Package inputenc Error: Keyboard character used is undefinedin inputencoding ‘applemac’See the inputenc package documentation for explanation.You need to provide a definition with or before using this key.4.232Examples of relations extracted after preprocessingtable.4.2

Conversation Timeline: Tracing Tête-à-Tête in News Media

A DISSERTATION
submitted towards the partial fulfillment of the
requirements for the award of the degree
of

MASTER OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by
ANUNAYA SRIVASTAVA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE – 247667 (INDIA)
May, 2015

Candidate's Declaration

I declare that the work presented in this dissertation with title "**Conversation Timeline: Tracing Tête-à-Tête in News Media**" towards the fulfilment of the requirement for the award of the degree of **Master of Technology** in **Computer Science & Engineering** submitted in the **Dept. of Computer Science & Engineering**, **Indian Institute of Technology, Roorkee**, India is an authentic record of my own work carried out during the period **from July 2014 to May 2015** under the supervision of **Dr. Dhaval Patel**, Assistant Professor, Dept. of CSE, IIT Roorkee.

The content of this dissertation has not been submitted by me for the award of any other degree of this or any other institute.

DATE : .... SIGNED: ....

PLACE: .... (ANUNAYA SRIVASTAVA)

Certificate

This is to certify that the statement made by the candidate is correct to the best of my knowledge and belief.

DATE : .... SIGNED: ....

(Dr. DHAVAL PATEL)
Assistant Professor
Dept. of CSE, IIT Roorkee

PUBLICATION

Part of the work presented in this dissertation has been submitted for publication in the following conference:

Anunaya Srivastava, Dhaval Patel. **Conversation Timeline: Tracing Tête-à-Tête in News Media**, In 24th ACM International Conference on Information and Knowledge Management (**CIKM 2015**).

Acknowledgements

I express my sincere gratitude towards my guide **Dr. Dhaval Patel**, Assistant Professor, Computer Science and Engineering, IIT Roorkee for his able guidance and constant support. Through out my entire graduate research period, he has been consistently encouraging me to pursue quality research. I am thankful to him for this valuable advice, insightful criticisms and constant motivation that helped me to complete this dissertation in time.

I thank **Department of Computer Science and Engineering, IIT Roorkee** for providing lab and other resources for my graduate thesis work.

I would also like to thank **Sahisnu Mazumder** who has not only been a good friend but also a great mentor. He has always helped me to overcome setbacks and stay focused on my thesis work. I also thank **Abhay Prakash** for his support and valuable suggestions in the thesis work. Finally, I thank all my friends and lab-mates for their insightful discussions and for accompanying me during the entire journey and making it delightful.

Abstract

Today there is vast amount of information/news available on the World Wide Web in the form of news articles, blogs and microblogs. Everyday the news media is filled with news about various entities commenting about different events. Thus each event generates a series of conversations between different entities of interest. It would be very interesting for a user to discover how a specific entity has conversed with other entities in the news media about various events happened within a certain time period.We propose a novel visualization technique, “Conversation Timeline” that chronologically displays the conversations of a given entity with other unknown entities which have been reported in news media. We have used news headlines to extract these conversations since news headlines contain the key idea of the comments made by an entity. We use relation extraction algorithm to retrieve conversations from the text. We propose a pre-processing step that syntactically alter the headlines so as the relation extraction module can generate better quality relations. We believe that a Conversation Timeline will give a better understanding about how an entity interacts with other entities of interest.

DEDICATION

I lovingly dedicate this thesis and all my achievements

to

my family

for their endless love, support and encouragement.

Table of Contents

List of Figures

List of Tables

CHAPTER 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. Thus, to encounter the problem of information overload researchers have proposed various types of document understanding systems like (1) Document Summarization (2) Topic Detection and Tracking (3) Storyline generation. In the next sub-sections we give an overview of these techniques.

1.1 Document Understanding Systems

Researchers have proposed various techniques/systems to extract and represent information from a text corpus. We call such systems 'Document Understanding Systems'. We discuss here three of these systems - (1) Document Summarization [1] (2) Topic Detection and Tracking [2] (3) Storyline generation [3]. Summarization algorithms reduce information overload by generating a summary of the entire document. The generated summary, similar to a human-generated summary of the document, is presumed to represent the key idea in the document. These algorithms generate a summary 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 [4], 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 focused 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 dispatch of search parties, rescue attempts, search for the reason of mis-happening and so on.

The goal of Topic Detection and Tracking(TDT), intuitively speaking, is to break the textual data down into individual news stories so that one can monitor the stories for new, unseen events, and group the stories such that each group discusses a single news

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

Table 1.1: Most frequent topics found after processing 17,000 articles from the journal Science

topic. Thus it alerts the user about the new events happening around the world. For example, Table 1.1 shows four of the most frequent topics found after processing 17,000 articles from the journal Science [5]. Here a topic is represented as a bag of inter-related frequently occurring words that correspond to a single topic. In the example shown in Table 1.1, 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 structure means 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.

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 1.2 shows how a one-dimensional storyline connects different events to give an idea of evolution of story. Figure 1.2 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 1.2, it can be easily noticed how the event of clashing of protesters 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.

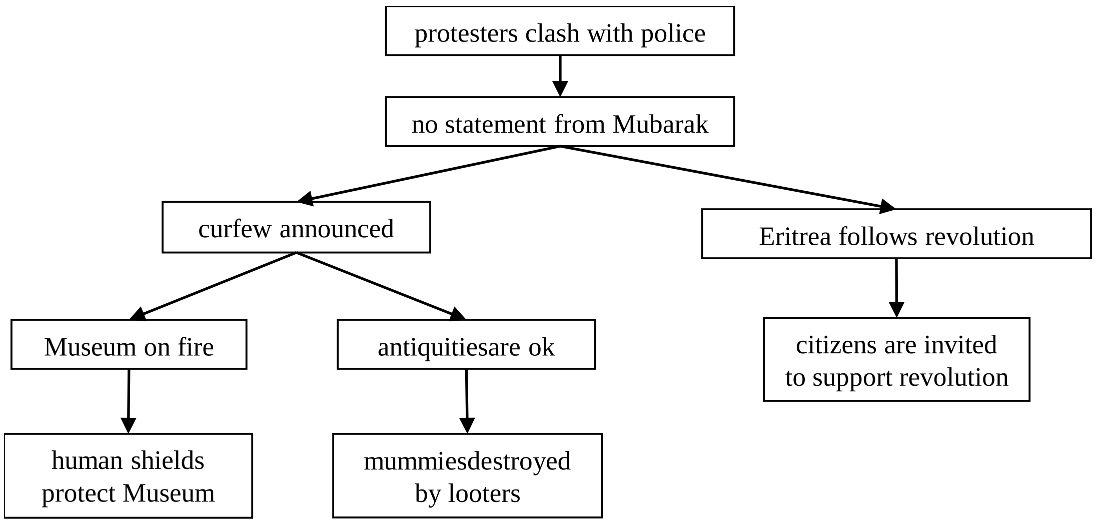

protesters clash with police
no statement from Mubarak
curfew announced
Eritrea follows revolution
Museum on fire
antiquitiesare ok
citizens are invited to support revolution
human shields protect Museum
mummiesdestroyed by looters

Figure 1.2: A storyline for query 'Egypt Revolution'

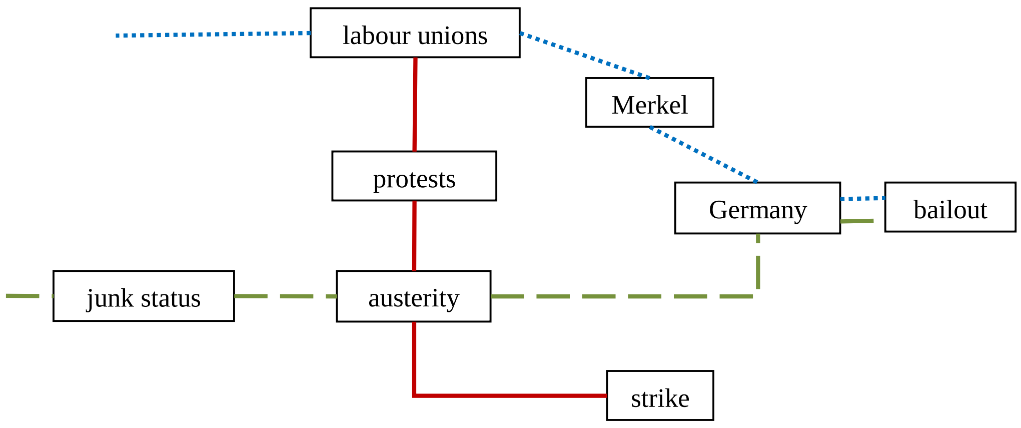


Figure 1.3: A simplified metro map on 'Greek debt crisis'

1.2 Thesis Contributions

The above mentioned techniques are applied on literary textual works, news articles, blogs, microblogs etc. to extract and represent the information more efficiently. However, we were unable to find a technique that represents the interactions between various entities i.e. what and when an entity said about another entity. We aimed at this research gap and planned to develop a technique that can represent how an entity interacts/converses with another entities. Our proposed system takes as input an entity; searches the news media to discover other unknown entities that the given the entity interacts with; and displays these interaction/conversations in a structured way to provide a better understanding to the user.

