*A Seminar Report submitted in partial fulfilment of the requirements*
*for the award of degree*

**Master Of Technology**
**In**
**Computer Science And Engineering**

# GENERATING STORYLINES


*Submitted by*
**Anunaya Srivastava**
(13535010)

*Under the guidance of*
**Dr. Dhaval Patel**
*Assistant Professor*




**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**
**INDIAN INSTITUTE OF TECHNOLOGY**
**ROORKEE – 247667**

# ABSTRACT

Today there is vast amount of information present on the internet. There are a number of news media, blogs, social networks like Facebook, Google Plus etc. and microblogs such as Twitter that provide information to the user who is trying to extract some specific knowledge from the world wide web. Thus the user gets burdened with large amount of data which makes it difficult for the user to find the desired information. In this seminar report, we discuss the various techniques that are used to represent data in a more meaningful manner to counter the problem of Information Overload. We focus primarily on various algorithms used for Storyline Generation which is a technique to capture the underlying temporal and casual dependencies amongst the news events.

# CONTENTS

# LIST OF FIGURES



# LIST OF TABLES

# 1. INTRODUCTION

Today the internet has become a rapidly growing repository of information. On the internet information is present in various forms such as online libraries/books, research papers, news articles, blogs, tweets, comments, social media posts. Online books are very long, news articles and blog posts are relatively shorter. Tweets and social media posts are even shorter, and are also called as microblogs. The content is not only in the form of text, but also in the form of images and videos. This vast amount of information often leads to information explosion. Thus navigating through such a large collection of web documents becomes challenging and users can easily miss the bigger picture. This problem is known as 'Information Overload' which has been long recognized in the industry. Organisations like Harvard Business Review, Google etc. also acknowledged the problem of having too much information and stated, "*the abundance of information people are exposed to through e-mail and other technology-based sources could be having an impact on the thought process, obstructing deep thinking, understanding, impedes the formation of memories and makes learning more difficult*". Eric Schmidt, former Google CEO, says, "*Between the dawn of civilization through 2003 about 5 exabytes of information was created. Now, that much information created every 2 days.*" Therefore, we need techniques to represent data in an effective and meaningful way.

To encounter the problem of information overload researchers have proposed various types of document understanding systems like (1) Removing duplicate information (2) Document Summarization (3) Topic Detection and Tracking (4) Storyline generation. The former removes repeated data from the corpus, but still the user has the tedious task of reading long texts. Summarization algorithms reduce information overload further by choosing representative sentences from a document such that these sentences collectively convey the principle idea presented in the document.

For example, a summary of the text written above, generated using an online tool [1], may look like :

*This is called Information Overload problem which has been long recognized in the computing industry.*
*To encounter the problem of information overload researchers have proposed various types of document understanding systems like Removing duplicate information Document Summarization Storyline generation.*
*Summarization algorithms reduce information overload further by choosing representative sentences from a document such that these sentences collectively convey the principle idea presented in the document.*

Figure 1.1: An example of summarization

However, the user still has to explore all the summarized information manually and understand how a particular chain of events have evolved. There is need of an automated system which can capture the evolution of events over time. To address this issue, recent work has focussed on topic detection and tracking (TDT) and storyline generation which are models used to extract useful summarized information from news media, blogs and microblogs such as Twitter. A **topic** is defined as "a set of news stories that are strongly related by some seminal real-world event", where **event** is defined as "something that happens at a specific time and location". For example, when the Malaysian airline MH370 went missing, that is a seminal event that triggers the topic.There are other events in the topic which include despatch of search parties, rescue attempts, search for the reason of mishappening and so on.

The goal of *Topic Detection and Tracking(TDT)*, intuitively speaking, is to break the textual data down into individual news storiesso that one can monitorthestoriesfornew, unseen events,andgroupthe storiessuchthateach groupdiscussesasinglenewstopic. Thus it alerts the user about the new events happening around the world.For example, following are the most frequent topics found after processing 17,000 articles from the journal Science[2]. Here a topic is represented as a bag of inter-related frequently occurring words that correspond to a single topic. In the example shown below, the words human, genome, dna, genetic, genes etc. belong to a single topic which can be called 'Genetics'. But the results of TDT give the topics and the documents related to that topic. The results do not give a structure of the topic in terms of its events. Here structuremeans identifying the events that make up a topic and establishing dependencies amongst them. If event B is dependent on event A, then event B happened after event A and is related to event A and may (or may not) be a consequence of event A. Thus, we need algorithms that can establish these dependencies among the events belonging to the

same topic. Such a chain of dependent events is called a **story**. These algorithms which can generate a story are called storyline generation algorithms.

| Disease | Computers | Genetics | Evolution |
|---|---|---|---|
| bacteria | computers | human | common |
| infectious | system | sequencing | life |
| parasites | computer | genetics | evolution |
| control | information | genetic | organisms |
| tuberculosis | methods | genes | species |
| united | parallel | sequences | group |
| disease | model | genome | biology |
| parasite | simulations | dna | groups |
| new | systems | gene | living |
| strains | data | sequence | evolutionary |
| bacterial | new | information | diversity |
| resistance | software | molecular | new |
| diseases | network | map | phylogenetic |
| host | models | project | origin |
| malaria | networks | mapping | two |

Figure 1.2: Most frequent topics found after processing 17,000 articles from the journal Science

*Storyline generation* algorithms select representative documents and use the temporal information of the documents to represent the evolution of events in the topic.It generates a meaningful and inter-related chain of events which is called a **storyline**.Figure 1.3 shows how a one-dimensional storyline connects different events to give an idea of evolution of story.Figure 1.4 shows a two-dimensional storyline called a *metro map*. A metro map displays multiple stories and how they inter-connected with each other.. From the storyline shown in Figure 1.3, it can be easily noticed how the event of clashing of protestors with police leads to different events of arson at museum, loot of antiques and increase in support of revolution. This event based dependency structure reflects the inter-connectivity of events in a more accurate and summarized manner, thus giving a better understanding to the user.Figure 1.4 shows a metro map with three storylines and how these storylines interact with each other.

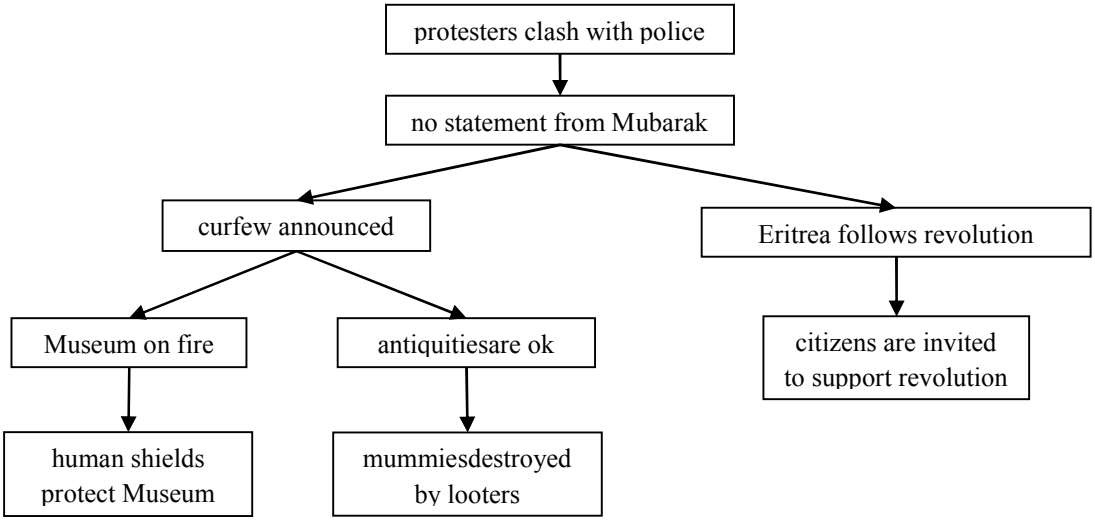


Figure 1.3: A storyline for query "Egypt Revolution"

