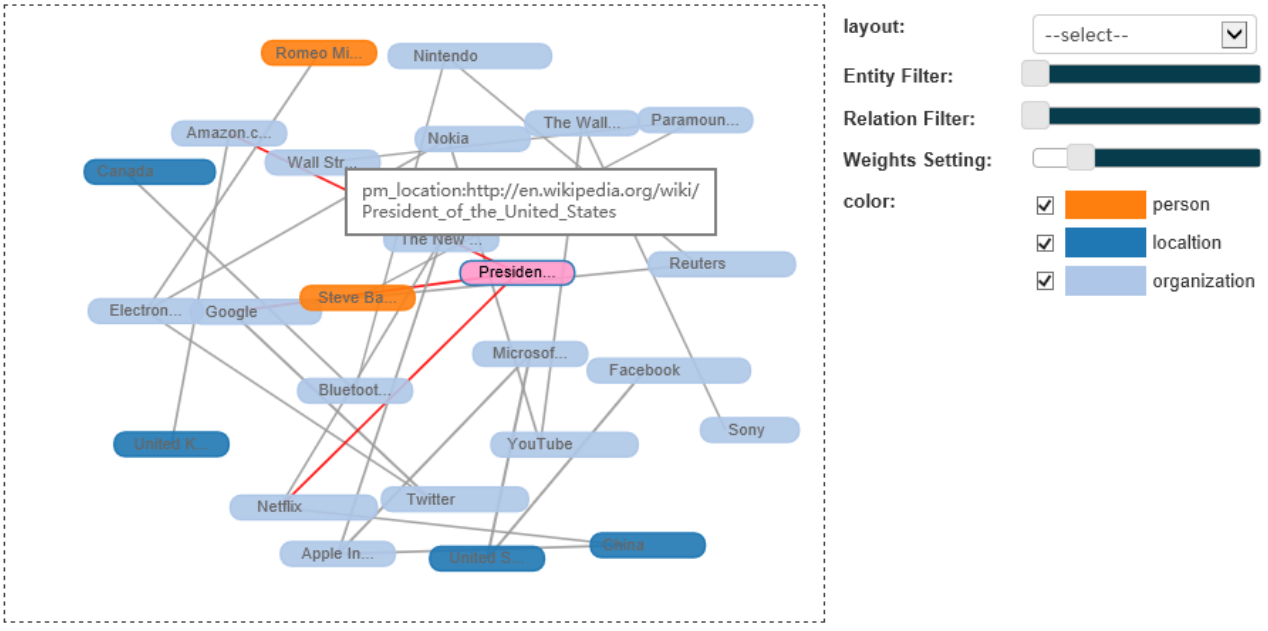

Figure 5 Highlight in Entity Graph

As shown in Figure 5, when user move mouse to any node, the node and all the relevant edges will be highlighted with red color. Also, the value of url field of the selected entity will be popped out in a small text box.

3.1.2 Radar Chart

Functionality

Radar chart is visualization control which provides graphical method of displaying multivariate data in the form of a plate-like chart. It automatically generates a polygon for an instance in a multi-axes coordinate system where each axis represents a component. Usually, the axes (components) are considered as independent with each other.