The proposed system extracts information from a set of news headlines which reduces the computation cost. Every news article published online or otherwise is associated with a news headline. A news headline is a summarized, human-audited textual information representing the key idea of the corresponding news article. Working on the news headlines saves the cost to parse the entire article. For example, the headline '*Arvind Kejriwal invites Kiran Bedi for swearing in ceremony*' is sufficient to determine that the corresponding news article is about Arvind Kejriwal inviting Kiran Bedi.

Secondly, a news headline is syntactically different from a sentence. Unlike well-formed sentences headlines do not follow the standard grammatical rules of English language. The relation extraction algorithm that we use requires that the input text must be well-formed according to the grammatical rules of the language. Thus, the headlines must be altered syntactically before feeding them to the relation extraction algorithm.

The major research contributions in our effort to form Conversation Timeline in this thesis are summarized in the following points:

- We have used news headlines to extract information. This saves the computation time of parsing the entire article. Since, there are multiple news articles on the event/story, parsing each article will consume a lot of time.
- We propose a novel technique to extract and represent conversations between various entities that are reported in the news media. The proposed visualization technique gives a better understanding to the user about how and when an entity interact with other entities.
- We propose pre-processing steps for news headlines so that the relation extraction algorithm can extract more meaningful relations.

1.3 Organization

The rest of this thesis is organized as follows. Chapter 2 discusses about the related works done in the area of summarization, topic detection and storyline generation. We have modelled the conversation extraction problem as a relation extraction problem. Hence, we also mention the various types of relation extraction algorithms in Chapter 2.

Chapter 3 presents the details of the proposed system. We first introduce the various modules present in the system and how they interact with each other. In the following sub-sections we describe each module in detail.

Chapter 4 presents the experimental evaluation of the proposed system. We do quantitative as well as qualitative analysis to show that the pre-processing steps help in extracting better quality relations. Finally, chapter 5 presents the concluding remarks about the thesis and provides suggestions for future research.

Chapter 2

Related Work

The problem of information overload has motivated the research community to develop various document understanding systems. In this chapter we discuss some of the well-known document understanding systems and their strengths and weaknesses. Also, researches have suggested algorithms to discover relationships between various entities. Such algorithms are known as *Relation Extraction algorithms*. We also discuss different types of relations extraction algorithms.

2.1 Understanding Textual Data

2.1.1 Summarization

Summarization systems takes as input a document and produce a concise and fluent summary conveying the key information in the input document. Summarization technique can be applied on a single document or on a multiple documents. In the later case, the method applied is called multi-document summarization. In this sub-section, we discuss some of the famous works done on extractive summarization systems which identify the representative sentences in the input which can be a single document or a set of documents; and string them together of form a summary. [1], discuss various summarization techniques. In all the techniques discussed in [1] the summarization task can be divided in three parts (1) Topic representation (2) Score sentences (3) Select summary sentences.

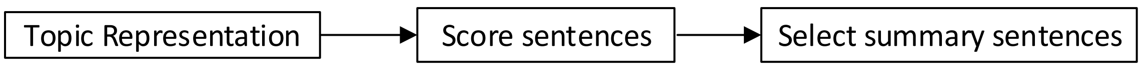


Figure 2.1: Process of summarization

Topic representation methods convert the textual data into 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.

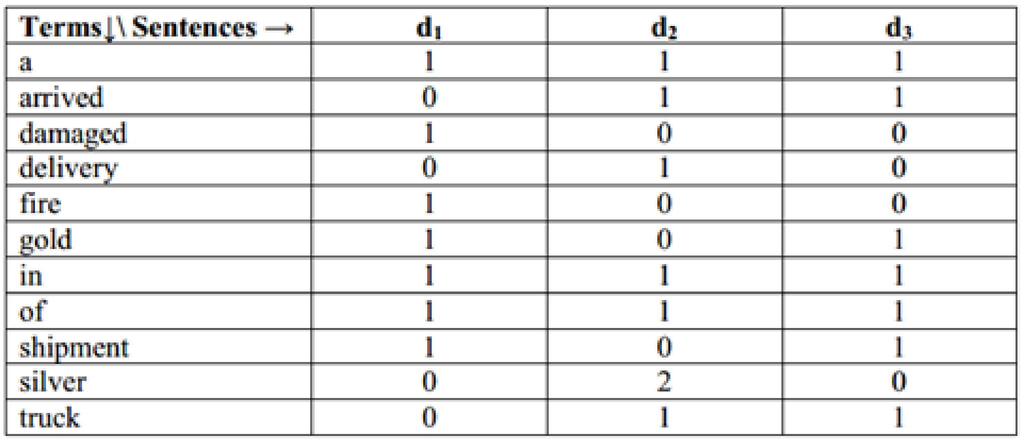
Multi-document summarization techniques commonly use clustering algorithms to generate a summary. A set of documents is treated as a set of sentences. Clustering algorithms are used to cluster these sentences where each cluster consists of sentences pertaining to a single event. Different summarization techniques can be used to summarize the documents. We will discuss some of the techniques here.

1. Latent Semantic Analysis (LSA)

   LSA[6] 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.

$$X_{m\times n} = A_{m\times k} \times \Sigma_{k\times k} \times B_{k\times n} \tag{2.1}$$



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

Figure 2.2: Term-sentence matrix for given example

Here A and B are orthogonal matrices and Σ 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.”

Using SVD we calculate A, Σ 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, Σ_2, B_2^T

$$A_2 \; or \; A_{11\times 2} = \begin{bmatrix} -0.4201 & 0.0748 \\ -0.2995 & -0.2001 \\ -0.1206 & 0.2749 \\ -0.1576 & -0.3046 \\ -0.1206 & 0.2749 \\ -0.2626 & 0.3794 \\ -0.4201 & 0.0748 \\ -0.4201 & 0.0748 \\ -0.2626 & 0.3794 \\ -0.3151 & -0.6093 \\ -0.2995 & -0.2001 \end{bmatrix} \tag{2.2}$$

$$\Sigma_2 \; or \; \Sigma_{2\times 2} = \begin{bmatrix} 4.0989 & 0 \\ 0 & 2.3616 \end{bmatrix} \tag{2.3}$$

$$B_2 \; or \; B^T_{2\times 3} = \begin{bmatrix} -0.4945 & -0.6458 & -0.5817 \\ 0.6492 & -0.7194 & 0.2469 \end{bmatrix} \tag{2.4}$$

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 Σ_2 corresponds to the weight of the event, which is represented by the i^{th} column of A_2. Matrix B_2^T is the new representation of the sentences, where each sentence is expressed as a combination of the events given in Σ_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 $\Sigma \times B^T$ 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.

2. Minimum Weight Dominant Set (MWDS):

   In [7], 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 [3], 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 non-negative real parameters $\alpha, \tau_1, \tau_2, \tau_1 < \tau_2$. There is edge from v_i to v_j iff tweet similarity is greater than α and $\tau_1 <= t_j - t_i <= \tau_2$. 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 [7]. 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

$$v^* = \underset{v}{\operatorname{argmin}}\, s(v) \tag{2.5}$$

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

$$D^* = \operatorname*{argmin}_{D \subseteq G} \Sigma_{s \in D} d(s, q) \tag{2.6}$$

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 that the 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

$$v^* = \operatorname*{argmin}_{v} \frac{w(v)}{s(v)} \tag{2.7}$$

Limitation of Summarization. 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.