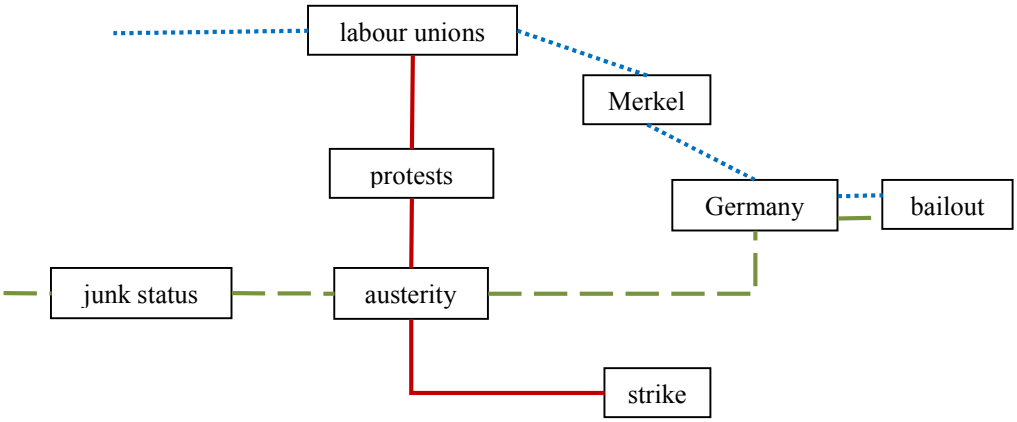


Figure 1.4: A simplified metro map on 'Greek debt crisis'

Why is the problem of storytelling interesting and important?Storytelling can be applied in many fields such as document modelling, social network analysis, computational linguistics,counterterrorism and bioinformatics. In all of these domains, storylines reveal various forms of insights. Unlike a real story, we can think of storytelling as a carefully argued process of adding and removing participants. Knowing exactly which objects mustbe displaced, and the orderin which they must be displaced, helps expose thecomplex mechanics of underlying relationships which otherwise need considerable manual effort to be determined.

In this seminar, we discuss about therecent research developmentsin the area of storyline generation from varioustypes of datasets(news articles, blogs, tweets etc.).The report has been organized into various sections and sub-sections. In section 2, wegive an overview of the various systems that have been developed to understand the set of documents. In section 3, we discuss the characteristics of a

good storyline. In section 4, we discuss a generalised framework followed in the recent years to generate storylines and different algorithms that have been used in each phase of this framework.

# 2. DOCUMENT UNDERSTANDING SYSTEMS

To encounter the problem of information overload, researchers have developed various methodologies or systems to extract useful information from large volumes of text. We refer to these systems as document understanding systems. In this section, we discuss about the various document understanding systems − summarization, topic detection and tracking, and storyline generation algorithms.

## 2.1. *Summarization*

In multi-document summarization [3], the objective is to summarize the document corpus by extracting a list of relevant sentences which should represent a synopsis of the entire set of documents. An example of document summarization has been mentioned in Figure 1.1.The summarization task can be divided in three parts (1) Topic representation (2) Score sentences (3) Select summary sentences.

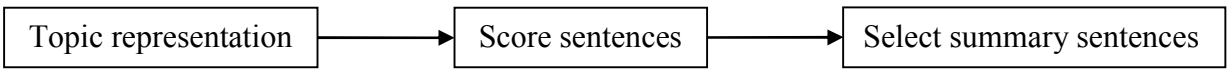


Figure 2.1: Process of summarization

Topic representationmethods convert the textual datainto an intermediate representation that captures the topic discussed in the input document. Some of the most popular topic representation methods include the commonly used term frequency (TF) and TF-IDF weights, which assign weights to all the words in a document. Words having higher weight are indicative of the topic. Topic representation also includes Latent Semantic Analysis(LSA) which works on the concept that two words used in the same concept tend to have similar meanings.After deriving an intermediate representation of each sentence, each representation is assigned a score indicating its significance. The scores represent how well a sentence expresses the most significant topics in the document. Finally a summary of the documents is produced by selecting sentences in a greedy way or by globally optimizing the selection to choose the best set of sentences. But the summary of the documents consists of multiple topics. The user may be interested in getting information about a single topic and how that topic has evolved over a period of time. A summary doesn't provide topic-specific information; hence we have a second category of document understanding system known as the Topic Detection and Tracking.

## 2.2. *Topic Detection and Tracking*

Topic Detection and Tracking (TDT), unlike multi-document summarization, aims to thread streams of text. Each story consists of several events and discusses a topic. Thus a topic is discussed by multiple stories, and each story is formed by multiple inter-related events. For a given document corpus, TDT recognises the different topics discussed in them and tracks the development of different stories. TDT consists of five main tasks –

a) *Story Segmentation* is the problem of dividing the document corpus into individual stories. It clusters the documents such that documents in a single cluster talk about the same story.
b) *First Story Detection* is the problem of recognizing the emergence of a new topic in the stream of news stories. It is detecting the first story about a topic which signifies the emergence of that topic.
c) *Topic Detection* is the problem of grouping or clustering of news stories as they arrive, dependingupon the topic they are based on.
d) *Topic Tracking*is the monitoring or tracking of news stories to determine how a specific topic evolves with time.
e) *Link Detection* is the problem of deciding whether two stories, which have been randomly selected, discuss the same topic i.e. it detects the link between different stories based on the topic they discuss.

TDT automatically detect salient topics from a given document corpus and associate each document with one of the detected topics. We can consider TDT as a special case of document clustering. Most of the TDT systems have been developed by adapting various document clustering techniques. An example of most frequent topics found after processing 17,000 articles from the journal Science is given in Figure 1.2.

### 2.3. *Storyline*

One of the shortcomings of TDT is that it views news topics as a flat collection of stories. For example, the topic detection part of TDT arranges a collection of news stories into different topic clusters. However, considering a topic as a mere collection of stories is an oversimplification. A topic is characterized by a definite structure of related events. For example, declaration of EVD(disease) outbreak as an International Health Emergency is a seminal event that triggers the topic and lead to other subsequential events such as rescue attempts, fund allocation etc. As discussed earlier, a storyline is a chain of inter-related events forming a story such that it gives a better understanding to the user about the underlying structure of events related to the topic.

Storylines can be one dimensional portraying how a single event leads to emergence of other events, and can also be two-dimensional which is known as Metro Map. A metro map consists of a set of lines(chain of documents) which have intersections and overlaps. Each line represents a coherent chain of events and different lines focus on the different aspects of the story. This visualization helps the user to understand the information at a holistic level.Figure2.2 shows a metro map representing Greek Debt Crisis. The map consists of four main storylines about austerity plans, the riots, the role of Germany and the role of IMF in the crisis. It also shows how the four storylines are inter-connected with each other through specific events.

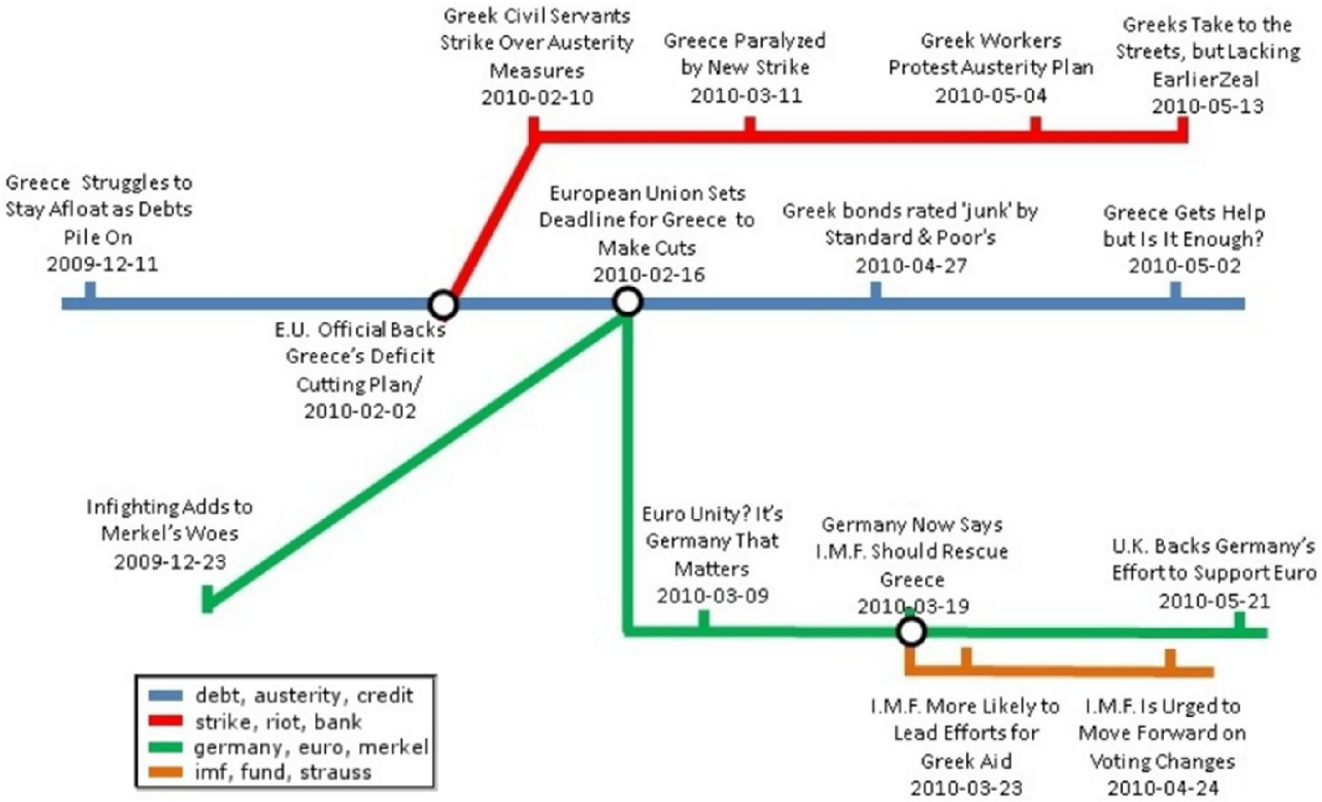


Figure 2.2: An illustration of Metro Map showing stories related to 'Greek debt crisis'
Courtesy: Research paper [15]