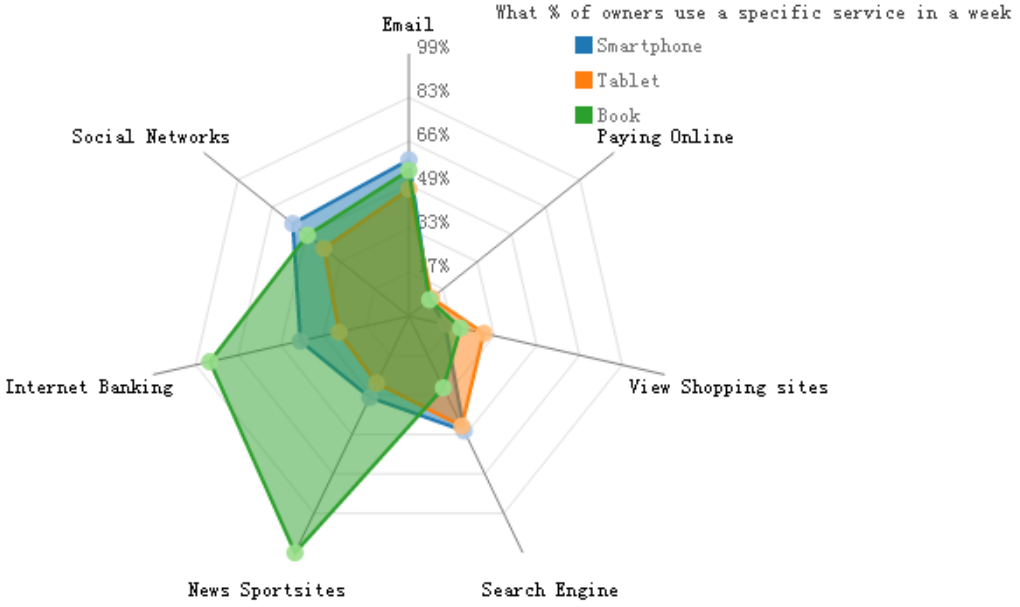

Figure 6 A Demo of Radar Chart

The radar chart is a useful way to display multivariate observations with an arbitrary number of variables. In general, it split the surface into several equi-angular areas with a star-like coordinate system with each axis representing one of the variables. In this coordinate system, the polygon of an instance is draw by:

- Plotting one point in each axis according to the magnitude of corresponding variable of the instance;
- Connecting points in nearby axes.

Figure 6 gives a typical example of radar chart with six variables, for each represents the percentage of a specific service that user use in a week. There are three instances shown in this chart, and three kinds of colors are used to distinguish them. The blue one represents the usage rate of smartphones distributing among these services, yellow for tablets, and green for books.

To compare these observations, the most obvious feature shown in this chart is that they have something in common on the distribution. They cluster in the left side of this chart, which means that people hardly use a mobile device to pay online. Except for this, it's not hard to see that people use books more frequently than the other two, especially for Internet Banking and News Sportsites. Smartphone and tablet distribute similarly and evenly among Email, Social Networks, Internet Banking, News Sportsites, Search Engine, and view shopping sites.

In conclusion, radar charts are primarily suited for strikingly showing bias and commonality. One possible application of radar charts in news data visualization is bias visualization. By assigning an entity/keyword in each axis, the radar chart will give user an intuitive image that how two or more event/news report differ from each other.

Interactions

Figure 6 shows an example of the main user interface, which contains the star plot of three kinds of devices. The variables list for the sample star plot is:

1. Email
2. Paying online
3. View shopping sites
4. Search engine
5. News sport sites
6. Internet banking
7. Social networks

We can look at these plots individually or we can use them to identify clusters of devices with similar features. For example, we can look at the star plot of the book device (the plot filled with green) and see that it is used approximately as frequently as the other two devices, little for paying online, but has a high usage rate in Internet banking and News sport sites. We can then compare the smartphone (the plot filled with blue) with the smartphone (the plot filled with orange). This comparison shows similar patterns. Both of these two devices are used evenly in all the services except paying online.