2.1.2 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 ,Äì

1. *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.

2. *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.

3. *Topic Detection* is the problem of grouping or clustering of news stories as they arrive, depending upon the topic they are based on.
4. *Topic Tracking* is the monitoring or tracking of news stories to determine how a specific topic evolves with time.
5. *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.

The most famous algorithms for detecting topics from a document corpus is *Latent Dirichlet Allocation (LDA)*. LDA is a generative probabilistic model. A diagram of the graphical model showing how the different random variables are related is shown in Figure 2.3 the diagram, each random variable is represented by a circle (continuous) or square (discrete). A variable that is observed (its outcome is known) is shaded. An arrow is drawn from one random variable to another if the the outcome of the second variable depends on the value of the first variable. A rectangular plate is drawn around a set of variables to show that the set is repeated multiple times, as for example for each document or each token.

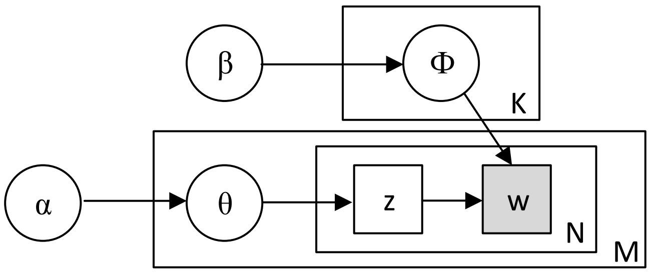


Figure 2.3: LDA Graphical Model

- α is the parameter of the Dirichlet prior on the per-document topic distributions,
- β is the parameter of the Dirichlet prior on the per-topic word distribution,

- θ_i is the topic distribution for document i,
- ϕ_k is the word distribution for topic k, z_{ij} is the topic for the j^{th} word in document i, and
- w_{ij} is the specific word.

LDA works on the idea that if two words co-occur multiple times, it is probable that they belong to the same topic. Since it is a generative process, it assumes that each document has a topic distribution (θ); and each topic has a word distribution (ϕ). θ and ϕ are latent variables which we need to calculate. The words present in a document is the only observed variable.LDA assumes the following generative process for a corpus D consisting of M documents each of length N_i.

1. Choose $\theta_i \sim \text{Dir}(\alpha)$, where $i \in \{1,\ldots,M\}$ and $\text{Dir}(\alpha)$ is the Dirichlet distribution for parameter α
2. Choose $\phi_k \sim \text{Dir}(\beta)$, where $k \in \{1,\ldots,K\}$
3. For each of the word positions i,j, where $j \in \{1,\ldots,N_i\}$, and $i \in \{1,\ldots,M\}$
   (a) Choose a topic $z_{i,j} \sim \text{Multinomial}(\theta_i)$.
   (b) Choose a word $w_{i,j} \sim \text{Multinomial}(\phi_{z_{i,j}})$.

The output is a bag-of-words representation of each topic.

Limitation of TDT. 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. To better understand and represent this inter-dependence of events we have stoylines have been proposed.

2.1.3 Storyline Generation

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 linear ([8]) which is made by ordering various events such that they form a coherent and meaningful chain of events. Storyline can also be branched [3] portraying how a single event leads to emergence of other events. Different storylines can also intersect with each other, thus forming a two-dimensional structure is known as Metro Map [9]. 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.

In [8], the input to the system is a set documents which should be ordered in a meaningful way to make sentence to the user. For n documents, there are $n!$ permutations/chain of the documents. The author proposes an objective function that 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 ‚Äì *coherency* and *influence*. A chain is coherent if it has a global coherent theme from the starting document to the ending document i.e. the documents in the chain talk about the same topic. *Influence* is defined for a pair of documents w.r.t. a word w. 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. The author says that a good storyline must be highly coherent and for high coherency each consecutive pair of documents in the chain must have high influence summed over all words in the document. Thus coherence of a chain is given by the formula –

$$Coherence(d_1,...,d_n) = \max_{activations} \min_{i=1..n-1} \sum_{w} Influence(d_i,d_{i+1}|w) \times 1(w active in d_i,d_{i+1}) \tag{2.8}$$

$Influence(d_i,d_{i+1}|w)$ is calculated using a random walk over a bipartitie graph where vertex set is the union of set of documents and set of words. 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:

$$\pi_i(v) = \epsilon.\mathbb{1}(v = d_i) + (1-\epsilon) \sum_{(u,v)\in E} \pi_i(u)P(v|u) \tag{2.9}$$

Where $\pi_i(v)$ is the stationary distribution of random walk starting from d_i. $P(v|u)$is the probability of reaching v from u and ϵ is random restart probability. Let œÄ$_i^w(v)$ 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, $\pi_i(d_j) - \pi_i^w(d_j)$.

Limitation of Storyline. Storyline relates various events using temporal dependency but still it doesn't tell about anything about the entities involved in the events. Any event comprises of various entities which make statements about each other related to the specific event. To shed light on the interaction between the entities we have proposed a new system which tell us how an entity has been interacting with other entities.

2.2 Relation Extraction

Relation extraction is the task of detecting and characterizing the semantic relations between entities in text. For example, from the sentence fragment, *Facebook co-founder Mark Zuckerberg* we can extract the relation *FounderOf(Mark Zuckerberg,Facebook)*. A relation can also be expressed in the form of an ordered triplet, for example $\langle MarkZuckerberg, FounderOf, Facebook\rangle$. In this sub-section, we discuss the various kinds of algorithms that have been proposed for relationship extraction. The relation extraction algorithms can be classified as -

1. Classified Relation Extraction
2. Weakly Supervised Relation Extraction
3. Unsupervised Relation Extraction

2.2.1 Classified Relation Extraction

A typical approach to relation extraction is to treat the task as a classification problem [10]. Specifically, any pair of entities co-occurring in the same sentence is considered a candidate relation instance. The goal is to assign a class label to this instance where the class label is either one of the predefined relation types or nil for unrelated entity pairs. Classification approach assumes that a training corpus exists in which all relation mentions for each predefined relation type have been manually annotated. Each candidate relation instance is represented by a set of features that are carefully chosen. Standard learning algorithms such as support vector machines and logistic regression can then be used to train relation classifiers.

2.2.2 Weakly Supervised Relation Extraction

In contrast to classification methods for relation extraction which rely on a large amount of training data, weakly supervised learning methods rely on much less training data. Bootstrapping is the most widely used relation extraction method that uses weakly supervised learning. It starts with a small set of seed relation instances and iteratively learns more relation instances and extraction patterns.

Agichtein and Gravano in [11] suggest a system called 'Snowball' which uses bootstrapping strategy for relation extraction. We choose a target relation (such as `SonOf`) and a set of seed entity pairs that are already related through the target relation. For example, if the target relation is `SonOf`, we may use seed pairs such as $\langle RahulGandhi, RajivGandhi\rangle$, $\langle AbhishekBachchan, AmitabhBachchan\rangle$, $\langle AkhileshYadav, MulayamSinghYadav\rangle$. Given a large corpus, we then look for co-occurrences of these entity pairs which are similar to these. We assume that if two entities that are related through the target relation co-occur closely, the text that contains these entities is likely to be a pattern for the target relation. For example, we may find sentence fragments such as 'Rahul Gandhi, son of Rajiv Gandhi' and 'Abhishek Bachchan, son of Amitabh Bachchan' and extract patterns like `SON`*, son of* `FATHER`. We can use these patterns to find more $\langle SON, FATHER\rangle$ pairs in the corpus. We add these pairs to the set of seed entity pairs and repeat the process. More patterns and entity pairs are added to the results until a certain condition is satisfied.

2.2.3 Unsupervised Relation Extraction

The two methodologies described above require that the relation types and the entity types must be defined in advance. They also need a good amount of training data. Also, defining the structure of information and annotating documents according to the defined structures require human expertise and are time consuming. Unsupervised relation extraction helps in overcoming these problems. It is also called ‘Open Information Extraction’ or Open IE. The goal is to find all the potentially useful relations from a large and diverse corpus such as the Web.

