Specifications and Architectures of Federated Event-Driven Systems
Outline

- Goals
  - Architecture
  - The Event-Driven Web
- Data Analysis
- Algorithms
- Conclusions & Future Work
Demonstrate an Architecture

• Modular distributed event-based systems enable:
  ○ simple reuse of code written in any language/specification
  ○ transparent addition/removal of distributed components

• This leads to libraries of event processing agents
Local ESB Facilitates Communication

- Multi-threaded sensor
- Multi-threaded processing component
- Multi-threaded responder

Local ESB

Creates queues and connects them to components

Shared input/output queue
Local ESB Facilitates Location Transparency

System 1

- Multi-threaded sensor
- Shared input/output queue
- Multi-threaded sender

System 2

- Multi-threaded receiver
- Shared input/output queue
- Multi-threaded processing component
- Shared input/output queue
- Multi-threaded responder
Local ESB Facilitates Distributed Processing

System 1

Multi-threaded sensor

Thread
Thread
Thread

Shared input/output queue

Multi-threaded sender

System 20

Multi-threaded sensor

Thread
Thread
Thread

Shared input/output queue

Multi-threaded sender
Distributed Component Architecture

- Web Service
- Feds

Enterprise Service Bus

- Processing Component
- Serialization/Deserialization
- Data Transformer
- Local ESB

Physical Systems

- Process
- Feds
- Web Service
Modular Components Enable Mashups

A New System

Processor Components

Sensor Components

Responder Components
Single Process Feds Example

Spinn3r

Blog Sensor

Metadata Retrieval

RDF Parser

Event Engine Responder

Event Engine

News Sensor

Spotlight
Single Process Feds Example

- Spinn3r
- Calais
- Event Engine
- Spotlight

Diagram:
- Local ESB
  - News Sensor
  - Blog Sensor
  - Event Engine Responder
  - RDF Parser
  - Metadata Retrieval
Outline

- Goals
  - Architecture
  - The Event-Driven Web
- Data Analysis
- Algorithms
- Conclusions & Future Work
Demonstrate The Event-Driven Web

- **What is the Event-Driven Web?**
  - Massive numbers of sources
    - Blogs, News, Facebook, Twitter, Flickr . . .
  - Events defined by consumers
    - What is the end user interested in?
  - Heterogeneous responders
    - Cell phone, iPod, laptop, TV, car, calendar . . .

- What can be done with sources?
- How can consumers define events?
- What is the most appropriate response?
Demonstrate The Event-Driven Web

- Value-add from sequences of push
  - resource conservation
  - timeliness

- Detection of events on Internet streams
  - news
  - blogs
  - twitter
Demonstrate Internet Event Detection

- **Online algorithms to generate events**
  - events are based on meaningful changes in frequency or correlation with and between *any* entity or pair of entities

- **Events in relationships**
  - relationships are formed, strengthened, or destroyed
Examples

- **Simple Event Detection**
  - Frequency change in volume of posts about:
    - an entity (company, location, person, etc.)
    - a relationship between entities
    - a topic

- **Complex Event Detection**
  - Combine two or more different sources:
    - change in price of stock and change in volume of news
WaMu files for bankruptcy
Wells Fargo Buys Wachovia

Number of posts

Outline

- Goals
  - Architecture
  - The Event-Driven Web
- Data Analysis
- Algorithms
- Conclusions & Future Work
Data Analysis – Entities

- Any person, thing, or concept identified in a post

- 27 million posts
  - 261,036 per day
  - Weekends have 76% of the posts of weekdays

- 77 million entities
  - 1 million unique entities per day
  - 1.9 million entity observations per day
Data Analysis – Relationships

- Defined as a co-occurrence of entities within a post
  - an entity can’t be related to itself
  - order (currently) doesn’t matter

- Mean entities per post: 7 -> 21 relationships
- Std deviation: 11, 18 entities -> 153 relationships
- Max entities per post: 2,595 -> 3,365,715 relations

- Median relationships per day: 18,890,658
Histogram of Entities in a Post
Unique Entity Occurrences in One Hour

$y = 0.71x - 2.25 \times 10^3$
Unique Entity Occurrences in One Day

\[ y = 0.584x - 5.73 \times 10^4 \]
Relationships Observed in One Hour
Outline

- Goals
  - Architecture
  - The Event-Driven Web
- Data Analysis
- Algorithms
- Conclusions & Future Work
Algorithmic Challenges

- Too much data -> cannot store interarrival data
  - moving averages and similar statistics difficult or impossible

- Post volume varies:
  - between entities (Google versus CEP)
  - between dates (weekdays versus weekends)
First Algorithm – Exponential Smoothing

- Maintain a single summary number
  - small memory footprint
  - constant time updates

- \( s_t = \) summary number at time \( t \), \( s_0 = 0 \)
- \( o_t = \) occurrences during time \([t, t+1]\)
- \( \alpha = \) rate of decay, \( \alpha > 0 \) (\( \alpha = 0 \) is just a counter)

- The summary number is then given by:
  \[
  s_{t+1} = e^{-\alpha} s_t + o_t
  \]
Exponential Smoothing

- If we haven’t updated since a given time \( n \), then:
  \[
  o_i = 0 \quad \forall i \in [n, t - 1]
  \]
  \[
  s_{t+1} = e^{-(t+1-n)\alpha} s_n + o_{t+1}
  \]

- Setting \( n = t \) returns the previous equation
Expected Time of Irrelevance (ETI)

- Calculated when summary numbers are updated

- \( \gamma \) = threshold value of interest
  - discard entities with a summary number less than \( \gamma \)

- \( h \) = time units until object is discarded

\[
h = \left\lceil \frac{\ln(s_t) - \ln(\gamma)}{\alpha} \right\rceil
\]
Object Buckets

- Each object is slotted into a bucket based on its ETI
  - object can be moved between buckets when ETI is updated

- Objects remaining when time arrives are discarded
  - implies estimate was accurate, no need for update
Concurrent Computation

- Natural labor divisions:
  - Post processing
  - Relationship tracking by entity class
Outline

- Goals
  - Architecture
  - The Event-Driven Web
- Data Analysis
- Algorithms
- Conclusions & Future Work
Conclusion

- Event detection on arbitrary entities and relationships between entities is possible with modest hardware.
Future Work

- **Source examination:**
  - Do sources vary in their ability to explain/predict events?
  - Are some sources better for certain classes of events?
  - Can you read a small subset with similar accuracy?

- **Algorithm examination:**
  - How do different algorithms compare with respect to:
    - accuracy
    - timeliness
    - efficiency
  - What can we add to improve algorithms
    - reputation
    - learning