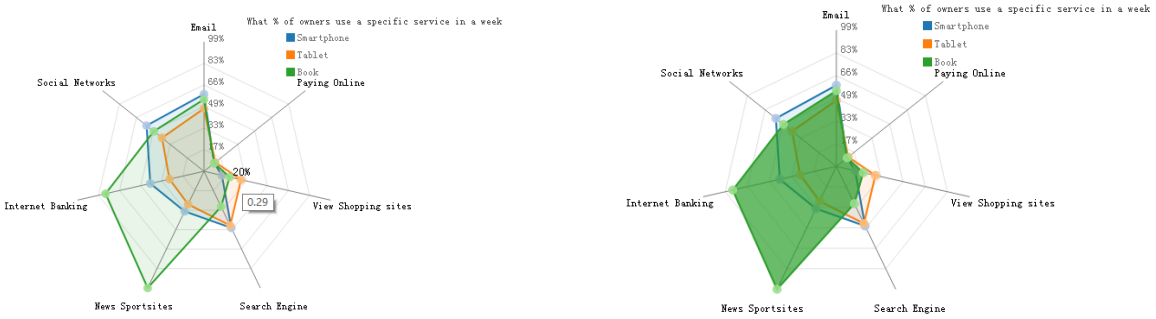

Figure 7 Highlight in Radar Chart

Figure 7 shows two examples of the highlight interaction. When user moves mouse to the data point, the chart will change to the left part of Figure 7. All the star plot become light, and the specific value displays floating.

The right part presents another way of highlight interaction to help user concentrate on one observation. When user move mouse in the area of one observation, the corresponding plot will becomes dark, and meanwhile the other plots become light

3.2 Article-oriented components

3.2.1 Timeline

Functionality

A timeline is a method of displaying a list of news or events in chronological order, which is often used to visualize time lapses between events, durations, and the simultaneity or overlap of spans and events.

This component provides a formalized interface to generate a timeline. The main user interface is shown in Figure 8. The entire page is split into two area laterally, of which the upper part shows the main body of news or events, and the lower part is a long bar labelled with dates alongside itself and events and news labelled on points where they would have happened.

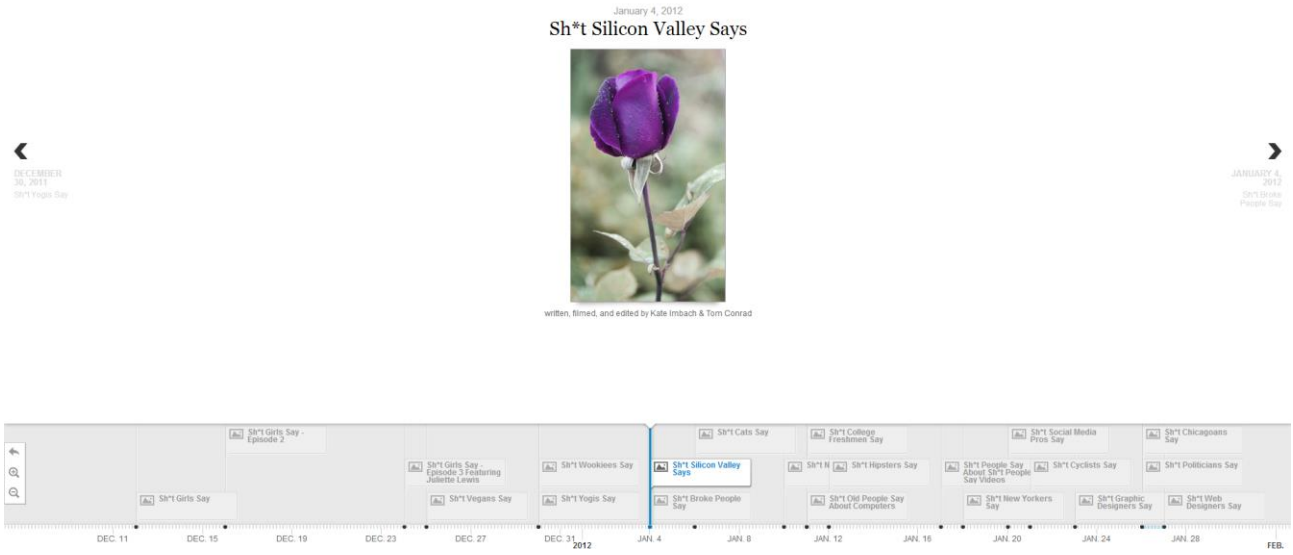

Figure 8 A Demo of Timeline

As mentioned above, there are two functionality areas laterally:

- Events body

This area shows the main body of events, which could be a picture, a piece of news article or even a video. It should be noticed that the video is embedded in web page, so that the supported video formats depend on web browser.

