Design and Implementation of Timeline Application for News Documents

Mastan Vali Shaik
Wright State University

Follow this and additional works at: https://corescholar.libraries.wright.edu/etd_all

Part of the Computer Sciences Commons

Repository Citation
https://corescholar.libraries.wright.edu/etd_all/212

This Thesis is brought to you for free and open access by the Theses and Dissertations at CORE Scholar. It has been accepted for inclusion in Browse all Theses and Dissertations by an authorized administrator of CORE Scholar. For more information, please contact corescholar@www.libraries.wright.edu, library-corescholar@wright.edu.
DESIGN AND IMPLEMENTATION OF
TIMELINE APPLICATION FOR NEWS DOCUMENTS

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science

By

MASTAN VALI SHAIK
B.Tech., Acharya Nagarjuna University, 2005

2007
Wright State University

Krishnaprasad Thirunarayan, Ph.D.
Thesis Director

Thomas Sudkamp, Ph.D.
Department Chair

Committee on Final Examination

Krishnaprasad Thirunarayan, Ph.D.

Mateen M. Rizki, Ph.D.

Thomas Hartrum, Ph.D.

Joseph F. Thomas, Jr., Ph.D.
Dean, School of Graduate Studies
The growing News document archive emphasizes the need for efficient techniques to retrieve and visualize its content. We present a timeline based graphical interface for this purpose. The timeline is a graph of number of documents supporting association between entity (event, country, person, etc) etc and event (entity, country, person, etc) with respect to dates. The query is formulated based on entity, event, country, and person metadata extracted from the text of the News documents by analyzing the documents using proprietary name-entity recognizers. The timeline also provides a means to index and access relevant documents.

Associations inferred on the basis of document-level metadata are not always correct in the presence of News documents with multiple News stories. The mis-associations can be eliminated by requiring paragraph/sentence level co-occurrence of the corresponding phrases. Our refined timeline points are also annotated with cluster labels generated from headlines and sentences. We have decoupled document archive from the GUI by generating metadata for timelines offline, and provided two separate renderings of the timeline using Java and Adobe Flex.
# TABLE OF CONTENTS

I. INTRODUCTION. ..........................................................1
   1.1 Information Retrieval System Interfaces ...............................1
   1.2 Indexing Documents .....................................................2
   1.3 Nature of News Documents .............................................3
   1.4 Our Approach ..........................................................4

II. RELATED WORK ..........................................................9
   2.1 Google Trends ...........................................................9
   2.2 Google and Yahoo Finance ............................................10
   2.3 Clusty .................................................................11
   2.4 Excite .................................................................12
   2.5 Cluster Label Generation .............................................13

III. DESIGN AND IMPLEMENTATION OF INDEX GENERATION ..........14
   3.1 Why do We Need Indexing? .........................................14
   3.2 How do We Index the Dataset? ......................................14
   3.3 How does the Index Look Like? ......................................16
   3.4 How the Index was Generated? ......................................17
      3.4.1 News Document Dataset ........................................17
      3.4.2 Parsing the Dataset to Extract Metadata terms .............18
IV. METADATA GENERATION USING RECOGNIZERS ..............................................23

4.1 Generating Metadata terms using Recognizers .........................................23

4.2 Extraction of Metadata terms from News Documents ..................................24

4.3 Metadata Generation using Entity, Event, Country, Person Recognizers ....25

4.4 Timeline Metadata ..................................................................................27

4.4.1 Algorithm to Generation Timeline Metadata ..........................................28

V. DESIGN AND IMPLEMENTATION OF TIMELINE GENERATOR ...........31

5.1 Timeline GUI Design ................................................................................31

5.1.1 Query Input Panel ................................................................................31

5.1.2 Timeline Graph Panel ...........................................................................32

5.1.2.1 How to Generate Timeline Using Metadata .....................................33

5.1.2.2 Timeline Varieties ...........................................................................33

5.1.2.2.1 Entity – AllEvents case .................................................................34

5.1.2.2.2 Entity-Event Case .......................................................................34

5.1.2.2.3 Entity-Multiple Events Case .......................................................35

5.1.3 Events List Box Panel ...........................................................................35

5.2 Scaling and Resizing the Components with Application Window Size

VI. CLUSTER LABEL GENERATION .................................................................38

6.1 Need for Cluster Label .............................................................................38

6.2 Cluster Label Generation .........................................................................38

6.2.1 Headline Based Label generation .........................................................40
ACKNOWLEDGEMENTS

I would like to thank my thesis advisor, Dr Krishnaprasad Thirunarayan for his guidance and endless patience.

I thank Dr Mateen M. Rizki and Dr Thomas Hartrum for serving on my committee.

I thank Trivikram Immaneni for his guidance and support throughout this project.
DEDICATION

To my parents Khaja Vali and Subhanbi, in appreciation for all the values they have instilled in me.
1. Introduction

1.1 Information Retrieval System Interfaces

News articles from the news agencies are being generated at enormous rate. Consequently, overwhelmed with these news documents for a given topic search, search engine retrieve a large set of news articles. The user is then required to read all the documents located at these URLs to acquire the needed information. Unfortunately, many query retrieval systems do not provide a good interface and queries are often not focused. The search interfaces need to be changed. Such that interaction with a large data archive will be more efficient if some sort of graphical display is provided that can help with query formulation and answer analysis.

The three phases of information seeking are searching, indexing and accessing. Most of the information retrieval systems concentrate on searching and indexing. Good visualization of huge data archives helps the users to gain more information in less time with less effort.

Our work was confined to News documents Domain. Time plays a major role in organizing documents in News domain as the topics in the News articles evolve over time. Thus we came up with a graphical interface built around timelines in presenting these news documents search results. Including the time attribute helps the user to search news within a given time window and see how the news develops over time.
This thesis will discuss the design and implementation of a graphical interface based on timelines. Timeline interface helps the user to search the archive and access the news documents more quickly than a usual search interface.

1.2 Indexing Documents

A Search mechanism should drive the application to get the desired results from the news document archive. Searching through the whole document repository (say 150GB) for documents matching a given query is not a good idea and in real time it is unacceptable to process the whole document set for each query. Instead the application should be “intelligent” enough to access only the relevant document set quickly for a given query. Search engines use some kind of indexing mechanism to access the desired results for a given query. Likewise, an indexing mechanism should be used to index the document archive to access the documents quickly from the archive.

What index terms help the system access the documents more conveniently? Most of the search engines do their text query and retrieval using keywords. A keyword is a significant word that reflects the content of the document text. The keywords are recognized and added to the search engine’s index when a document is being processed for indexing. Later this index is used to retrieve the relevant documents for a given query. As stated, keywords capture document subject and context.
In News documents domain, the ordinary keywords are not sufficient to generate indexing information or to represent the document. The usage of words varies with context in news documents and the choice of the keywords needs to account for user interest and news context. Thus, additional semantic or conceptual information is required to extract the keywords from news documents. Representing the documents as a set of these indexing terms makes the process easy and powerful in searching and accessing the relevant documents from the News archives.

1.3 Nature of News Documents

News articles have the primary purpose of conveying information about who, what, when and where. A news document may speak on an issue or topic, an entity (e.g: company, person, organization,...) or a set of events (e.g: mergers & acquisitions, company earnings, takeovers, new products,...). From the user’s perspective, the entities and events are interesting. An automatic system should analyze the documents with a user’s perspective, thus entities and events make sense in selecting a document. Thus indexing the news documents on these entity, event, country, person terms makes the search approach more powerful and natural as the users show interest in getting information on these terms. Let’s say people want to know about Microsoft’s Windows release. The key terms in the query could be MICROSOFT, COMPUTER OPERATING SYSTEM, WINDOWS etc. An entity, event, country, person recognizer which can also apply conceptual information was used to determine these indexing terms from the text of the document. Technically
speaking as these index terms represent the document content, the indexing terms are called “metadata terms”.

We assume news documents are time based information. For example, news articles will also discuss topics for a period of time and the topics might vary with time. For example, a deal may be initiated on a certain day, followed by talks on the deal and then the deal may get finalized later. Thus we can benefit from time based progression to access the documents pertaining to a story.

1.4 Our Approach

The following steps discuss our approach to build the graphical interface based on timeline for a news document dataset.

1) The timeline application will take an entity term and time period as query and retrieves the answer set using the two-level file-based index.

2) We implemented a two level file-based index for news documents to access the answer sets using date and metadata terms. The two levels were: date at the first level and metadata terms, at the second level. This will be discussed in Chapter 3. Using this index, the desired answer set for a query can be accumulated to show it on a timeline.

3) The timeline is a graph showing number of documents on vertical axis versus date on horizontal axis.

4) The News document dataset is XML formatted, that is parsed to extract the text. The parsing mechanism will be discussed in Chapter 3.
5) An entity, event recognizer is required to generate the metadata terms and their corresponding scores. The recognizer also assigns a magic score to these terms analyzing the concept information of the document. The scores explain the significance of metadata terms in representing a given document. These entity-event metadata terms abstracts the news documents content. These metadata terms that were generated can be used in lieu of the news documents.

6) The metadata terms can be used to create a data structure for visualization of the dataset through the timeline. A data structure was created to represent the answer set documents with the timeline interface. The data structure should contain the metadata terms and the number of documents related with the metadata terms per day. The data structure also hold on to the headlines and file paths of the documents to provide access to the original documents. The data structure will be written as an XML file for each query to decouple the backend process from the graphical interface (GUI). The data structure and its generation will be discussed in Chapter 4.

7) The metadata can be generated for given dataset sample offline. A customized GUI can render timelines solely from this metadata without processing the dataset repeatedly as the metadata abstracts and normalizes the dataset. Thus we can say that the metadata generated can be used to decouple the actual GUI interface from the time consuming step of processing the News documents for indexing and metadata generation. This approach further improves the performance and efficiency of the application.
8) The timeline application is generalized so that we can visualize timelines with any combination of metadata terms. Like entity vs event timeline, event vs person timeline, person vs country timeline, country vs entity timeline etc.

9) The metadata contains the pair (entity, event), the number of documents that contain both these terms, their headlines and file paths. Assuming the document discusses on the pair (entity, event), the entity and event are said be associated. These associations help in analyzing the quality of information represented through the timeline. Though the thesis was generalized to various timelines the entity-event pair timeline will be discussed in detail throughout the document.

10) A document might contain multiple entities and events. Let’s say there are n entities and k events in a document. The metadata terms are generated with the textual appearance in the document text. So, all the n x k associations may not be valid. For example, consider a news document discussing multiple news stories "Proctor and Gambles announces third quarter profits" and "Microsoft unveils details on next windows release". Here PG can be associated with product release and Microsoft with annual profits. But the actual document is talking about two different stories. The entity, event associations that are not related are called mis-associations. These mis-associations should be removed to increase the reliability of timeline. The mis-associations can be decreased with the refinement of the entity, event associations. The co-occurrence of entity, event terms with in the paragraph or sentence is a better yardstick of semantic relationship than just the positional
proximity in the document. The refinement process will be attempted at two levels: one with the paragraph level and other with sentence level entity-event associations. The refinement process will be discussed in Chapter 7.

