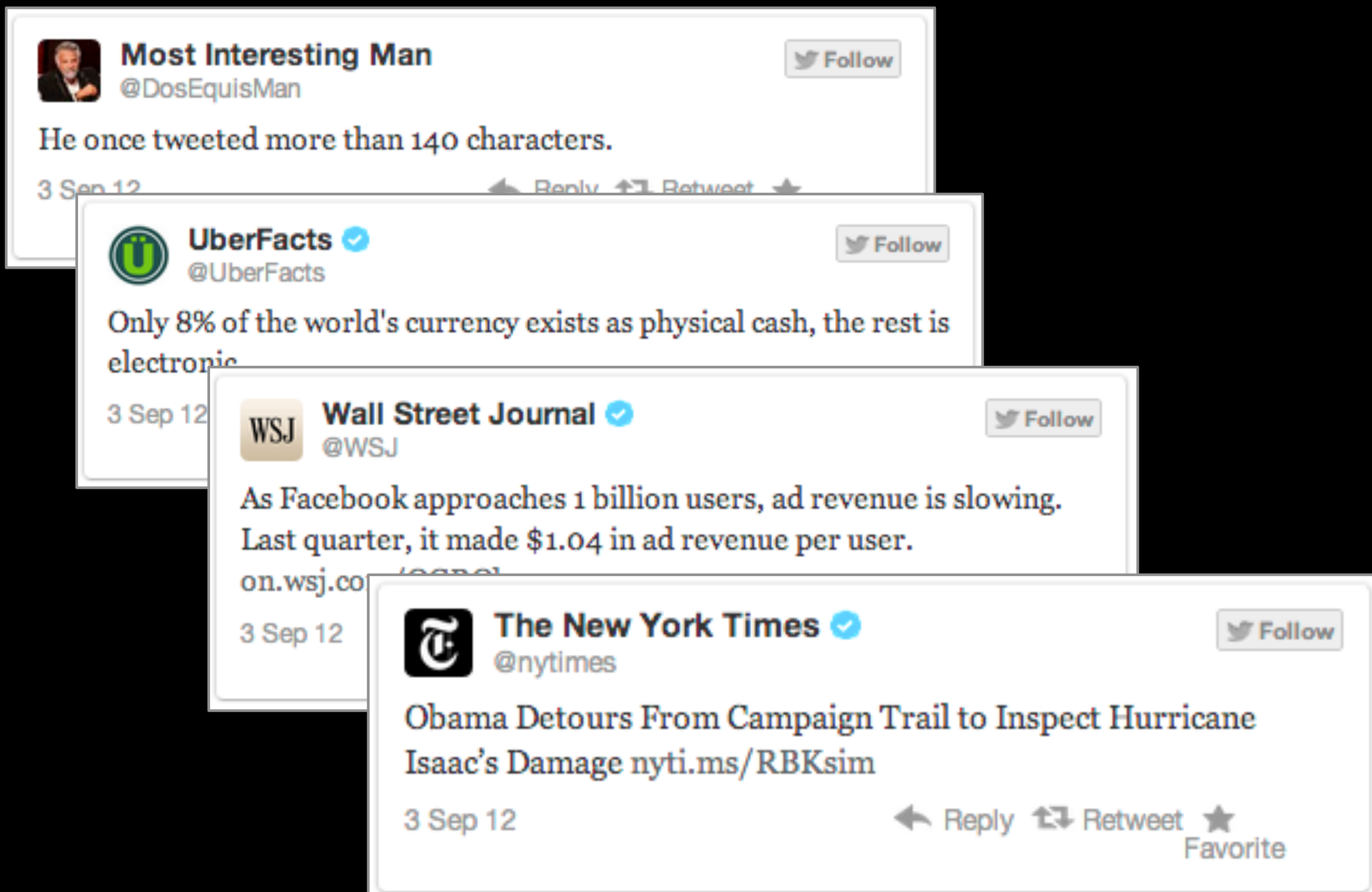
TREND AND EVENT DETECTION IN SOCIAL STREAMS

Kostas Tsioutsiouliklis
@kostas
September 2012

Outline

- Trending topic detection
  1) Simple counting
  2) Chi-square test
  3) Topic-specific models
- Event detection
  1) Online clustering
  2) Online clustering using MinHash
- Natural language processing for tweets

1st approach: Simple Counting



- >400M tweets per day, or >4600 tweets per sec

Tokenization, phrase extraction

- What is a topic?

