NIFTY: A System for Large Scale Information Flow Tracking and Clustering

Caroline Suen, Sandy Huang, Chantat Eksombatchai, Rok Sosic, Jure Leskovec
Standford University

Presentation by- Ankita Joshi

Organization

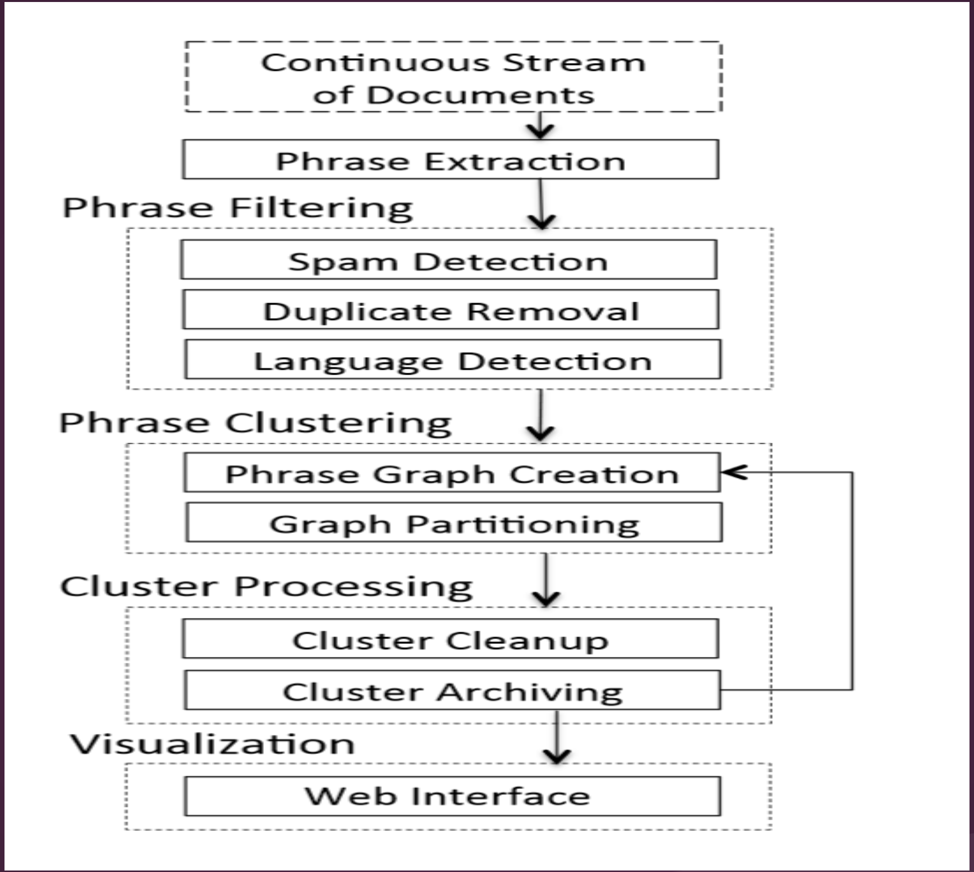
NIFTY: News Information Flow Tracking, Yay!

Meme: Short textual phrases that travel and mutate through the Web. They remain relatively intact as they propagate from website to website.

Peaks[6]: Different participants in online media space shape the dynamics of attention the content receives, across a time period. This observation of a time period shows peaks. (Number of mentions of a topic over time).

Introduction (Need)

1. Online information content has taken increasingly dynamic form -> real time aspect
2. Granularity -> coarse-grained, fine grained
3. Sheer volume and time span of Web data
4. Efficient online incremental algorithms



Overview of The NIFTY Pipeline

Data

1. Data set covers online media activity since August 1, 2008.
2. 6.1 billion documents and 2.8 billion unique quoted phrases
3. Total size of data set : 20 TB
4. They used Spinn3r, a service that monitors 20 million internet resources, to obtain new documents.

Phrase Extraction

Each document D comprises of a news report or a blog post.

Phrases: According to “Memetracker” [1], phrase is a quoted string that occurs in one or more documents.

Document and Phrase Filtering

1. First Pass

Eliminate documents if they are duplicates or URLs are from among the blacklist.

Phrases having less than 3 or more than 30 words are eliminated.

At least 50% of the characters in the phrase should be ASCII characters for the phrase to be considered.

cont..

2. Second pass:

Infrequent phrases: Appearing in fewer than 5 distinct documents

Language Filtering: Phrases which have less than 75% English words.

