Fast Data in the Era of Big Data: Twitter's Real-Time Related Query Suggestion Architecture

Gilad Mishne, Jeff Dalton, Zhenghua Li, Aneesh Sharma, Jimmy Lin

Presented by: Rania Ibrahim

AGENDA

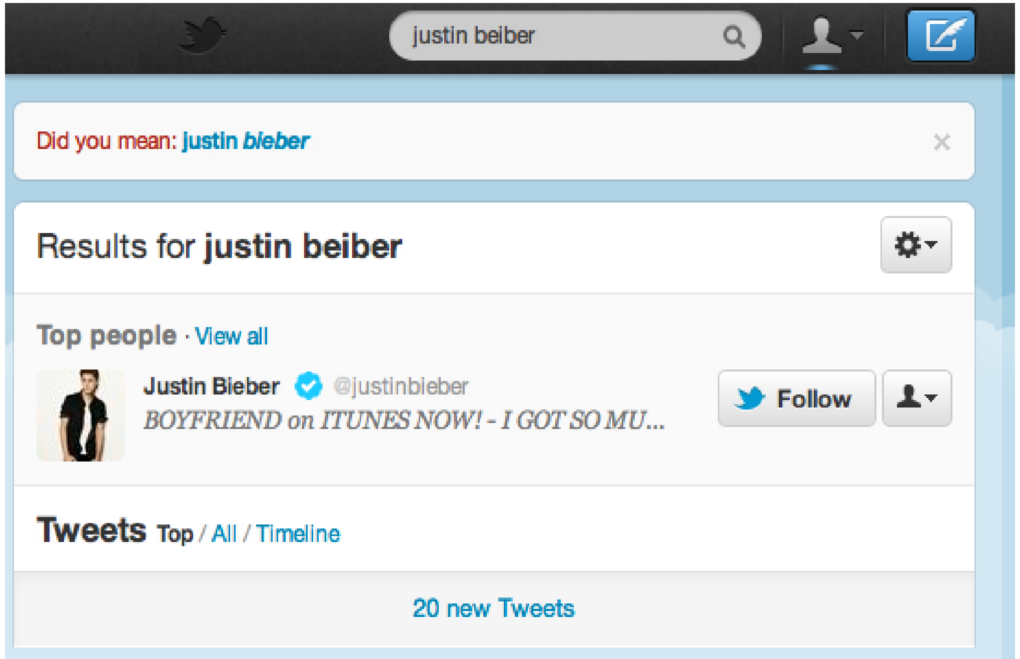
- Motivation & Background
- Contributions
- Real-Time Query Suggestion
- First Solution
- Second Solution
- Future work
- Conclusion
- Discussion

AGENDA

- Motivation & Background
- Contributions
- Real-Time Query Suggestion
- First Solution
- Second Solution
- Future work
- Conclusion
- Discussion

Motivation & Background

- Develop a real time query suggestion system



The figures are taken from https://blog.twitter.com/2012/related-queries-and-spelling-corrections-search

Motivation & Background

- Develop a real time query suggestion system
- Example:
  - When Marissa Mayer was in Google

Motivation & Background

- Develop a real time query suggestion system
- Example:
  - When Marissa Mayer was in Google

  - When Marissa Mayer scheduled for Yahoo CEO

Motivation & Background

Goal

- Provide Relevant Related Query Suggestions within 10 Minutes of Major Events

AGENDA

- Motivation & Background
- Contributions
- Real-Time Query Suggestion
- First Solution
- Second Solution
- Future work
- Conclusion
- Discussion

Contributions

- Introduce real time related query suggestion problem
- Explain two solutions:
  - First Solution: using Hadoop
  - Second Solution: using in memory processing engine
- Suggest future work to reduce the gap between **big data** and **fast data**

AGENDA

- Motivation & Background
- Contributions
- Real-Time Query Suggestion
- First Solution
- Second Solution
- Future work
- Conclusion
- Discussion

Real Time Query Suggestion

- **Good** related query suggestions provide:
  - Topicality
  - Temporality
- Topicality: capture same topic
- Temporality: capture temporal connection
  - #SCOTUS suggestions: healthcare and #aca
  - Marissa Mayer example