- n-gram
  - Simple, low precision/high recall, large space
- Dictionary-based phrases (wikipedia entries, named entities)
  - Simple, high precision/low recall, stale
- Noun-phrases (NP) extracted via part-of-speech tagging
  - Difficult for short texts

Term frequencies

- Tokenize text and count term frequencies
- Assume, for now, unigrams

Term	Count
Obama	1
Detours	1
…	…
Hurricane	1
…	…

Term	Count
Obama	1
Detours	1
…	…
Hurricane	2
…	…

Term frequencies

- Periodically, every few minutes, or every certain number of tweets, sort terms by decreasing frequency

Term	Count
the	2,000,000
a	1,200,000
is	800,000
of	600,000
…	…

- Problem: Stopwords dominate
- Solution: Remove them
- Problem: Common, not trending, words dominate

Background model

- Establish baseline of expected frequencies based on history
- Compare current frequencies to baseline

Term	Past freq (per time unit)	Present freq (per time unit)	Ratio
Hurricane	10	200	20
Giants	50	500	10
Obama	1,000	1,200	1.2
Bieber	20,000	23,000	1.15

Limitations of simple frequencies

- Works great, in general, but low past frequency terms could get artificially inflated
  - If a term is new, the past frequency is 0
    - usually memes: #ReplaceMovieTitleWithFavoriteDrink, #BestReasonToStayHome, but sometimes goldmines
  - Low vs. high frequencies
    - What is more trending: A term that goes from 20 to 25, or one that goes from 20,000 to 25,000?
- Solutions:
  - Sandboxing, thresholds, smoothing.
    - Drawback: Latency.
  - Need a better statistic, than simple ratio, to capture relative growth

2nd approach: Fit of a distribution

- Assume that terms are drawn independently at random from a static distribution, where each term has a fixed prior likelihood of being selected (multinomial distribution).

- The following samples are observed:

- What is the probability that they were drawn from the theoretical distribution?

2^{nd} approach: Fit of a distribution

- A common test for such a goodness-of-fit experiment is the chi-squared test.

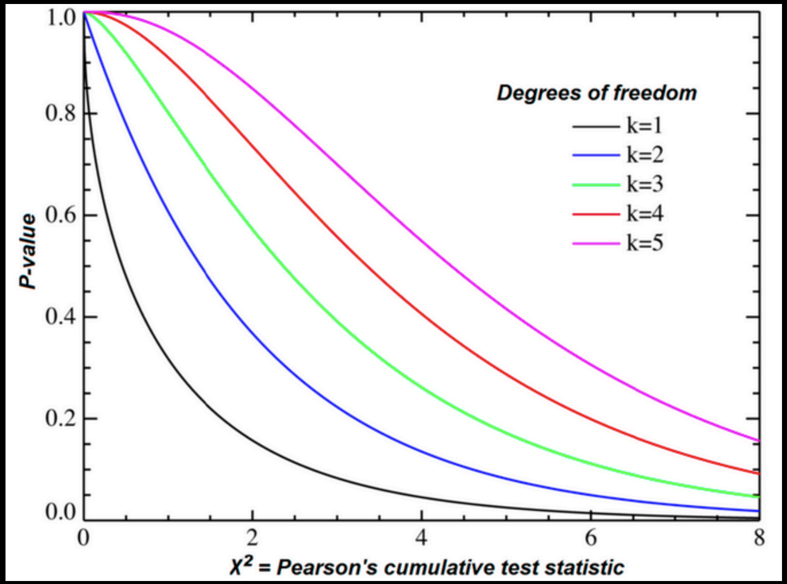
	Expected	Observed
Category 1	70	68
Category 2	30	32
Total	100	100

$$\chi^2 = \sum \frac{(O-E)^2}{E}$$

$$\chi^2 = \frac{(68-70)^2}{70} + \frac{(32-30)^2}{30} = 0.19$$

2^{nd} approach: Fit of a distribution

- A chi-square value of 0.19 corresponds to a p-value of 0.663.
  - Not statistically significant, because p-value > 0.05
  - Null hypothesis is not rejected.



(Graph from Wikipedia)

2nd approach: Chi-square test

- For trend detection, use the chi-square value to determine trendiness.
- Example
  - Of N (large) total past terms, 20 where "Hurricane". Of N present terms, 30 are "Hurricane". 20 is the expected frequency and 30 is the observed frequency.
  - O > E, and the chi-square value is:

$$\chi^2 = \frac{(30-20)^2}{20} + \frac{((N-30)-(N-20))^2}{N} = \frac{10^2}{20} + \frac{10^2}{N} \approx 5$$

  - If the frequency of "iPhone" goes from 40 to 60, O > E, and:

$$\chi^2 \approx \frac{(60-40)^2}{40} = 10$$

  - So 2nd term is more trending.

Using the chi square score

- In a nutshell:
  - If (Observed > Expected) then the trend score is equal to:

    $$\frac{(O-E)^2}{E}$$

    else 0.
- What if E=0?
  - Add-one smoothing.

    $$\frac{((O+1)-(E+1))^2}{E+1} = \frac{(O-E)^2}{E+1}$$

- If low frequencies still dominate, use thresholds or Yates's correction:

  $$\frac{(|O-E|-0.5)^2}{E}$$

3rd approach: Per-topic models

- Previous assumption: single static multinomial distribution, where samples are independent.
- But topic frequencies follow time series. For example, they are periodic.
  - "Good morning" has higher frequency every morning.
  - #FF (follow Friday) has higher frequency every Friday.
- We can estimate what the expected current frequency should be using multiple features from the past.