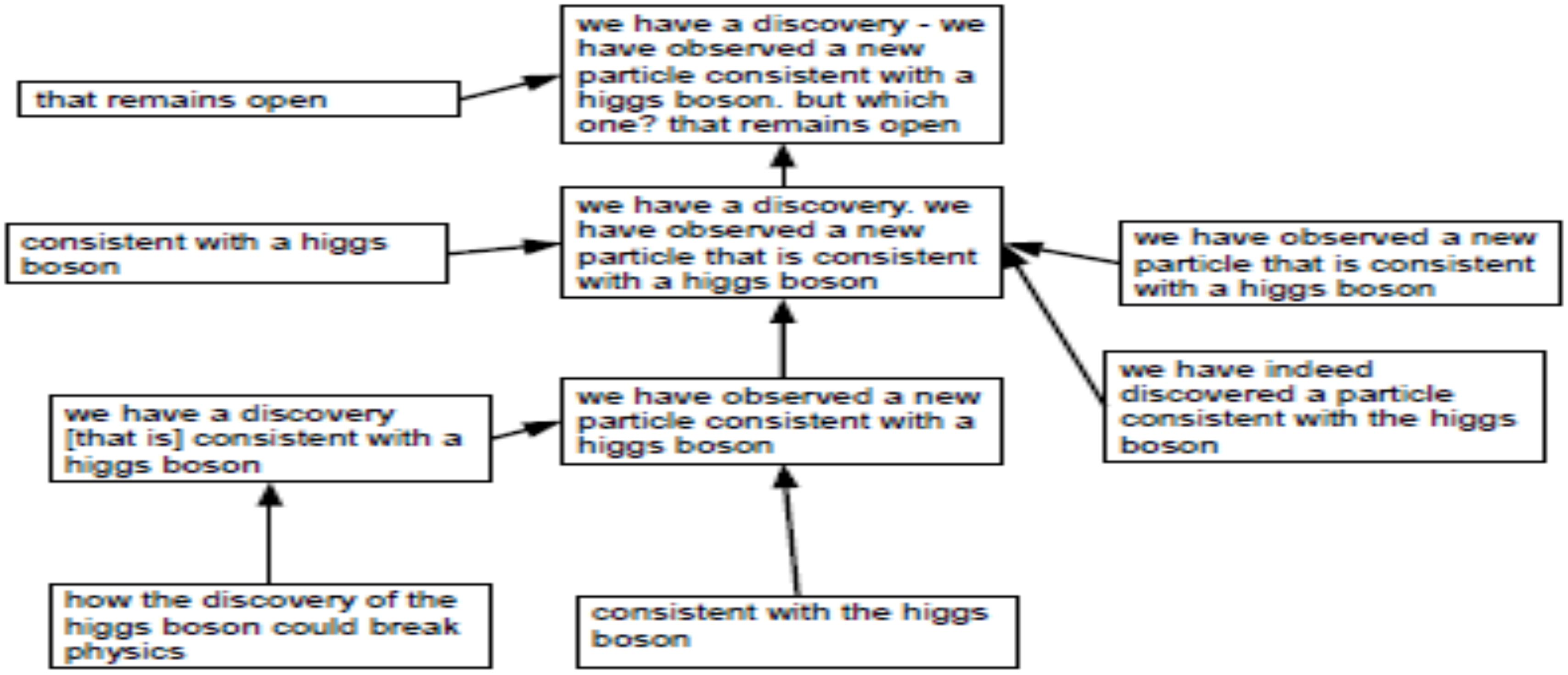
URL to domain name ratio: unique URL/ number of unique domain names > 6

Output of Filtering

Sanitized set of documents D => Document Base

Filtered set of phrases P => Phrase Base

Example of Meme Evolution


that remains open
we have a discovery - we have observed a new particle consistent with a higgs boson. but which one? that remains open
consistent with a higgs boson
we have a discovery. we have observed a new particle that is consistent with a higgs boson
we have observed a new particle that is consistent with a higgs boson
we have indeed discovered a particle consistent with the higgs boson
we have a discovery [that is] consistent with a higgs boson
we have observed a new particle consistent with a higgs boson
how the discovery of the higgs boson could break physics
consistent with the higgs boson

Phrase Clustering

1. Phrase Graph Creation

Purpose : To create a directed weighted acyclic graph.

What is phrase distance??

Levenshtein Distance Algorithm

We use this algorithm to find the substring edit distance.

Minimum number of word insertions, deletions or substitutions needed to transform one string into a substring of another string.

For example[2],

For example, the Levenshtein distance between "kitten" and "sitting" is 3, since the following three edits change one into the other, and there is no way to do it with fewer than three edits:

1. **k**itten → **s**itten (substitution of "s" for "k")
2. sitt**e**n → sitt**i**n (substitution of "i" for "e")
3. sittin → sittin**g** (insertion of "g" at the end).

Edge Creation

1. Pair of nodes(phrases) <-> their substring edit distance
2. Task is to determine whether one is derived from the other
3. An edge is created between two phrases in the graph, if the decision tree returns True.

Algorithm 1 Decision tree to determine if an edge should be created between phrases in the phrase graph

```
Require: Phrases $p_1$ and $p_2$
  (1) Set $l_p$ to the minimum number of words in $p_1$ or $p_2$.
  (2) Compute $s_1$ and $s_2$ by stripping the stop words from $p_1$ and
  $p_2$, respectively.
  (3) Set $l_s$ to the minimum number of words in $s_1$ or $s_2$.
  (4) Set $d$ to the substring edit distance between $s_1$ and $s_2$.
  if $l_s \geq 2$ and $d = 0$ then
      return $True$
  else if $l_p = 4$ and $l_s = 4$ and $d \leq 1$ then
      return $True$
  else if $l_p = 5$ and $l_s > 4$ and $d \leq 1$ then
      return $True$
  else if $l_p = 6$ and $l_s \geq 5$ and $d \leq 1$ then
      return $True$
  else if $l_p > 6$ and $l_s > 3$ and $d \leq 2$ then
      return $True$
  else
      return $False$
```

Speeding up Phrase Graph Creation

NIFTY reduces the number of pair-wise distance calculations by using Locality Sensitive Hashing.