As the name indicates, it is unsupervised and does not assume any specific target relation type. It tries to generate as many relations as possible in a single pass over the corpus. The relations extracted are in the form of triplets - $\langle arg1, reln, arg2\rangle$. A famous unsupervised relation extraction algorithm is WOE [12]. It uses Wikipedia infobox values for the automatic construction of training examples. There are two kinds of relation extractors, one WOE$^{\text{parse}}$ using features from dependency-parse trees and the other WOE$^{\text{pos}}$ limited to shallow features like POS tags. WOE$^{\text{parse}}$ uses a pattern learner to classify whether the shortest dependency path between two noun phrases indicates a semantic relation.

2.2.4 WOE Relation Extraction Algorithm

WOE [12] is an unsupervised relation extraction algorithm that uses Wikipedia for the construction of training examples. These examples are used to generate an unlexicalized, relation-independent (open) extractor. WOE has three main components : pre-processor, matcher and learner.

(a) **Pre-processor**: It converts the raw Wikipedia text into a sequence of sentences, attaches NLP annotations (POS tags and NP-chunk annotations), and builds synonym sets for key entities (Indian Institute of Technology, Roorkee is equivalent to IIT Roorkee). It uses Apache’s Open NLP library for NLP annotations. For compiling synonyms, it uses Wikipedia redirection pages and backward links to automatically construct synonym sets.

(b) **Matcher**: The matcher constructs training data for the learner component by heuristically matching attribute-value pairs from Wikipedia articles containing infoboxes with corresponding sentences in the article. Given the article on “Indian Institute of Technology, Roorkee” for example, the matcher should associate $\langle established, 1857\rangle$ with the sentence “The institute was founded in 1857 by ....”. Given a Wikipedia page with an infobox, the matcher iterates through all its attributes looking for a unique sentence that contains references to both the subject of the article and the attribute value; these noun phrases will be annotated arg1 and arg2 in the training set. The matcher considers a sentence to contain the attribute value if the value or its synonym is present.

(c) **Learner**: There are two kinds of extractors, one (WOE^{parse}) using features from dependency-parse trees and the other (WOE^{pos}) limited to shallow features like POS tags. WOE^{parse} uses a pattern learner to classify whether the shortest dependency path between two noun phrases indicates a semantic relation. In contrast, WOE^{pos} trains a conditional random field (CRF) to output certain text between noun phrases when the text denotes such a relation.

In the paper, author concludes that WOE^{parse} was able to extract more relations than WOE^{pos}. We discuss WOE^{parse} in details below. The author used Stanford Parser to generate collapsed dependencies since these are simplified patterns which are useful for relation extraction. Consider the sentence *Dan was not born in Berkeley.* The Stanford Parser dependencies are:

`nsubjpass`(born-4, Dan-1)
`auxpass`(born-4, was-2)
`neg`(born-4, not-3)
`prep` in(born-4, Berkeley-6)

These dependencies form a directed graph $\langle V, E\rangle$ where each token is a set of vertex in V, and E is a set of dependencies. For any pair of tokens, such as *Dan* and *Berkeley*, the author uses the shortest connecting path to represent the possible relation between them. It is called *corePath*.

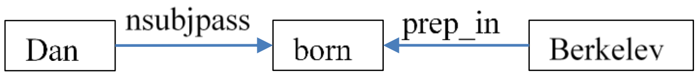


Figure 2.4: Example of corePath

corePaths indicate that a relation exists between tokens, but they don't necessarily capture the semantics of that relation. For example, the corePath illustrated above doesn't show the existence of negation. To capture the semantics of the relation, the authors

Words identified by	**Replaced by**
Noun POS tags and “PRP” tags	“N”
Verb POS tags	“V”
Adverb POS tags	“RB”
Adjective POS tags	“J”
Preposition dependencies such as “prep in”	“prep”

Table 2.1: Tag replacement scheme

suggests adding all adverbial and adjectival modifiers as well as dependencies like ‘neg’ and ‘auxpass’. It is called *expandPath*.

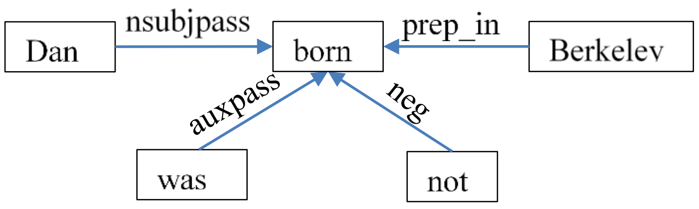


Figure 2.5: Example of expandPath

Since the aim is to extract “subject, relation, object” triples, the learner rejects corePaths that don’t start with subject-like dependencies, such as `nsubj`, `nsubjpass`, `partmod` and `rcmod`. Next, the corePaths are generalised by removing the domain specific information i.e. by extracting the syntactic information from the corePath. Each word is the corePath is replaced by its POS tags. The replacement procedure follows the following rules –

This new path is called *generalised-corePath*. The corePath shown in Figure‘2.5 becomes

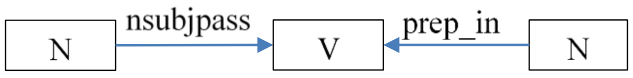


Figure 2.6: Example of generalised-corePath

The author calls this generalised-corepath an *extraction pattern*. In total, WOE builds a database (called DB_p) such distinct patterns and each pattern p is associated with a frequency ‚Äî the number of matching sentences containing p.

Once WOE prepares a database of extraction patterns, it needs to provide learning parameters to a pattern classifier so as to extract the relevant extraction patterns. The learner builds a simple pattern classifier, named WOE$^{\text{parse}}$, which checks whether the generalized-corePath from a test triple is present in DB_p, and computes the normalized logarithmic frequency as the probability $w(p) = \frac{\max(\log f_p - \log f_{min}, 0)}{\log f_{max} - \log f_{min}}$, where f_{max} is the maximal frequency pattern in DB_p and f_{min} is the controlling threshold that determines the minimal frequency of a valid pattern. The expandPath stores the semantic relation between the two tokens. Hence to output the final relation, WOE$^{\text{parse}}$ traverses the triple's expandPath to produce the final expression $\langle Dan, wasNotBornIn, Berkeley \rangle$.

CHAPTER 3

Proposed Work

In this section we discuss our proposed system - **Conversation Timeline**. Before describing the system, we should give a couple of definitions.

Real-world entity: A real-world entity is the name of a real-world person or organisation, such as 'Arvind Kejriwal', 'Indian Institute of Technology, Roorkee' etc.

Relation: A relation is a triple ⟨`ent1, reln, ent2`⟩ where `ent1` and `ent2` are real-world entities, and `reln` is a semantic relationship between them such that `ent1` is related to `ent2` by the relationship `reln`. For example, ⟨`Rahul Gandhi, SonOf, Rajiv Gandhi`⟩ `Rahul Gandhi` is related to `Rajiv Gandhi` by the relationship `SonOf`. Relations can also be represented as `reln(ent1, ent2)`, for example, `SonOf(Rahul Gandhi, Rajiv Gandhi)` or `ParentOf(Rajiv Gandhi, Rahul Gandhi)`.

We have modelled the conversation extraction problem as relation extraction problem which is a common research area. We represent each conversation as a relation i.e. a triplet ⟨`ent1, reln, ent2`⟩ which means that `ent1` has said something (`reln`) about/to `ent2`. An entity can be a person or an organization. For example, '*Arvind Kejriwal invites Kiran Bedi for swearing in ceremony*' can be represented as the relation ⟨`Arvind Kejriwal, invites, Kiran Bedi`⟩ . But relation extraction algorithms are designed for sentences following the grammatical structure of English language. They are unable to extract meaningful relations from headlines which have a unique syntactic structure. For example, the headline 'Arvind Kejriwal: We will apologise to the people of Delhi' will generate the relation ⟨`We, apologise to, the people of Delhi`⟩ while we want ⟨`Arvind Kejriwal, apologise to, the people of Delhi`⟩.