3^{rd} approach: Per-topic models

- [H.R. Varian and Choi] tried to predict sales using a seasonal autoregressive (AR) model. Let S_t be the sales S at time t. Then:
- $\log(S_t) \sim \log(S_{t-1}) + \log(S_{t-12}) + x_t + e_t$

  where S_{t-1}, S_{t-12} are sales 1 month ago and 12 months ago, x_t is the number of queries for this item, and e_t is an error term.
- Linear regression estimation:

  $$\log(S_t) \sim w_1 \log(S_{t-1}) + w_2 \log(S_{t-12}) + w_3 x_t + w_4 e_t$$

3rd approach: Per-topic models

- This translates directly to the trend detection problem:
  $$\log(f_t) \sim \log(f_{t-24\,hours}) + \log(f_{t-1\,week}) + e_t$$

- Pros/cons:
  - Richer feature set
  - Harder to compute and update.
- Intermediate approach: Maintain more statistics than just expected value, e.g. periodicity, standard deviation, and model them.

Event detection

- Trend detection looks at topics in isolation.
- But topics are not independent, e.g.
  – Bratt Pitt to marry Angelina Jolie
  – Bill Clinton visits Haiti
- Need to cluster trends with their respective tweets.
- These clusters can provide rich context about trends, including links.

1^{st} approach: Online clustering

- In 1998 NIST initiated the Topic Detection and Tracking (TDT) Project.
- Goal: To discover the topical structure of a news stream, including event detection and event tracking.
- Corpus: 15,836 news stories containing 25 events.
- Definition of event very broad: Something (non-trivial) happening in a certain place at a certain time.
- Several papers (e.g. [Allan et al.], [Yang et al.]) came out of TDT.

TF-IDF weighting scheme

- Let d be a document in a corpus of N documents.
- Let t be a term in document d.
- Then TF-IDF(t,d) is defined as:

$$TFIDF(t,d) = (1 + \log(TF(t,d))) \times IDF(t)$$

Where TF(t,d) is the term frequency of t in d, and IDF(t) is the inverse document frequency, i.e. N / n(t), where n(t) is the number of documents containing the term.

- A document can be represented by a vector of all its term weights. The weights are normalized (L2 norm) and often only the top k terms are kept.
- Similar function for clusters.

1st approach: Online clustering

- Both [J. Allan] and [Y. Yang] use similar one-pass on-line clustering algorithms:
- For each new article:
  - Find the cosine similarity of its TFIDF vector with that of each cluster.
  - Assign the document to the cluster with highest similarity if above a certain threshold.
  - If all similarities are below the threshold, create a new cluster with that document.

1st approach: Online clustering

Challenges:

- Online IDF computation: IDF is unknown unless all documents are processed.
  - Solution 1: Compute IDF from a similar corpus.
  - Solution 2: Compute IDF from the first few documents. This is often sufficient.
- Updating representative vector for a cluster
  - Easy
- Finding which cluster a document should be assigned to
  - Linear in the number of clusters.
  - Too slow for a rate of thousands of tweets per second.
    - Solution: min-hash, locality sensitive hashing

2^{nd} approach: MinHash

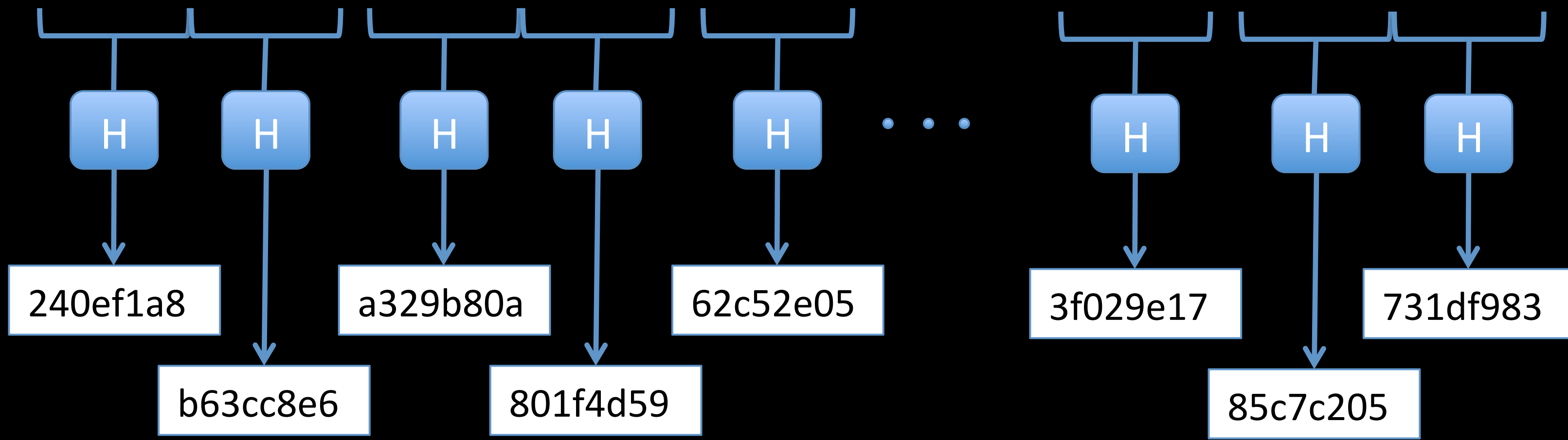
- MinHash is an algorithm for fast set similarity detection
- Outline
  - [A. Broder] for duplicate detection
  - [A. Das, et al.] for Google news article clustering
  - [S. Petrovic] event detection with application to Twitter