In this paper to find candidate pairs of phrases, they have used Shingling + Min Hashing

Shingling[3]

A k-shingle for a document is a sequence of k characters that appears in the document.

Example: For k=2 doc= abcab

Set of shingles = { ab, bc, ca }

Min Hashing[4]

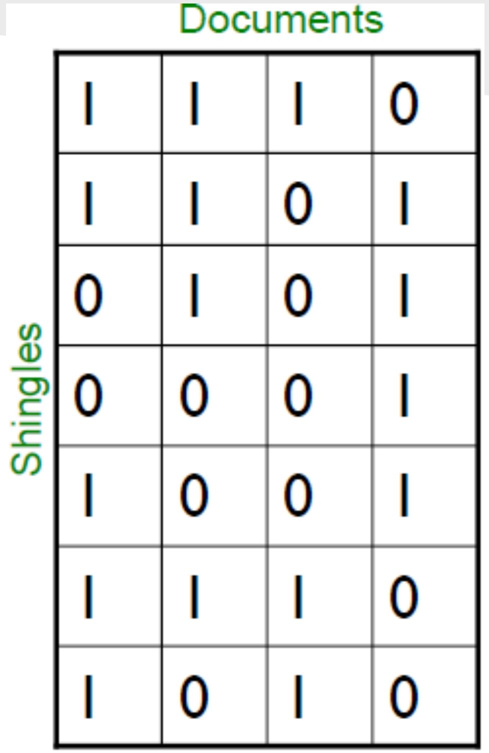
1. We have the set of k-shingles
2. Encode the set into a Boolean Matrix

Shingles => Rows

Documents => Columns

Jaccard Similarity : 3/6

Jaccard Distance : 1-(3/6)



	Documents			
Shingles	1	1	1	0
	1	1	0	1
	0	1	0	1
	0	0	0	1
	1	0	0	1
	1	1	1	0
	1	0	1	0

Min Hashing cont..