Through a JavaScript API, all the events are displayed with a readable structure automatically. Figure 12 shows a type of structure including its data at the top, a title below the data, a picture in the middle, and the relevant introduction at the bottom.

On both sides of this page, there are also two arrow headed buttons, labelled with a brief introduction, providing a functionality of switching to the next event or the previous one conveniently.

- Time bar

All the events list as a rectangle span begins at their start time and ends at the end time in this area. The label of each span gives a brief information about its title and whether it contains a picture (a small icon suggests that there is a picture). Except for browsing events and news one by one through the buttons on both sides of the upper area, clicking a rectangle span is another way of replacing the current event with the new one.

Time scale can be easily changed through the toolbar at the right side of this area, or the roller of the mouse. There are three buttons, of which the uppermost one provides the functionality of back to the begin date, the other two represent zoom up and out respectively.

Due to the chronological distribution of events, it is clearly to see the life circle of a news, what time is the breaking out point, and so on.

Interactions

There are two main ways to browse the events.

- Browsing in sequence

On both sides of the page, there are two arrow headed buttons, labelled with a brief introduction, providing a functionality of switching to the next event (the right button) or the previous one (the left button) conveniently.

- Switching quickly

Each rectangle span listed at the time bar indicate an event. User can browse this event by clicking the matching span.

3.2.2 Time-Geo

Time and Location are two very important elements of news articles, or events. In the visualization prototype (D5.2.2) and news event registry, we have designed techniques to visualize news articles on the time dimension (Time Trend) and the geographic distribution (Map [3]). It is useful for user to learn about how an event develops as the passing of time and the where the event has spread. However, in the map, users can only get a static image of the geographic distribution of news articles. The Time-Geo Visualization controls aims to showing the development of an event dynamically.

Functionality

The Time-Geo Visualization control contains two controls. One is the map showing the locations of articles and another is a Time-Trend chart showing the number of reports on a sequence of time stamps. Different from static visualization, the map can show the geographic distribution of news articles from one time stamp to another. In this way, users will know where the event is firstly reported and how it spreads over the world.

Figure 9 A Demo of Time-Geo

Interactions

The Time-Geo control is designed to visualize the spreading or diffusion of information on geographic aspect. To help user tracing the trend, this control provides two interactions.

- Manual interaction

Users can click the nodes in the time trend chart to see the geographic distribution of news reports published at the selected time point. Or,

- Automatic interaction

The control can automatically generate a short animation which shows the spreading of news reports according to the given data. This will show the diffusion of information though over the world vividly.

3.3 Relation-oriented components

Relation is a basic element hidden in articles, entities and topics, but it is worth to be displayed. In stressing different types of relations, there are two main methods, i.e., hierarchy and network. In general, relations increase exponentially as the amount of elements, which suggests more space to show clearly. For the reason of saving space, two visualization components are developed representing hierarchical and network relations respectively.

3.3.1 Chord Diagram

Functionality

The Chord diagram shows direct relations among a group of entities, topics, or indirect relations among articles through their own features. The data is arranged radially around a circle with the relationships between the points typically drawn as arcs connecting the data together. The size of sector in circle implies the number of its features, and different colors are used to split sectors explicitly. In addition, it is should be noticed that sector can connect to itself to display its independence proportion of features.

Figure 10 gives a simple example of several twitter groups. Each of the groups contain many persons sharing the same interest, and one person can take part in several groups at the same time. The relations among groups indicates the communication of them through the feature of the number of common persons.