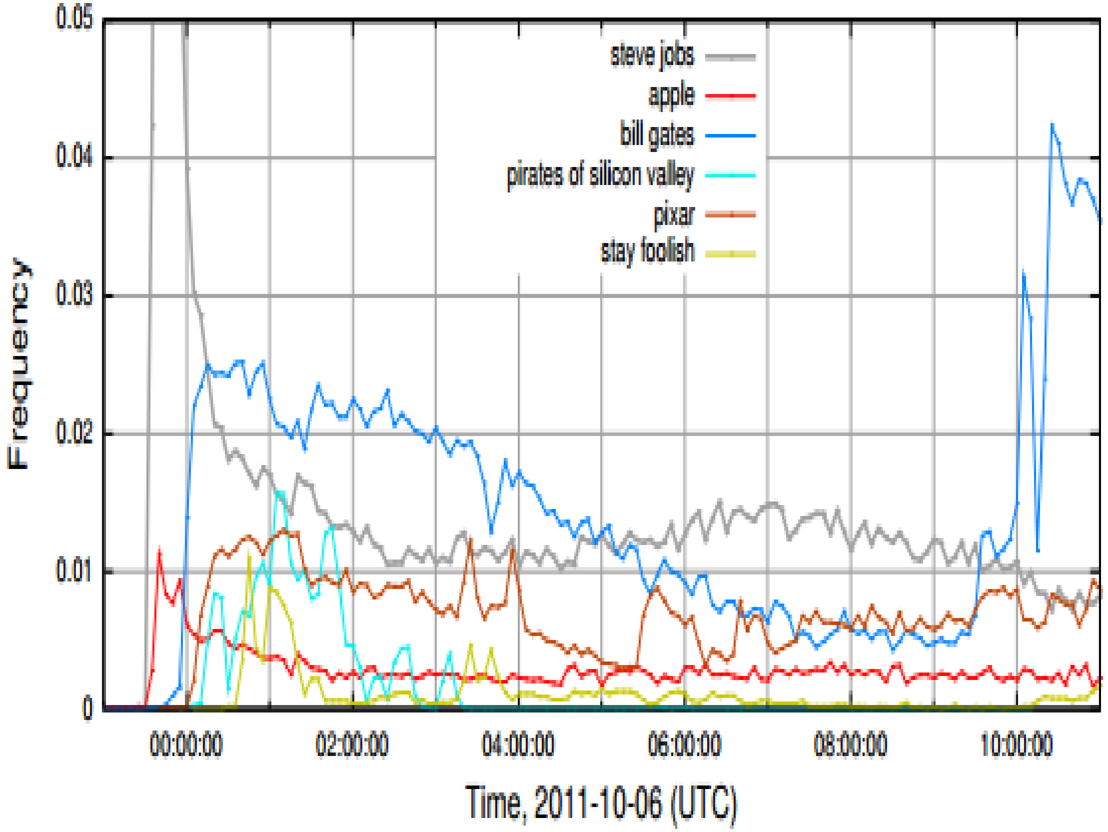
Real Time Query Suggestion

- Time constrain to include news breaks

- When is the best time to make suggestions ?
  - Too early: No enough evidences
  - Too late: User experience

Real Time Query Suggestion

- Steve Jobs died:
  - “steve jobs” becomes 15%
  - “stay foolish” and “apple” ↑
- Window size:
  - 10 minutes



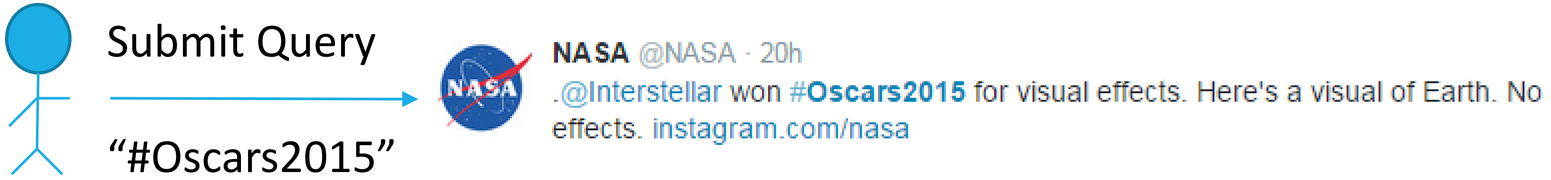
The figure is taken from the paper “ Fast Data in the Era of Big Data: Twitter’s Real-Time Related Query Suggestion Architecture ”