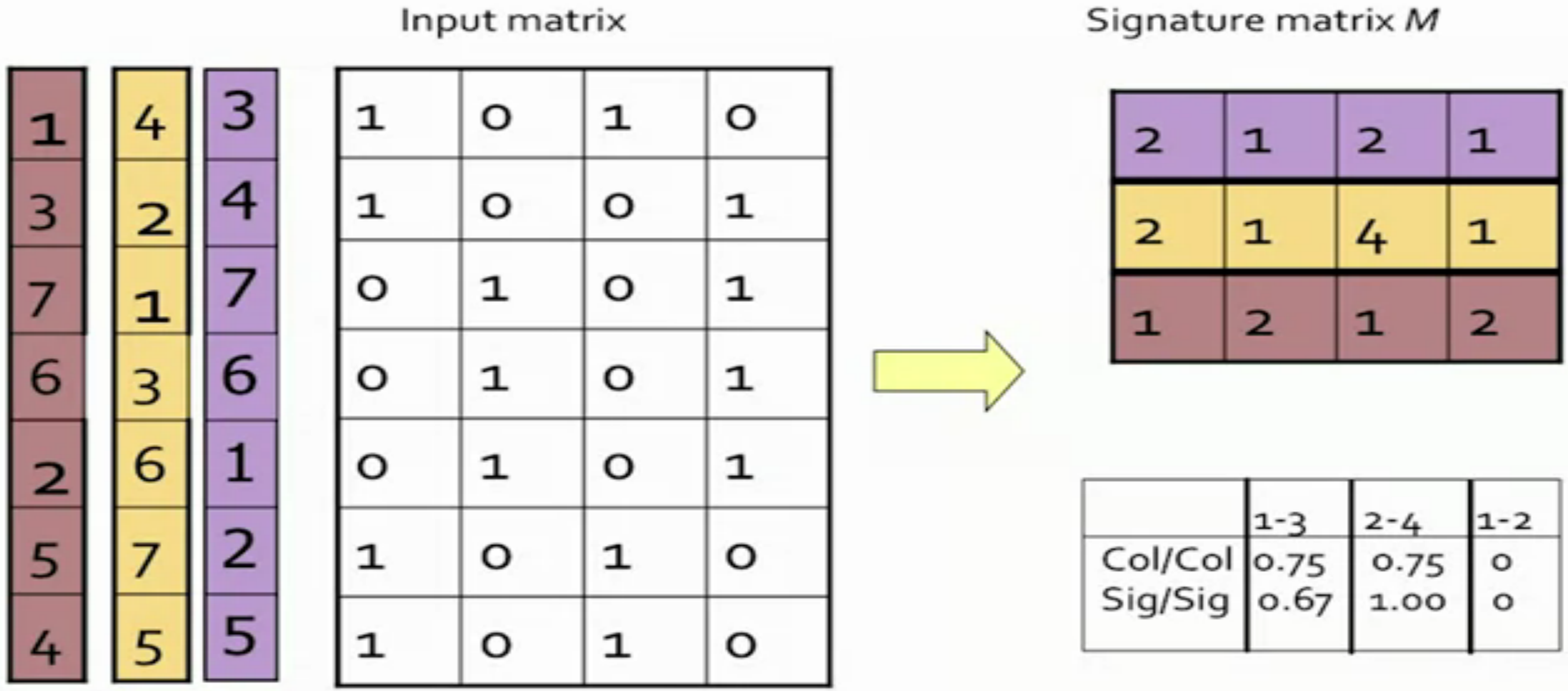
Imagine rows are permuted randomly

minhash function h(C)= number of the first row in which the column has 1

Create a signautre through each column.

Locality Sensitive Hashing of the signatures! => candidate pairs

Example[5]



1	4	3
3	2	4
7	1	7
6	3	6
2	6	1
5	7	2
4	5	5

Input matrix

1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

Signature matrix *M*

2	1	2	1
2	1	4	1
1	2	1	2

	1-3	2-4	1-2
Col/Col	0.75	0.75	0
Sig/Sig	0.67	1.00	0

Locality Sensitive Hashing [6]

From signature matrices hash columns to buckets, and elements of the same bucket become candidate pairs.

FOR NIFTY : 4 - character shingles from phrases => Perform min - hashing => compute phrase signatures => each pair of phrases having a common signature is tested for edge existence.

Edge Weights

$$w(p_s, p_d) = c \cdot \frac{|p_d|}{(D_{edit}(p_s, p_d) + 1) \cdot (T_{peak}(p_s, p_d) + 1)}.$$

Edge from node ps to pd,

| pd | is the number of documents containing phrase pd

D_{edit}(ps,pd) : Substring edit distance between ps and pd

T_{peak}(ps,pd) : Time difference between the first volume peaks between each of the two phrases.

2. Phrase Graph Partitioning

1. Given a weighted directed acyclic graph the goal is to delete a set of edges with a minimum total weight.
2. Start with a working set that includes all root phrases, nodes with outdegree 0.
3. For each node not in the working set, we find the cluster its neighbour is assigned to, then for each cluster we sum up the edge weights for all neighbours in the cluster.The node is attached to the cluster with the largest sum, edges to other clusters are removed.
4. Algorithm terminates when all nodes are in the working set.

Output of partitioning..

Set of clusters referred to as the **cluster base** C.

Cluster Processing

Filtering by Phrase Mutation: Remove all clusters including a single phrase mutation.

Filtering by peaks: Most news follows a predictable popularity cycle with at most 2 peaks. Remove clusters with more than 5 peaks.

Incremental Phrase Clustering

1. Phrase Graph Creation

a. Daily creation of phrase graphs
b. Only consider edges between phrases where at least one phrase is new.
c. Edges to new phrases can be freely added to graph.

Incremental Phrase Clustering cont..

2. Phrase Graph Partitioning

a. Preserve existing clusters by preserving all edges that existed a day before.
b. Edge is selected over the other edges and kept in graph if both its phrases already existed the day before.
c. Only edges of newly added nodes can thus be removed.

Incremental Phrase Clustering cont..

3. Cluster completion and removal

a. *Completed cluster* : Average document frequency within last 3 days is less than 20% of its frequency at its highest peak.
b. No new phrases are added to completed clusters.
c. A cluster is *removed* from the phrase graph, if its highest peak is more than 7 days old.

Visualization

1. Rank clusters by popularity.
2. Each cluster c is assigned a time based popularity score S(c) based on the number of document mentions and cluster correctness using the exponential decay formula:

$$S(c) = \sum_{p \in c} \sum_{t=t_p}^{t_c} \exp\left[-\left\lfloor \frac{t_c - t}{48} \right\rfloor\right] \cdot M_p(t)$$

p is a phrase in c, t is the time of the earliest mention of phrase p, tc is the current time, and $M_p(t)$ is the number of document mentions of p at time period t. 48 in the formula corresponds to 2 days.

NIFTY LINK

Evaluation of NIFTY

1. **Locality Sensitive Hashing**

Using LSH speeds up the algorithm

Experiments done on one week of data with 38,000 phrases.

Without LSH: 721 million pairwise comparison

With LSH: 26 million

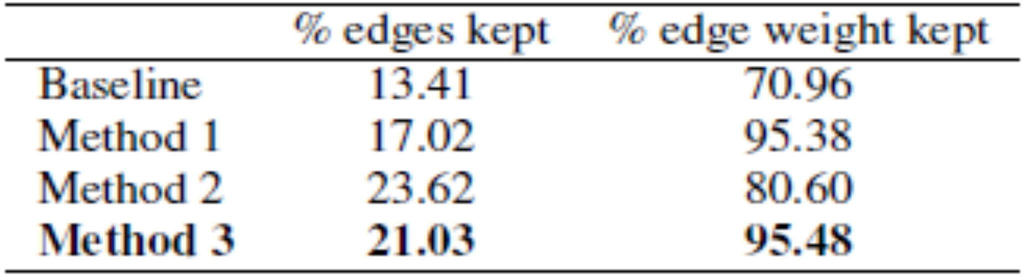
Evaluation of NIFTY cont..