11) The timeline graphical interface is used to represent the search results for a query. Giving labels to the news document clusters enables us to provide useful and concise summary of each timeline point. For instance, Google trends present the cluster labels as callout labels. Clusty is a cluster based search engine that categorizes the search results as a hierarchical tree structure. A simple technique to generate cluster labels for the news documents cluster will be discussed in Chapter 6.

12) The Timeline application was also extended to show multiple timelines for a given query. The necessity in going for multiple timelines was that a user can compare among multiple events and analyze their behavior or trends. The news document dataset is a good example dataset to find trends about entities, events, countries etc. The multiple timeline application allows the user to generate multiple timelines by selecting multiple events simultaneously. The basic idea of generating multiple timelines was to improve the visualization experience of the application and giving an idea of how easy and quick to compare, analyze and find the behavior among multiple entities and events. The same metadata generated for single timeline can be used to generate multiple timelines too.

13) One more feature of the timeline application was its pop-up interface for showing headlines, file paths and label. On the horizontal axis the answer set
documents for the given query are categorized with date. The set of documents on each date will be treated as a cluster and a cluster label is generated as discussed in Chapter 7 with headline based or sentence based approach. This feature takes the timeline application to next stage of improvement in visualizing the answer set news documents.

14) There are no metrics to quantify this kind of a work till now. We will also propose an approach to quantify the results. We introduce a quality factor (QF) for each level (document/paragraph/sentence) of entity-event pair associations. What level of refinement process is required to reduce the mis-associations can be estimated by comparing the quality factors at each level. We formalized the quality criteria by adopting traditional notions of precision and recall. This approach will be further explained to better understand the nature of the news document dataset in Chapter 8. This Chapter also describes a fragment of the experimental results obtained for News documents for the year 2005.
2. Related Work

We now discuss some timeline-based web applications that present the search results information in the form of a chart or hierarchical structure categorizing them into groups with callout labels.

2.1 Google Trends

Google trends [8] is a tool from Google labs that represents the most popularly searched terms from the beginning of 2004 to now in the form of a timeline with callout labels referenced through letters. The application charts how often a particular search term is entered relative to the total search volume across various regions of the world and in various languages. The horizontal axis of the chart represents the time and vertical axis represents how often the search term is searched relative to the total search volume. The search volume can be normalized into regions, languages etc. The application also allows the user to compare the search volumes between two or more terms. The additional feature is its ability to show how new events affect search popularity of news related to the search term.
2.2 Google and Yahoo Finance

Yahoo finance [9] and Google finance [10] both are finance websites providing business and enterprise headlines for many corporations including their financial decisions and news events about the company searched. Another feature is presenting stock information in the form of a stock price chart with time on horizontal axis and stock price values on the vertical axis.
2.3 Clusty

Clusty [11] is a meta search engine developed by Vivisimo which offers clusters of results. The aim of this search engine is to organize the search results into several meaningful clusters. This presents the search results in the form of hierarchical tree structure representing the nodes with the cluster labels. The cluster labels on these nodes represent all the children nodes information. This presents the results in a categorized fashion going down the tree of sub-categories. The user’s scope of searching for the
desired results will be normalized to a particular category of results. This helps the end user to overview of all the result topics. Giving the tree structure cluster organization helps to view similar results in the same cluster rather than scattering throughout the list of results.

Figure 3: Clusty Showing search results for windows categorizing into clusters like Microsoft & Doors

2.4 Excite

Excite was the general purpose search engine site that relied on concept-based searching. AOL is now using the Excite search engine, both for internal and external web search (http://search.aol.com/aol/webhome). Unlike keyword based search systems, concept based search systems try to determine what you mean, not just what you say. A concept based search engine returns hits on documents that are about the subject or theme that a user is exploring, even though the keywords precisely do not appear in the search index. Excite also uses a numerical approach to score the quality of the concept information. Excite analyses the meaning by calculating the frequency with which certain important words appear. When several words or phases are tagged to a single concept term then these scores will help in concluding the relevancy of the search information subject. For example, the word Jaguar, when used in a automotive context, would likely appear with the words cars, automotives, auto motors etc. If the word appears in a document with other words like big cat, mammal, predator etc then a different context is
established and the concept based search engine will return the document on the subject of animals.

2.5 Cluster Label Generation

Cluster labeling concept will arise in the context of clusters or chart (graph) based information representation applications to present a group of documents or data using the cluster labels. Large result datasets can be categorized and represented in the graphical charts. Thus, the labels will help in giving minimum information about these categories. These cluster labels also improve the visualization capabilities of the applications. Cluster labels act as an eye catchy to a group of information. But generating cluster labels is not that simple because the labels should be good enough to give meaningful information of the group of documents that the label should represent. The label should be very simple and elegant. To generate a cluster label from the document content [2] presents a simple technique to extract the cluster label form the headlines and sentences of the News documents that are conceptually clustered using entity-event-date associations. A frequent terms sets based approach was discussed in [3] to cluster the documents and to form a cluster label from the text of the supporting documents.

In order to visualize the information over time using tags entered by the Flickr (online image sharing community) users for the photos in their albums [5] calculates the interestingness of the tags over a time interval and characterizes them. Instead of representing a document using keywords [6] proposes a graph based text representation, which attempts to capture the structure and semantics of the documents more effectively.
3. Design and Implementation of Index Generation

The three important features for any information accessing system are indexing, searching and accessing.

3.1 Why do We Need Indexing?

As the information should be accessed and presented to an end user within seconds, an indexing mechanism should exist for any information accessing system to access the documents quickly. As the data repository can be very huge in size (say 150GB), it is unacceptable in real time to process this data repository for each given query to retrieve the desired results. Thus an indexing mechanism is required to narrow the search and access the desired results from the data archive for a given query.

We have implemented an indexing mechanism to access news documents dataset using metadata terms.

3.2 How do We Index the Dataset?

Most of the information retrieval systems use keywords to index the document set. The keywords are not significant words from the text of the documents. For example, from the first sentence of this paragraph “information” “retrieval” “systems” are good keywords for indexing this document. The irrelevancy of information retrieval issues arises when the indexed keywords are weak in representing the content of the document. Most of the search engines ignore the (stop) words “a, an, the, this, when, where, of” etc.
as keywords to index because these words do not signify the content of the documents. The keywords indexing fails in retrieving relevant results for a query if many words or phrases are indexed using a single keyword. The issue arises with the contextual usability of the words or phrases. For example “Windows” should be tagged with WINDOWS if it appears with words furniture, house, wood etc. and should be tagged with COMPUTER OPERATING SYSTEMS if it appears with OS, Windows 95, and other OS related terms.

So, the extra functionality needed for the keywords generation is the concept definition. What is a concept definition? The concept definitions are used to extract the semantic information from the text of the document to generate the keywords from it. These conceptually generated keywords can abstract the actual document context. Thus we can call these keywords as metadata terms. Using these metadata terms helps to determine the context of use and resolve ambiguity. A score can be calculated for these indexing terms depending on the concept information and the frequency of words within given context. Thus the score for a metadata term indicates the quality of representing the document with these metadata terms.

We need an entity, event recognizer which have the capability to generate the indexing terms and their respective scores with concept definitions for the news documents dataset. For these specific news document dataset we used Global Indexing API (GIAPI) to generate the metadata terms and their corresponding scores.
The News domain is time based information. The news evolves with time so a news domain information system should account for the time in retrieving the information from a news document repository. Narrowing the news document repository search using time window will also further reduce the overhead of accessing the desired document set for a given query. This improves resource utilization and performance issues with the indexing step because this is the major step for an information retrieval system.

We have two indexing mechanisms one: with metadata terms and other with date, to take advantage of both the indexing approaches. With this hybrid index, the document filtering process reduces with metadata terms and increases the performance in accessing the documents from the news archive. The key design issue with this hybrid index would be what index should be at the first level and at the second level. Comparing the total number of dates to the average number of metadata terms per document, conclude that using the date at the first level and metadata terms at the second level makes it simple and powerful in accessing the relevant document set from the news archive for a given query. So the final index was designed as two level index having date at the first level and metadata terms at the second level.

3.3 How does the Index Look Like?

The indexing and retrieving of news documents from the news repository is not the actual theme of the thesis. To make the indexing process much simpler a file based index was built with dates at the first level as directory names and metadata terms at the
second level as text file names to store the actual paths of the news documents that fall in the index criteria. As the index was a file based index, the accumulation of news document answer set with this index for a given query might be more time consuming when compared to standard index mechanisms.

3.4 How the Index was Generated?

3.4.1 News Document Dataset

The News document repository that we have access to, was an XML formatted CurrNews dataset for the year 2005. These News documents were already tagged with metadata terms. The metadata terms are the entities, events, countries, person etc. generated by corresponding entity, event, country, person recognizers from the text of the documents. The recognizers also assign corresponding scores to these terms after analyzing the context of the document. These scores capture the strength of metadata terms in representing a given document conceptually. These entity - event metadata terms abstracts the news documents content. The existing metadata terms are associated with the entire document.

The News document consists of a number of sections delimited by XML element tags for headline, lead, body, etc. The XML elements relevant for generating timelines are:

- Headline of the document.
- Lead (first) paragraph of the document.
- Body of the document.
• Paragraph with 1-3 sentences.
• Subject metadata terms with weights.
• Entity metadata terms with weights.
• Country metadata terms with weights.
• Person metadata terms with weights.
• Date of the document.

Each document usually contains subject, entity, country, person metadata terms. Normally, subject metadata terms were consulted for events, entity metadata terms for entities, country metadata terms for countries and person metadata terms for persons.

(Note that a document can have several other tags such as publication, language, tickers etc.)

3.4.2 Parsing the Dataset to Extract Metadata terms

The News document dataset is XML formatted. To fetch the text of the documents, the document needs to be parsed. We implemented three (3) approaches to extract the metadata terms from the XML files.

1) Parsing using SAX Parser

The SAX parser is an event driven API. Call back methods are called when an element is encountered occur during parsing. This approach use less memory and parsing time when compared to DOM as DOM builds the whole XML tree in memory while SAX parser does not. For any parser to read the content from the XML files, the files
should be well formed and valid. DTD defines how the XML should be formatted to make the XML file well formed.

We abandoned this approach in general, due to certain corrupt input documents on which the SAX parser will raise an exception without parsing the file.

2) Parsing using String operations

To overcome the above problem, we read the XML files using String operations. String API's are used to read the text from the required XML elements of the document. This process was able to read the XML dataset. As mentioned earlier, the documents are already tagged with document level metadata terms. Substring API was used to read the metadata terms from the corresponding elements of the XML files. This approach fails with the fabrication issues of existing metadata terms in the news documents. This required extracting the metadata terms from the text of the XML files on the fly using the corresponding entity, event, country, person recognizers as explained below.