Real Time Query Suggestion Algorithm

- Query A and B are seen in same context
    - A and B are related queries
- Context can be:
    - User search session
    - Tweet itself

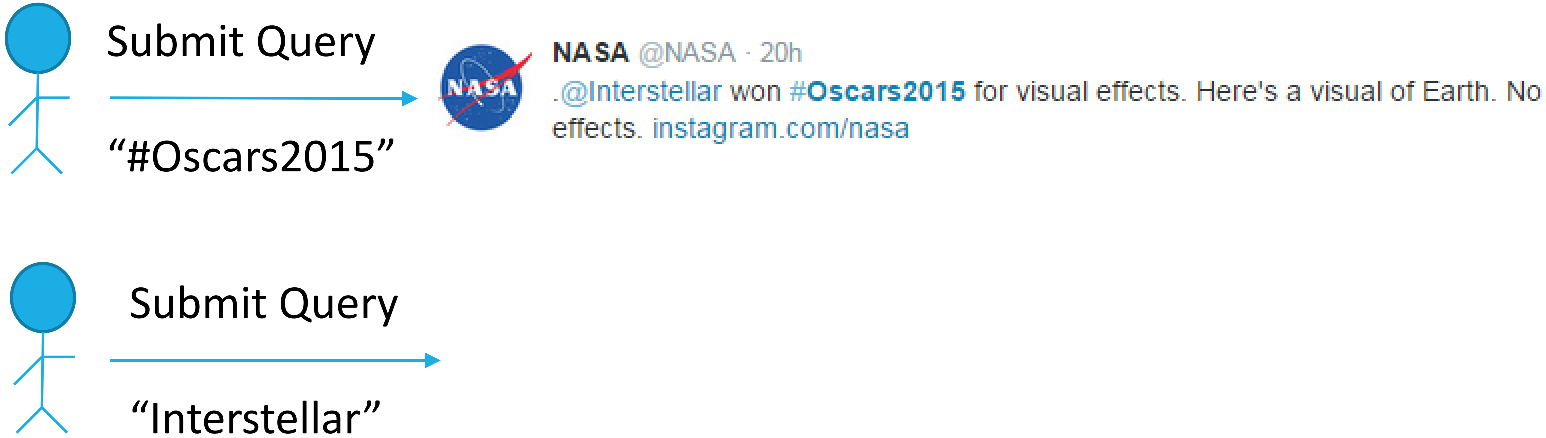
Real Time Query Suggestion Algorithm

- User search session



Real Time Query Suggestion Algorithm

- User search session



- Tweet: Terms in the tweet are related

Real Time Query Suggestion Algorithm

- A is before B in time
  - B is interested to users who liked A
- A and B are similar and B has more results
  - B is spelling correction of A
- Measures relatedness between query A and B
- Decays measurement with time

AGENDA

- Motivation & Background
- Contributions
- Real-Time Query Suggestion
- First Solution
- Second Solution
- Future work
- Conclusion
- Discussion

First Solution (Hadoop)

First

Why to use Hadoop ?!

First Solution (Hadoop)