2. **Phrase Graph Partitioning**

Different Edge selection methods

a. Pick outgoing edge with highest edge weight
b. Pick outgoing edges from the cluster with the most neighbours to the node.
c. Pick outgoing edges from the cluster of the node having the highest total weight.

Evaluation of NIFTY cont..

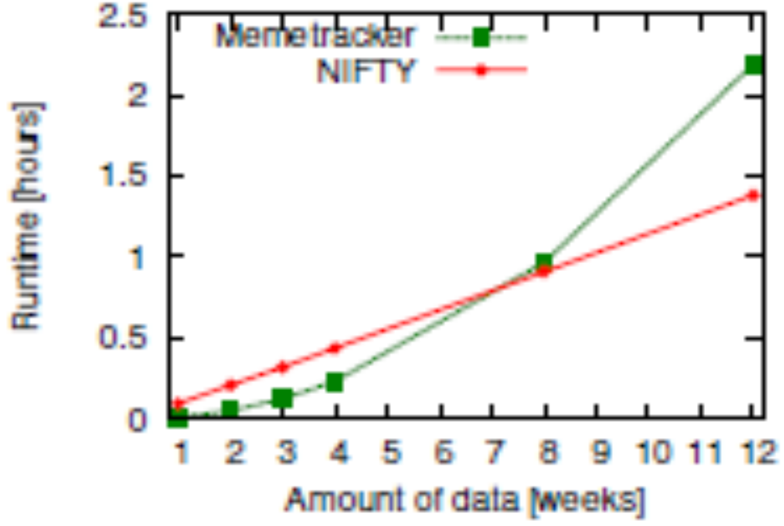


	% edges kept	% edge weight kept
Baseline	13.41	70.96
Method 1	17.02	95.38
Method 2	23.62	80.60
Method 3	**21.03**	**95.48**

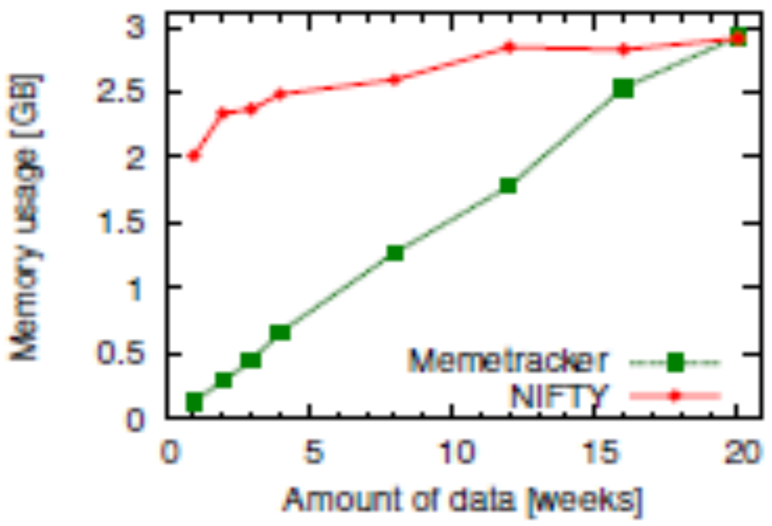
Method 3 performs best. Method 2 better at retaining edges and Method 1 gives a better edge weight.