# 3. STORYLINE

The main emphasis of this report is to study the various techniques used for storyline generation. As discussed in Section 2, a storyline is a chain of inter-related events that represents the development of a single story. In this section we will discuss the characteristics of a good storyline, how different datasets give rise to different kinds of storylines and the methods used for storyline generation.

## 3.1. *What Makes a Story Good?*

A storyline generation algorithm aims to connect two given events - start event, *s* and end event, *t*, by finding the intermediate events such that after reading it the user should get a better understanding of the evolution of the story. Events *s* and *t* can belong to a story which revolves around a single event, person, place or institution. Such stories are called simple stories. Events *s* and *t* may seem unrelated at first since they may be connected by multiple intermediary events. Such stories which span through multiple events are categorized as complex stories. To generate an optimum storyline we need a way to define and measure the quality of a generated storyline. A good storyline should have the following properties -

a) **Relevance**: The articles used to form the chain must be relevant to the events connecting the start event and the end event.
b) **Coherence**: The transition between different events on the storyline should be smooth such that the user can understand the relation between two adjacent events.This helps in maintaining a global coherent theme across the storyline.
c) **Low redundancy**: Same event can be reported by many media outlets, thus our document set can have repeated documents about the same event. The userwould not prefer to read a storyline that contains mulitple articles for the same events.
d) **Coverage**: Every important event of the story should be covered by a good storyline. In case of a metro map, the map should contain topics which are diverse in nature.
e) **Connectivity**: This property is exclusive to a metro map. There is an underlying structure in a metro map. Connectivity of a map measures how different aspects of a story interact with each other thus forming a meaning structure in the map.

Coherence is a very important property for any storyline generation algorithm. The idea of coherence was first suggest in [4] where the author suggests that all the events in the chain should belong to a

consistent theme and there should not be any concept-jumping or jitteriness. Consider the example in Figure 3.1, which shows two stories connecting the same end points. In chain A, each consecutive transition is reasonable, but the erratic storyline passes through Microsoft trial, Palestinians, and European markets. Thus, it lacks a global or coherent theme which is present in chain B.

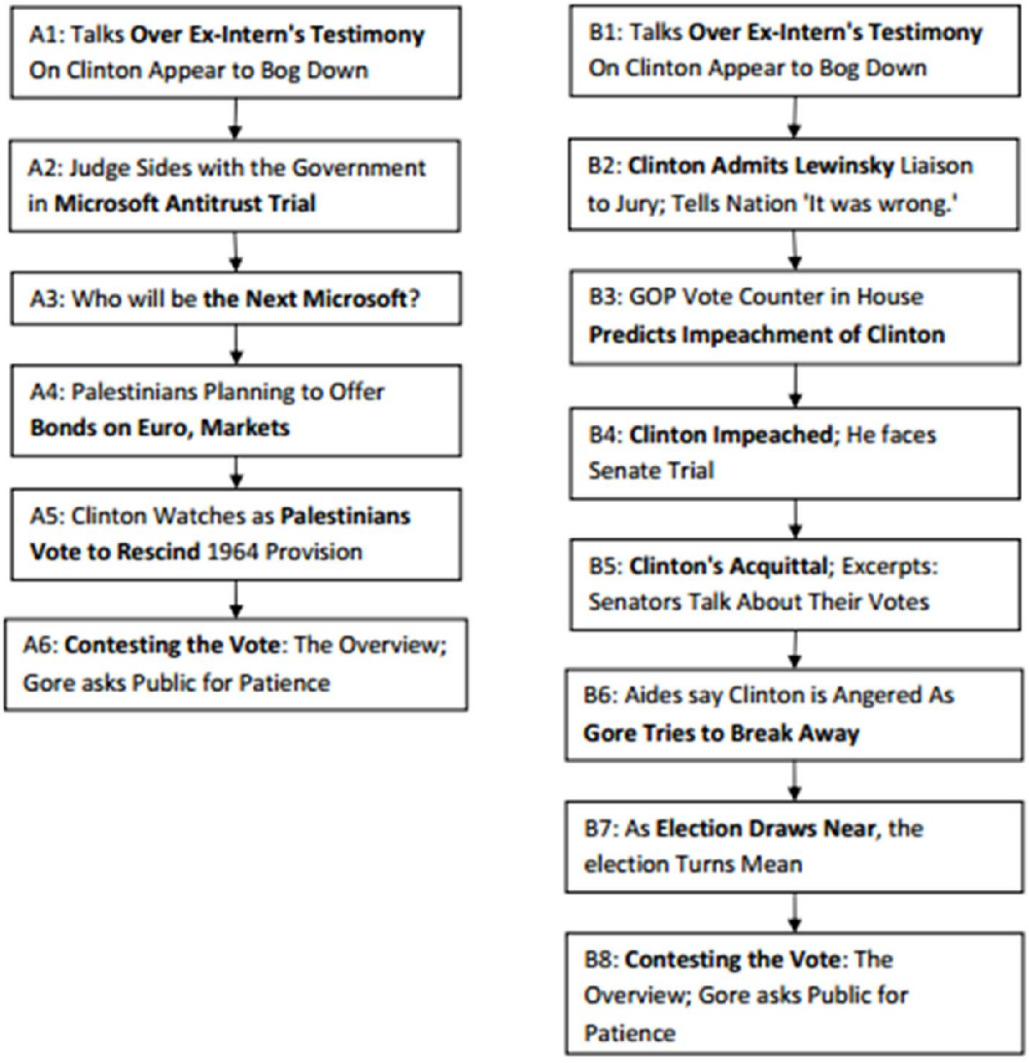


Figure 3.1: (a) Chain A is non-coherent. (b) Chain B is coherent.

3.2. *Storyline Configuration*

We have discussed storyline as a chain of events connecting two distinct events. For the given set of documents, a start event and an end event, the aim is to determine how the two given events are connected with each other. However, the user may be interested in knowing the consequences of the single event that may onset more than one chain of events. In such a case, for a given set of documents the user gives a single event as the query and the result is a branched storyline where each branch represents a single chain of events.We refer to such a storyline as an evolution storyline. Also, the user may want to know about the different stories in the document corpus and how they interact with each other. Then the user gives a user query and the result contains all the stories related to the query. The resultant storyline is a metro map which has been explained earlier.Figure 3.2 shows an illustration of different storyline configurations.

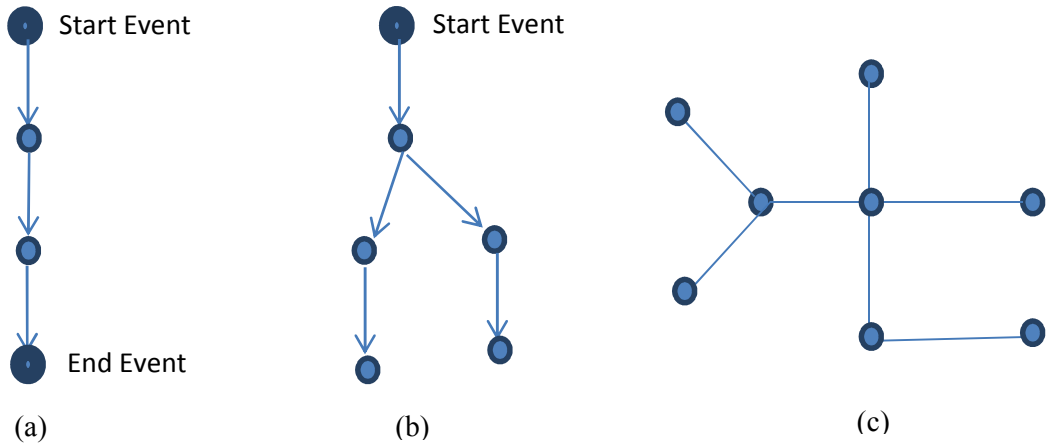


Figure 3.2: (a) A connecting dots (b) An evolution storyline(c) A 2-D metro map

Events of storyline and metro map can be extracted from a set of news articles or blogs; or from a set of microblogs or tweets. Single news is reported by the multiple news sources. Thus when we query the news media to get information about a set of events, the result set will have duplicate data for the same event. Similarly, there are multiple tweets and re-tweets for a single event.Thus the algorithm needs to choose a representative set of sentences for each distinct event. These distinct events are then joined to form a storyline.