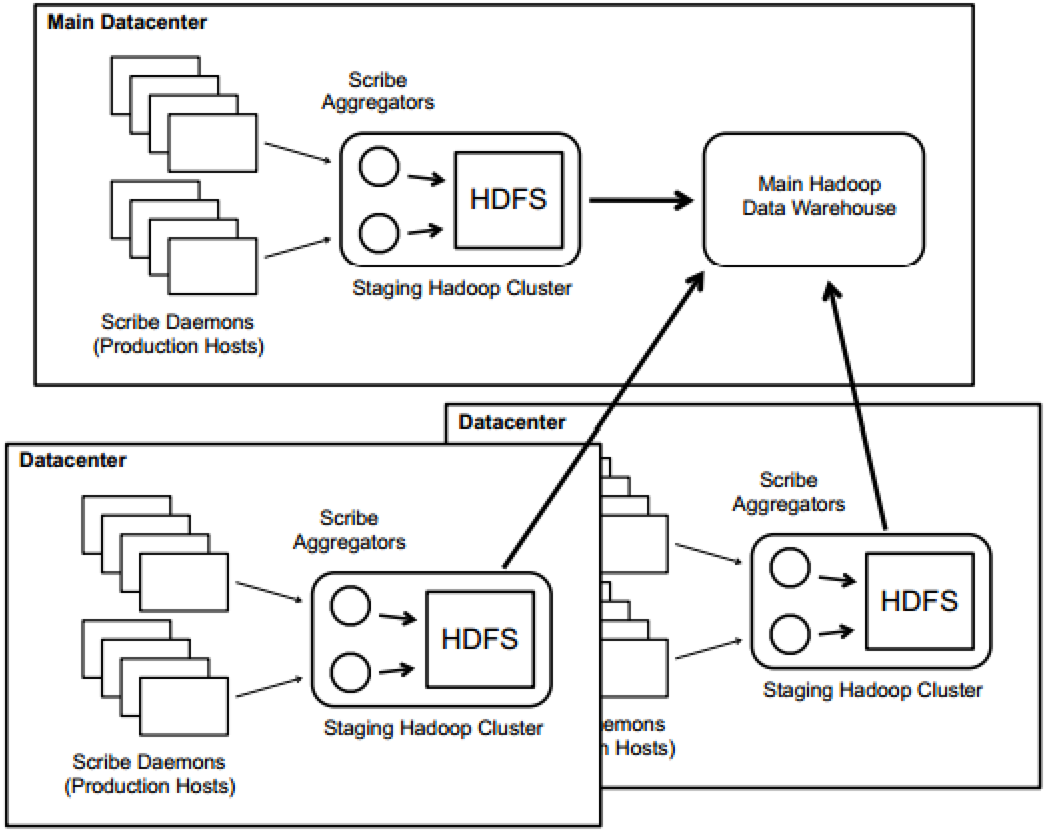
- Twitter has robust and production Hadoop cluster
- Twitter has incorporated components on top of Hadoop
  - Pig, Hive, ZooKeeper and Vertica
- Use Oink pig flow manager
- The first version was developed in Pig and Java UDF

First Solution (Hadoop)

- Using pig script to:
  - Aggregate user search session
  - Compute term and co-occurrence statistics
  - Rank related queries and spelling correction
- Frontend loads outputs and serves requests
- Unacceptable latency (several hours!)

First Solution (Hadoop)

- Two bottlenecks
  - Log Import
  - Hadoop



The figure is taken from the paper " Fast Data in the Era of Big Data: Twitter's Real-Time Related Query Suggestion Architecture "

First Solution (Hadoop)

- Hadoop delay
  - Resource contention
  - Mapreduce jobs took 15-20 minutes
  - Stragglers

First Solution (Hadoop)