3) Parsing using String operations plus Entity - Event Recognizer

Some of the documents do not have metadata terms with correct syntax. This results in missing important content that represents the actual document in an abstract form. So getting the metadata terms on the fly from the text of the document using the entity-event recognizer was implemented. As a result, there will be time overhead in processing the text of the document and generating the metadata terms.
This is a two step process: one to extract the text of the document using String operations and the second to feed the extracted text to corresponding entity, event, country, person recognizers to generate metadata terms and their scores.

3.4.3 Generating Index

Once the dataset is parsed and the date and metadata terms per document are populated, the two level file based index can be generated using the date and metadata terms. Recall that, metadata term may be an entity, event, company, person etc. The timeline application takes a metadata term and time period as query and retrieves the answer set that satisfies the given query. The documents can be filtered from the answer set with some threshold value for the metadata terms scores. As discussed earlier, a high scoring metadata term is very relevant to the document that it represents. A separate index should be generated for each metadata term type. The following example was presented with entity as the metadata term and filtered the metadata terms at a threshold 90%.

Example:

Portions of Input XML Data File Format for indexing the documents

1) The date of document is in LONG format as follows:

   <date>April 19, 2005</date>

2) The Entities appear as

   <entities>

       MICROSOFT CORP (91%); ELECTRONIC ARTS INC (91%);
APPLE COMPUTER INC (90%);
</entities>

**Output:** The index is generated in the output directory as shown below:

April 19, 2005

|-- MICROSOFT CORP.txt
|-- ELECTRONIC ARTS INC.txt
|-- APPLE COMPUTER INC.txt

where April 19, 2005 is a directory which contains the txt files named after entities in a document and containing the full path names of the files in which the entities appear with weight > (say 90%).

Figure 4. Date-based First-Level Index
Figure 5. Entity-based Second-Level Index
4. Metadata Generation using Recognizers

Metadata terms are the terms generated from the text of the news documents using corresponding entity, event, country, person recognizers. The recognizers also assign scores to these terms taking into account the context of the news documents. We used Global Indexing API (GIAPI) for the news documents in our experiments. The GIAPI takes concept definitions to analyze the content of the news documents. The metadata terms can be used to represent the news documents instead of traditional keywords. The scores reflect the significance of the metadata terms in representing the documents. These metadata terms abstract and normalize the actual news documents.

4.1 Generating Metadata terms using Recognizers

The text from the XML formatted news documents should be extracted and fed to the entity, event, country, person recognizers to generate the metadata terms. This chapter discusses only the document level extraction of metadata terms and their scores the paragraph level, sentence level metadata terms generation will be discussed in Chapter 6.

The news documents contain different sections like headline, lead and body. The headline summarizes the document, the lead is the first paragraph of the document and body contains the textual description. The treatment of the metadata terms for a document varies with the section it appears in. That is, the recognizer will accord different
significance to the words in the text in each situation. Similarly, the frequency of occurrence of a word makes a difference in the weight computation.

### 4.2 Extraction of Metadata terms from News Documents

As mentioned earlier, the news document dataset that we are dealing with is already tagged with metadata terms and their scores at document level context. Due to the errors in metadata terms, we dynamically generated the metadata terms using the GIAPI from the document text.

The text of each document section will be created as a DocumentSection instance to plug into the entity, event recognizer.

**Usage:** “DocumentSection(sectionType, elementType, elementText)”

where sectionType is either headline, lead or body, elementType is a tag such as headline, lead, body, p1, p2, etc, and elementText is the text fragment for the document section (usually delimited by tags such as headline, lead, body, p, etc). The elementType markers for the document sections is immaterial for document level analysis.

For example, consider a document containing the sections headline, lead, body

The following DocumentSection instances can be created.

```java
DocumentSection ds1 = new DocumentSection(HEADLINE, HEADLINE, 
    textOfDoc.substring(headlineStart, headlineEnd));

DocumentSection ds2 = new DocumentSection("BODY", "BODY", 
```
Note: we treat the lead and the body sections as body for this news dataset.

4.3 Metadata Generation using Entity, Event, Country, Person Recognizers

We now present and explain a sample piece of code that generates the metadata terms from the above instances at document level context.

```java
public String buildEntries(List documentSections, ConceptDefinitionInput CDI, String _indexType, IndexingSearcher inds, String id, DocumentContext docContext, String textOfDoc) throws Exception {
    IndexingScorer insc = IndexingScorerFactory.newInstance(_indexType);
    Vector termHits = new Vector();
    Iterator docSectionIter = documentSections.iterator();
    while (docSectionIter.hasNext()) {
        DocumentSection ds = (DocumentSection) docSectionIter.next();
        termHits.addAll(inds.search(ds, CDI, id, "XML"));
    }
    Vector entriesMatched = insc.score(termHits, docContext);
    Vector greaterThan90 = new Vector(10);
    for(int i=0;i<entriesMatched.size();i++) {
        Entry entry = (Entry)entriesMatched.get(i);
        if(entry.getScore() >= THRESHOLD) greaterThan90.add(entry);
    }
    entriesMatched = greaterThan90;
```
• In line 1, IndexingScorer is instantiated with the indexType (i.e 'Entity'/'Topical'). The indexType is set to ‘Entity’ for generating company names and people names, and to ‘Topical’ for generating events and countries.

• In line 7, the entries or terms collected by IndexingSearcher from each documentSection are added to the termHits vector. The termsHits vector consists of a list of terms that match the concept definition input. The terms are characterized by the elementType where the term appears (headline, lead, body…), and the word token number in the corresponding document section (‘0’ assigned to the first word), etc.

• In line 9, the IndexingScorer consolidates the scores for the terms in the termHits vector using the document context, and returns the entriesMatched vector. The indexing scorer will assign the score > 85 for the terms in the headline, 50-85 for the terms in the lead, 40-50 for the terms in the body. If a term appears more than once in any of the section(s), its score will be > 85. The entriesMatched vector contains all the matched terms belonging to the CVT list (which contains normalized metadata terms). (The getCvt() method extracts the human readable CVT from a structure returned by getCVTList().)

• In line 13, we eliminate the entries with score less than the THRESHOLD value (< 90).

The entities and events, whose weights cross the given programmable threshold value (say 90%) are called strong entities and strong events because these entities, events contribute more support to represent the actual news document.
4.4 Timeline Metadata

Till now we generated the metadata terms from the news documents to represent them in an abstract form. What additional information is required to visualize the timelines using the metadata terms?

The theme of the thesis is to visualize the news documents through timelines and help the user to get the helpful information on an entity, event, country or person information. We use a combination of metadata terms to represent a useful association such as entity-event, person-event etc. The timeline was mentioned to have time on the horizontal axis and number of documents on the vertical axis. Thus we use dates in given time window on the horizontal axis and for the vertical axis, need to compute the number of documents that get support from the given entity and event on a date. The headlines and file paths of the documents are also required to present them in a popup window on the timeline. Thus the following metadata data structure was designed to hold onto the information to visualize the news documents through timeline:

1) Date
2) StrongEntity : Entity with weight > threshold (90%)
3) StrongEvent : Event with weight > threshold (90%)
4) EntityDocCount: number of documents associated with an entity on the given date.
5) EventDocCount: number of documents associated with an event on the given date to an entity.
6) Filepaths_OnADay : contains filePaths separated by "@" on the given date.
7) HeadLines_OnADay : contains headLines separated by "@" on the given date

4.4.1 Algorithm to Generation Metadata Data Structure

1) Consider a set of documents \{D_1, D_2, ..., D_m\} and let \{En_1, En_2, ..., En_n\} and \{Ev_1, Ev_2, ..., Ev_k\} refer to unique set of strong entities and events.

2) for all \(i \in 1,2 \ldots n\) of \(En_i\):
   
   for all \(j \in 1,2 \ldots m\) of \(D_j\):
   
   entityDocCount = 0;
   
   for all \(k \in 1,2 \ldots m\) of \(D_k\):
   
   \[
   \text{if}(D_j.date == D_k.date \&\& D_j,D_k \text{ contains } En_i) \]
   
   entityDocCount++;
   
   for all \(e \in 1,2 \ldots k\) of \(Ev_e\):
   
   eventDocCount = 0;
   
   filePaths, Headlines = "";
   
   for all \(k \in 1,2 \ldots m\) of \(D_k\):
   
   \[
   \text{if}(D_j.date == D_k.date \&\& D_j,D_k \text{ contains } En_i,Ev_e) \]
   
   entityDocCount++;
   
   filePaths += D_j.filePath+"@";
   
   headlines += D_j.headline+"@";
   
   metadataRecord = new metadataRecord(D_j.date,En_i, entityDocCount,Ev_e, eventDocCount, headlines, filePaths);
   
   if(!metadata contain metadataRecord)
   
   metadata.add(metadataRecord);
Date can be used to sort the METADATA records in a chronological order and to filter the METADATA to generate timeline for a given time window. The generated timeline metadata will be dumped into an XML format. The XML formatted timeline metadata alone can be used to generate the timelines without the need for news documents. The purpose of creating this XML file is to decouple the GUI module from the timeline related essentials. The same metadata file can be used to generate single and multiple timelines. As the metadata contains the headlines and file paths the popup window to show headlines and file paths can also be populated from the metadata file.

A sample metadata XML file is as shown below:

```xml
<graphdata>
  <guidata>
    <entity>INTEL CORP</entity>
    <entity-doccount>4</entity-doccount>
    <event>COMPUTER OPERATING SYSTEMS</event>
    <event-doccount>3</event-doccount>
    <filepaths>C:\example\Microsoft_ComputerOperatingSystems\ID23020584.BL23020585.2073.xml@...@FEG@</filepaths>
    <headlines>Microsoft ships Windows for 64-bit computers@...@...@</headlines>
    <date>April 26, 2005</date>
  </guidata>
  <guidata>
    <entity>INTEL CORP</entity>
    <entity-doccount>4</entity-doccount>
    <event>MICROPROCESSORS</event>
    <event-doccount>2</event-doccount>
    <filepaths>C:\example\Microsoft_ComputerOperatingSystems\ID23020584.BL23020585.2073.xml@FEG@
    <headlines>Microsoft ships Windows for 64-bit computers@...@</headlines>
    <date>April 26, 2005</date>
  </guidata>
...
<graphdata>
```
For each entity-event pair on a given date, <entity-doccount>-element contains the total number of documents containing the entity on that date, <event-doccount>-element contains the total number of documents (containing the entity-event pair), <filepaths>-element contains @-sign delimited list of document file names (containing the entity-event pair), and <headlines>-element contains @-sign delimited list of document headlines (containing the entity-event pair). Note that file FEG has both event tags.