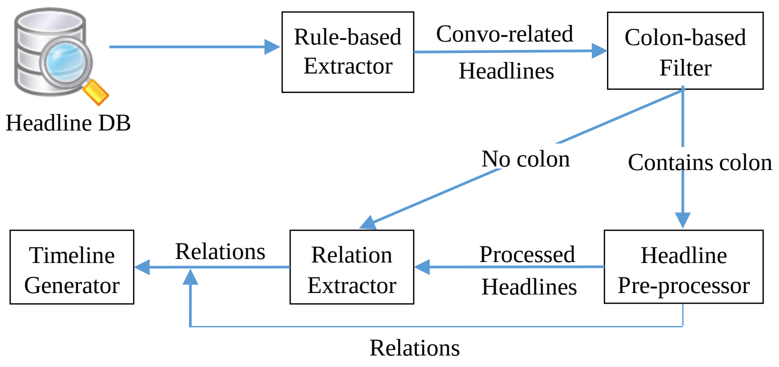


Figure 3.1: Conversation Timeline Architecture

To handle the unique syntactic structure of headlines we propose the following system. The rule-based extractor searches for the headlines which may contain conversation involving the input entity. Let this set be called S. These conversation related headlines then go to colon-based filter. We noticed that headlines containing a colon, ':' frequently depicts that an entity is making a statement. Colon-based filter partitions S into S_{CC} (headlines containing colon) and S_{NC} (headlines not containing colon). Headlines in S_{CC} then goes to headline pre-processor, which in some cases can directly (without using relation extraction algorithms) extract the relation, if not it alters the syntactic structure of headline so that the relation extraction algorithms (relation extraction module) can extract better quality relations. Headlines in S_{NC} directly goes to relation extraction module. We use WOE^{parse} algorithm for relation extraction. The generated relations are used by timeline generator module to form the conversation timeline.

Our contributions are as follows –

1. We have used the news headlines to extract information. This saves the computation time of parsing the entire article. Since, there are multiple news articles on the event/story, parsing each article will consume a lot of time.

2. We give a novel document understanding technique (*Timeline Generator Module*)

that is implemented by extracting interactions or relations between different entities along with their timestamps using news headlines.

3. We propose (*Rule-based Extractor Module*) that identifies if a headline contains conversation or not.
4. We give steps to pre-process the news headlines (*Headline Pre-processor module*) so that the relation extraction algorithms can extract better quality relations from the headlines.

The details of each module are explained in the following sub-sections.

3.1 Rule-based Extractor

We need to extract the relevant headlines from the web so that they can be further parsed to extract entities and relationships. We have used the news headline dataset mentioned in [13]. This dataset has been built using a web crawler that crawls each of a predefined list of 87 news websites at an interval of 30 minutes and then, collects information about the news headlines published during that time interval. It runs on an Intel Xeon Machine with 32 processors and 64 GB RAM with an Ethernet access of 100 Mbps.

The records from the news headline database are extracted using the following rules –

Rule 1: For the given user query, extract records that contain any of the words in the user query. For example, for the user query 'arvind kejriwal' extract records containing 'arvind' OR 'kejriwal'.

Since, we want only the records that depict some kind of interactions between entities, we look out for specific verbs in the sentence. These verbs are called 'Verbs of Attribution'[14]. These verbs are used when the writer is quoting, paraphrasing or referring to another source. For example, says, accepts, addresses, advices etc. are verbs of attribution. We compiled a list of such verbs of attribution, and we call it *VList* (short for Verb List). A list of verbs that make the VList is as follows ‚Äì

Also, a colon ':' is frequently used in news headlines which involves an entity making a statement. So, we form the next rule.

says	expresses	indicates	exclaims	answers	concurs with
calls	hypothesizes	questions	mentions	compares	supports
retorts	offers	advises	responds	echoes	agrees
flays	states	cites	assumes	lists	confirms
blasts	acknowledges	defines	considers	replies	verifies
tells	categorizes	insists	finds	wonders	concedes
tweets	illustrates	uses	notes	asks	counters
posts	points out	allows	reveals	concedes	disagrees
requests	suggests	claims	assures	emphasizes	opposes
accuses	addresses	describes	contends	maintains	criticizes
threatens	challenges	interprets	grants	reports	disputes
praises	proposes	reasons	observes	writes	refutes
accepts	charges	utilizes	shows	asserts	denies
believes	declares	analyzes	argues	concludes	objects
introduces	offers	comments	explains	warns	rejects
remarks	speculates	discusses	holds	affirms	

Table 3.1: List of verbs of attribution

Rule 2: If the news headline contains any of the verbs mentioned in VList or a colon ‘:’, we use it for further analysis.

3.2 Headline Pre-processor

News headlines have different syntactical structure. They do not follow the standard grammatical rules a language (in our case English). For eg. colon ‘:’ is extensively used in headlines to state the statement made by a person or organization. For eg. ‘*Kejriwal on Elections: We will apologize to Delhi*’. Also, ‘to’ is used to state that a person has said something about another person/organization. For eg. ‘*Discom creating ugly situations, govt. may step in: Kejriwal to Jung*’

A relation extraction algorithm such as WOE assumes that the input sentence follows the standard grammatical rules. WOE uses a dependency parser, which is based on the grammatical rules, to extract relations from the sentence. This specific syntactical structure of news headlines hinders the relation extraction algorithms to extract relations efficiently.

Hence, keeping in mind the specific syntactical structure of news headlines we propose certain rules to pre-process the headlines so as to make the relation extraction process more efficient. Our rules apply specifically on headlines containing colon ':'. Such headlines can be divided into two parts - *header* and *body*, which are separated by a colon. Header tells about the topic or the entity which the headline is about; and body tells the details or the statement from the entity. For eg. for the headline, '*Kejriwal on Elections: We will apologize to Delhi*' header is '*Kejriwal on Elections*' and body is '*We will apologize to Delhi*'. Header can precede body or vice-versa. We assume that header is shorter in length than the body, and use this property to identify the header and the body in a headline.

This module consists of two types of mechanisms - (1) Those that pre-process the text so as to enable the relation extraction algorithm to work effectively. (2) Those that utilizes the headlines unique syntactic structure to directly identify the relations, i.e. without using relation extraction algorithm. We noticed that news headlines contain specific patterns or templates that can be used to extract useful information from the headline. We discuss below the proposed mechanisms to identify and utilize these templates –

1. Headlines state if an entity makes an statement about another entity. For eg. *Discom creating ugly situations, govt. may step in: Kejriwal to Jung*. These can be represented by the template where 'to' appears in the header.
   $$\langle ent1 \rangle \textbf{ to } \langle ent2 \rangle : \langle statement \rangle$$
   It gives the relation
   $$\langle ent1, statement, ent2 \rangle$$
   No further processing (relation extraction algorithm) is required.

2. Headlines state if an entity is speaking about a topic. For eg. *Ex-CM Kejriwal on Elections: We will apologize to Delhi*. Such headlines can be represented by the following template where 'on' appears in the header:
   $$\langle entity \rangle \textbf{ on } \langle topic \rangle : \langle statement \rangle \text{ or } \qquad NNP \textbf{ on } NNP : \langle statement \rangle$$
   Here entity and topic is connected with the preposition 'on'. It can be reduced to
   $$\langle entity \rangle : \langle statement \rangle$$
   which will be further processed.

The above two rules are to process the header part of the headline. Following two rules are to process the body part of the headline.

3. The body starts with a pronoun. For eg. *Arvind Kejriwal: I can sacrifice CM seat 100 times*. It can be reduced to $\underline{\textit{Arvind Kejriwal can}}$ *sacrifice CM seat 100 times*.

4. The body starts with a proper noun. For eg. *Arvind Kejriwal: Reliance is accused of looting public*. It can be represented by the template
$$ent1 : ent2\langle statement\rangle$$
It can be reduced to the relation
$$\langle ent1, statement, ent2\rangle$$
No further processing(relation extraction) is required.

5. The header contains words like 'live', 'opinion', 'watch', 'read' etc. which do not contain entities Such headers are not useful to us and can be safely ignored. We ignore the header of such headlines and pass the body to relation extraction algorithm.

The steps explained above are summarized in in Table 3.2 and are to be executed in the order as mentioned. Step 1 and Step 4 gives direct relations that doesn't need further processing. Step 2 and Step 4 transforms the headline to match the syntax of English grammar.