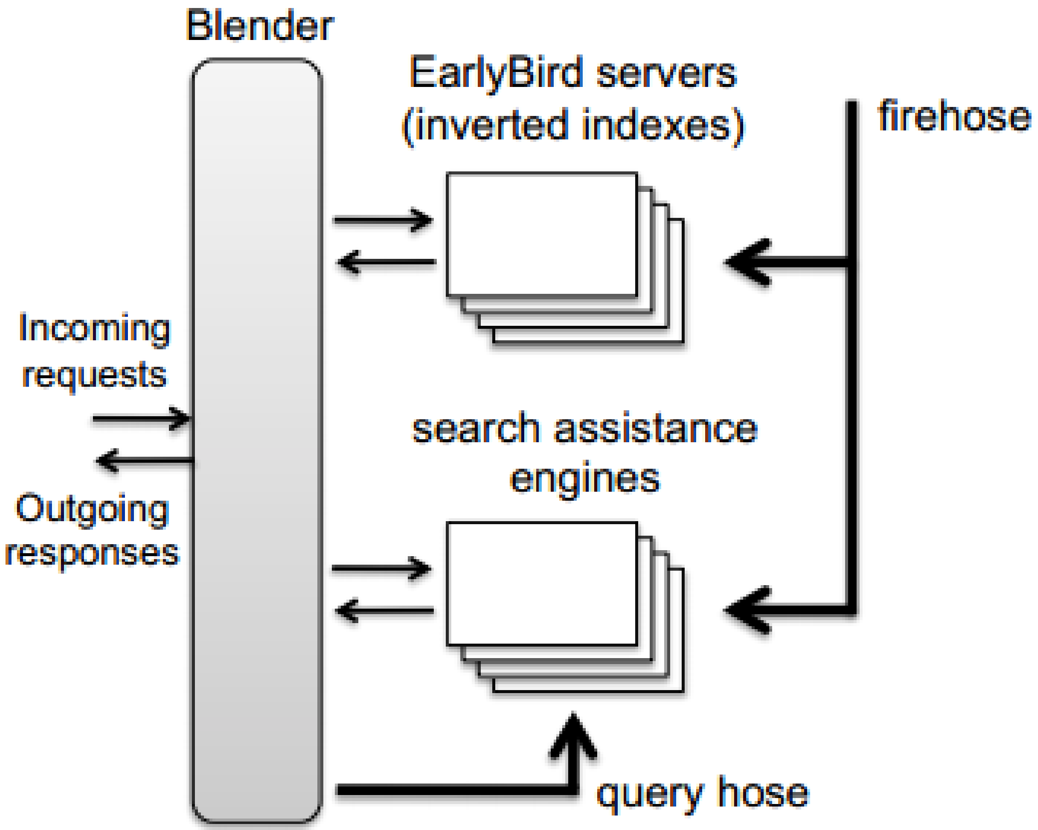
- Hadoop delay
  - Resource contention
  - Mapreduce jobs took 15-20 minutes
  - Stragglers

Hadoop is not designed for latency sensitive jobs

AGENDA

- Motivation & Background
- Contributions
- Real-Time Query Suggestion
- First Solution
- Second Solution
- Future work
- Conclusion
- Discussion