2nd approach: MinHash

- The main idea behind MinHash is the use of multiple hash functions over tweets to find similar ones.
- Hash functions are fast to compute and produce small signatures.
  - If two documents have many common signatures, then they are similar.
- Hash functions define random permutation over sets.

Duplicate and near-duplicate detection

"Obama Detours From Campaign Trail to Inspect Hurricane Isaac's Damage"



Sorting the signatures of the hash function leads to a random permutation of all tokens:

Obama → 240ef1a8

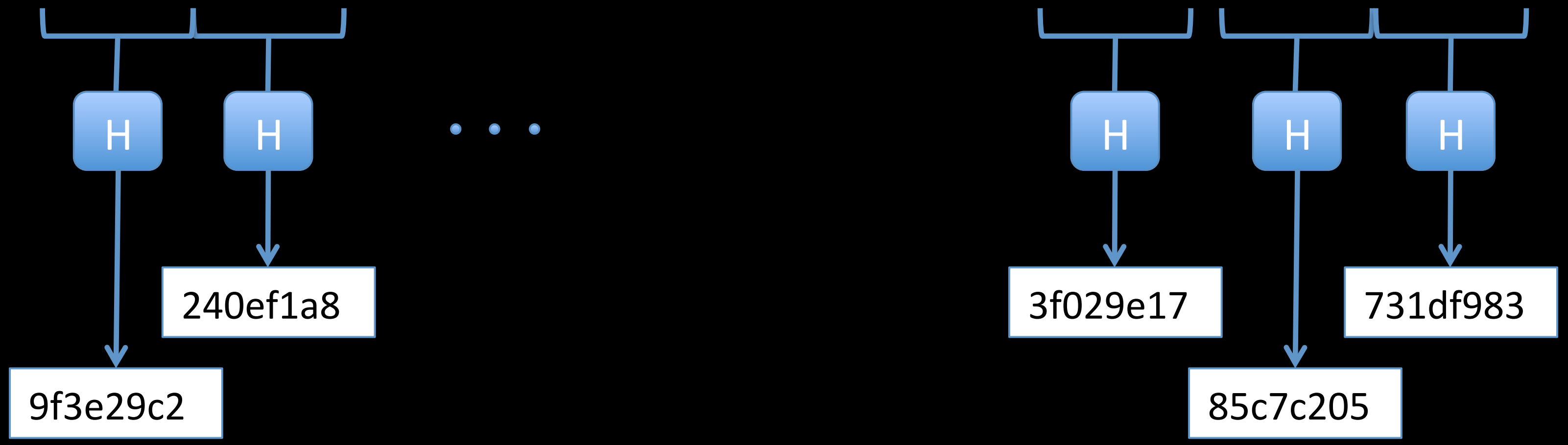
Hurricane → 3f029e17

Trail → 62c52e05

…

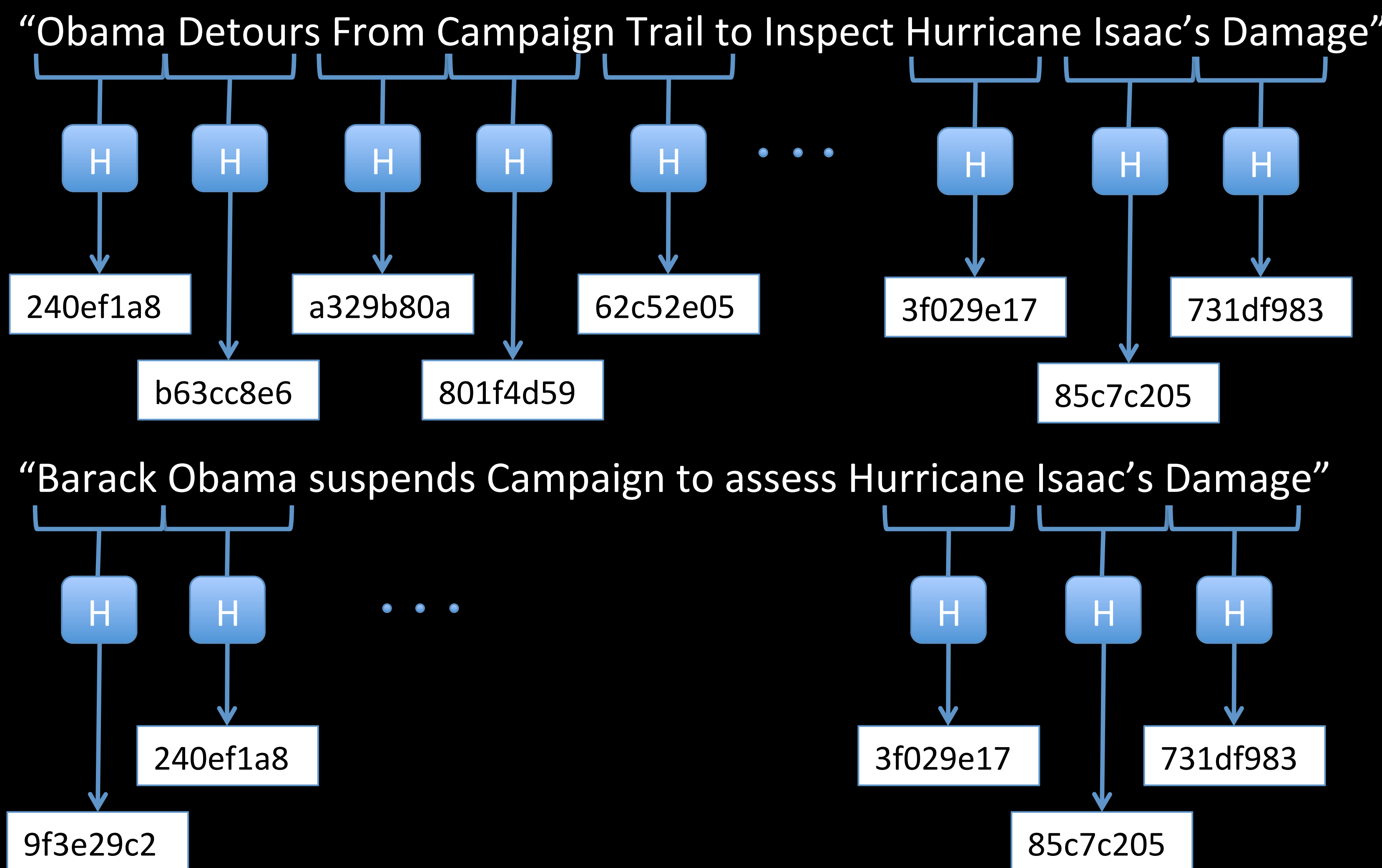
Duplicate and near-duplicate detection

"Barack Obama suspends Campaign to assess Hurricane Isaac's Damage"