Storylines can be formed not only using textual data, but also the image data [5], where the dataset consists of the images and corresponding text associated with each image.Figure 3.3 shows an example

of pictorial storyline. Also, storyline generating algorithms can be used to connect two distinct entities[6] which are extracted from a set of relevant documents.

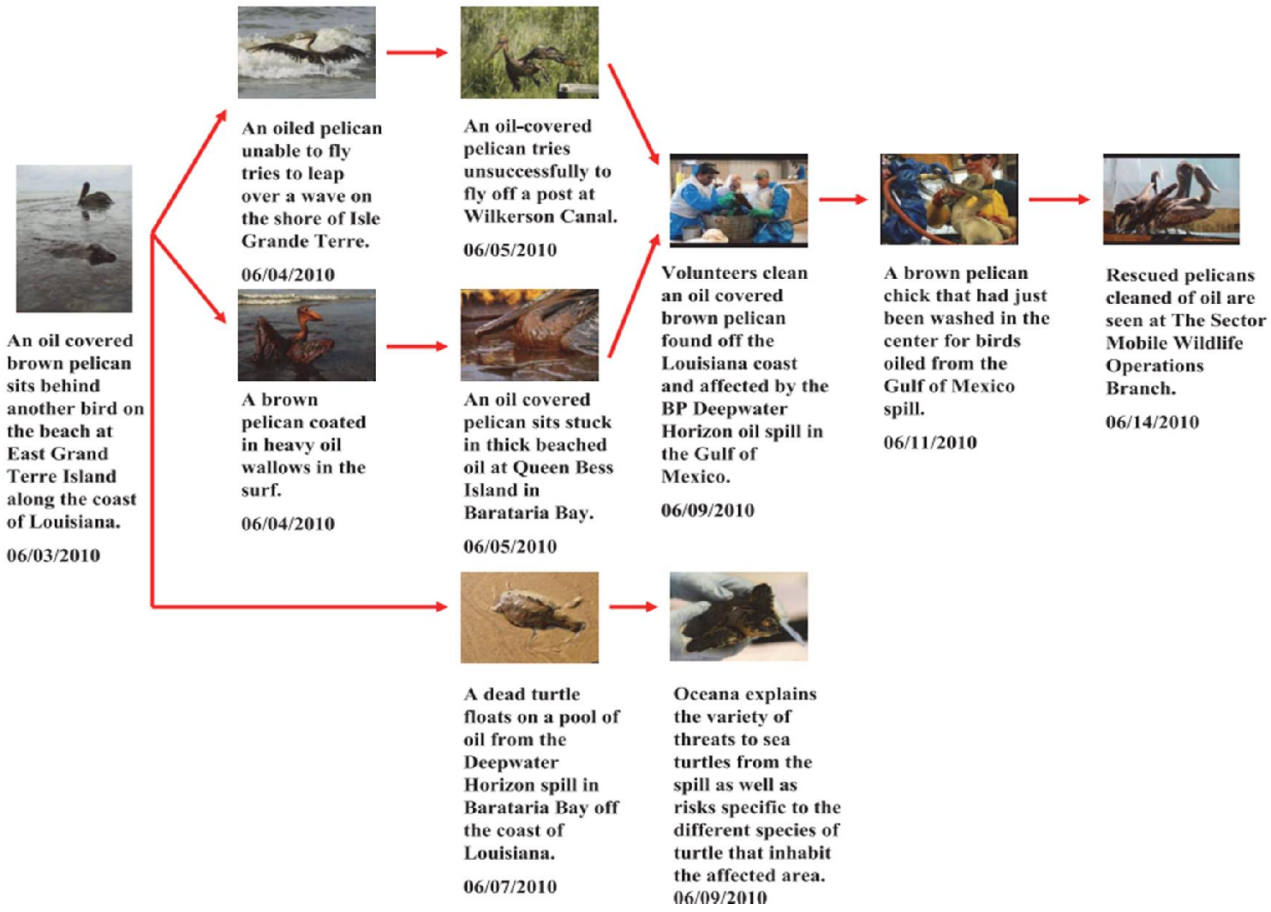


Figure 3.3: An illustration of pictorial storyline
Courtesy: Research paper [5]

# 4. STORYLINE GENERATION

Storyline generating process can be broadly divided into 3 stages (1) Extract relevant documents, (2) Summarization to find event-wise representative sentences, (3) Connecting events to generate storyline.

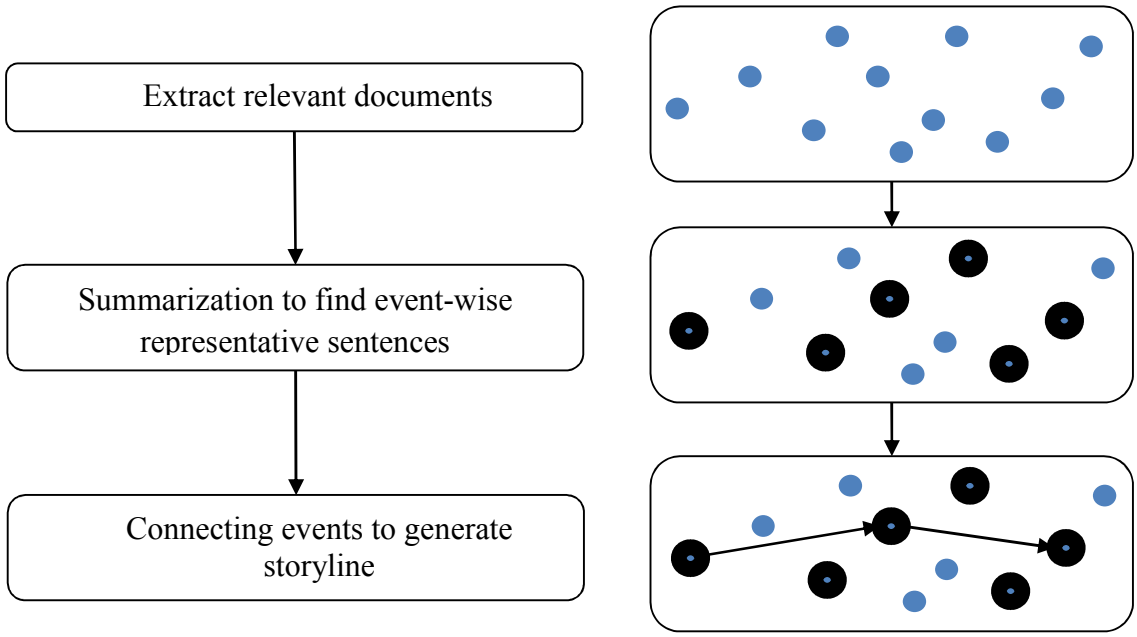


Figure 4.1: Stages for generating storylines

In this section, we discuss the different stages of storyline generation mechanism. For each phase, we discuss the different algorithms that have been used to process the information in that phase.

## 4.1. *Extract Relevant Documents*

The user gives a query telling the system the story he wants to know about. An example of a user query can be 'Egypt Revolution'. First of all, the system has to extract the documents relevant to the user query from the news media or blogs or Twitter. We refer to each news article, blog and tweet as a document. Let the dataset or corpus of relevant documents extracted from news media be C. When the dataset consists of news articles or blogs, they are extracted from the world wide web along with their corresponding timestamps using key word search. The challenge arises when the dataset consists of tweets and we have to extract related tweets from the Twitter datastream. An effective method to extract tweets from the datastream is Dynamic Pseudo Relevance Feedback (DPRF) which has been given in [7].

