Continuous Keyword Search on Multiple Text Streams

Vagelis Hristidis[*] Oscar Valdivia Michail Vlachos Philip S. Yu

School of Computing and Information Sciences
Florida International University
{vagelis, oscar.valdivia}@cis.fiu.edu

IBM T. J. Watson Research Center
mvlachos@cs.ucr.edu, psyu@us.ibm.com

ABSTRACT

In this paper we address the issue of continuous keyword queries on multiple textual streams. This line of work represents a significant departure from previous keyword search models that assumed a static database. In our model the user poses a query comprised by a collection of keywords, which is subsequently applied on multiple text streams (these can be RSS news feeds, TV closed captions, emails, etc). A result to a query is a combination of streams "sufficiently correlated" to each other that collectively contain all query keywords within a specified time span.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]

General Terms

Algorithms, Performance, Experimentation

Keywords

streams, keyword search, correlation

1. OVERVIEW

In this work we present techniques for extracting useful information from a collection of text streams. This paper represents, to our best knowledge, the first approach that performs keyword search on a multiplicity of textual streams. The scenario that we consider is quite intuitive; let's assume that a professional analyst would like to continuously be informed about occurrences of a topic, a topic that can be described by several keywords. Information is provided by various feeds in a streaming fashion. The running example we use is the streams of emails in the ENRON dataset, where each stream corresponds to an email session (sequence of send, reply, forward and so on). Suppose we are looking for information on "arbitrating problem", within a stream or across multiple streams; e.g., Stream *A* may mention "arbitrating" whereas another stream, Stream *B*, which is somehow correlated to Stream *A*, may mention "problem".

Our methodology shares apparent commonalities with two technologies. The first one is keyword search on databases and the second is subscription/alert services (eg Google Alerts). We briefly elaborate on the differences of the approach that we propose. Keyword search on databases [ACD02, BNH+02, HP02] provides support for discovery of associations between the query keywords on textual sources; however these sources are static or updated in a batch fashion. Additionally, the posed queries address only a specific time snapshot of the data. The left side of Figure 1 shows how a set of text sources, which can range from relational tuples, to XML elements, to Web pages, is associated by weighted edges, which can represent any semantic connection like primary-to-foreign keys, XML containment edge or hyperlink. On the other hand, alert systems over text sources do provide support for streaming sources, but each stream is typically processed separately, hence inter-correlation between the different streams is completely lost. On the right side of Figure 1 we see a set of streaming text sources and a continuous query.

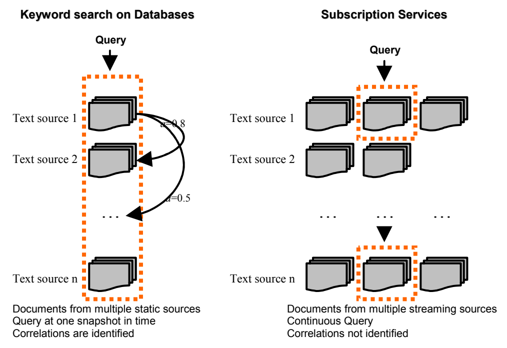
Keyword search on Databases
Query
Text source 1
Text source 2
Text source n
a=0.8
a=0.5
Documents from multiple static sources
Query at one snapshot in time
Correlations are identified
Subscription Services
Query
Text source 1
Text source 2
Text source n
Documents from multiple streaming sources
Continuous Query
Correlations not identified

Figure 1: *Left:* Keyword search on Databases, *Right:* Structure of Alert/Subscription services.

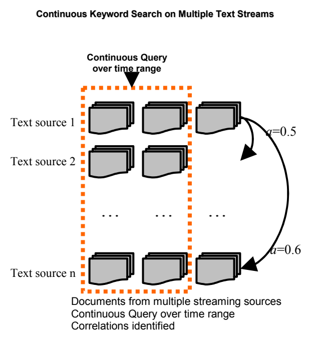
Continuous Keyword Search on Multiple Text Streams
Continuous Query over time range
Text source 1
Text source 2
Text source n
a=0.5
a=0.6
Documents from multiple streaming sources
Continuous Query over time range
Correlations identified

Figure 2: Overview of the proposed methodology.

Our work bridges the above two technologies, by supporting correlation and keyword search over a time span on multiple textual streams, as shown in Figure 2. Therefore, the proposed system allows the inclusion of the inherent temporal dimension

[*] Partly supported by NSF grant IIS-0534530.

into the problem, enabling the execution of more complex keyword queries with temporal constraints.

Problem definition A *stream* is a sequence of *events* (e.g., emails for email streams) e_i, where each event has a textual description. A *keyword query* is a set of keywords w_i. A *result* (*event tree*) to a keyword query is a tree of streams and their corresponding events, whose events within the specified time window collectively contain all query keywords. An edge is created between two streams if they are correlated, that is, they have similar topics. The intuition behind requiring that two streams in a result have to be correlated is that the user typically tries to combine information from related sources, e.g., email sessions involving overlapping participants or topics. Removing the correlation requirement would also overwhelm the user by producing too many results.

Technical challenges There are two challenges in developing a system to perform continuous keyword search on multiple search streams. First, the correlations between the streams must be maintained in an incremental manner as new events arrive and streams are created and destroyed. Second, we need to incrementally check for results by on-the-fly joining the new events with the previous events from multiple streams, within the specified time window.