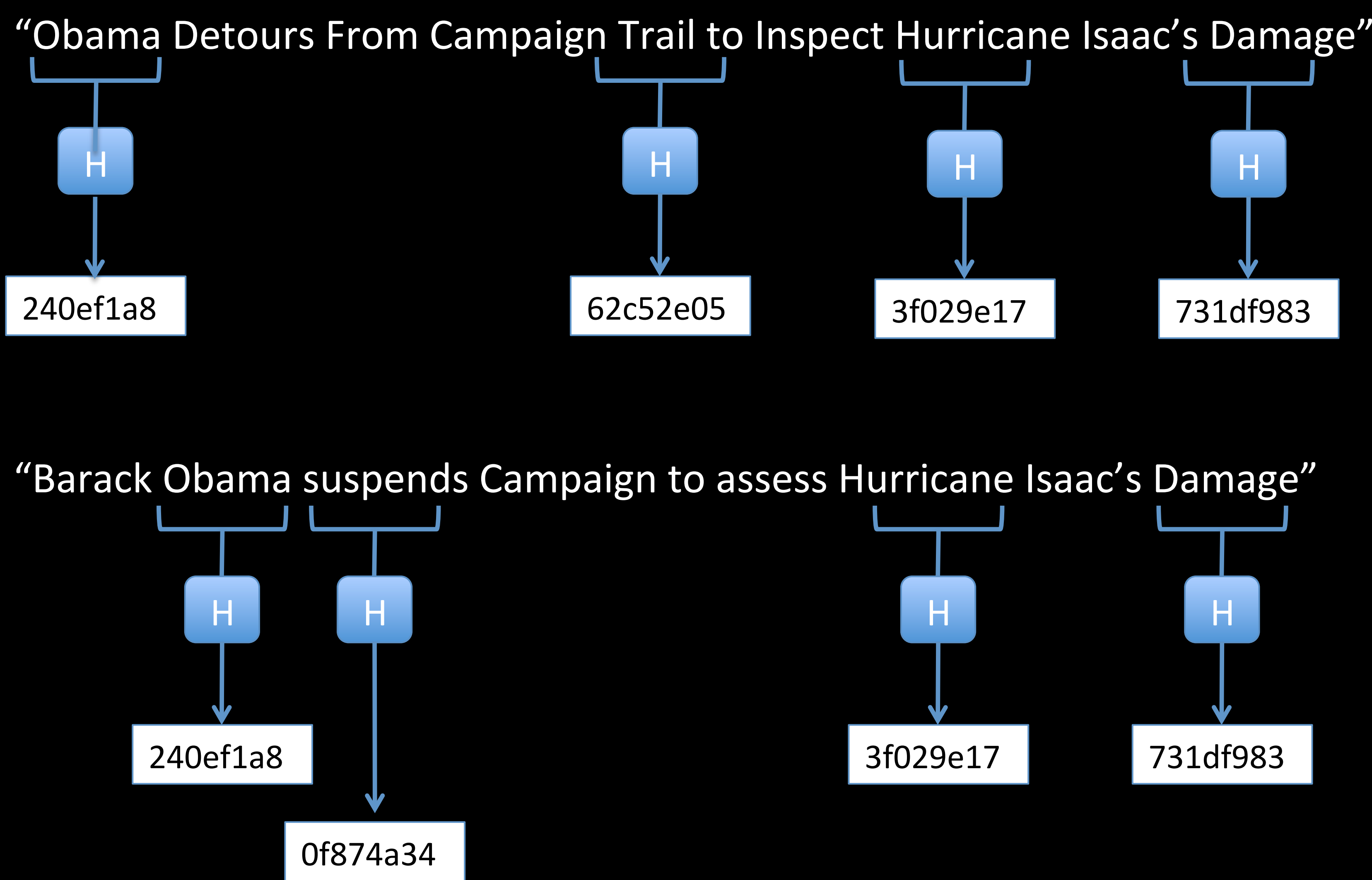


- What is the probability that the element with the minimum signature appears in both texts?
  - Assume the union of all tokens.
    Obama Detours From Campaign Trail To Inspect Hurricane Isaac's Damage Barack Suspends Assess
  - Sort by signatures.