4.1.1. *Dynamic Pseudo Relevance Feedback (DPRF)*

In [7], author has used DPRF to mine relevant tweets for a given event query Q, which is a set of user defined keywords or phrases describing an ongoing event in real life. An example of query is 'India Election 2014'. To mine such information, language modelsuses probability distribution $\theta_d$ over the vocabulary W for each document d in the corpus C. By modelling the query as $\theta_Q$, relevant documents can be ranked according to their query likelihood. But, the user queries are short and vague, and are unable to fully express the information needed by the user. For example, the term *plane* may refer to *airplane* and also to a *geometric plane*.Thus to enhance the query expressibility, the author uses query expansion to generate a new high quality query Q' which is used in place of the original query Q.Query expansion is adding context relevant words to the query so that it can better express the information required.

Relevance Feedback (RF) is a method to improve the information retrieval process by involving user intervention in the process. The user gives feedback on the relevance of the documents in an initial set of results. User feedback is taken by asking the user that which documents are relevant and which are not. Then the system utilizes the user feedback to compute a better representation of the information. RF goes through one or more iterations of this sort. Pseudo Relevance Feedback (PRF) automates the manual part of relevance feedback, so that the user gets improved retrieval performance without anyuser intervention. The method assumes that the top k rankeddocuments are relevant, and performs relevance feedback as done previously.

To retrieve relevant tweets for the model using PRF, supposea relevant model $\theta_F$is build by using the few top ranked documents d+ which were generated by the initial query Q. We can represent the new query as a linear combination of relevant model $\theta_F$and original query Q as explained in [8]:

$$p(w|\theta_{Q'}) = (1- \alpha)p(w| \theta_Q) + \alpha p(w| \theta_F) \quad (4.1)$$

where the degree of coherence is controlled by α.Relevance model defines term distribution in $\theta_F$ as the likelihood of generating terms from pseudo relevance:

$$p(w|\theta_F) \propto \sum_{d+\in F} p(d+)p(w|d+)p(Q|d+) \quad (4.2)$$

where $p(Q|d+) = \prod_{q\in Q} p(q|d+)$. Usually in PRF, the prior p(d+) is assumed to be uniformly distributed.. However, in an instant broadcast medium like Twitter this assumption doesn't hold.

For example, if the queries for "Egypt Revolution", tweets that are published on 25/01/2011, which is the starting date of the revolution, would be more relevant than those published on 01/01/2011. The Twitter datastream is not continuous and uniform, but have bursts within certain time periods. Thus we can say that the prior distribution of relevant tweets should be concentratedaround each burst period.DPRF is dynamic i.e. prior probability of relevant document is given by a probability distribution.

### 4.2. *Summarization to Find Representative Documents*

Once we have extracted the relevant documents, based on the user query, we need to summarize the documents in order to reduce redundancy in the storyline. Multi-document summarization techniques commonly use clustering algorithms to generate a summary. A set of documents is treated as a set of sentences.Clustering algorithms are used to cluster these sentences where each cluster consists of sentences pertaining to a single event.Different summarization techniques can be used to summarize the documents. We will discuss some of the techniques here.

#### 4.2.1. *Latent Semantic Analysis (LSA)*

LSA[9] is used for discovering the hidden concepts in text data. It uses Singular Value Decomposition (SVD) to discover the patterns. LSA utilizes the basic concept that two words used in the same context tend to have similar meanings.

The given document corpus is considered as a corpus of sentences. The aim to cluster these sentences according to the event they discuss. In LSA, we construct a term-sentence matrix X where each row represents the word and each column represents the sentence in the corpus. An element $X_{i,j}$ represents the frequency of term i in sentence j. Next we apply SVD on matrix X to the following three matrices.

$$X_{m\times n} = A_{m\times k}\sum_{k\times k}B_{k\times n} \tag{4.3}$$

Here A and B are orthogonal matrices and $\sum$ is a diagonal matrix. For example, consider the following sentences.

d1: "Shipment of gold damaged in a fire."

d2: "Delivery of silver arrived in a silver truck."

d3: "Shipment of gold arrived in a truck."

| Terms↓\ Sentences → | $d_1$ | $d_2$ | $d_3$ |
|---|---|---|---|
| a | 1 | 1 | 1 |
| arrived | 0 | 1 | 1 |
| damaged | 1 | 0 | 0 |
| delivery | 0 | 1 | 0 |
| fire | 1 | 0 | 0 |
| gold | 1 | 0 | 1 |
| in | 1 | 1 | 1 |
| of | 1 | 1 | 1 |
| shipment | 1 | 0 | 1 |
| silver | 0 | 2 | 0 |
| truck | 0 | 1 | 1 |

Table 4.1: Example of term-sentence matrix

Using SVD we calculate A, ∑ and B for some value of k, suppose k = 2.k is the number of sentence clusters we want to form. Thus the semantic vector space consists of 2 dimensions. After using SVD we get $A_2$, $\sum_2$, $B^T_2$.

$$A_2 \; or \; A_{11\,x\,2} = \begin{bmatrix} -0.4201 & 0.0748 \\ -0.2995 & -0.2001 \\ -0.1206 & 0.2749 \\ -0.1576 & -0.3046 \\ -0.1206 & 0.2749 \\ -0.2626 & 0.3794 \\ -0.4201 & 0.0748 \\ -0.4201 & 0.0748 \\ -0.2626 & 0.3794 \\ -0.3151 & -0.6093 \\ -0.2995 & -0.2001 \end{bmatrix} \tag{4.4}$$

$$\Sigma_2 \; or \; \Sigma_{2\,x\,2} = \begin{bmatrix} 4.0989 & 0 \\ 0 & 2.3616 \end{bmatrix} \tag{4.5}$$

$$B_2 \; or \; B^T{}_{2\,x\,3} = \begin{bmatrix} -0.4945 & -0.6458 & -0.5817 \\ 0.6492 & -0.7194 & 0.2469 \end{bmatrix} \tag{4.6}$$

Each column of $A_2$ can be considered as an event description i.e. each event is modelled as a bag-of-words where the number assigned to each word represents its weight in the input.An entry in $i_{th}$ row of matrix $\sum_2$corresponds to the weight of the event, which is represented by the $i^{th}$ column of $A_2$. Matrix $B^T_2$is the new representation of the sentences, where each sentence is expressed as a combination of the events given in $\sum_2$. In the semantic space generated by SVD, the three sentences can be represented as the following.

$d_1$(-0.4945, 0.6492)

$d_2$(-0.6458, -0.7194)

$d_3$(-0.5817, 0.2469)

The matrix $\sum^* B^T$ combines the event weights and the sentence representation to indicate to what extent the sentence conveys the event.An element $x_{ij}$ of the matrix indicates the weight for the event i in sentence j. Once we have represented the sentences in the latent semantic space generated by SVD, we can apply a clustering algorithm such k-means to cluster the data-points into k clusters. Here each cluster will represent a sentence.k-means clustering algorithm is explained next.

4.2.2. *k-means*

k-means is a clustering algorithm which takes the number of clusters, k, as the input parameter and partitions a set of n objects into k clusters such that the similarity between different clusters is high but the similarity within each cluster is low. Cluster similarity is measured with respect to the mean value of the objects in a cluster, hence the name k-means. The working of k-means clustering can be explained as follows –

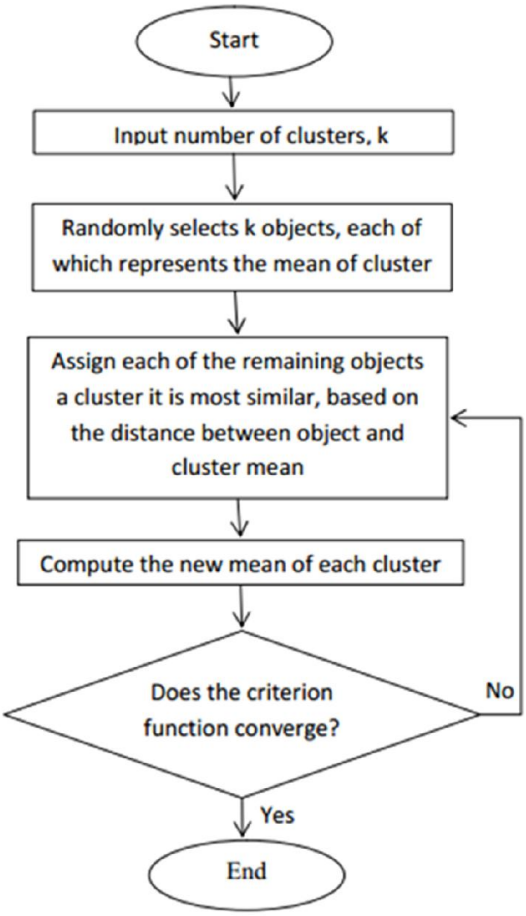


Figure 4.2**:** Flowchart to show k-means clustering

Square-error criterion function is generally used to check for the convergence. Square-criterion function is defined as

$$E = \sum_{i=1}^{k} \sum_{p \in Ci} |p - m_i|^2 \tag{4.7}$$

where*E* is the sum of square error for all objects in the data set; p is the point in the space representing a given object; and $m_i$ is the mean of cluster $C_i$. In other words, the criterion tries to make the resulting k clusters as compact and as separate as possible.