Steps	**Template**	**Transformation**
Step 1	$\langle ent1\rangle to\langle ent2\rangle : \langle statement\rangle$	Relation extracted $\langle ent1, statement, ent2\rangle$
Step 2	$NNPonNNP : \langle statement\rangle$	$\langle entity\rangle : \langle statement\rangle$ To be processed further
Step 3	$NNP : PRP\langle restofbody\rangle$	$NNP\ VB/MD\langle restofbody\rangle$ To be processed further
Step 4	$NNP : NNP\langle statement\rangle$	Relation extracted $\langle ent1, statement, ent2\rangle$

Table 3.2: Steps in Headline Pre-processor

3.3 Relation Extractor

This module is responsible for extracting relation triplets from the headlines. We use WOE$^{\text{parse}}$ [12] algorithm for relation extraction. WOE$^{\text{parse}}$ processes each headline using NP-chunking software and dependency parsing software. Due to the use of dependency parse features WOE$^{\text{parse}}$ performs better than other relation extraction algorithms.

Since dependency parsers use the grammatical sentence formation rules for finding dependencies in a sentence, it is pre-requisite that the input sentences must follow the grammatical rules of the language. In the proposed system, the headline pre-processor module alters the grammatical structure of the headline so that the relation extraction module can give better results.

3.4 Timeline Generator

This module takes as input relation triplets and forms the conversation timeline. Each triplet also stores the timestamp when the headline first appeared in the news media. This timestamp is used to order the relations chronologically. We order the entities (represented by vertical lines) from left to right in decreasing order of their frequency of interaction. The left most entity is the input entity with which every other entity interacts.

Chapter 4

Experimental Study

4.1 Dataset

To extract the conversations between entities we have used the news headlines dataset developed by Sahisnu et. al. in [13]. This dataset has been built using a web crawler that crawls each of a predefined list of 87 news websites at an interval of 30 minutes and then, collects information about the news headlines published during that time interval. The crawler stores the following fields -

1. News headline text
2. Start timestamp: The time stamp when the news headline appeared in the news media
3. End timestamp: The time stamp when the news headline disappeared from the news media

The dataset contains news headlines published from 28-01 -2014 to 04-01 -2015. There are 3773677 headlines records in the dataset. Table 4.1 shows the sample news headline data.

News Headline Text	Start Timestamp	End Timestamp
Aziz Qureshi takes oath as Uttar Pradesh governor	2014-11-28 21:17:01	2014-11-29 14:17:01
Hughes died as a result of vertebral artery dissection: Doctor	2014-11-28 21:17:01	2014-11-29 14:17:01
Sharif says Pak ready to discuss all issues with India	2014-11-28 21:17:01	2014-11-29 14:17:01

Table 4.1: Sample of news headline data

4.2 Experimental Study

In this section we discuss the experiments that we performed to access the outputs and affects of pre-processing the headlines on the output. We anyalyse the results quantitatively and qualitatively to record the improvement in the relations generated. First we discuss the statistical analysis performed on 4 entities, where we show the improvements in the relation extraction due to pre-processing. Next we discuss 2 case studties each of which shows a conversation timeline and how it can convey useful information.

A naive approach is to directly use the relation extraction algorithm on the news headlines without pre-processing them. We used WOE^{parse} relation extraction algorithm to extract relation from conversation-related headlines i.e. headlines that are filtered out by *Rule-based Extractor*. We observed that the relations extracted do not make much sense to a user. Due to the ungrammatical syntax of the headlines the relation extraction algorithm is unable to extract relations in most of the cases or if it is able to the extracted relations are not meaningful. We use this method as a baseline approach to evaluate the performance of the proposed system. We show that pre-processing enables the relation extractor to extract more meaningful relations from the headlines.

We preformed the experiments on 4 entities to show that pre-processing enables to extract more relations from the headlines. TABLE 4.3 shows these the statistics of the experiment. The column *Related headlines in DB* shows the number of headlines in the dataset that are related to the input entity. All of these headlines are not used for conversation extraction. The conversation related headlines are filtered out by *Rule-based Extractor*. Its output is shown in the `Column III`. Since the pre-processing step applies only to headlines containing a colon, `Column IV` shows the number of colon containing headlines. It also shows the percentage of colon containing headlines w.r.t. the total number of user query related headlines i.e. $\frac{ColumnIV}{ColumnII}$. The next 4 columns shows the number of headlines processed by each step of the headline-processor. The percentages shown are w.r.t. the number of colon-containing headlines i.e.`Column IV`. In the headline pre-processor `Step 1` and `Step 4` directly gives the relation without using the relations extractor algorithm. `Column A` and `Columm D` gives the number of headlines that are affected by `Step 1` and `Step 4`. `Column B` and `Column C` show the number of headlines that are altered syntactically by the pre-processor. These syntactically altered headlines are expected to generate more meaningful relations.

The high values of colon containing headlines i.e. `Column IV` shows that a significant portion of the total entity-related headlines contain colon. 20% - 30% of the headlines that are related to the input entity contain a colon. `Column A` + `Column D` gives the number of relations that are extracted without using the relation extraction algorithm. This figure is around 50% of the colon-containing headlines. This shows that pre-processing will help in considerable increase in the number of relations extracted without even using the relation extractor. `Column B` + `Columm C` shows that a significant part of the headlines is affected by `Step 2` and `Step 3` of the pre-processor. We feed the syntactically altered headlines to the relation extractor. The number of relations extracted thereafter is added to the number of relations directly extracted in `Step 1` and `Step 4`. These are the relations that are extracted after pre-processing the headlines. To quantify the effect of pre-processing on the relation extraction process we calculate the percentage increment in the number of relations extracted after pre-processing the headlines. Let the number of relations extracted without pre-processing and with

pre-processing step be M and N respectively. Percentage improvement is calculated using the formula (N-M)/M. The percentage improvement in mentioned in the last column of TABLE 4.3. It is clearly evident that there is considerable increase in the number of relations extracted after the pre-processing step. To qualitatively access the effect of the pre-processing we compared the relation extracted from some of the headlines before pre-processing and afrer pre-processing. They are shown in TABLE 4.2. In the first 4 cases, the relation extractor was unable to extract any relation without pre-processing the headline, however after pre-processing meaningful relations were extracted. In the last case, the headline text is *Anna Hazare: I will protest against Arvind Kejriwal if he indulges in wrongdoing*. Without pre-processing the relation extracted was ⟨ `I, will protest against, Arvind Kejriwal` ⟩. The pronoun `I` does not make sense and should be replaced with `Anna Hazare`. After pre-processing the relation extractor was able to extract the correct relation – ⟨ `Arvind Kejriwal, will protest against, Arvind Kejriwal` ⟩.

Headlines	**Relations before pre-processing**	**Relations after pre-processing**
Court to Arvind Kejriwal: Appear on May 24 or face action	No relations extracted	⟨Court , Appear on May 24 or face action, Arvind Kejriwal⟩
Is it right to use such foul language: Arvind Kejriwal to Narendra Modi	No relations extracted	⟨Arvind Kejriwal, Is it right to use such foul language, Narendra Modi⟩
BJP on power crisis: Arvind Kejriwal should quit	No relations extracted	⟨BJP, should quit, Arvind Kejriwal⟩
BJP: Arvind Kejriwal unable to govern, looking for escape route	No relations extracted	⟨BJP, unable to govern, looking for escape route, Arvind Kejriwal⟩
Anna Hazare: I will protest against Arvind Kejriwal if he indulges in wrongdoing	⟨I, will protest against, Arvind Kejriwal⟩	⟨Anna Hazare, will protest against, Arvind Kejriwal⟩

Table 4.2: Examples of relations extracted after pre-processing

Input Entity (Column I)	**Related headlines in DB (Column II)**	**Conv. related headlines (Column III)**	**% hdl with colon (in complete DB) (Column IV)**	**Step 1 (A)**	**Step 2 (B)**	**Step 3 (C)**	**Step 4 (D)**	**% Impv. in Reln. Ext. (Column V)**
Arvind Kejriwal	3238	1424	969	37	135	29	445	31.5 %
			30 %	3.8 %	14 %	3 %	45.9 %	
Barack Obama	1463	560	350	20	17	8	201	14.6 %
			23.9 %	5.7 %	4.9 %	2.3 %	57.4 %	
Congress	18418	7943	5693	232	1727	74	3170	15.9 %
			31 %	4.1 %	30.3 %	1.3 %	55.7 %	
Narendra Modi	14885	6692	4606	235	544	101	2222	22 %
			30.94 %	5.1 %	11.8 %	2.3 %	48.2 %	

Table 4.3: Statistics for various user queries

Parameters	**Value**
Input entity	Arvind Kejriwal
Date Range (DD/MM/YYYY)	28/01/2014 ‚Äì 04/01/2015
No. of records in database	3773677
No. of records related to input entity	3238
No. of records after rule-based headline extractor	1411
No. of relations extracted	281

Table 4.4: Statistics for query ‘Arvind Kejriwal’

4.2.1 Timeline for ‘Arvind Kejriwal’

We used the developed system to generate the timeline for the input entity ‘Arvind Kejriwal’. The details of the experiment are given in Table 4.4.