NIFTY vs MEMETRACKER

1. Resource Usage



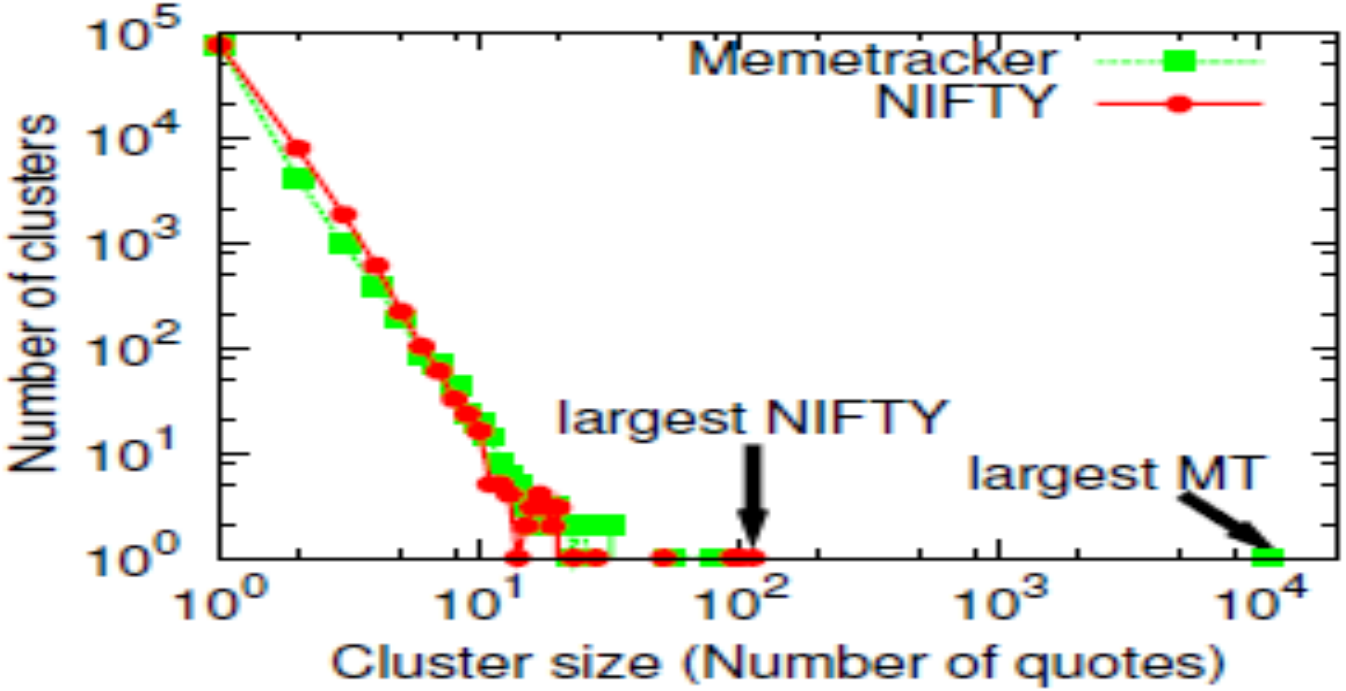
(a) Running time



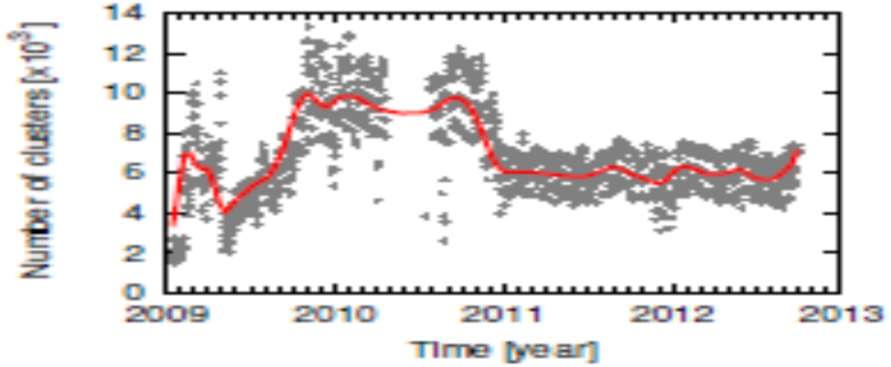
(b) Memory usage

NIFTY vs. MEMETRACKER cont..