In the above example, the second guidata-record has entity-doccount = 4 implying that there are 3 documents with entity = INTEL CORP with at least one event with weight > 90%, and event-doccount = 2 because there are 2 documents with the entity = MICROSOFT CORP and event = MICROPROCESSORS, on the date April 26, 2005.
5. Design and Implementation of Timeline Generator

The theme of thesis is to present the retrieved results in the form of a graphical interface to help the user to analyze and understand the results quickly compared to the present search engine interfaces in news domain. For any input query the retrieval system will analyze the query and retrieve a set of results. Likewise the timeline application interface also contains query input panel and a timeline panel to present the retrieved results in the form of timelines. This chapter will discuss designing the timeline application interface.

5.1 Timeline GUI Design

The timeline application interface contains three (3) components.

1) Query input panel
2) Timeline graph panel
3) Events list box panel

5.1.1 Query Input Panel

The Query input panel was designed as per the need of the news document domain. The panel has a query input where a user can enter an entity, event, country, person etc. These terms are used for retrieving the relevant documents containing the entity, event, country, person crossing a programmable threshold value using the indexing mechanism discussed earlier. As we are dealing with news domain, the input query should also have the time window that a user wants to narrow the search results.
Thus the query input panel has an input component for time window. (As mentioned earlier, the scope of the thesis was generalized to any combination of the entity, event, country and person terms). Both the inputs: the entity term and the time window, will make a complete input query for timeline application. The panel also contains a button "Generate Timeline", to fire an event to retrieve the results for the given input query and generate initial timeline. The final input query input panel was designed to also contain a button to generate the initial Entity vs All-Events timeline.

### 5.1.2 Timeline Graph Panel

The Timeline graph panel has a graph with dates on the horizontal axis and the number of documents accumulated on a day from the query results on the vertical axis. The timeline graph is drawn using the metadata data structure generated from the result news documents.

The key features in generating timelines using timeline graph panel are:

- Drawing a timeline with the given entity and a selected event.
- Generating multiple timelines for the given entity and selected list of events.
- Showing a popup window with headlines and file paths for a click on the timeline for both single and multiple timelines.
- Retrieving the user selected document from the popup window.
- Resizing the timeline application components to show all the components and the timeline with respect to the change in the application window size.
The metadata data structure was used to visualize the timelines for a given entity and selected events. The design of timeline metadata was discussed in Chapter 4. The metadata data structure was designed in such a way that the above mentioned key issues can be addressed as simply as possible.

5.1.2.1 How to Generate Timeline Using Metadata

Rendering a timeline on the timeline graph panel is similar to drawing a line on a graph having vertical and horizontal axis. To draw a line on the graph a set of points needs to be plotted and subsequently the plotted points needs to be joined by a line. The metadata contains time related information: date, entity-doccount & event-doccount numbers. This information can be used to make points: Divide the horizontal axis on dates for the given time window and the vertical axis on number as the number of supporting documents should be shown. Thus the points(date, entity-doccount or event-doccount) are used to plot on the graph and a line will be drawn through these points to generate a timeline. The next section will discuss on the usage of entity-doccount or event-doccount to show various timeline types that the timeline application can generate.

5.1.2.2 Timeline Varieties

The timeline application was designed to visualize three different types of timelines:

1) Entity – AllEvents case
2) Entity – Event case
3) Entity – Multiple Events case
As mentioned earlier, the timeline application was not confined to entity, event pair. The application was generalized to any combination of entity, event, country, person terms. The AllEvents case is different from selecting individual events. The following discussion explains the difference among the three approaches. For the three varieties of timelines, the same metadata data structure can be used to generate timelines.

5.1.2.2.1 Entity – AllEvents case:

The summation of entity-doccount on a date will give the total number of documents that contain the given entity having at least one event, both having a threshold value (say > 90%). This value is used to show the entity vs AllEvents timeline. This timeline shows the information from all the events that happened in a given time window for a given entity.

A click on the timeline in the vicinity of a date a popup will show a list of headlines and file paths from all the documents accumulated on that date, that account from all the events.

5.1.2.2.2 Entity-Event Case:

The summation of event-doccount on a date will give the total number of documents that contain the given entity and selected event having weights greater than the given threshold value. This value is used to show the entity vs event timeline. This timeline shows the information of a selected event in the given time window.
A click on the timeline in the vicinity of a date a popup will show a list of headlines and file paths from the documents accumulated on that date, that account from the selected event.

**5.1.2.2.3 Entity-Multiple Events Case:**

The Entity Vs Multiple events timeline was an extended behavior from the single timeline. The timelines will be drawn for the given entity Vs a selected list of events where as in single timeline only one timeline will be drawn for the given entity Vs selected event. The only difference between single and multiple timelines is the timelines are rendered on the same graph panel iteratively for a list of selected events. The same event-doccount values are used to show the information from all the selected events. Multiple timelines can help the user to compare and analyze the behavior of an entity with respect to the selected list of events.

A click on the timeline in the vicinity of a date a popup will show a list of headlines and file paths from the documents accumulated on that date, that account from the selected list of events.

**5.1.3 Events List Box Panel**

Event list box panel contains a list box that will be populated with all the unique events crossing a threshold value from the result set for a given query. A user can see only 5 events in the list and needs to scroll to go through all the events in the list. The events are sorted in alphabetical order. Selecting any of the event fires an event to
generate the timeline in the timeline graph panel. Holding a CTRL key, a user can select multiple events resulting generating multiple timelines for the selected events with the given entity. All-Events event has a peculiar functionality, selecting all-events in the events list box generates a timeline showing the number of documents contributed by all the events for the given query.

5.2 Scaling and Resizing the Components with Application Window Size Change

Recall that, the timeline application takes an input query and visualizes the retrieved news documents through timelines and gives access to them. The application supports up to one full year of time window. The input time window changes with query. The timeline graphs are driven from the back metadata. The timelines are generated by selecting the events from the events list box. The input to the graph panel changes with events selected.

As the horizontal axis was divided by the number of dates in the given time window it is difficult to show all the date labels on the horizontal axis. Similarly the vertical axis labels do suffer when a large set of labels needs to be rendered. Thus the nature of labels changes with the size of the backend data dynamically. Both the axis’s lengths also changes with user interaction to fit the timeline in the graph panel.

In addition to that, we designed the application to dynamically calculate its size and make all the components visible by adjusting the size and position of the components.
with the resize of the application window. By this a user can interact with all the components of the application at any size of the application window.
6. Cluster Label Generation

6.1 Need for Cluster Label

Cluster label is very useful in the context of summarizing a collection of data. Today, most of the search engines dump all the search results for a given query in a big list. This lacks organization of the search results by which the end user may comprehend the contents of the search results. We will extend the timeline application to present the cluster labels at (the peaks of) the data points on the timeline.

As the timeline presents the search results in the form of a chart with date on horizontal axis and number of relevant documents on vertical axis the user may not comprehend the contents of the search results shown in the timeline unless the user goes through all the documents. To improve presentation and for easy comprehension of results in the timeline, cluster label functionality was added to the timeline application. This chapter will discuss generation of label for a cluster of documents. The generated cluster label will be shown on the top of the headlines and file paths in the popup window.

6.2 Cluster Label Generation

The News documents of the CURRNEWS dataset have multiple sections like headline, lead and body. Body is the actual content of the news discussed, the lead is the
first paragraph and the headline summarizes the whole document content with in a sentence. All these sections are human composed.

The entity event recognizer assess the significance of metadata terms with these section types and scoring mechanism also varies from section to section. Same phrases is assigned different weights if it appears in different sections. Comparatively headline will be given more weight among the three, next will be lead and then the body. It means entity event recognizer generates more metadata terms at headline and assigns higher scores with respect to other sections.

Two strategies of generating cluster labels will be discussed.

1) Headline based label generation and

2) Sentence based label generation

Headline based label generation algorithm elects a promising and concise headline as a cluster label among the headlines associated with documents on a given date for an entity & event query.

Sentence based label generation algorithm elects a well supported sentence having both the entity and event along with required additional words from the document set of the cluster, and extracts a fragment from it as cluster label.

Refer [2] for clear understanding of the approaches.
6.2.1 Headline Based Label generation

The headline based label generation solely elects a headline as label from the headlines of the news documents. The headlines are human composed, small and meaningful, and unravel the news topic discussed in news documents. The advantage of generating label using this approach is that it does not need body of news document. This approach would be useful, where there was no access to the actual news documents and this process is very fast when compared to sentence based.

A set of headlines will be taken as input and processed as follows: Break each headline into a set of word tokens. Remove all the stop words. The design issue is why do we need to remove stop words? The stop words are prepositions and some helping words for the subject keywords to build a complete meaningful sentence. Conceptually these stop words does not account to the details of the document. Thus these stop words can be omitted. That's why many search engines remove the stop words and index on keywords. From recent past search engines like Google are also making use of stop words for indexing. For now we will remove the stop words. This step results in associating a set of keywords to each headline. Then stem the keywords to their roots. Thus the stop word elimination and stemming of keywords steps, plus treating the headline as a set removes overly discriminating word sequencing information from a headline and generates words that are more amenable to syntactic manipulation for gleaning semantics. The keywords are stemmed using scalable Porter Stemmer algorithm. These raw set of stems are used to calculate the support for a headline from rest of the headlines.
To compute the support, a document Dj accords to a headline hi we use the following scoring strategy:

$$\text{score}(hi, Dj) = \frac{|M(hi) \cap M(hj)|}{|M(hi)|}$$

And for the cumulative support for the headline hi due to entire cluster

$$\text{score}(hi) = \sum_{j=1}^{m} \text{score}(hi, Dj)$$

where hi is the headline of document Di

$M(hi)$ denotes the set of stems of headline hi

$Dj$ is a document from the cluster

The headline with the maximum score will be elected as a cluster label for the document cluster. If two or more headlines get the same cumulative score, Any one of the headlines can be used for cluster label.

### 6.2.1.1 Headline Based Label Generation Algorithm:

1) Consider a set of documents $\{D_1, D_2, \ldots, D_m\}$ and let $\{h_1, h_2, \ldots, h_m\}$ refer to the headlines of the documents.