We present a portion of the timeline for ‘Arvind Kejriwal’ here.

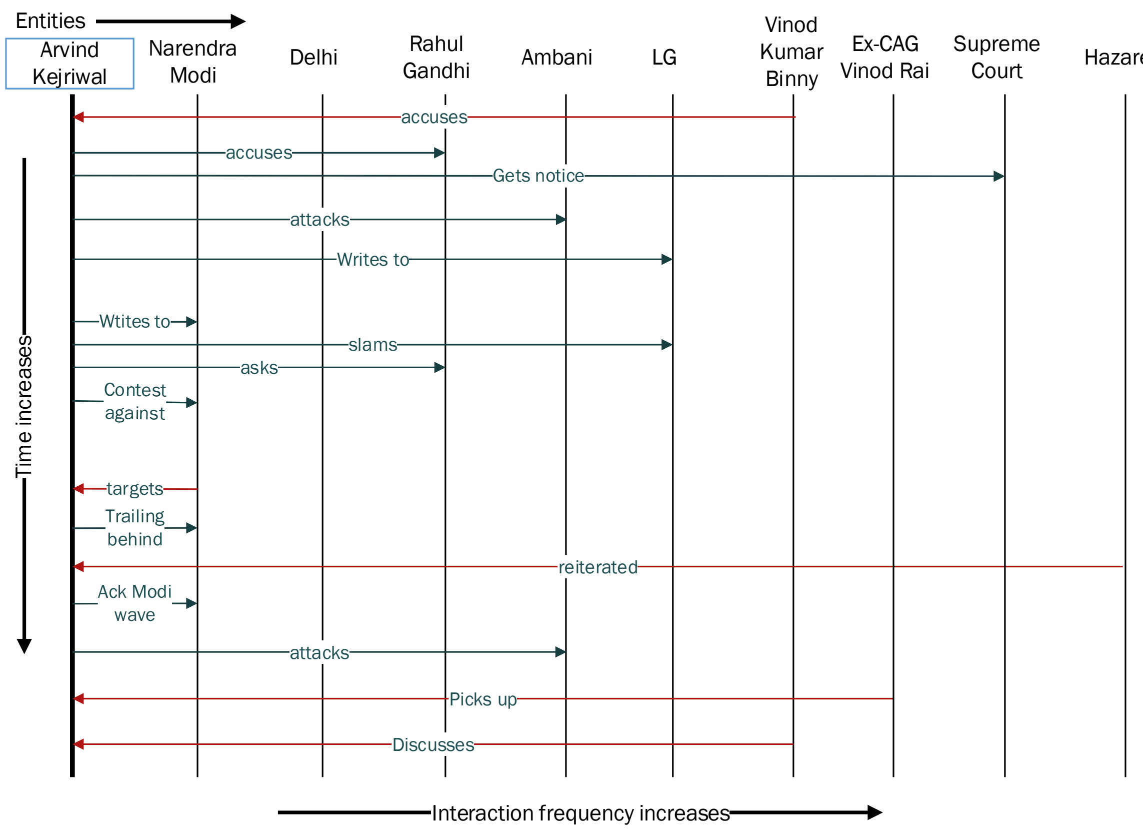


Figure 4.1: Part of the conversation timeline for entity ‘Arvind Kejriwal’

By looking at the timeline we can extract the following information -

1. Arvind Kejriwal has been interacting the most with Narendra Modi, then with the people of Delhi and so on.

2. In the starting of the year, Vinod Kumar Binny accused Mr. Kejriwal. Also, Mr. Kejriwal accused Rahul Gandhi.

3. It is interesting to determine that Arvind got a notice from Honourable Supreme Court.

4. Politicians have been alleging each other in the past, but Mr. Kejriwal took the bold step to attack Ambani brothers and Reliance. This is unusual.

5. During the campaigning of the Lok Sabha Elections 2014, there had been dialogue exchange between Arvind Kejriwal and Narendra Modi. This conversation can be spotted in the timeline. It shows that Mr. Modi attacked Mr. Kejirwal who was then lagging behind Mr. Modi. Mr. Kejriwal also acknowledged the existence of Modi wave.

6. We can also notice that Mr. Kejriwal alleged Ambani brothers *before* the elections as well as *after* the elections.

4.2.2 Conversations about 'Jayalalithaa'

In the previous sub-section, we showed how a Conversation Timeline can be used to better understand the interaction of an entity (*Arvind Kejriwal*) with other entities. In this sub-section, we take up a recent case of acquittal of *Jayalalithaa* to determine what other entities has been saying about this event.

Recently Former Tamil Nadu Chief Minister Jayalalithaa has been acquitted of all the corruption charges against her. To know more about this event we queried the recent news headline database that we have used till now. The dataset that we use now contains news headline data till 14-May-2015. The details of the experiment are given in Table 4.5.

The major events related to this case given in Wikipedia [15] along with the dates when they occurred. We use this temporal data to evaluate the quality of the timeline for *Jayalalithaa*. To analyse the statements made by various entities about *Jayalalithaa* we consider specific events in this case and then look at the conversations extracted in that specific time periods. As per the data given on Wikipedia, the major case decisions and their dates have been listed in Table 4.6. Also, in the dataset we calculated the number

Parameters	**Value**
Input entity	Jayalalithaa
Date Range (DD/MM/YYYY)	28/01/2014 ‚Äì 14/05/2015
No. of records in database	4579530
No. of records related to input entity	3413
No. of records after rule-based headline extractor	1255
No. of relations extracted	315

Table 4.5: Statistics for query ‘Jayalalithaa’

Date(YYYY-MM-DD)	**Event**
2014-09-27	*Jayalalithaa* was convicted and sentenced to prison
2014-10-07	Bail plea rejected by Karnataka High Court
2014-10-17	Supreme Court grants bail to *Jayalalithaa*
2015-05-11	*Jayalalithaa* gets acquitted

Table 4.6: Major Events in Jayalalithaa’s case

of times *Jayalalithaa* has been mentioned each day in a news headline. The graphical representation is shown in Figure 4.2. It clear that during the dates given in Table 4.6, *Jayalalithaa* was frequently talked about in the news media. We expect these events and the reactions that follow should be captured in the timeline generated.

The relation triplets that have been extracted using the system are shown in Table 4.7.

The entities that have commenting on the *Jayalalithaa* case and their statements can be clearly seen in the Table 4.7. After *Jayalalithaa*’s conviction on 2014-09-27, BJP leader *Subramanian Swamy* was commenting on the unrecoverable loss to *Jayalalithaa*’s credibility. Where as blogger *Arun Ram*, lawyer *Jethmalani* and *Venkaiah Naidu* were in support of *Jayalalithaa*. After *Jayalalithaa* was granted bail on 2014-10-17, BJP leader *Arun Jaitley* called *Jayalalithaa* to congratulate her. *Panneerselvam*, leader of *AIADMK* and present Chief Minister of Tamil Nadu announced that *Jayalalithaa* will be CM again. *Jayalalithaa* was acquitted of all charges by the Supreme Court on 2015-05-11. *AIADMK* was celebrating across Tamil Nadu. PM *Narendra Modi* called *Jayalalithaa* to congratulate her. *BJP* had to ask Tamil Nadu govt. to maintain law and order in the state. The famous political commentor *Neerja Chowdhury* even said that *Jayalalithaa* is set to become India’s most powerful Chief Minister.