Here the set of objects is the set of all the sentences given in the document corpus. Each sentence is represented in the semantic space generated by SVD. k-means algorithm clusters the sentences where each cluster represents a event.

### 4.2.3. Non-negative matrix factorization(NMF)

Non-negative factorization is used to factorize a matrix X into (usually) two matrices V and U, with the property that all three matrices have no negative elements.As shown in [10], NMF can be used for clustering of documents using the term-sentence matrix derived from the document corpus. In the latent semantic space derived by the NMF, each axis captures the base topic of a particular sentence cluster, and each sentence is represented as an additive(since the two matrices V and U have all positive elements) combination of the base topics.The cluster to which a sentence belongs can be is represented by the axis on which it has the maximum projection value.

NMF differs from LSA, which uses Singular Value Decomposition (SVD), in that the latent semantic space derived by NMF does not need to be orthogonal and each sentence will necessarily have only non-negative values in all the latent semantic directions. Thus each axis in the space derived using NMF has a much more straight-forward correspondence with each document cluster than in the space derived by the SVD. When an overlap exists among clusters, NMF can find the latent semantic direction for each cluster, while this is less likely in the latent semantic space derived by SVD due to its requirement of orthogonality. Also, there is no need to apply clustering methods such k-means(which is needed in LSA) after using NMF on a term-sentence matrix since NMF auto-clusters the data points.

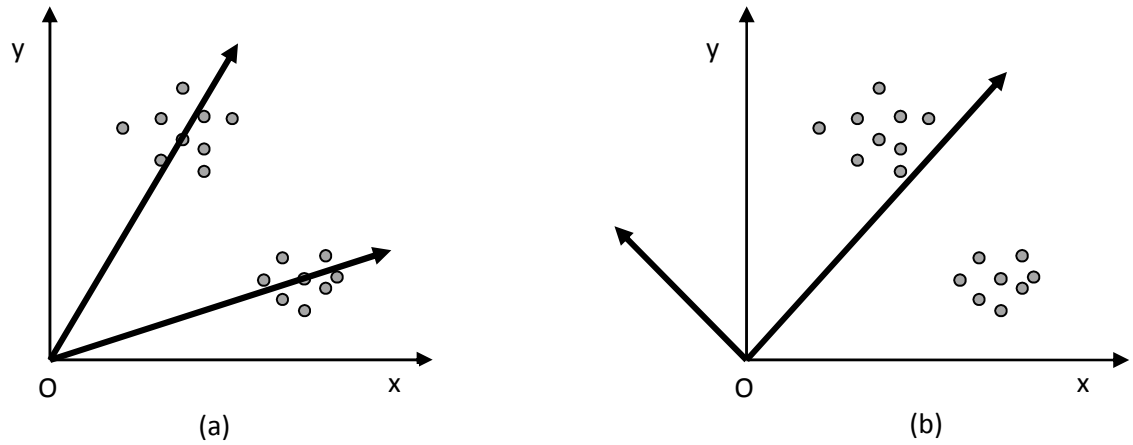


Figure 4.3**:** (a) Directions given by NMF (b) Directions given by LSA

NMF algorithms takes input the number of clusters, k. The algorithm projects each sentence in a k-dimensional semantic space in which each axis corresponds toa particular topic. Consider a term-frequency txdmatrix X, such that each column $X_i$ represents the term-frequency vector for sentence $d_i$. $X_i$is defined as

$$X_i = [x_{1i}, x_{2i}, \ldots, x_{di}]^T \tag{4.8}$$

Where $x_{ij}$ represents the TF-IDF weight of word j in sentence i. Using NMF, we derive two matrices U and V such that

$$X_{t\times d} = U_{t\times k} {V^T}_{k\times d} \tag{4.9}$$

Where k is number of clusters given as a parameter to the algorithm. While factorizing the matrix, we want to minimize objective function J which is defined as

$$J = \frac{\mathbf{1}}{2}\|X - UV^T\| \tag{4.10}$$

Where $||\,.\,||$ denotes the squared sum of all the elements in the matrix. It is proven by Lee[11], that the objective function J is non-increasing under the iterative rules mentioned below, and the convergence of the iteration is guaranteed.

$$\mathrm{u_{ij}} = \mathrm{u_{ij}}\frac{(XV)_{ij}}{(UV^TV)_{ij}} \tag{4.11}$$

$$\mathrm{v_{ij}} = \mathrm{v_{ij}}\frac{(X^TU)_{ij}}{(VU^TU)_{ij}} \tag{4.12}$$

Each element $u_{ij}$of matrix U represents the degree to which term $t_i$ belongs to cluster j.We use matrix V to determine the cluster label of each data point. Assign sentence $d_i$ to cluster xif $x = \underset{j}{\operatorname{argmax}}\, v_{ij}$. A large value of $v_{ix}$ while rest of the elements in i$^{th}$ row vector of V have small values, indicates that sentence i belongs solely to cluster x.

4.2.4. *Minimum Weight Dominant Set (MWDS)*

In [12], the author has used MDWS for multi-document summarization. Given a set of documents a sentence graph G=(V,E,W) is first generated where V is set of sentences in the document corpus, E is the set of edges representing similarity between the sentences and W is set of weights assigned to each vertex. The sentences are represented as vectors based on TF-IDF weights, and then obtain the cosine similarity for each pair of sentences. If the similarity between the sentences $v_i$ and $v_j$ is greater than a threshold, say α, then there is an edge between $v_i$ and $v_j$.MWDS algorithm is applied on the sentence graph to generate a summary.

Summarization can be of two types - (1) General summarization, which generates a summary of the document corpus; (2) Query-focused summarization, which generates summary of the documents based on the user query. For generic summarization, we use all the sentences in the document corpus to build the sentence graph, while for query-focused summarization, we only use the sentences containing at least one term in the query. In addition, for generic summarization the vertices of the sentence graph are not assigned any weights. For query-focused summarization where q is the user query, each node $v_i$ is assigned a weight, $w(v_i) = distance(v_i,q) = 1 - cos(v_i,q)$, to indicate distance between the sentence and the query q.

In [7], the author has used MWDS on multi-view tweet graph to extract the most relevant tweets from Twitter dataset. A multi-view graphis a quadruple G=(V,W,E,A), where V is a set of tweets, W is the weights of V, E is a set of undirected edges, which represents the similarities between tweets, and A is a set of directed edges (arcs), which represents the time continuity of the tweets. Construction of such a graph is controlled by three nonnegative real parameters$\alpha$, $\tau1,\tau2$, $\tau1<\tau2$. There is edge from $v_i$ to $v_j$iff tweet similarity is greater than $\alpha$ and$\tau1 <= t_j - t_i <= \tau2$. $t_i$ and $t_j$are the timestamps of $v_i$ and $v_j$ .Similarity between user query Q and a vertex $v_i$ is given by $score(Q,v_i)$ which is calculated using cosine similarity. Vertex weight,$w(v_i) = 1 - score(v_i)$.

Now we discuss how MWDS can be applied on the sentence graph or mutli-view tweet graph, henceforth referred only as graph G. MWDS is a greedy approximation algorithm to find a minimum weight dominant set. A dominant set(DS) of a graph G is a set of vertices such that every vertex either belongs to DS or is adjacent to a vertex in DS. A minimum dominating set (MDS) is a dominating set with the minimum size. MDS can be naturally as a summary of the document corpus, since each sentence is either in MDS or connected to vertex in MDS. Finding a MDS for a graph is an NP-hard problem, hence a greedy approximation algorithm is suggested in [12]. Starting from empty set, if the current subset of vertices is not in MDS, a new vertex which highest number of adjacent vertex that are not adjacent to any vertex in the current set will be added.The vertex to be added, v*, is determined using the following formula

$$v^* = \underset{v}{\operatorname{argmin}}\, s(v) \tag{4.13}$$

Where s(v) are the vertices adjacent to v but not in MDS. This method is for choosing v* when the vertices are unweighted i.e. in generic summarization. For query-focused summarization, the problem can be modelled as

$$D^* = \underset{D \subseteq G}{\operatorname{argmin}} \sum_{s \in D} d(s, q) \tag{4.14}$$

where D is a dominant set of G. Thus we want to minimize the weight of dominant set, which is the sum of weights of all vertices in the set.Thus we want to add a vertex to MWDS such thatthe weight of a newly added vertex is shared among its newly covered neighbours and selects the node which minimizes this load for each round of iteration. Hence, we choose v* the node to be added as

$$v^* = \underset{v}{\operatorname{argmin}} \frac{w(v)}{s(v)} \tag{4.15}$$

## 4.3. *Connecting Events To Generate Storyline*

Once we have the representative documents we need the connect them appropriately capturing the temporal and structural information of the documents. Some of the effective methods used for the purpose are explained below.