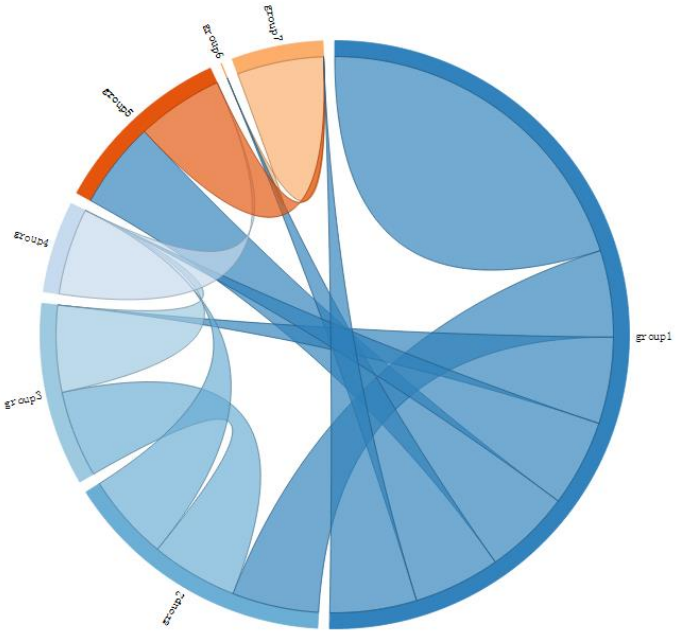

Figure 10 A Demo of Chord Diagram

Interactions

Chord diagram shows data in a static way.

3.3.2 Radial Tree

Functionality

Radial tree is a method of displaying a tree or hierarchical structure in a way that expands outwards, radially. Figure 11 shows a simple example of the class hierarchy. Each class is represented as a node, and each edge implies an inheritance relation from children node to its parent node. The root node is in the central of radial tree, and the depth of nodes represents the distance from the root.