2) for all $i \in 1,2 \ldots m$ of $h_i$: Eliminate stop words

3) for all $i \in 1,2 \ldots m$ of $h_i$: Stem the keywords

4) for each headline $h_i$:
for each headline $h_j,(i!=j)$ and let $M(h_j)$ be the set of stems

$$\text{score}(hi, Dj) = \frac{|M(hi) \cap M(hj)|}{|M(hi)|}$$

$$\text{score}(hi) = \sum_{j=1}^{m} \text{score}(hi, Dj)$$

5) The headline which got the maximum cumulative score will be the well supported headline for cluster label
Examples and Observations

We now discuss a concrete example to bring out the nature of the News documents and the behavior of our prototype. For instance, on April 12, 2005, for the entity Microsoft, and for the event Computer Operating Systems, the generated headline cluster label is: *In next Windows release, Microsoft to use hardware for security*, based on the headlines:

- Microsoft unveils more details of next Windows release
- In next Windows release, Microsoft plans to use hardware to lock down security
- In next Windows release, Microsoft to use hardware for security
- Microsoft ships Windows for 64-bit computers
- Microsoft Gives Details on Windows Release
- New Windows Operates on 64-Bit Computers
- Microsoft ships Windows for 64-bit computers
- In next Windows release, Microsoft to use hardware for security
- Microsoft unveils more details of next Windows release
- Microsoft ships Windows for 64-bit computers
- Microsoft unveils more details of next Windows release
- In next Windows release, Microsoft will use hardware for security
- Microsoft plans to use hardware to lock down security in Windows
- Microsoft ships Windows for 64-bit computers
- In next Windows release, Microsoft to use hardware for security
- Gates shows off features of next-generation Windows system
Our criteria effectively chooses the most frequent short headline such as *In next Windows release, Microsoft to use hardware for security* (or *In next Windows release, Microsoft will use hardware for security*) as the cluster label, while ignoring other headlines such as *Microsoft ships Windows for 64-bit computers*. (The majority criteria was chosen to eliminate "noise"). Several documents share a headline due to correlated News sources (such as Associated Press, Reuters, AFX News, etc.). Each such document can be viewed as providing an independent endorsement. Unfortunately, this approach can miss multiple headlines for different News stories that happen to have the same event and entity metadata tags, and occur on the same day. In fact, the best comprehensive headline for the above example is: *Microsoft unveils more details of next Windows release*. So, our approach can be further improved by clustering headlines on the basis of similarity, or ranking headlines on the basis of support and cutoff thresholds.

### 6.2.2 Sentence Based Label generation

Sentence based label generation generates a cluster label for a set of documents from the sentences of the documents. The cluster label is a small phrase of the sentence containing references to both the entity and event with additional information. The additional information abstracts the information supported by a maximum number of documents in the given cluster. Using the entity-event recognizer the sentences containing the references are fetched from the document cluster. In the earlier approach each document will contribute only a single headline where as in this approach each document will contribute multiple sentences. A well supported sentence can be obtained
by maximizing the number of documents that support the sentence, by maximizing the
degree of overlap with each sentence in the document and by minimizing its length,
subject to the constraint that the sentence contains phrasal references to entity and event.
This approach is shallow and scalable to extract a good label, as apposed to
understanding the content and synthesizing a label from its meaning.

Consider a document cluster that needs a cluster label generated using sentence
based cluster label generation. Populate all the sentences from the documents having
phrasal references to entity and event. We call them significant sentences of the
documents in the cluster. Let s be a sentence and collect all the words into a set.
Eliminate the stop words and then stem each word. This removes overly discriminating
word sequencing information from a sentence such as due to active/passive voice
changes. These raw set of stems are used to calculate the support for each sentence from
other documents.

To compute the support a document Dj accords to a sentence s ∈ sen(Di),
we use the following scoring strategy:

\[
\text{score}(s, Dj) = \text{MAX}_{t \in \text{sen}(Dj)} \frac{|M(s) \cap M(t)|}{|M(s)|}
\]

and for cumulative support score for sentence s due to the entire cluster

\[
\text{score}(s) = \sum_{j=1}^{m} \text{score}(s, Dj)
\]

where s is the sentence of document Di
M(s) denotes the set of stems of sentence s
M(t) denotes the set of stems of sentence t in document Dj
Dj is a document from the cluster
6.2.2.1 Sentence Based Label Generation Algorithm:

1) Consider a set of documents \{D_1, D_2, \ldots, D_m\} and 
   Let \text{sen}(D_i) refer to the set of sentences in document \text{Di} that each 
   contain phrasal references to the entity EN and the event EV.

2) for all \(i \in 1, 2, \ldots, m\) of \(D_i\) 
   for all \(i \in 1, 2, \ldots, k\) of \(s_k\) 
   Eliminate stop words

3) for all \(i \in 1, 2, \ldots, m\) of \(D_i\) 
   for all \(i \in 1, 2, \ldots, k\) of \(s_k\) 
   stem the keywords

4) for all \(i \in 1, 2, \ldots, m\) of \(D_i\) and Let \(s \in \text{sen}(D_i)\) 
   for all \(j \in 1, 2, \ldots, m\) of \(D_j\), \(i \neq j\) and Let \(t \in \text{sen}(D_j)\) 
   \(\text{score}(s, D_j) = \max_{t \in \text{sen}(D_j)} |M(s) \cap M(t)| / |M(s)|\)

   \(\text{score}(s) = \sum_{j=1}^{m} \text{score}(s, D_j)\)

5) A well-supported sentence \(s\) for cluster label for the cluster of 
   documents is the one that has the maximum cumulative support 
   score.

6.2.2.2 Phrase Selection for Cluster Label

A candidate cluster label is the shortest sequence of words of a sentence in a 
document that contains a phrasal reference to the entity EN, the event EV, and 
significant words that appear in all well-supported sentences. To generate a candidate 
cluster label clip the well-supported sentence at entity and event phrasal references of the 
sentence and add the significant additional keywords that appear in most of the cluster
document sentences. If multiple well-supported sentences have same cumulative support score from the document cluster any one of them can be used to create a candidate cluster label.

For example, consider the following set of document sentences

- Intellisync to be acquired by Nokia.
- Nokia’s (NOK) acquisition of Intellisync (SYNC) will not change the overall picture for company, says Greger Johansson at Redeye in Stockholm.
- The acquisition of Intellisync supports Nokia’s goal to be the leader in enterprise mobility and enhances the ability of its customers to connect devices to data sources, applications and networks.
- Nokia and Intellisync have signed a definitive agreement for Nokia to acquire Intellisync.

Besides a reference to the explicitly searched entity (that is, “Nokia”) and the event (that is, “acquires” etc.), the cluster label should contain other relevant information (such as “Intellisync”) about the queried subjects (analogous to answer extraction), to provide a concise highlight of the document collection. For the above example, electing “Nokia acquires Intellisync” seems reasonable for the following reasons:

- It is sound, containing “Nokia”, and a reference to “Mergers and Acquisitions” via “acquires”.
- It is complete, containing additional relevant information “Intellisync”.

46
It is well-supported, with majority of the document fragments providing supporting evidence for it.

Examples and Observations

The sentence based approach to cluster label generation elects a significant sentence from the cluster documents and clips it. For example, on April 4, 2005, for the entity BHP Billiton, and for the event Takeovers, the relevant document sentences from three separate News documents are:

- Anglo-Australian resources giant BHP Billiton has been given the green light by Treasurer Peter Costello for its $9.2 billion takeover of Australian miner WMC Resources.
- WMC had been the focus of a hostile takeover by Swiss-based Xstrata that had gained attention from the government backbench before BHP put in its bid.
- Mr. Costello, who had the ability to block the takeover or set impossible restrictions, only set two conditions on BHP and its proposal, both relating to uranium.
- BHP chief executive Chip Goodyear welcomed the decision, saying the treasurer's conditions were acceptable and the company would abide by them.
- BHP has offered $7.85 for each WMC share, with the takeover bid due to close at 7.30 pm (AEST) on May 6.
- The federal government had raised no objection to the proposed takeover of WMC Resources by BHP Billiton, Treasurer Peter Costello said today.
• In a statement, Mr Costello set two conditions for the proposed $9.2 billion takeover of WMC by BHP Billiton.

• BHP Billiton chief executive Chip Goodyear welcomed the government's approval of the WMC bid.

• The company said the conditions attached to the announcement by the Treasurer today were acceptable to BHP.

The well-supported sentence to summarize the cluster is: *In a statement, Mr. Costello set two conditions for the proposed $9.2 billion takeover of WMC by BHP Billiton.*, yielding the cluster label: *takeover of WMC by BHP Billiton.*

To see the limitations of the current sentence-based approach, it is instructive to consider cluster labels generated from the 2005 CURNWS dataset given below in Table ...

<table>
<thead>
<tr>
<th>Entity</th>
<th>Event</th>
<th>Date</th>
<th>Cluster Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toyota</td>
<td>Automotive Sales</td>
<td>June 3, 2005</td>
<td>Toyota Motor Corp. posted a 0.5% sales</td>
</tr>
<tr>
<td>Google</td>
<td>Mergers&amp;Acquisition</td>
<td>April 22, 2005</td>
<td>Google, (GOOG) the No. 1 search engine, said its first-quarter profit</td>
</tr>
<tr>
<td>Google</td>
<td>Internet&amp;WWW</td>
<td>June 22, 2005</td>
<td>Google, which depends upon online</td>
</tr>
<tr>
<td>Google</td>
<td>Search Engine</td>
<td>June 22, 2005</td>
<td>Google to sell content through its search engine</td>
</tr>
</tbody>
</table>
Table 1: Sentence-based cluster-labels

Table 1 summarizes the running time of the cluster-label generation algorithm for various configurations. There is a steep increasing in the running time for sentence-based labels primarily due to the extra work involved in determining the exact location of the phrases in a sentence that correspond to the entities and the events. This information is necessary to clip a “supported” sentence to obtain the label.
7. Metadata Refinement and Subsequent Analysis

So far, an entity-event association (EN, EV) was inferred based solely with the presence of the metadata Entity term EN and the metadata Event term EV with weights crossing a given threshold value. In a document containing strong references to multiple entities EN₁, EN₂,..., ENₙ and multiple events EV₁, EV₂,..., EVₖ not all n × k entity-event associations can reasonably be "correct". In order to discover support for an entity-event association (EN, EV), the document content should be considered. For example, consider a news document that unravels the information of stock price information of various corporations. So the keywords like Microsoft can account for the entity term MICROSOFT CORP and can also account for the event term SOFTWARE MAKERS, Likewise Rexam plc can be tagged for an entity term REXAM PLC and for an event term CAN MANUFACTURER associations such as (MICROSOFT CORP, CAN MANUFACTURER) association can be generated is invalid semantically as the document is discussing the annual profit and stock information of the entities. Thus using the cross product of the entities and events with the document level metadata terms to represent the association between the terms is not correct. These unacceptable entity, event associations are called mis-associations.

This chapter presents an approach to eliminate the mis-associations. It also shows experimental examples and observations to improve result quality.
7.1 Metadata Refinement Approach

- The co-occurrence of entity-event (EN, EV) within a sentence or within a paragraph is a better yardstick of semantic relationship than just the incidental positional proximity in the document text that may be the result of co-occurrence in two neighboring sentences or in two neighboring paragraphs.

As proposed in the approach the existing metadata tags with their respective weights were generated by the entity, event recognizers at document level context. The entity En, event Ev terms are said to be associated with the incidental proximity in the document text. Our approach refines the of entity-event (EN, EV) pair associations by associating the entity-event (EN, EV) that appear in the proximity of same paragraph or sentence.