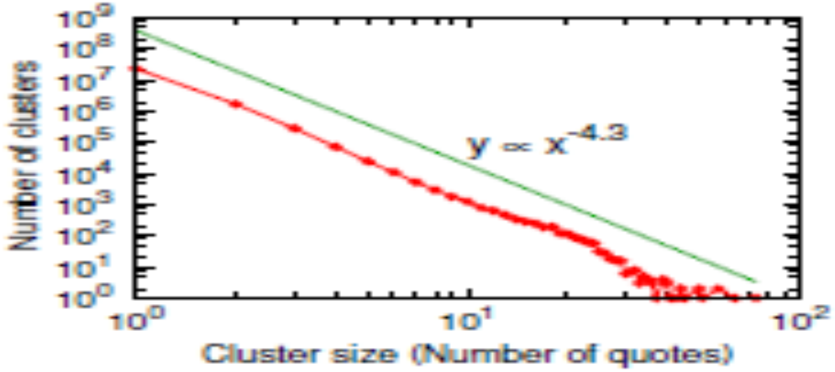
2. Meme Custer Quality



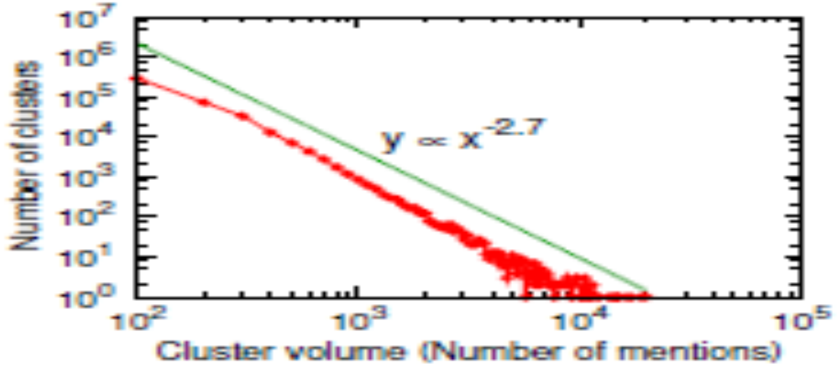
ANALYSIS OF MEMES



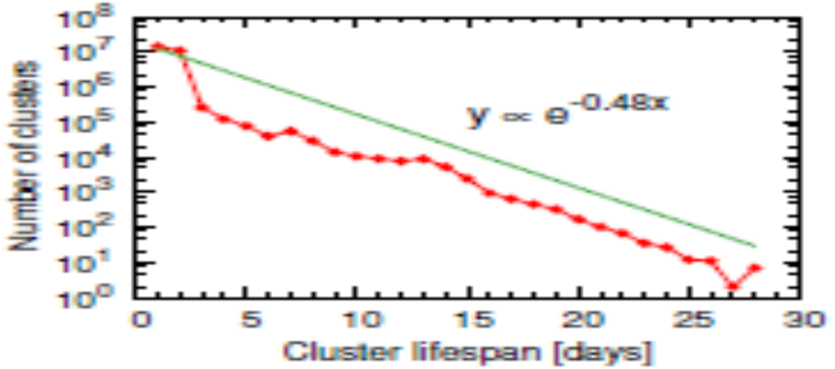
(a) Clusters over time



(b) Cluster size



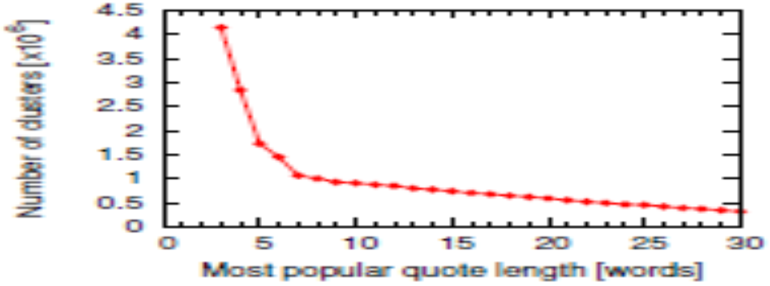
(c) Cluster volume



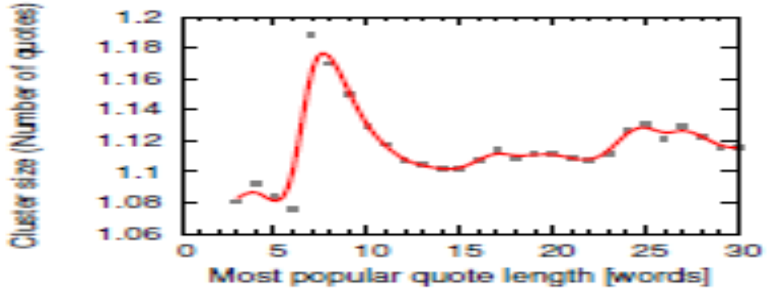
(d) Cluster lifespan

Properties of meme clusters

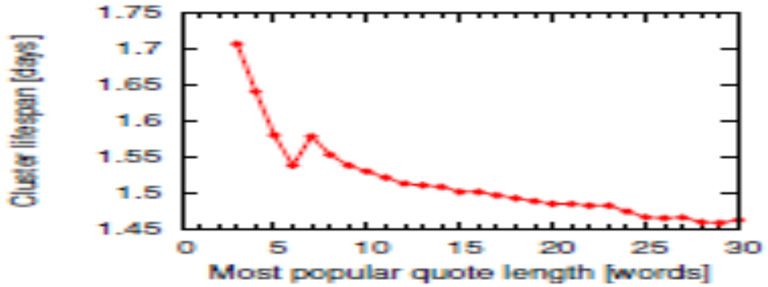
Most popular phrase length



(a) Count

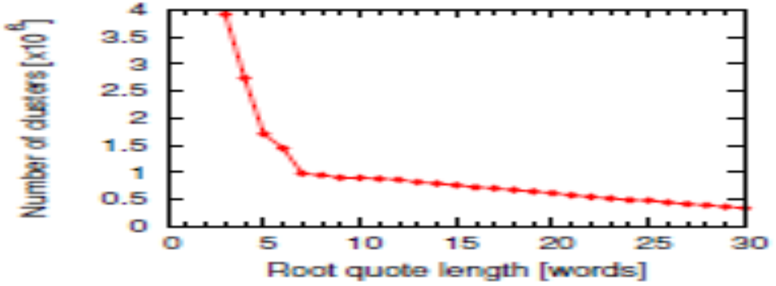


(c) Cluster size

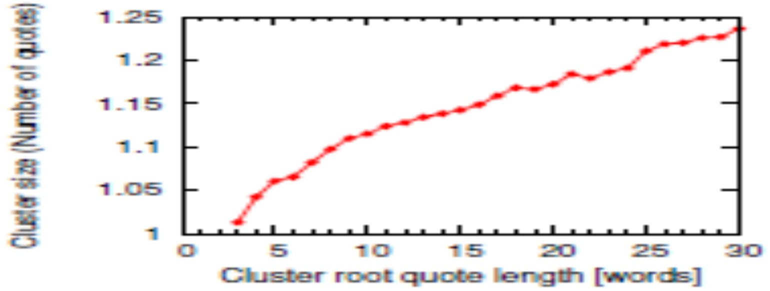


(e) Cluster lifespan

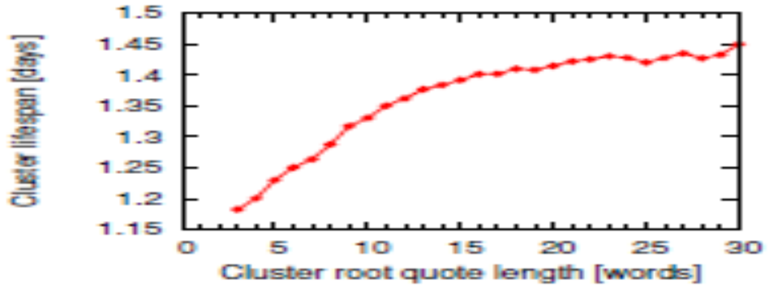