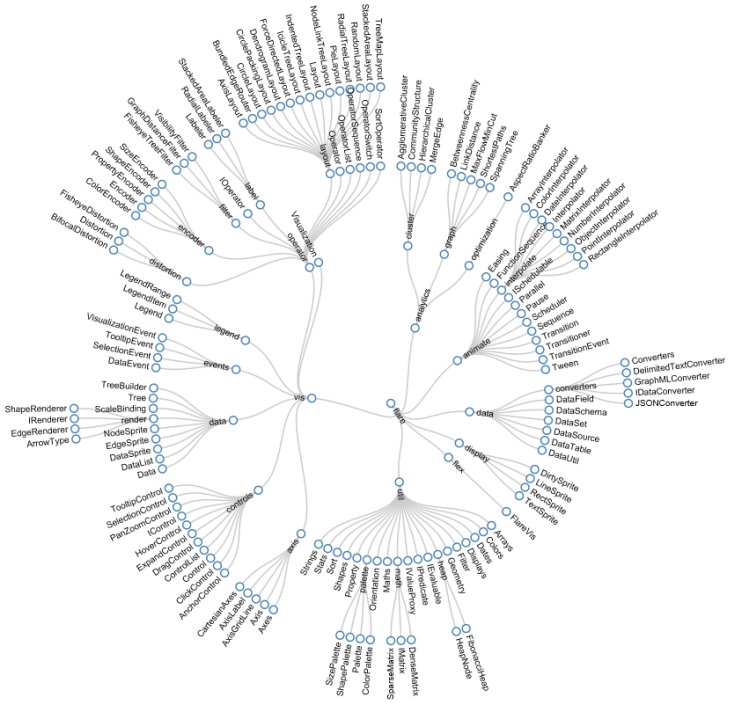

Figure 11 A Demo of Radial Tree

Interactions

Radial tree shows data in a static way

3.4 Topic-oriented components

3.4.1 Stream Chart

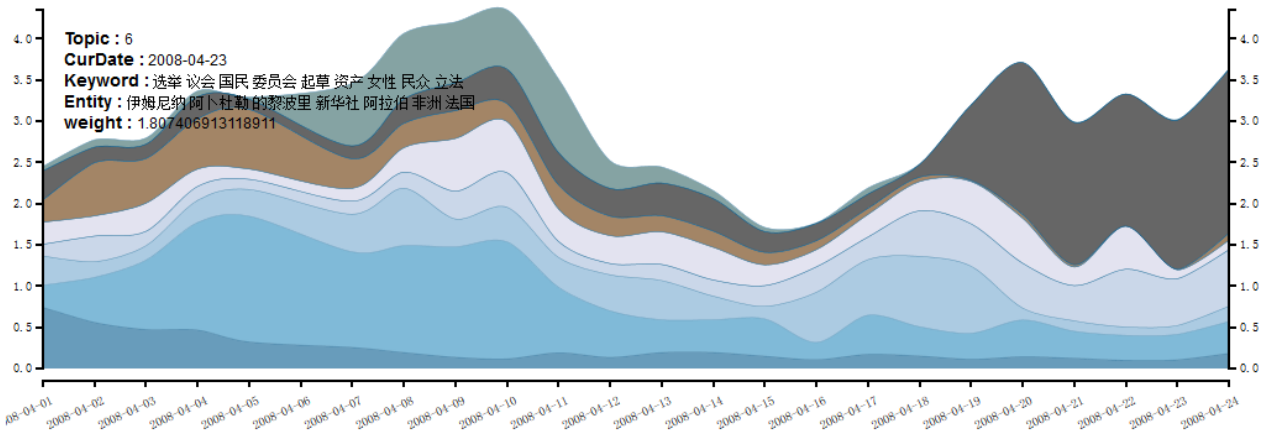

Figure 12 A Demo of Stream Chart

Functionality

Stream chart shows the topic changing as the passing of time. Figure 12 takes some politics news as an example. The horizontal axis represents the dimension of time, and the vertical axis implies the total weight of the topics controlled by a parameter in the left list. Different colors split the stream into several layers with its thickness according to the topic weight.

Interactions

The users can see the topics flow up and down roughly. Furthermore, a more detailed information will be listed in the left side, when the mouse moves in the figure area after a click awakes the information representing function, where a vertical line occurs in the corresponding date.

4 Data format and Library

The toolkit provides a standard data interface for a fast prototyping. Using Google V8 JavaScript engine, the components are programmed based on Node.js, a platform that allows for running a web server without the use of external software. Besides, D3.js [4], as a dependent library, provides support through a JavaScript API. This section describes the data format of each visualization components, and their dependent JavaScript libraries and web services that support the controls in the final prototype front-end. Currently Node.js, D3.js and Google Map are included.

4.1 Data format

- Entity Graph

To deploy entity graph in target system, user must provide a list of entities which satisfy the predefined format. Figure 13 shows an example entity, of which the data fields' meaning can be found in **错误!未找到引用源。**.

```
{
   uri: "pm_org:http://en.wikipedia.org/wiki/Microsoft",
   id: "23605",
   type: "pm_org",
   label: "Microsoft",
   label_eng: "Microsoft",
   label_deu: "Microsoft",
   label_spa: "Microsoft",
   label_zho: "微软",
   label_slv: "Microsoft",
   score: 79
},
```

Figure 13 Data Format of Entity Graph

Table 1 Data Filed of Entity Graph

url	Source url of entity
id	entity id
Type	Classification of entity
label	Node's text shown in the graph
Label_eng	Node's label in English
Label_deu	Node's label in German
Label_spa	Node's label in Spanish
Label_zho	Node's label in Chinese
Label_slv	Node's label in Slovenian
score	Weight of entity

Except an entity list, each relation, edge of the graph, requires a standard input data format. shows an example of relation data structure that links two nodes: id of 233 and 23605.

```
{
   Id1: "233",
   Id2: "23605",
   score: 2605
```

```
},
```

Figure 14 Data Format of Relation

Table 2 details the meanings of parameters of relation

Table 2 Data Fileds of Relation

Id1	Id of the head of the edge
Id2	Id of the tail node of the edge
score	Weight of edge

- Radar Chart

```
[
   {axis:"Email",value:[0.59,0.48,0.55]},
   {axis:"Social Networks",value:[0.56,0.41,0.49]},
   {axis:"Internet Banking",value:[0.42,0.27,0.77]},
   {axis:"News Sportsites",value:[0.34,0.28,0.99]},
   {axis:"Search Engine",value:[0.48,0.46,0.30]},
   {axis:"View Shopping sites",value:[0.14,0.29,0.20]},
   {axis:"Paying Online",value:[0.1,0.11,0.10]},
]
```