7.2 Paragraph/Sentence Level metadata Generation

So far, the metadata terms are generated at the document level scope. At document level the whole document text will be plugged into the entity, event recognizers to generate the metadata terms and their scores are generated with document level context. Similarly, the paragraph/sentence level metadata tags can be generated by plugging paragraph/sentence into the entity, event recognizers and their scores are also generated within the paragraph/sentence context. Consequently the scores will decrease compared to document level as the context was limited to paragraph/sentence and are not consolidated.
As discussed earlier, the DocumentSection instances are used to plug in the extracted paragraph/sentence to the entity, event recognizers to generate the metadata terms and their weights.

### 7.2.1 Extracting Paragraphs from the documents

The fundamental change in going from document-level to paragraph-level is the usage of the elementType tag in determining co-occurrence of entities and events within a paragraph. Each paragraph is identified with a different `<p>`-tag, and each matched entry carries the p-tag type marker in the entriesMatched’s termInfo object and each such element will contribute for a document section as shown below.

```
“DocumentSection("BODY", “p”+pTagCount, textOfDoc.substring(pStart, pEnd))”
```

```java
String body = "";
if(bodyStart < bodyEnd)
{
    body += textOfDoc.substring(bodyStart+10, bodyEnd);
    int pTagCount=0;
    body = body.replaceAll("<p>", "@");
    body = body.replaceAll("</p>", "@");
    String[] paras = body.split("@");
    for(int i=0;i<paras.length;i++)
    {
        if(!paras[i].equals("") && (!paras[i].contains("<")))
        {
            DocumentSection ds = new DocumentSection("BODY",
"p”+pTagCount, paras[i]);
            documentSections.add(ds);
            pTagCount++;
        }
    }
}
```

52
7.2.2 Extracting Sentences from the documents

To determine co-occurrence of entities and events at sentence level, similar to paragraph-level we generate a separate instance of DocumentSection for each sentence, and then determine termHits. To recognize a sentence, we use the fact that a paragraph (<p>-tag delimited fragment) is made up of sentences, separated by “period-followed-by-blanks-followed by an upper-case-letter-and-not-preceded-by-upper-case-letter”-pattern. Each sentence will contribute for a document section as shown below:

```

"DocumentSection("BODY", "s"+sTagCount, textOfDoc.substring(sStart, sEnd))"

Pattern pat1 = Pattern.compile("\p{Lower}\.\s\p{Upper}");
Pattern pat2 = Pattern.compile("@");
String body = "";
if(bodyStart < bodyEnd)
{
    body += textOfDoc.substring(bodyStart, bodyEnd);
    int sTagCount=0;
    body = body.replaceAll("</p><p>", "@");
    body = body.replaceAll("<p>", "@");
    body = body.removeListener("</p>", "@");
    String[] paras = body.split("@");
    for(int i=0;i<paras.length;i++)
    {
        if(!paras[i].equals("") && (!paras[i].contains("<")))
        {
            Matcher mat = pat1.matcher(paras[i]);
            StringBuffer sb = new StringBuffer();
            boolean result = mat.find();
            while(result) {
                mat.appendReplacement(sb,
                    "@"+(mat.group()).replace(". ", " ").trim());
                result = mat.find();
            }
            mat.appendTail(sb);
            String[] sen = pat2.split(sb.toString());
            for(int j=0;j<sen.length;j++)
            {
                DocumentSection ds = new DocumentSection("BODY", "s"+sTagCount, sen[j]);
                documentSections.add(ds);
                sTagCount++;
            }
        }
    }
}
```

53
7.2.3 Metadata generation using Entity, Event, Country, Person Recognizers

with Paragraph/Sentence level context

We now present and explain a sample piece of code that generates the metadata terms from the paragraph/sentence document sections at paragraph/sentence level context.

Note: The scores will decrease compared to document level as the context was limited to paragraph/sentence and are not consolidated.

```java
1) IndexingScorer insc =
2) IndexingScorerFactory.newInstance(_indexType);
3) Vector termHits = null;
4) Vector entriesMatched = null;
5) Vector greaterThan90 = new Vector();
6) Iterator docSectionIter = documentSections.iterator();
7) while (docSectionIter.hasNext())
8) {
9)    termHits = new Vector();
10)   DocumentSection ds = (DocumentSection) docSectionIter.next();
11)   termHits.addAll(inds.search(ds, CDI, id, "XML"));
12)   entriesMatched = insc.score(termHits, docContext);
13)   greaterThan90.addAll(entriesMatched);
14) }
15) entriesMatched = greaterThan90;
16) greaterThan90 = new Vector();
17) for (int i=0; i<entriesMatched.size(); i++)
18) {
19)    Entry entry=(Entry)entriesMatched.get(i);
20)    if (entry.getScore() >= THRESHOLD)
21)       greaterThan90.add(entry);
22) }
23) entriesMatched = greaterThan90;
```

- In the while loop line 7 the IndexingScorer consolidates the scores for the terms in the termHits vector using the paragraph/sentence context, and returns the entriesMatched vector. The indexing scorer will assign the score with paragraph/sentence context. The entriesMatched vector contains all the matched
terms belonging to the CVT list (which contains normalized metadata terms).

(The getCvt() method extracts the human readable CVT from a structure returned by getCVTList().)

- In line 20, we eliminate the entries with score less than the THRESHOLD value.

### 7.3 Examples and Observations

Consider the document fragment: *News of radical cost-cutting at carmaker General Motors and remarks from Federal Reserve chairman Alan Greenspan, which seemed to suggest the American central bank has finished raising interest rates, saw the Dow Jones Industrial Average chalk up an early three-digit gain.* The metadata term recognizer enable generation of entity (person/company) tag “ALAN GREENSPAN”, and event (subject/country) tags such as “INTEREST RATES”, “AUTOMOBILE MFG”, “CENTRAL BANKS”, “NORTH AMERICA” etc. To determine the most appropriate tags to associate with an event phrase in the text, we require the tag weights to be greater than 0.

<table>
<thead>
<tr>
<th>Entity</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEREST RATES</td>
<td>29</td>
</tr>
<tr>
<td>AUTOMOBILE MFG</td>
<td>7</td>
</tr>
<tr>
<td>AUTOMOBILE MFG</td>
<td>8</td>
</tr>
<tr>
<td>CENTRAL BANKS</td>
<td>13</td>
</tr>
<tr>
<td>CENTRAL BANKS</td>
<td>24</td>
</tr>
<tr>
<td>CENTRAL BANKS</td>
<td>29</td>
</tr>
<tr>
<td>NORTH AMERICA</td>
<td>17</td>
</tr>
<tr>
<td>NORTH AMERICA</td>
<td>23</td>
</tr>
<tr>
<td>AUTOMOBILE MFG</td>
<td>9</td>
</tr>
<tr>
<td>AUTOMOBILE MFG</td>
<td>19</td>
</tr>
<tr>
<td>STOCK INDEXES</td>
<td>33</td>
</tr>
<tr>
<td>STOCK INDEXES</td>
<td>44</td>
</tr>
<tr>
<td>AUTOMAKERS</td>
<td>7</td>
</tr>
<tr>
<td>AUTOMAKERS</td>
<td>8</td>
</tr>
</tbody>
</table>
Now consider another document fragment: A day after Google (GOOG:Nasdaq) delivered another dazzling quarter of earnings far ahead of Wall Street’s expectations, analysts rushed to boost their estimates as investors hustled to bid up the share price.

The metadata term recognizer enable generation of entity tag “GOOGLE INC” in response to the text “Google” and “GOOG”. However, for event tags, we draw a blank if the tag weight threshold is greater than 0. On the other hand, if we permit with weight greater than or equal to 0, the event tags are all over the place. Observe that the same phrase can be tagged multiple times and with a weight of 0. To determine the most appropriate tags to associate with an event phrase in the text, we use the global tags as a filter, thereby incorporating contextual information implicit in the document content. The bold face event tags below are also global (document-level) tags.

To better understand the nature of News documents and the effect of using different co-occurrence granularity as criteria for inferring entity-event associations, consider the following example fragments:
Document-1:

…

<p>When Kellogg Co. announced plans to bring back $1 billion in foreign profits to the United States this year thanks to new corporate tax breaks, it said it would use those funds to do such things as develop new products and explore potential acquisitions. It didn’t mention directing that money toward creating jobs.</p>

…

Document-2:

…

<p>Rexam eyes expansion in Eastern Europe and Asia</p>

<p>REXAM, which produces a billion cans a week for the likes of Coca-Cola and Anheuser-Busch, is looking for bolt-on acquisitions to grow in Eastern Europe and Asia after recent success in Russia. Underlying pre-tax profits rose 26% to £300 million on a 4% increase in sales from ongoing operations to £3.12 billion.</p>

<p>Sankyo and Daiichi confirm £4bn merger</p>

<p>*** News Story ***</p>

<p>VT radars in on £30m Royal Navy contract</p>

<p>*** News Story ***</p>

<p>Medisys axes boss after £6.8m assets writedown</p>

<p>*** News Story ***</p> ...

• A paragraph is delimited by <P>-</P> tags in an XML-version of a News document, while a sentence is separated using “period-followed-by-blanks-followed by an upper-case-letter-and-not-preceded-by-upper-case-letter”-pattern.
• **Document-1** associates KELLOGG CO with JOB CREATION. This can be determined by inspecting consecutive sentences within a paragraph, to resolve the co-reference “it”.

• **Document-2** is a classic example of a document with multiple News stories that can give rise to mis-associations using document-level tags. Specifically, document-level tags such as COMPUTER & ELECTRONICS MFG, COMPUTER MAKERS, DEFENSE CONTRACTING, NAVAL CONSTRUCTION & RETROFIT, NAVIES, etc is mis-associated with REXAM PLC. (The entity tag is part of the first story, while the event tags arise from the second and the third stories.) However, these mis-associations can be eliminated by using paragraph-level analysis.

• Paragraph-level association of tags such as COMPANY PROFITS with ANHEUSER-BUSCH COS INC and REXAM PLC can be eliminated by going to sentence-level. However, the net effect is *mixed* because this refinement is correct w.r.t. the former but incorrect w.r.t. the latter.

• The sentence-level association of MERGERS & ACQUISITIONS with ANHEUSER-BUSCH COS INC and REXAM PLC is tricky. If tags are filtered based on a threshold of > 0, then both associations are eliminated. However, if the threshold is changed to >= 0 and the generated tags are further filtered through document-level tags, then both associations are retained. In reality, only the association of
MERGERS & ACQUISITIONS with REXAM PLC is sound.

- Pragmatically, document-level analysis is scalable, while paragraph-level and sentence-level analysis, which is more time-consuming, can improve precision. However, there are cases where a clause-level analysis may be necessary and sufficient, such as in the last example. As discussed later, we propose to see if going to paragraph-level may provide a reasonable trade-off between precision improvement and recall reduction.