    Obama Hurricane Trail Damage Campaign Isaacs Barack From Detours Assess Inspects To Suspends
  - Probability is equal to the Jaccard coefficient: |Intersection| / |Union|

Duplicate and near-duplicate detection



Duplicate and near-duplicate detection



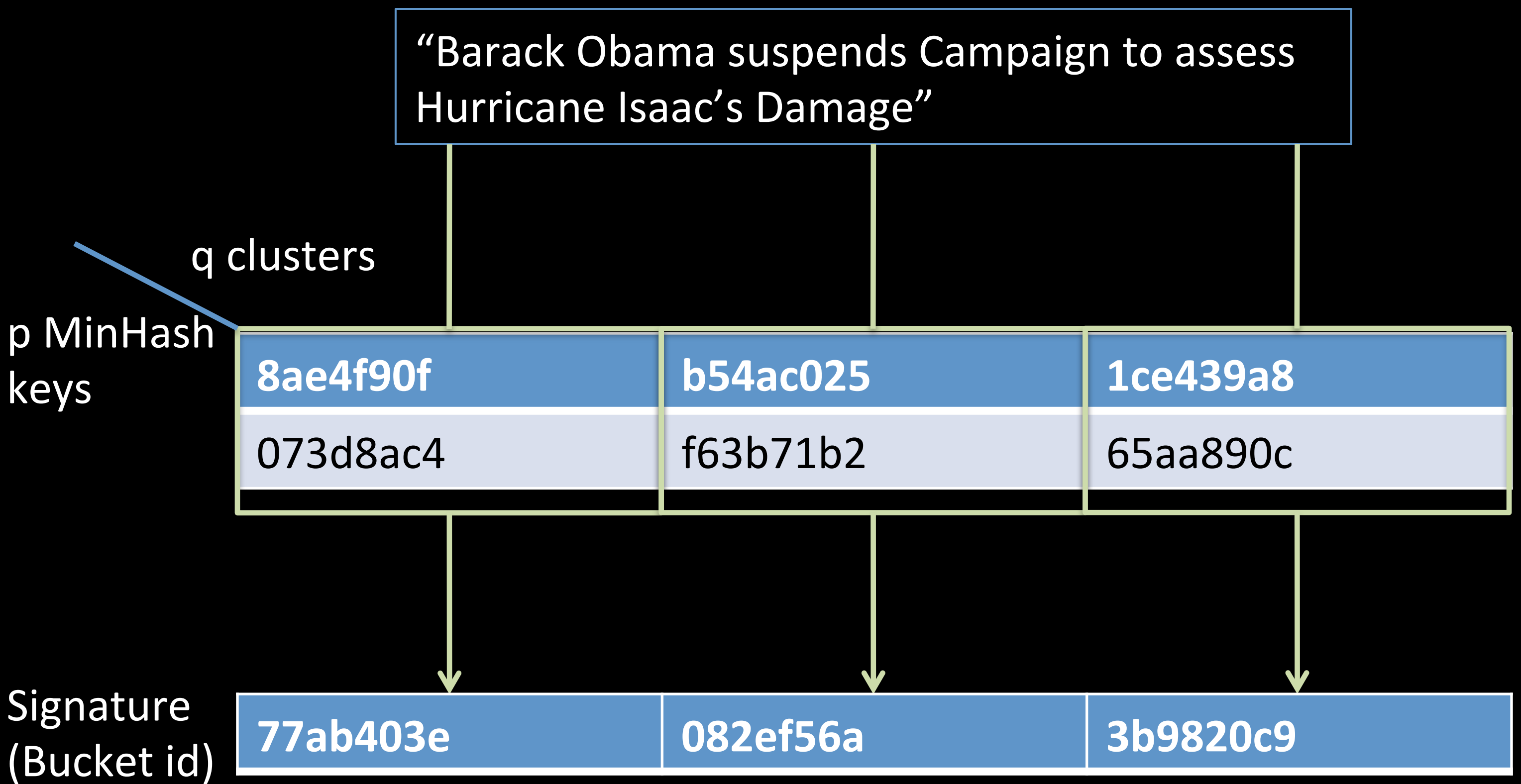
Locality Sensitive Hashing (LSH)