Figure 15 Data Format of Radar Chart

Figure 15 shows a sample data of the same radar chart in Figure 6. Each line represents a variable/axis, where:

Table 3 Data Fields of Radar Chart

Axis	The label of the variable, a specific service in this sample
value	The corresponding value array of the observations

It should be noticed that there are three observations in this chart, each one corresponds to a number in the value field by the order of colors listed in the upper right corner. For example, all the first numbers of each line will be connected to generate a star plot representing one observation, and so does the second and third numbers.

- Timeline

As a dependent library, Timeline.js gives support to this component, which requires a specific data format for every event or news shown in timeline. Figure 16 shows an example of the data structure for one event, and in Table 4, each domain will be detailed.

```
{
   "startDate":"2011,12,25",
   "headline":"Vegans Say",
   "text":"text to be shown",
   "asset":
      {
         "media":"http://g.hiphotos.baidu.com/pic/w%3D230/sign=1f6136aa38dbb6fd255be2253925aba6/
         b8014a90f603738de7cc8e73b21bb051f919ecd9.jpg",
         "credit":"",
         "caption":""
      }
}
```

Figure 16 Data Format of Timeline

Table 4 Data Fields of Timeline

startDate		Start date of event
endDate		End date of event
Headline		Title of event
Text		Body of event
asset	Media	Media embedded in the event
	Credit	
	Caption	Caption of the media

- Time-Geo

The data inputted to the Time-Geo control should be a list of articles with publish time and publish location.

```
[
  {
    "id":1334938,
    "url":"http://www.eluniversal.com.mx/finanzas-cartera/2013/abren-con-resultados-
    mixtos-principales-bolsas-europeas-972749.html",
    "title":"Abren con resultados mixtos principales bolsas europeas",
    "language":"spa",
    "date":"2013-12-13T09:23",
    "source":"www.eluniversal.com.mx",
    "location":[
        23,
        -102
    ]
  },
  …
]
```

Figure 17 Example Data for Time-Geo Control.

Table 5 Data Fields for Time-Geo Control.

Id	Id of the article
url	url of the article
Title	Title of the article, will be shown in the article list
Date	The publish time of the article
Source	The publisher of the article
Location	The location of the publisher, or where the news happens

- Chord Diagram

To deploy Chord diagram in target system, user could input a list of data satisfy the predefined format. Figure 18 shows an example entity, of which the data fields' meaning can be found in Table 6.

```
[
  {
    "name":"flare.analytics.cluster.AgglomerativeCluster"
    "size":"3938",
    "imports":["flare.animate.Transitioner","flare.vis.data.DataList","flare.util.math.IM
```

```
    atrix","flare.analytics.cluster.MergeEdge","flare.analytics.cluster.HierarchicalClust
    er","flare.vis.data.Data"]
  },
  …
]
```

Figure 18 Example Data of Chord Diagram

Table 6 Data Fields of Chord Diagram

name	Name of the element
size	Size of the features
imports	The elements have relations

- Radial Tree

To deploy Radial Tree in target system, user could input a list of data satisfy the predefined format. Figure 19 shows an example entity, of which the data fields' meaning can be found in Table 7.

{

"name": "cluster",

"children": [

{"name": "AgglomerativeCluster", "size": 3938},

{"name": "CommunityStructure", "size": 3812},

{"name": "HierarchicalCluster", "size": 6714},

{"name": "MergeEdge", "size": 743}

]

},

```
[
  {
    "name":"cluster",
    "children":[
          {"name": "AgglomerativeCluster", "size": 3938},
          {"name": "CommunityStructure", "size": 3812},
          {"name": "HierarchicalCluster", "size": 6714},
          {"name": "MergeEdge", "size": 743}
    ]
  },
  …
]
```