In order to see the reliability difference between paragraph level vs sentence level co-occurrence for inferring associations, consider the document for entity Microsoft and event Mergers & Acquisition on April 19, 2005 containing the fragment:

... Amazon already offers e-books and more than 1 million e-documents on its site, using downloadable software from Microsoft Corp. and Adobe Systems Inc. The purchase of Mobipocket will allow Amazon to use its own software to diversify product distribution methods, rather than relying on third-party providers. ...

The indexing metadata tags Microsoft and Mergers & Acquisition are associated with the phrases Microsoft and purchase. If document-level or paragraph-level co-occurrence of phrases is used for inferring associations, we get a false positive. As the phrases appear in successive sentences, sentence-level co-occurrence can improve reliability.
8. Quantifying and Analyzing Results

This Chapter discusses analyzing the improvement over the baseline system. We will discuss the observations made working with the News Document dataset and why we need a refinement over baseline system and how we improved the system with our approach.

8.1 Analyzing the Metadata Refinement Approach

The issue that we are dealing with for timeline refinement is how to eliminate mis–associations between entities and events. We cannot judge, without looking at the document context whether two metadata terms are related or not. For example to understand the relationship between the metadata tags (such as OIL & GAS PRICES, DISEASES & DISORDERS)), the relative locations of the text phrases (such as fuel costs, heart attacks) and the overall document content, we generated triples of numbers of supporting documents (nd, np, ns) (corresponding to document-level, paragraph-level and sentence-level support), for every “significant” entity-event pair, given a year’s worth of News document (2005), in a table (one per month) of the form:

|-----|--------|-------|----------|----------|----------|------|
(Rank captures the degree of interestingness of a specific entity-event query for determining granularity of co-occurrence of phrases for determining associations. E.g., rank can be \((nd - ns)\).)

The following table shows a number of entity-event queries and the number of documents retrieved based on document-level, paragraph-level, and sentence-level association. In general, more stringent criteria improves precision and reduces recall.

<table>
<thead>
<tr>
<th>Entity Event</th>
<th>#Doc</th>
<th>#Para</th>
<th>#Sent</th>
</tr>
</thead>
<tbody>
<tr>
<td>WILLIAM WRIGLEY JR CO MERGERS &amp; ACQUISITIONS</td>
<td>5</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>NORTHWEST AIRLINES CORP OIL &amp; GAS PRICES</td>
<td>8</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>AMERICAN BROADCASTING COS INC TERRORISM</td>
<td>10</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>J C PENNEY CO INC BUYINS &amp; BUYOUTS</td>
<td>8</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>R R DONNELLEY &amp; SONS CO MERGERS &amp; ACQUISITIONS</td>
<td>9</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>US DEPARTMENT OF ENERGY FUEL CELL TECHNOLOGY</td>
<td>13</td>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td>J P MORGAN CHASE &amp; CO MERGERS &amp; ACQUISITIONS</td>
<td>9</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>R J REYNOLDS TOBACCO CO SUITS &amp; CLAIMS</td>
<td>6</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>VOLKSWAGEN AG JUSTICE DEPARTMENTS</td>
<td>9</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>REXAM PLC NAVIES</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2: Refinement of association using different co-occurrence criteria

The analysis of the table and the corresponding documents yielded a number of examples supporting the following tentative observations:

(a) There are documents with document-level entity tags with weights > 90% where these entities do not seem to be the subject of the document. For instance, a document may carry an ABC NEWS tag not because the document is about the News reporting agency, but because the document is a TV News broadcast transcript containing reference to the reporter’s affiliation such as “Diane Sawyer of ABC News …”, “…, Graphic: Flooding by Nancy Weiner, ABC News”, etc. Another example is the
redundant occurrence of stock exchange names (NYSE, AMEX, NASDAQ) in a document about stocks. Sentence-level analysis and special treatment of publisher tags can remedy this problem. Similar situation does not seem to arise for Reuters, may be because there is no analog of TV News transcripts. That is, Reuters-tagged documents seem to be talking about Reuters business as a subject. *(Publisher/Stock Exchange Tags Case)*

(b) There are documents for which entity-event associations derived from document-level entity-event tags are unsound but paragraph-level entity-event phrase co-occurrence are sound. This is especially the case when a document contains fragments of multiple independent short News stories in print or multiple TV News stories (such as about Inaugural function for Pope Benedict XVI followed by Martha Stewart’s house arrest) or multiple Newsitem-elements. Furthermore, moving to sentence-level co-occurrence may cause incompleteness in the presence of co-references (Cardinal Ratzinger vs Pope Benedict XVI vs Holy Father, or GM Cars vs Chevrolet/Cadillac). *(Multiple Independent Stories)*

(c) There are documents describing stock prices and changes in stock quotes (with headlines such as “Big movers on the stock market”) that involve companies (such as IBM, GE, Eli Lilly, Citigroup, GM, etc) from different market sectors (such as Hardware, Pharmaceuticals, Automotive, etc) and fluctuations due to different reasons (such as Oil Prices, Litigation, etc). This can potentially be a rich source of erroneous associations. Derivation of entity-event associations solely on the basis of sentence-level entity-event co-occurrence can improve precision in this case. *(Stock News documents)*
(d) There are some Question-Answer documents that potentially contain description of a wide variety of subjects whose document level tags can lead to erroneous associations. The associations can be improved by using paragraph level or sentence level analysis. (*Q&A documents*)

(e) There are documents where document-level tags result in subtle erroneous associations in search context. For instance, there is a Lockheed Martin – Nuclear Weapons tagged document where the latter is related to collaborative research between University of Texas and Sandia National Lab, and is not associated with Lockheed Martin. Or a Microsoft – Search Engine tagged document where Search Engine is clearly related to Google. Or a Google - Electronic Mail tagged document where Laplink Software Inc and Email are associated with Google Desktop, but the document is not about Gmail. (*Content-based Errors*)

(f) A sentence containing “Kate Snow of ABC News” gets erroneously tagged with WEATHER and ABC News showing that inferred sentence level associations can be unsound too in general.

To improve information displayed on the Timeline GUI, we have developed headline-based and sentence-based cluster-label generation algorithms discussed in Chapter 7. The former seems to work well given the nature of correlated headlines, while the latter seems to generate meaningful labels that do not always “look” pleasing. (See NLDB 2007 Conference paper: “Selecting Labels for News Documents Cluster”.) In the future, we will consider the issue of clustering headlines of a day resulting from an entity-event
query, to deal with multiple stories involving the entity and the event, and to exploit metadata tags for clustering documents.

The following table shows how associations can be manipulated using different co-occurrence criteria. Specifically, note that document-level associations can be refined by moving to sentence-level, but the latter are still far from sound.

<table>
<thead>
<tr>
<th>Entity Tag</th>
<th>Event Tag</th>
<th>Association Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALAN GREENSPAN</td>
<td>TELECOMMUNICATIONS SERVICES</td>
<td>Document</td>
</tr>
<tr>
<td>ALAN GREENSPAN</td>
<td>GOVERNMENT BONDS</td>
<td>Document</td>
</tr>
<tr>
<td>ALAN GREENSPAN</td>
<td>TALKS &amp; MEETINGS</td>
<td>Document</td>
</tr>
<tr>
<td>ALAN GREENSPAN</td>
<td>WIRELESS INDUSTRY</td>
<td>Document</td>
</tr>
<tr>
<td>ALAN GREENSPAN</td>
<td>BEVERAGE INDUSTRY</td>
<td>Document</td>
</tr>
<tr>
<td>ALAN GREENSPAN</td>
<td>FOOD &amp; BEVERAGE</td>
<td>Document</td>
</tr>
<tr>
<td>ALAN GREENSPAN</td>
<td>WIRELESS TELECOMMUNICATIONS CARRIERS</td>
<td>Document</td>
</tr>
<tr>
<td>ALAN GREENSPAN</td>
<td>TRADE SHOWS</td>
<td>Document</td>
</tr>
<tr>
<td>ALAN GREENSPAN</td>
<td>BROKER RECOMMENDATIONS</td>
<td>Document</td>
</tr>
<tr>
<td>ALAN GREENSPAN</td>
<td>CONFERENCES &amp; CONVENTIONS</td>
<td>Document</td>
</tr>
<tr>
<td>ALAN GREENSPAN</td>
<td>BOND MARKETS</td>
<td>Document</td>
</tr>
<tr>
<td>ALAN GREENSPAN</td>
<td>NORTH AMERICA</td>
<td>Sentence</td>
</tr>
<tr>
<td>ALAN GREENSPAN</td>
<td>AUTOMOTIVE MFG</td>
<td>Sentence</td>
</tr>
<tr>
<td>ALAN GREENSPAN</td>
<td>STOCK INDEXES</td>
<td>Sentence</td>
</tr>
<tr>
<td>ALAN GREENSPAN</td>
<td>CENTRAL BANKS</td>
<td>Sentence</td>
</tr>
<tr>
<td>ALAN GREENSPAN</td>
<td>AUTOMOBILE MFG</td>
<td>Sentence</td>
</tr>
<tr>
<td>ALAN GREENSPAN</td>
<td>INTEREST RATES</td>
<td>Sentence</td>
</tr>
<tr>
<td>ALAN GREENSPAN</td>
<td>AUTOMAKERS</td>
<td>Sentence</td>
</tr>
<tr>
<td>ALLIED DOMEQ PLC</td>
<td>BEVERAGE INDUSTRY</td>
<td>Sentence</td>
</tr>
</tbody>
</table>
8.2 Approach to Quantify Results

In order to uncover an optimal granularity for co-occurrence of an entity phrase and an event phrase (that is, one of document-level, paragraph-level, sentence-level) to determine when a document satisfies an entity-event query, we propose a criteria based on a variant of the traditional notions of precision and recall. By document-level, we mean the existence of entity tag and event tag in the same document. By paragraph-level (resp. sentence-level), we mean the occurrence of entity phrases and event phrases within the same paragraph (resp. sentence). Specifically, to establish paragraph-level as the appropriate granularity for co-occurrence when computing entity-event query results, we need to establish that going from document-level to paragraph-level shows far greater increase in precision compared to reduction in recall, while going from paragraph-level to sentence-level diminishes recall much more than it improves precision. We focus on the manual inspection of documents that are eliminated when going from document-level (resp. paragraph-level) to paragraph-level (resp. sentence-level), to determine estimates for changes in precision and recall. Our approach tries to minimize the manual effort.
required to analyze the News documents to compare the three alternatives for granularity of co-occurrence.

Let nd, np, and ns be the number of documents that support an entity-event association at document-level, paragraph-level and sentence-level respectively. We assume that the recall for an entity-event query on the basis of the document-level existence of the corresponding tags is 100%, and the precision for an entity-event query on the basis of the sentence-level co-occurrence of the corresponding phrases is 100%.