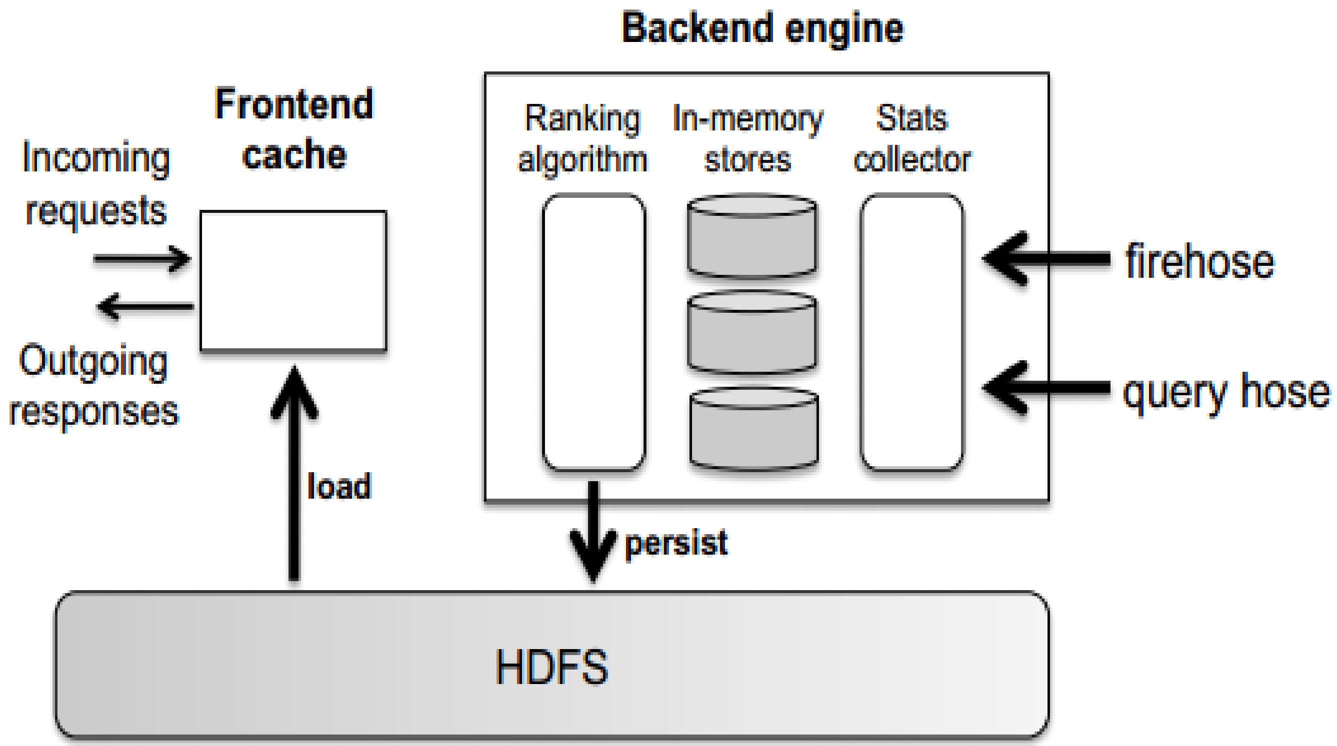
Second Solution



The figure is taken from the paper " Fast Data in the Era of Big Data: Twitter's Real-Time Related Query Suggestion Architecture "

Second Solution

- Every 5 minutes:
  - Results are stored in HDFS
- Cold Restart:
  - Read from HDFS
- Replication



The figure is taken from the paper " Fast Data in the Era of Big Data: Twitter's Real-Time Related Query Suggestion Architecture "

Second Solution (In-Memory Stores)

- Session stores (sliding window):
  - User session: Queries and co-occurrence queries
- Query statistics stores:
  - Query statistics and decay weights
- Query co-occurrence statistics stores:
  - Query pairs statistics
  - Store query before\after in user session

Second Solution (Data Flow)

- When new query arrives (Query Path)
  - Update query statistics
  - Add query to sessions store
  - For each previous query in user session & the new query
    - Update query co-occurrence statistics store

Second Solution (Data Flow)

- When new tweet arrives (Tweet Path)
  - Retrieve its n-grams
  - Check if they occurred before as queries
  - Repeat query Path for each query

Second Solution (Data Flow)

- Decay/Prune Cycles
  - Decay all weights periodically
  - Remove queries and co-occurrence queries <= threshold
  - Remove users sessions with no recent activities

Second Solution (Data Flow)

- Ranking Cycles
  - Periodic process to rank queries
  - Uses queries statistics
  - For each query: it generates suggestions

Second Solution (Scalability)

- CPU limitation
  - One server needs to consume query hose and fire hose
  - Turn out not a limitation
- Memory limitation
  - Memory size vs. Coverage

Second Solution (Background Models)

- Previous model limited to temporal coverage
- Solution: Run background process over older data
- For spelling correction:
  - Form pairwise edit distance between all queries
- Results are stored in HDFS
- Frontend cache combines real time & background results

AGENDA

- Motivation & Background
- Contributions
- Real-Time Query Suggestion
- First Solution
- Second Solution
- Future work
- Conclusion
- Discussion

Future Work

- Automatically perform pruning when memory is needed
- Single unified data platform to deal with real time and slower moving suggestions (fast data + big data)

AGENDA

- Motivation & Background
- Contributions
- Real-Time Query Suggestion
- First Solution
- Second Solution
- Future work
- Conclusion
- Discussion

Conclusion

- The paper proposed two solutions for real time related query suggestion
- The first solution was using Hadoop
- The second solution was using in memory approach

Thank you ☺
Any Questions

Discussion

- No experimental results?
- Memory size vs. coverage trade off, how to reduce the gap?
  - A distributed in memory system? (challenges)
- How to decide automatically which data to prune?
- Would sampling help to solve log import bottleneck in first solution ? How ?
- How to use other information like click graph with in memory structures to enhance the ranking?