### 4.3.1. *Linear Programming*

In [4], the author has used linear programming to find a good chain. He first defines coherence which is used to develop the objective function. The objective function is used to score a possible chain. Then the highest scoring chain is chosen as the final storyline.

Author has proposed two metrics for scoring a chain – *coherency* and *influence*. A coherent chain has a global coherent theme across the storyline i.e. all the documents in the chain belong to a single theme. Two consecutive documents in a chain has high influence(for a given word w) if the two documents are highly connected and w is important for connectivity.

Suppose D be a set of articles, and W a set of features (typicallywords or phrases). Each article is a subset of W. Given a chain ($d_1$, ...,$d_n$) of articles from D. An intuitive way to form a coherent chain is that every time a word appears in two consecutive documents we score a point.

$$Coherence(d_1, \ldots, d_n) = \sum_{i=1}^{n-1} \sum_{w} 1(w \in d_i \cap d_{i+1}) \tag{4.16}$$

Thus similar documents are placed next to each other. But it has 4 drawbacks – weak links, missing words, importance, jitteriness.

**Weak links:** A chain can have high coherence is most of the links are strong while few links are weak.Thus summing over the transitions will not give the true coherence value. A more reasonable way is to consider the minimum transition score instead of the sum.

$$Coherence(d_1, \ldots, d_n) = \min_{i=1\ldots n-1} \sum_{w} 1(w \in d_i \cap d_{i+1}) \tag{4.17}$$

**Missing Words**: There are cases in which some words donot appear in an article, although they should have.Suppose a document can contain words'court' and 'lawyer'but not 'defence'.However 'defence' can still be a highly-relevant word. Thus only considering words that have appeared in the article will lead to a faulty value of coherence.

**Importance:**On a corpus level as well as on a document level some words are more important than others. Two documents can have numerous common words but more important words must have influence on the transition between the two documents.

To address above two problems, the author suggests the concept of influence of $d_i$ on $d_{i+1}$ through the word w. The calculation of $Influence(d_i, d_{i+1}|w)$ has been discussed later. Intuitively, $Influence(d_i, d_j|w)$ is highif (1) the two documents are highly connected, and (2) w isimportant for the connectivity.

$$Coherence(d_1, \ldots, d_n) = \min_{i=1\ldots n-1} \sum_{w} Influence(d_i, d_{i+1}|w) \tag{4.18}$$

**Jitteriness:** Jitteriness is appearance and disappearance of topics throughout the chain.One way to avoid jitteriness is that we consider the longest continuous stretch of each word. But words can have high influence on a transition even if they do not appear in the documents.The authorarbitrarily defines an activation pattern for each word and then uses these activation patterns to compute the objective function. Now using the best activation patterncoherence can be defined as

$$Coherence(d_1, \ldots, d_n) = \max_{activations} \min_{i=1\ldots n-1}$$

$$\sum_{w} Influence(d_i, d_{i+1}|w) \times 1(w\ active\ in\ \mathrm{d_i}, \mathrm{d_{i+1}}) \tag{4.19}$$

Word activations can be binary or continuous. The author has considered word activation in the range [0,1]. The above mentioned equation is called '**Objective function**'.

To calculate $\text{Influence}(d_i, d_j|w)$ consider the bipartite graph shown in Figure 4.5. Intuitively, if the two documents are connected, a short random walk starting from $d_i$should reach $d_j$ frequently. The stationary distribution is the fraction of the time the walker spends on each node:

$$\pi_i(v) = \epsilon.1(v = d_i) + (1 - \epsilon) \sum_{(u,v)\in E} \pi_i(u)P(v|u) \tag{4.20}$$

Where $\pi_i(v)$is the stationary distribution of random walk starting from $d_i$. $P(v \mid u)$is the probability of reaching v from u and $\epsilon$ is random restart probability.Let $\pi^w{}_i(v)$ be the stationary distribution for graph which has as a sink node. The stationary distribution of $d_j$ would decrease considerably if w was influential. The influence on $d_j$ w.r.t. the word w is defined as the difference between the two distributions, $\pi_i(d_j) - \pi^w{}_i(d_j)$.

Once we have the Objective Function, it can be formalized as a linear program(LP). To do so, we put the following restrictions. We limit the total number of active words and the number of words that are active per transition. We allow each word to be activated at most once. The objective function and the restrictions are sufficient to formulate the problem as LP which is available as a standard module. Further detailed description of LP is beyond the scope of this report.

4.3.2. *Steiner Tree Algorithm(ST)*

In [7], the author has used Steiner Tree Algorithm to generate storylines from tweets. He first uses DPRF followed by MWDS algorithm (discussed previously) to extract the most relevant tweets. These tweets form the vertices of the Minimum Weight Dominant Set graph, that is used as an input to the ST algorithm to form a storyline. A Steiner tree of a graph G with respect to a vertex subset S, known as terminals, is the edge-induced subtree ofG that contains all the vertices of S having the minimumtotal cost. Here the total cost is the minimum total weight of the vertices. We can define the Steiner tree problem as follows -

Consider a directed graph G = (V,W,A) where V is set of tweets/sentences, W is the weights of V, A is set of directed edges(arcs), which represents time continuity of edges. Given a graph G, a set S of

vertices(terminals), and root q ∈ S from which every vertex of S is reachable in G, steiner tree is the subtree of G rooted at q containing S with the smallest total vertex weight.

This problem is known to be NP-hard. An intuitive solution for the problem is to find the shortest path from the root, q, to all the terminals and then merge all the shortest paths. But combining the shortest paths does not guarantee to minimize the total cost of the tree.Charikar and Chekuri in 1999, proposed an approximation algorithm in [13]. The input to the algorithm are vertex-weighted directed graph G = (V,A), a level parameter i ≥ 1, the target terminal set S, the root q, and the required number of nodes to cover in S, k. When i = 1 is a default case where algorithm selects k vertices in S that are closest to root and returns the union of the shortest paths. The algorithm is given below. The initial call of Ai(k,q,S) is made where q is set to the vertex with the earliest timestamp. k is set to the size of S. The output of the tree is interpreted as the storyline transitioning from the root vertex to all the vertices in S.

$T \leftarrow \phi$
**while** k >0 do
  $T_{best} \leftarrow \phi$
  $cost(T_{best}) \leftarrow \infty$
  **for** $v \in V, (v_0, v) \in A, 1 \leq k' \leq k$ **do**
    $T_p \leftarrow A_{i-1}(k',v,S) \cup \{(v_0, v)\}$
    **if** $cost(T_{best} > cost(T')$ **then**
      $T_{best} \leftarrow T'$
    $T \leftarrow T \cup T_{best}$
    $k \leftarrow k - || S \cap T_{best} ||$
    $S \leftarrow S \setminus V(T_{best})$

Figure 4.4: Steiner Tree Algorithm

4.3.3. *Probabilistic Approach*

In [14], the author models the storyline generating problem as a divide and conquer bisecting search problem. The input to the algorithm is a set of documents with their timestamps, a start document(*s*) and an end document(*t*). The author does not intend to generate summaries but re-organize articles in a meaningful and coherent manner.The initial story chain contains only one link*s*-*t*, where *s* is start article and *t* is end article. We need to add nodes to the initial story chain to generate a storyline. Each time we insert a node on a link, the link will be divided into two sub-links. The bisecting search adds a node on each sub-link recursively. The algorithm consists of two parts - (1) Searching an optimum

node to add to the chain. (2) Prune articles to automatically remove least relevant and redundant articles. To do so the author proposes a random walk algorithm as explained below.