Let the documents \((nd – np)\) containing entity-event phrases in different paragraphs be partitioned into two groups: those that are found relevant to the entity-event query (say of size \(n_{gP}\)) and those that are found irrelevant to the entity-event query (say of size \(n_{bP}\)). \([nd – np = n_{gP} + n_{bP}\]) (Recall that relevancy is based on human judgment.) Then the “relative” precision at the document level is \((nd – n_{bP}) / nd\), or equivalently, \((np + n_{gP}) / nd\), assuming optimistically that np documents are relevant. The “relative” recall at the paragraph-level is \(np / (np + n_{gP})\). In other words, if there are very few irrelevant documents in which entity phrases and event phrases occur in different paragraphs, then the precision at document-level will be high. Similarly, if there are many documents in which entity phrases and event phrases occur in different paragraphs and are relevant, then the recall at paragraph-level will be low. On the other hand, to prefer paragraph-level query results over document-level query results, the overall improvement in precision should offset the overall reduction in recall. Thus, we
can define a quality factor QF for each entity-event query (or for that matter any document set) as equal to \[ \frac{\text{Recall at paragraph-level}}{\text{Precision at document-level}} = \left( \frac{(np+1) * (nd+1)}{np+1+n_gP} \right)^2 \]. (The increment helps in not giving any unpleasantness when the values are ZERO).

A similar analysis can be carried out to determine if we should prefer sentence-level associations over paragraph-level associations as follows: Let the documents \((np – ns)\) containing entity-event phrases in different sentences be partitioned into two groups: those that are found relevant to the entity-event query (say of size \(n_gS\)) and those that are found irrelevant to the entity-event query (say of size \(n_bS\)). The precision at the paragraph-level is \((np – n_bS) / np\), or equivalently, \((ns + n_gS) / np\), assuming optimistically that \(ns\) documents are relevant. The recall at sentence-level is \(ns / (ns + n_gS)\). On the other hand, to prefer sentence-level query results over paragraph-level query results, we can define a quality factor QF for each entity-event query (or for that matter any document set) as equal to \[ \frac{\text{Recall at sentence-level}}{\text{Precision at paragraph-level}} = \left( \frac{(ns+1) * (np+1)}{ns + 1 + n_gS} \right)^2 \].

Experimentally, through manual analysis of a small subset of documents, we can determine the Quality Factor over a wide range of queries, or over specific (well-circumscribed) document clusters, to determine the most effective granularity of co-occurrence in a given context. Relevant information can be summarized as follows:
We are expecting that this analysis will provide quantitative support to our earlier qualitative observations about the nature of documents that can be benefited by paragraph-level analysis and by sentence-level analysis. For instance, precision of the results of an entity-event query (and its efficiency of computation) can be improved by considering document-level co-occurrence of entity-event tags for general documents, paragraph-level co-occurrence of entity-event phrases for multiple News item documents and Questions-Answers documents, and paragraph-level co-occurrence of entity-event phrases for Stock Price Fluctuations documents.

The following table shows that queries involving REXAM PLC – NAVIES, GOOGLE – OIL & GAS PRICES, and KELLOGG CO – EMPLOYMENT GROWTH, etc benefit from paragraph-level co-occurrence criteria, while J C PENNEY CO INC – queries can benefit from sentence-level co-occurrence criteria. For REXAM PLC – NAVIES query, the result set contains documents with multiple News stories that are removed by paragraph-level analysis. For GOOGLE – OIL & GAS PRICES query, the only document that survives even sentence-level analysis contains an indirect contextual association of “current prices” with OIL & GAS PRICES via latter’s connection to Indonesia’s economy, and the relationship between Indonesia’s net worth to Google’s net worth. KELLOGG CO – EMPLOYMENT GROWTH query benefits from paragraph-level analysis that effectively infers association through co-reference. The sentence-level analysis benefits J C PENNEY CO INC – query by eliminating association with OIL & GAS PRICES, thereby improving precision. However, for J C PENNEY CO INC – COMPANY EARNINGS query, the GIAPI-COS does not interpret “Penney” as a reference to the entity J C. PENNEY, thereby reducing recall (shown in RED in Table 4).
If we were to change the document by updating “Penney” to “J. C. Penney”, we reach the conclusion that document-level analysis is adequate (shown in GREEN in Table 4).

<table>
<thead>
<tr>
<th>Entity</th>
<th>Event</th>
<th>nd</th>
<th>np</th>
<th>ns</th>
<th>QFp</th>
<th>QFs</th>
</tr>
</thead>
<tbody>
<tr>
<td>REXAM PLC</td>
<td>NAVIES</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>GOOGLE INC</td>
<td>OIL &amp; GAS PRICES</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>3.5</td>
<td>1</td>
</tr>
<tr>
<td>OAO GAZPROM</td>
<td>BANKRUPTCY COURTS</td>
<td>11</td>
<td>8</td>
<td>7</td>
<td>1.33</td>
<td>1.125</td>
</tr>
<tr>
<td>KELLOGG CO</td>
<td>EMPLOYMENT GROWTH</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>J C PENNEY CO INC</td>
<td>COMPANY EARNINGS</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0.18</td>
</tr>
<tr>
<td>J C PENNEY CO INC</td>
<td>OIL &amp; GAS PRICES</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0.18</td>
</tr>
<tr>
<td>J C PENNEY CO INC</td>
<td>COMPANY EARNINGS</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4. Quantifying improvement using different co-occurrence criteria

We can infer the refinement improvement over the baseline system comparing the quality factors. We assume the quality factor at document level QFd to be 1. If QFp > QFd going to paragraph level refinement is preferred. If QFs > Qfp going to sentence level refinement is preferred.

8.3 Performance Details

The following table gives the performance details to index, search and present the results in a timeline and to access the documents through timeline index.

<table>
<thead>
<tr>
<th>Time to index (on entity) for one month data (12 GB)</th>
<th>30 hrs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Query response for determining associations (for 1000 doc)</strong></td>
<td>2/4/6 min</td>
</tr>
<tr>
<td>Document/Paragraph/Sentence-level analysis generating Metadata</td>
<td>(using GIAPI)</td>
</tr>
<tr>
<td>Document/Paragraph/Sentence-level analysis given Metadata</td>
<td>Few Seconds</td>
</tr>
<tr>
<td>Time to show headlines</td>
<td>Few seconds</td>
</tr>
</tbody>
</table>

Table 5. Performance data on indexing and searching

**Machine Used:** HP xw9300 WS, 64-bit XP, Dual AMD Opteron 275/2.2GHz (dual-core), 12 GB MM, 880 GB SCSI
9. Decouple GUI using Metadata

The Timeline Application can be driven by metadata and timeline application's GUI can be decoupled from actual pre-processing of dataset and the time consuming metadata generation for the news documents. And more over decoupling the GUI enables choosing various GUI builder tools for the application. Two GUI interfaces were developed, one was the Java based and the other was Flex based that uses only the metadata XML file to generate timelines in lieu of the entire dataset.

The Java based timeline application can generate at most 10 multiple timelines per query while Flex based timeline applications supports any number of multiple timelines. A user will really experience the need for a Flex based version when compared with Java based because of the formers built in extended functionality in presenting the timeline interface.

In Java based application, a click is required in the vicinity of the date to show the popup window with label, headlines and file paths where as in Flex based application a mouse rollover on the data point is required to show the popup window with headlines and file paths. The label generation feature was not included in Flex based GUI.
9.1 Timeline Application Screen Shots

We now present the timeline application screen shots to give a sense of how they look like and how our approach of moving from a list of search results interface to a graphical timeline interface improves presenting and analyzing the search results for a given query.

Figure 6: Java based Entity-Event-Duration Timeline (Microsoft-All-Events Case)
Figure 7: Entity-Event-Duration Timeline (Microsoft-Software Makers with Headlines)

Figure 8: Entity – Multiple Events Duration Timeline Showing a retrieved document
Figure 9: Entity-Event Timeline with Sentence based Label and document headlines

Figure 10: Flex GUI showing Single Timeline, driven by Metadata XML file.
Figure 11: Microsoft Vs Computer operating systems with headlines and file paths

Figure 12: Multiple timelines showing Microsoft Vs Multiple Events for April 2005
Figure 13: Multiple timelines with headlines and file paths for Microsoft in April 2005

Figure 14: Multiple timelines with headlines and file paths for Google in April 2005
10. Conclusion and Future Work

We developed a Generalized Timeline Application which allows us to visualize search results in the form of a chart for News Documents dataset for the year 2005 for a given query and time window. In general, we can render single/multiple timelines for any pair of associations from entity, event, person, and country.

We designed a Baseline system to show the retrieved news documents in a timeline for a given query. The Baseline system uses the metadata terms that exist in the news documents which were generated using entity, event, country, and person recognizer’s offline. To visualize the answer set, we developed an algorithm to generate an XML formatted timeline metadata structure which can alone be used to generate timelines in lieu of news documents. We then refined the Baseline system by eliminating the entity-event mis-associations. To eliminate mis-associations of metadata terms, we explored two separate refinements – one that redefines support in terms of co-occurrence of entity-event pair in a paragraph, and another that redefines support in terms of co-occurrence of the entity-event pair in a sentence, instead of their mere appearance in the document.

To improve information displayed on the Timeline GUI, we have developed two label generation approaches one with headlines and other with sentences from the News
Documents. The former approach is necessary if only the timeline metadata is available and the later can be applied if the entire News Documents are available.

We proposed an approach to quantify the results using precision and recall. The document/paragraph/sentence level quality factors (QF) generated from this approach can help in judging what level of refinement is required for eliminating entity-event mis-associations.

We designed the application to decouple the GUI from the document content by generating timeline metadata offline. We then developed two timeline graphical interfaces: one is Java based and another is Flex based.

In future, we will consider addressing the following issues to improve the Timeline Application:

The physical dataset was used for both the paragraph and sentence level analysis. It is worth exploring other alternatives for abstracting documents.

The present form of the metadata can only generate timelines for any pair of entity, event, country and person. We need to explore alternative representations of timeline metadata to generate arbitrary timelines efficiently.
The metadata term generation approach suffers from co-reference problem and can benefit from robust co-reference resolution algorithms.

We will consider the issue of clustering headlines of a day resulting from an entity-event query, to deal with multiple stories involving the entity and the event, and to exploit the metadata for clustering documents.

As the size of the metadata increases, the Flex GUI takes a lot of time to render the timelines for a selected entity-event pair. So the performance of Flex GUI can be improved by adopting the MVC design pattern. Java or dot net objects can be used to pre-compute the metadata and respond to the Flex object. The advanced graphing facilities of Flex like SpringGraph or Roamer can improve visualization of relationships among various entities, persons, countries and events.
REFERENCES


[8] [www.google.com/trends](http://www.google.com/trends)

[9] [www.finance.yahoo.com](http://www.finance.yahoo.com)

[10] [www.finance.google.com](http://www.finance.google.com)

[11] [www.clusty.com](http://www.clusty.com)

[12] [www.wikipedia.org](http://www.wikipedia.org)