Figure 19 Example Data of Radial Tree

Table 7 Data Fields of Radial Tree

name	name of the element
children	list of the children elements
size	size of the element

- Stream Chart

The data inputted to the Stream chart should be a list of topics with time and keywords.

```
[
```

```
  {
    "key":"7",
    "value":"支持者 政府军 组织 人士 消息 领导人 当天 占领 民兵 国界:尼沃利 拜尼沃利 的黎波里 哈姆扎
    新华网 联合国 新华社: 0.0053307927256423",
    "date":"04/24/08",
  },
  …
]
```


Figure 20 Example Data of Stream Chart

Table 8 Data Fields of Stream Chart.

key	Id of topic
value	Keywords of topic with weight
Date	Time of the topic

4.2 Library and Tools

4.2.1 Node.js

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

4.2.2 D3.js

D3.js is a JavaScript library for manipulating documents based on data. D3 helps you bring data to life using HTML, SVG and CSS. D3's emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization components and a data-driven approach to DOM manipulation.

D3 allows you to bind arbitrary data to a Document Object Model (DOM), and then apply data-driven transformations to the document. For example, you can use D3 to generate an HTML table from an array of numbers. Or, use the same data to create an interactive SVG bar chart with smooth transitions and interaction.

D3 is not a monolithic framework that seeks to provide every conceivable feature. Instead, D3 solves the crux of the problem: efficient manipulation of documents based on data. This avoids proprietary representation and affords extraordinary flexibility, exposing the full capabilities of web standards such as CSS3, HTML5 and SVG. With minimal overhead, D3 is extremely fast, supporting large datasets and dynamic behaviours for interaction and animation. D3's functional style allows code reuse through a diverse collection of components and plugins.

4.2.3 Google Map

Google Maps is a set of services provided by Google. In the visualization component, the Google Maps Javascript API and Google Map API Geocoding web service are employed.

The Google Maps Javascript API lets developers embed Google Maps in your own web pages. Version 3 of this API is especially designed to be faster and more applicable to mobile devices, as well as traditional desktop browser applications.

The Google Map API Geocoding API provides interface to convert addresses (like "1600 Amphitheatre Parkway, Mountain View, CA") into geographic coordinates (like latitude 37.423021 and longitude -122.083739), which you can use to place markers or position the map. The Google Geocoding API provides

a direct way to access a geocoder via an HTTP request. Additionally, the service allows you to perform the converse operation (turning coordinates into addresses); this process is known as "reverse geocoding."

Google provides a free license to these APIs but with a condition that our service must be freely and publicly accessible to end users.

5 Conclusions

In this document, we have presented the final prototype front-end which contains seven visualization component integrated into a UI toolkit. The toolkit is supposed to be used in industrial show case thus one important requirements is to be easily deployed. In this toolkit, we have integrated visualization controls in different sources but packaged them with well-defined interfaces. One do not need to know how the controls work but only need to feeding formatted data to the interfaces. In addition, all the components in the toolkit are developed using Java Script therefore are independent with operating system.

In this deliverable, seven visualization components with distinguish functionalities are presented. The Entity Graph is suitable for the visualization of relations between different entities. The entities here do not need to be named entity such as Persons, Locations and Organizations, but can be any concepts as long as we can find a way to measure the relationships or affinities between them. The Radar Chart is suitable to show different dimensions of an entities or visualize the difference of a small set of entities in the same dimensions. The Timeline is suitable for the visualization of a sequence of articles/events. The Time-Geo control is suitable to visualize data with both time stamp and location information. Chord diagram and Radial tree are suitable for visualize network and hierarchical relation respectively. At last, Stream chart is suitable for the visualization of the changing of data stream

This deliverable have introduced the functionalities, data formats and interactions of these four controls in detail and the codes have been uploaded in Github.

This is the final prototype front-end, which is an update to the toolkit from D6.4.1.

6 References

[1] https://code.google.com/p/v8/
[2] http://nodejs.org/
[3] https://developers.google.com/maps/
[4] http://d3js.org/