- Using one hash function corresponds to a probabilistic clustering algorithm where two tweets u,v end up in the same cluster with probability equal to their item-set overlap similarity S(u,v)
- Using p hash functions leads to a probabilistic clustering algorithm where u,v end up in the same cluster with probability $S(u,v)^p$
  - Concatenate their signatures to generate a new signature.

LSH in Online Clustering

- With more hash functions, clusters are more refined.
  - High precision, low recall.
- To increase recall, repeat the above process q times and assign each tweet to q clusters.
- To speed up the process, pre-compute p x q seeds.
- Typical values for p are 2-4, and for q are 10-20.

LSH in Online Clustering



LSH in Online Clustering

```
1)  Set p,q, e.g. p=2, q=10.
2)  Create a pxq table S of seeds by taking the checksum
    of any pxq integers (not necessarily random). These
    seeds will be used in subsequent hash functions.
3)  For each tweet
       For each column q_j
          For each row p_i
              Set seed s = S(p_i,q_j)
              For each token in the tweet
                 Find the minimum MD5 checksum using s
          Concatenate the seeds and get their MD5 checksum.
          Assign the tweet to the bucket with id equal to
          the checksum.
4)  Iterate through all buckets to extract the clusters of
    tweets.
```

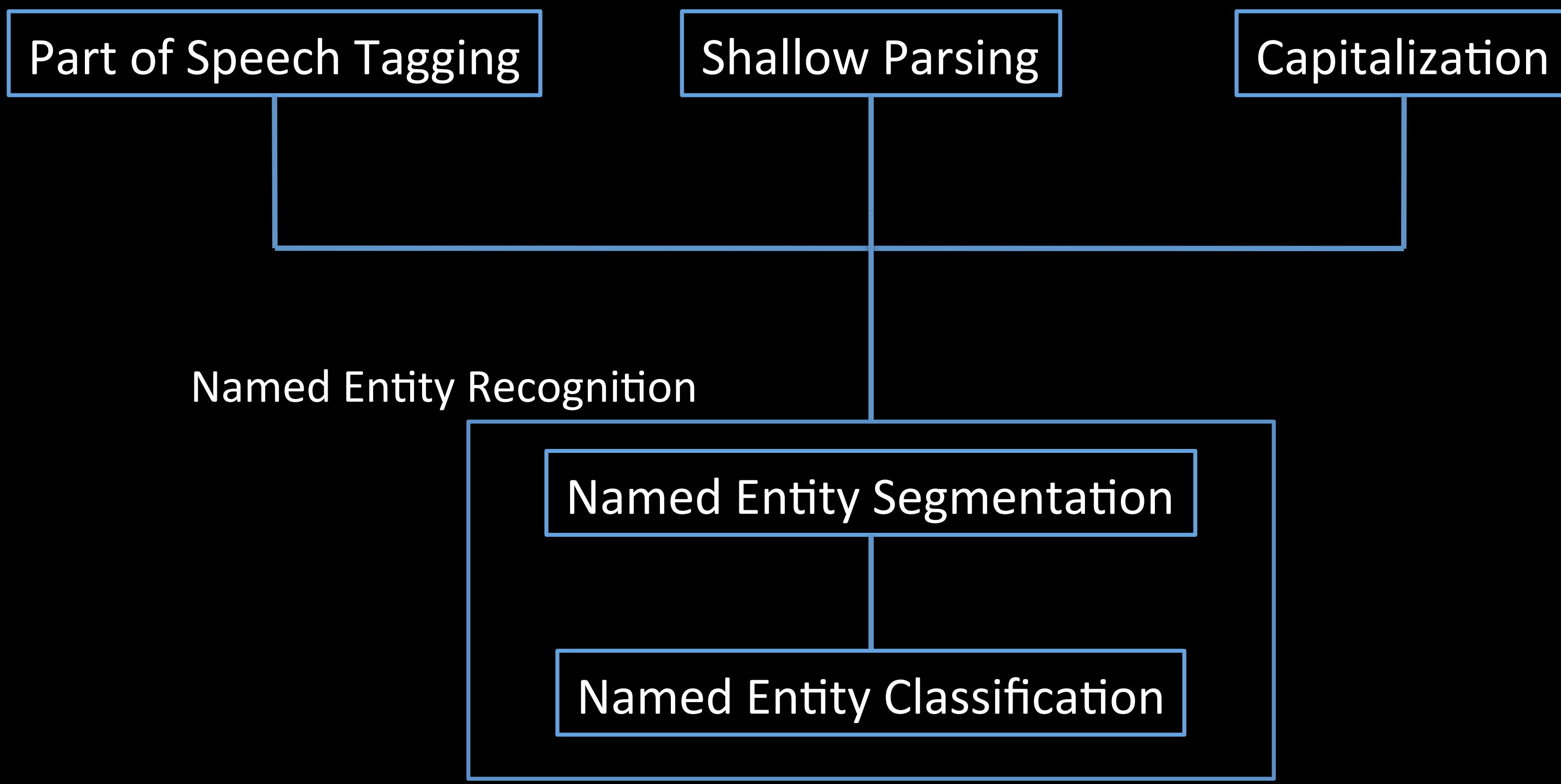
Online clustering

Challenges:

- The number of clusters is unbounded.
  - Need to fine-tune thresholds.
- Dominant subtopics. Topic splits into multiple clusters.
  - Balance between p, q, and length of signatures
- Spam and noise.
  - Train classifiers to remove.
  - Background model over clusters.

NLP for tweets

- Brief overview of [A. Ritter, et al.]



NLP for tweets

- Part of Speech (POS) Tagging
  - Baseline: Manually label 800 tweets. Then, assign each word its most frequent tag and each out of vocabulary (OOV) the most common POS tag (NNP). Accuracy: 0.76
  - Stanford POS tagger: Accuracy: 0.8 using the Penn Treebank WSJ (PTB).
  - T-POS: Accuracy: 0.883, using PTB and Twitter-specific tags, clusters of synonyms, and model using conditional random fields.

NLP for tweets

- Shallow Parsing
  - Identifying non-recursive phrases, e.g. noun phrases, verb phrases, and prepositional phrases.
  - Used T-POS and its features to outperform against off-the-shelf chunker.