It is evident from the above Case Study that the relations extracted for the entity *Jayalalithaa* gives us insight into what other noteable entities are saying about *Jay-*

Sr. No.	**Date of news publication**	**From**	**Conversation**	**To**
1	2014-09-28	Subramanian Swamy	Jayalalithaa Can t Recover From This Loss of Credibility	Jayalalithaa
2	2014-09-28	Arun Ram	Jayalalithaa ,Äì a woman of convictions	Jayalalithaa
3	2014-10-08	Karnataka HC	rejects bail plea	Jayalalithaa
4	2014-10-08	Jethmalani	Jayalalithaa must be granted bail	Jayalalithaa
5	2014-10-18	Supreme Court	grants bail to	Jayalalithaa
6	2014-10-18	Venkaiah Naidu	Jayalalithaa will not misuse interim bail granted by SC	Jayalalithaa
7	2015-01-19	Jaitley	calls on	Jayalalithaa
8	2015-03-25	Panneerselvam	Jayalalithaa will be CM again	Jayalalithaa
9	2015-05-09	Karnataka HC	plea filed by	Jayalalithaa
10	2015-05-11	Swamy	Jayalalithaa has come with unclean hands	Jayalalithaa
11	2015-05-11	Jayalalithaa	acquitted by	Karnataka HC
12	2015-05-11	BJP	asks TN govt to maintain law and order	Jayalalithaa
13	2015-05-12	PM Modi	calls	Jayalalithaa
14	2015-05-12	Neerja Chowdhury	set to become India s most powerful chief minister	Jayalalithaa
15	2015-05-12	AIADMK	celebrate acquitted	Jayalalithaa

Table 4.7: Relations extracted for query ‘Jayalalithaa’

alalithaa. Thus, the above two case studies show that Conversation Timeline can be helpful in understanding the interactions about an entity in the news media. The temporal information contained in the timeline gives us some idea about how the interactions have changed during a period of time.

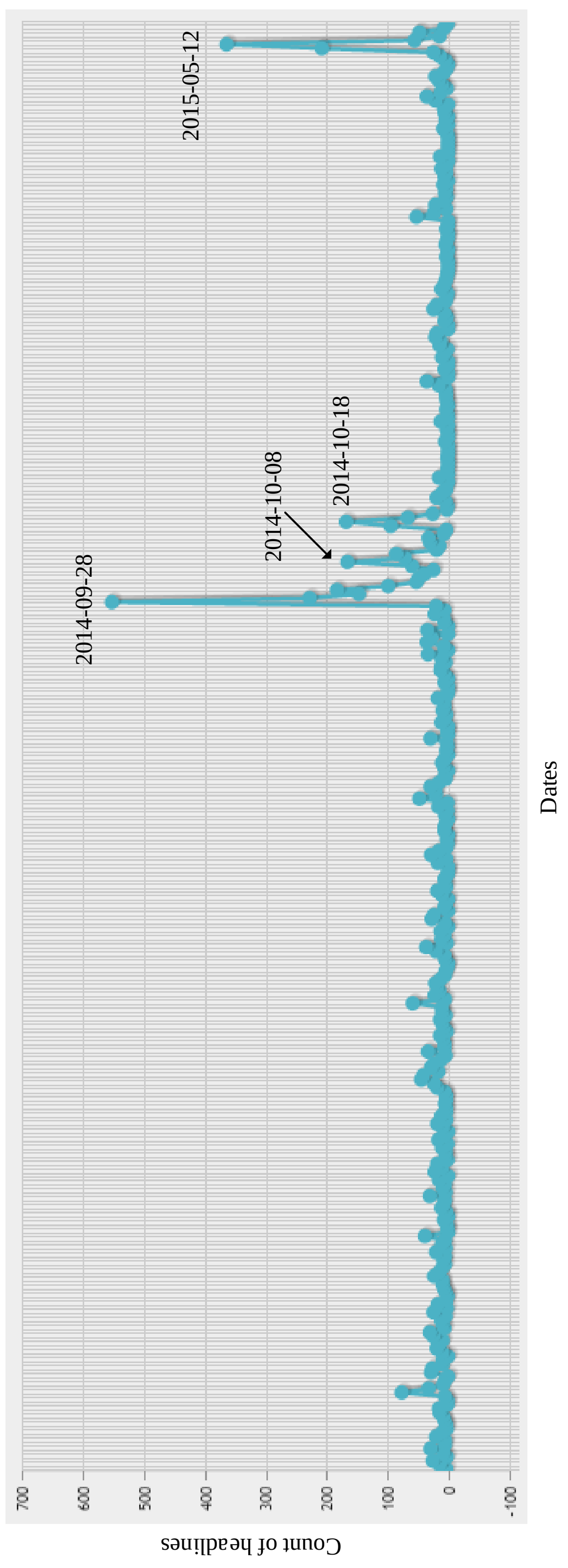


Figure 4.2: The frequency plot of mentions of ‘Jayalalithaa’ per day

CHAPTER 5

Conclusions and Future Work

The problem of *Information Overload* has compelled the researchers to propose *Document Understanding Systems* that can enable the user to have a better insight in the underlying structure of text without going through the entire text corpus. For this purpose, various document understanding systems have been proposed. We discuss three of the most widely used systems in this thesis – (1) Document Summarization, (2) Topic Detection and Tracking, and (3) Storyline Generation. We realised that none of them work on the conversations between entities. To fulfill this research gap, we decided to work on extracting conversations between entities from the news media.

In this dissertation, we present a novel document understanding technique called *Conversation Timeline*, which takes an entity as input and discovers various entities that the given entity had been conversing with along with the conversation between them. We use news headline dataset to form the timeline. News headline contains the key idea of the corresponding article and parsing them saves computational cost. We use relation extraction algorithms to retrieve the conversations between entities. Since headline do not follow the English grammatical rules, which is necessary for relation extraction; we propose pre-processing steps that syntactically alter the headlines for improve relation extraction. We quantitatively and qualitatively compare the relations extracted without using and after using the headline pre-processor.

The work presented in this dissertation can be extended to form a more interesting and informative timeline. Currently, we manually filter out the relations which do not make much sense to the user or convey the same information about a conversation. Further research can be done on designing a module that can automatically filter out the meaningless relations from the extracted relation set. Secondly, the entities are displayed according to the decreasing order of their interaction frequency with the input entity. One can try to figure out a more interesting way to order the entities such that the entities that are close to each other on the timeline are inter-related in a certain fashion such that it is more interesting to the user to look at these grouped entities.

Bibliography

[1] A. Nenkova and K. McKeown, "A survey of text summarization techniques," in *Mining Text Data*, pp. 43–76, 2012.

[2] J. Allan, *Topic detection and tracking: event-based information organization*, vol. 12. 2002.

[3] C. Lin, C. Lin, J. Li, D. Wang, Y. Chen, and T. Li, "Generating event storylines from microblogs," in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, (New York, NY, USA), pp. 175–184, 2012.

[4] "Automatic Text Summarizer. [Online]," 2015. `http://autosummarizer.com/`.

[5] D. M. Blei, "Probabilistic topic models," *Communications of the ACM*, vol. 55, no. 4, pp. 77–84, 2012.

[6] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, "Indexing by latent semantic analysis," *JAsIs*, vol. 41, no. 6, pp. 391–407, 1990.

[7] C. Shen and T. Li, "Multi-document summarization via the minimum dominating set," in *Proceedings of the 23rd International Conference on Computational Linguistics*, pp. 984–992, 2010.

[8] D. Shahaf and C. Guestrin, "Connecting the dots between news articles," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 623–632, 2010.

[9] D. Shahaf, C. Guestrin, and E. Horvitz, "Trains of thought: Generating information maps," in *Proceedings of the 21st international conference on World Wide Web*, pp. 899–908, 2012.

[10] J. Jiang, "Information extraction from text," in *Mining text data*, pp. 11–41, 2012.

[11] E. Agichtein and L. Gravano, "Snowball: Extracting relations from large plain-text collections," in *Proceedings of the fifth ACM conference on Digital libraries*, pp. 85–94, 2000.

[12] F. Wu and D. S. Weld, "Open information extraction using wikipedia," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 118–127, 2010.

[13] S. Mazumder, B. Bishnoi, and D. Patel, "News headlines: What they can tell us?," in *Proceedings of the 6th IBM Collaborative Academia Research Exchange Conference on I-CARE 2014*, pp. 1–4, 2014.

[14] "Verbs of attribution [online]." `http://uwc.utexas.edu/wp-content/handouts/Verbs-of-Attribution.pdf`.
Accessed: 2015-03-30.

[15] "Disproportionate asset case against jayalalithaa [online]." `http://en.wikipedia.org/wiki/Disproportionate_Asset_case_against_Jayalalithaa#Timeline`.
Accessed: 2015-05-14.