Root phrase length



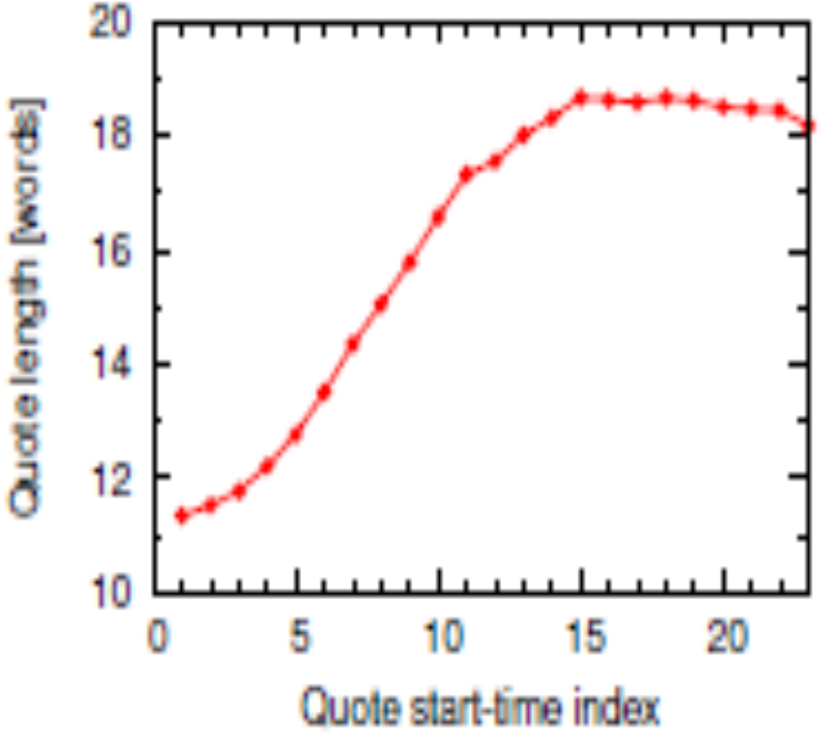
(b) Count



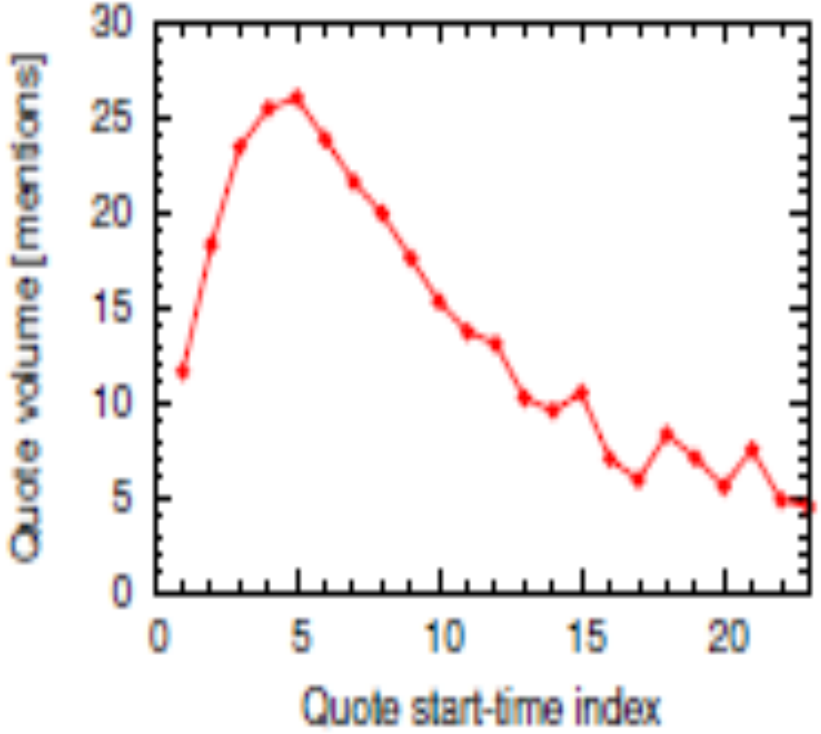
(d) Cluster size



(f) Cluster lifespan



(a) World length



(b) Volume

Properties of Phrases

References

1. J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle.
2. https://class.coursera.org/mmds-002/lecture/35 Massive Mining of Data Sets Coursera.
3. https://class.coursera.org/mmds-002/lecture/35
4. https://class.coursera.org/mmds-002/lecture/33
5. https://class.coursera.org/mmds-002/lecture/36
6. J. Yang, J Leskovec. Patterns of temporal variation in online media.

Thank You

QUESTIONS?