- Capitalization
  - Capitalization classifier whether or not a tweet is “informatively” capitalized.
  - Trained a Support Vector Machine (SVM) with features such as:
    - the fraction of capitalized words
    - fraction of words that mismatch compared to a dictionary of lowercase/uppercase words
    - number of times “I’ is capitalized
  - Outperforms the majority baseline.

NLP for tweets

- Named Entity Segmentation
  - Conditional random fields
  - Features included in-domain data (2400 labeled tweets with 34K tokens), POS, chunk, capitalization, dictionaries (including a set of type lists from Freebase). P/R: 0.7/0.6
  - Baseline: Stanford NER (P/R: 0.6/0.35)
- Named Entity Classification
  - Freebase baseline: broad coverage (70%), ambiguous.
  - Model: LabeledLDA where topics are distributed over types according to Freebase.
  - Experiment: 10 popular categories: Person, Geolocation, Company, Product, Facility, TV-show, Movie, Sportsteam, Band, Other. P/R: 0.7/0.6

References

CHI-SQUARE

- R. Swan, J. Allan, Automatic Generation of Overview Timelines, SIGIR 2000

TREND DETECTION

- H.R. Varian, H. Choi, Predicting the Present with Google Trends, Google Research Blog http://googleresearch.blogspot.com/2009/04/predicting-present-with-google-trends.html

ONLINE EVENT DETECTION

- J. Allan, R. Papka, V. Lavrenko, On-line New Event Detection and Tracking, SIGIR 1998
- Y. Yang, T. Pierce, J. Carbonell, A Study on Retrospective and On-Line Event Detection, SIGIR 1998
- S. Petrovic, M. Osborne, V. Lavrenko, Streaming first story detection with application to Twitter, HLT 2010

MINHASH

- A. Andoni, P, Indyk, Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions, Communications of the ACM, 2008
- A. Broder, On the resemeblance and containment of documents, In Compression and Complexity of Sequences, 1997
- A. Broder, M. Charikar, A.M. Frieze, M. Mitzenmacher, Min-wise independent permutations, STOC 1998
- A. Das, M. Datar, A. Garg, S. Rajaram, Google News Personalization: Scalable Online Collaborative Filtering, WWW 2007

NLP

- A. Ritter, S. Clark, M. Etzioni, O. Etzioni, Named Entity Recognition in Tweets: An Experimental Study, EMNLP 2011

OTHER

- R. Bandari, S. Asur, B. Huberman, The Pulse of News in Social Media: Forecasting Popularity, Arxiv preprint arXiv:1202.0332, 2012
- J. Kleinberg, Bursty and Hierarchical Structure in Streams, KDD 2002