Consider a bipartite graph (Figure 4.4) G = (V,E), where V = $V_D$ U $V_W$, $V_D$ is set of documents, $V_W$ is set of words. Edge weights represent the strength of the correlation between a document and a word. A word-document weighing scheme such TF-IDF can be used to assign importance to each word and use these weights for document-to-word edges. Since weights are interpreted as random walk probabilities, they are normalized over all words in thedocument.

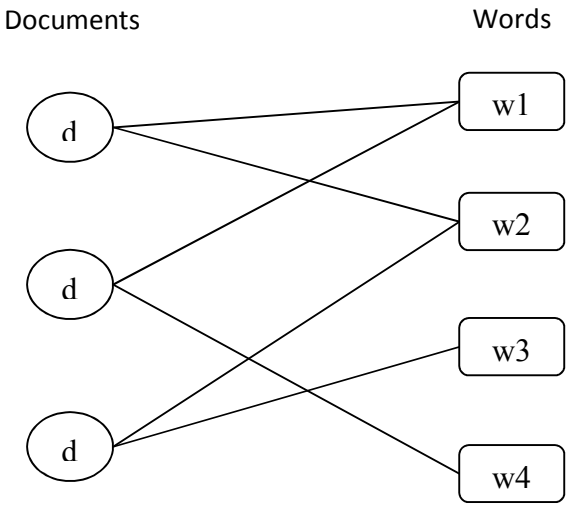


Figure 4.5: A bipartite graph to model word-document relationship

To search for an optimum node to add to the initial link, *s*-*t*, simulate random walks on the bipartite graphto calculate document relevance scores, which are used to select the best nodes to add to the link. To find the most relevant document, say *A*, between the end points *s* and *t* by using the following

$$A = \underset{j}{\operatorname{argmax}} \{r_s(d_i) \times r_t(d_i)\} \tag{4.21}$$

where for any document $d_i$,$r_s(d_i)$ is the probability that a random walk reaches $d_i$from s.Thus the most relevant article has the highest probability of reaching from s as well as from t. Now the problem is reduced to two sub-problems of finding a chain between *s* and *A*, and finding a chain between *A* and *t*.

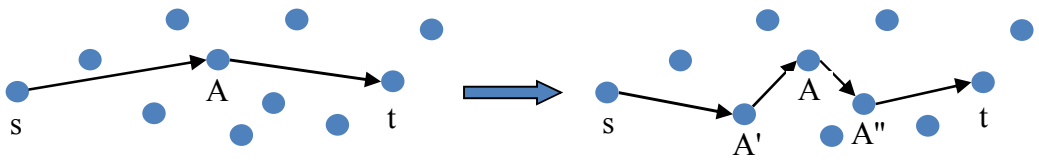

Figure 4.6: Illustration of Probabilistic approach

In actual application, a graph contains thousands of document-nodes and word-nodes. Simulating random walks on such a large graph is time consuming. We need to prune least relevant and redundant documents to reduce the graph size, thus improving the efficiency of the algorithm.First, we discuss the method used for pruning least relevant articles using document-word bipartite graph. The strength of a story chain is the strength of the weakest link in the chain. In [4], the author defines coherence of the chain i.e. strength of the chain as

$$Coherence(d_1,\ldots,d_n) = \min_{i=1..n-1} linkStrength(d_i,d_{i+1}) \tag{4.22}$$

This idea is used to prune least relevant articles. The author simulates a random walk starting from *s* and the walk will terminate at *t*. The algorithm prunes all articles $d_i$ such that

$$r_s(d_i) < r_s(t) \; or \; r_i(d_i) < r_t(s) \tag{4.23}$$

where $r_s(d_i)$ is the probability that a random walk reaches $d_i$ from *s*. Thus, it removes all the documents which are less probable to reach from *s* or from *t*. Next the author suggests a method to prune redundant articles.

We want to select the most representative article but don't want to remove similar articles with different timestamps. We add time nodes to the previous bipartite graph. Article-time weight is always 1 as an article belongs only to a single time frame, time-article weight is normalized over the number of edges from time to article.

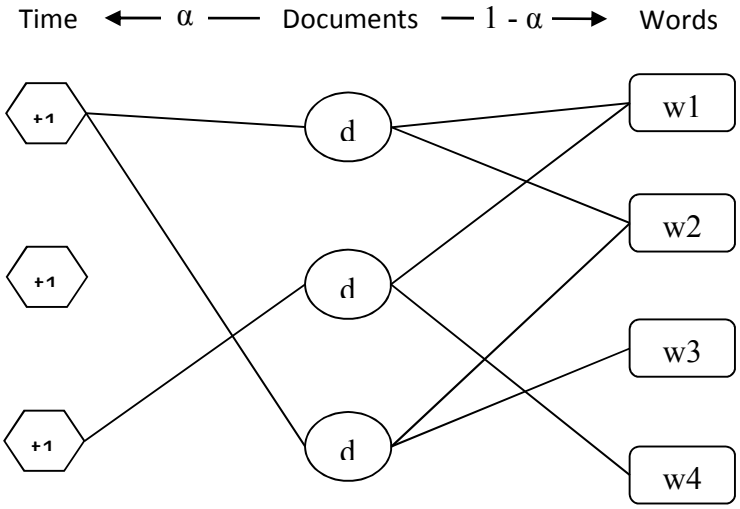


Figure 4.7**:** A tripartite graph to solve redundancy problem

The random walk starts from the middle article nodes.With probability α the walker moves to time-nodes and with probability (1-α) the walker moves to word-nodes. The influence of time nodes is decided by thevalue of α. Articles that are in the same time bin and are close in content are more frequently reached by the walker. One drawback of this graph is that is puts more weight on articles that are in the same time bin and ignores the difference between the bins that are close and bins that are far away.

# 5. CONCLUSION

As we have investigated in the report, Information Overload is major problem facing today's netizens. To encounter this problem we need better and smarter document understanding systems that can help the user to understand the bigger picture. Storyline is a remarkable way of representing the underlying structure of events since it portrays the causal dependencies amongst the different events belonging to a common storyline. Storylines can be used in various domains to connect different events and discover hidden relationships, which otherwise would have taken considerable human effort. Storytelling is a relatively new field of research. The algorithms presently used to generate storylines can be improved to give better and more accurate results.

# REFERENCES


[1] Automatic Text Summarizer. [Online]. http://autosummarizer.com/

[2] David M. Blei, *Probabilistic Topic Models*.: Communications of the ACM, 2012.

[3] Ani Nenkova and Kathleen McKeown, "A Survey of Text Summarization Techniques," *Mining Text Data, Springer*, pp. 43-76, 2012.

[4] Carlos Guestrin, Dafna Shahaf, "Connecting the dots between news articles," *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 623-632, 2010.

[5] Dingding Wang, Tao Li, and Mitsunori Ogihara, "Generating Pictorial Storylines via Minimum-Weight Connected Dominating Set Approximation in Multi-View Graphs," *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[6] M. Shahriar Hossain, Patrick Butler, Arnold P. Boedihardjo, and Naren Ramakrishnan, "Storytelling in entity networks to support intelligence analysts," *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1375-1383, 2012.

[7] Chen Lin et al., "Generating Event Storylines from Microblogs," *21st ACM International Conference on Information and Knowledge Management, CIKM'12*, pp. 175-184, 2012.

[8] ChengXiang Zhai, "Statistical Language Models for Information Retrieval: A Critical Review," *Foundations and Trends in Information Retrieval*, pp. 137-213, 2008.

[9] Scott Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman, "Indexing by Latent Semantic Analysis," *JASIS*, pp. 391-407, 1990.

[10] Wei Xu, Xin Liu, and Yihong Gong, "Document Clustering Based On Non-negative Matrix Factorization," *SIGIR 2003: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 267-273, 2003.

[11] Daniel D. Lee and H. Sebastian Seung, "Algorithms for non-negative matrix factorization," *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems*, pp. 556-562, 2001.

[12] Chao Shen and Tao Li, "Multi-Document Summarization via the Minimum Dominating Set," *COLING 2010, 23rd International Conference on Computational Linguistics*, pp. 984-992, 2010.

[13] Moses Charikar et al., "Approximation Algorithms for Directed Steiner Problems," *J. Algorithms*, pp. 73-91, 1999.

[14] Xianshu Zhu and Tim Oates, "Finding Story Chains in Newswire Articles," *IEEE 13th International Conference on Information Reuse & Integration, IRI, IEEE 2012*, pp. 93-100, 2012.

[15] Dafna Shahaf, Carlos Guestrin, and Eric Horvitz, "Trains of Thought: Generating Information Maps," *Proceedings of the 21st international conference on World Wide Web, WWW 2012*, pp. 899-908, 2012.