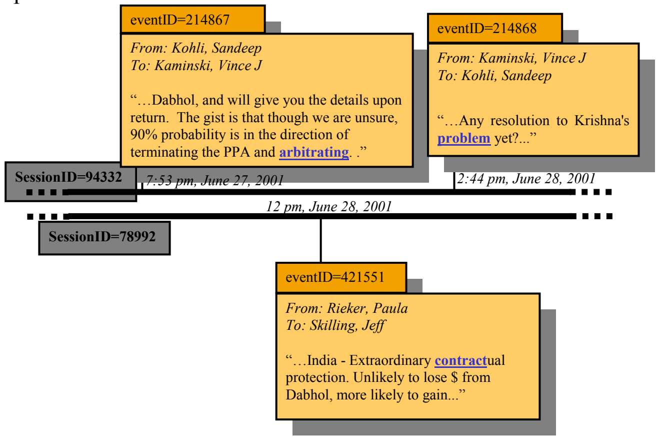
eventID=214867
From: Kohli, Sandeep
To: Kaminski, Vince J
"…Dabhol, and will give you the details upon return. The gist is that though we are unsure, 90% probability is in the direction of terminating the PPA and arbitrating. ."
eventID=214868
From: Kaminski, Vince J
To: Kohli, Sandeep
"…Any resolution to Krishna's problem yet?..."
SessionID=94332
7:53 pm, June 27, 2001
2:44 pm, June 28, 2001
12 pm, June 28, 2001
SessionID=78992
eventID=421551
From: Rieker, Paula
To: Skilling, Jeff
"…India - Extraordinary contractual protection. Unlikely to lose $ from Dabhol, more likely to gain..."

Figure 3: Snapshot from the ENRON emails stream.

Example 1: Figure 3 depicts a timeline snapshot of the ENRON email database, which captures a small subset of email events. Given the query *"arbitrating problem"*, the event tree $S_{94332}(e_{214867}, e_{214868})$ is output, whereas for the query *"arbitrating contract problem"* the event tree $S_{94332}(e_{214867}, e_{214868})$-$S_{78992}(e_{421551})$ is output. Sessions S_{94332} and S_{78992} (which conceptually correspond to different streams of events) are correlated because they share rare words like "Dabhol". Clearly, the above event trees convey useful information; the first shows how arbitration problems have arisen with Krishna, and is captured by keywords existing on the same stream but on different documents (emails) on the timeline. For the second case the various pieces of information are not only fragmented among different documents but also among different streams, and the example shows that Dabhol (area of India where the power plant was supposed to be built) was a conflicting and problematic topic among ENRON executives. □

Another recent, related field of research deals with the join of streaming data [DGR03, GO03], which involves only *exact* matches between multiple streams within a time window.

2. ALGORITHMS

Stream correlations To maintain the *Stream Graph*, where each node is a stream and each edge represents the degree of correlation between two streams, we can use an incremental computation of the Jaccard coefficients between the participants and the content of the streams. No more details are presented due to lack of space.

Results generation As events arrive, the system must output all created results in almost real time. Hence, an incremental algorithm is needed that when a new event e arrives, it outputs all event trees involving e. Only a brief overview of the *Tree algorithm* is provided here. The key idea of the algorithm is to maintain a forest C of query-related events (i.e., events that contain some query keyword), where each path from a root to a leaf represents a combination of events ordered by ascending timestamps. Each level of C corresponds to a single event e and each node of this level determines if e is considered in the corresponding root-to-leaf path. Each such path becomes a candidate result as we explain below.

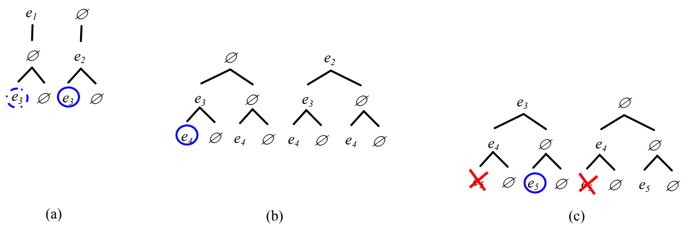

Figure 4: Snapshots of forest C in Tree algorithm.

Example 2: Consider query q=*(Florida, mortgage), and* three text streams S_1, S_2, S_3 with the following time-interleaved sequence of events (only the query-related events are shown):

e_1: in stream S_1, e_1*.content* = "Florida real estate is hot."
e_2: in stream S_2, e_2*.content* = "I live in Florida."
e_3: in stream S_3, e_3*.content* = "More people apply for mortgage."
e_4: in stream S_1, e_4*.content* = "Florida is growing."
e_5: in stream S_2, e_5*.content* = "People move to Florida."

Figure 4 shows snapshots of the forest C after e_3, e_4 and e_5, assuming that only event trees with up to 3 consecutive query-related events (events that contain a query keyword) are of interest. For every event e_i, the nodes e_i and $\varnothing$ denote the inclusion or not of this event in the event tree respectively. Solid-line circles denote leaf nodes that correspond to a result, and dotted-line circles are the nodes that do not output a result, because the score of the event tree is less than the specified association threshold. E.g., the first solid circle corresponds to the event tree $S_2(e_2)$-$S_3(e_3)$. Furthermore, we cross out leaf nodes that are pruned as redundant because they would produce a non-minimal result, that is, a result where an event does not contribute any keyword.

3. REFERENCES

[ACD02] S. Agrawal, S. Chaudhuri, and G. Das. DBXplorer: A System For Keyword-Based Search Over Relational Databases. ICDE, 2002

[BNH+02] G. Bhalotia, C. Nakhe, A. Hulgeri, S. Chakrabarti and S,Sudarshan: Keyword Searching and Browsing in Databases using BANKS. ICDE, 2002

[DGR03] A. Das, J. Gehrke, and M. Riedewald. Approximate join processing over data streams, SIGMOD, 2003

[GO03] L. Golab and M. T. Ozsu. Processing sliding window multi-joins in continuous queries over data streams. VLDB, 2003

[HP02] V. Hristidis, Y. Papakonstantinou: DISCOVER: Keyword Search in Relational Databases. VLDB, 2002