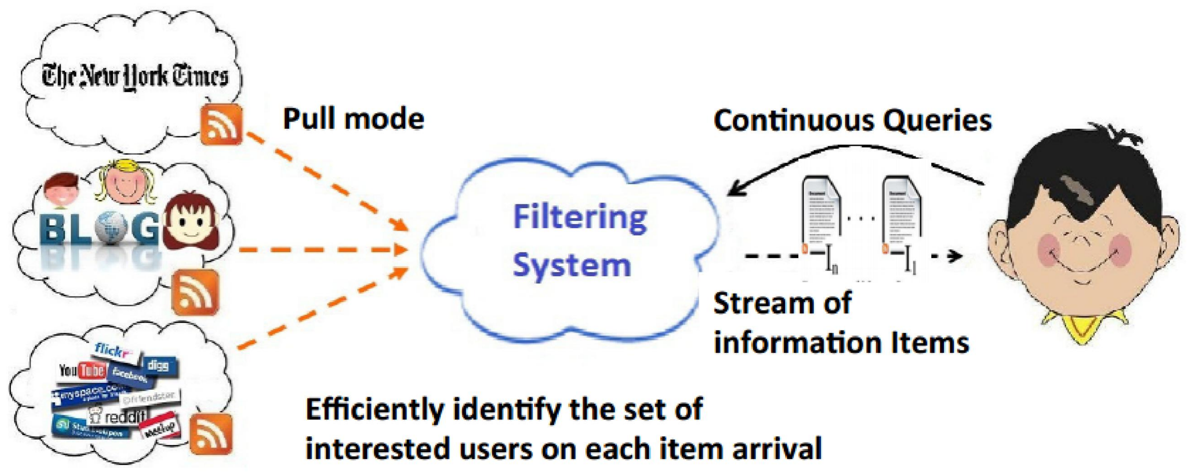
Cloud based publish/subscribe model for Top-k matching over continuous data-streams

Y.S. Horawalavithana, Dr. D.N. Ranasinghe
University of Colombo School of Computing

Introduction

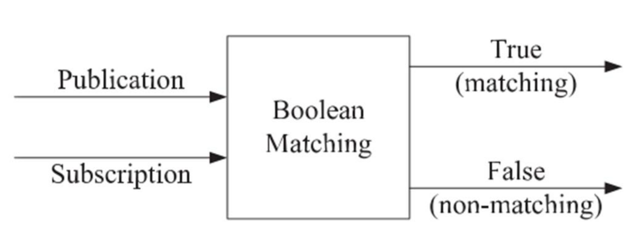


Why publish/subscribe?

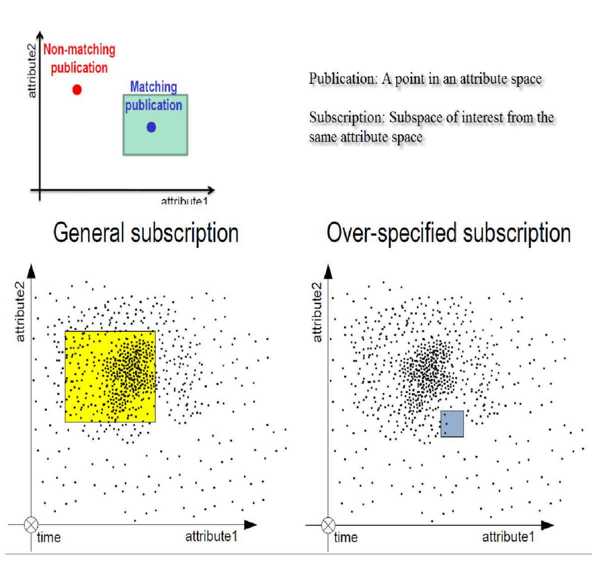
- Is the backbone among many modern day large scale applications emerged with ***"live" info*** production, too numerous applications behind:
  - ***The Web***: issue subscriptions for pages' updates, etc.
  - ***The network/system***: track requests with specific IPs,
  - (Multi-player Strategic Computer) ***Games***
- Decoupling of producers and consumers of info is valuable for more
  - Flexible, Lightweight, and Scalable systems

Boolean publish/subscribe model

Drawbacks

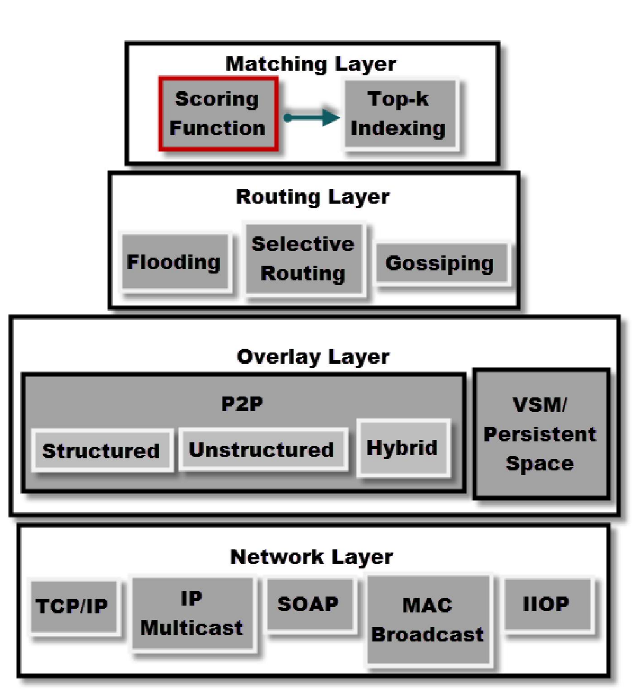


- A subscriber may be either overloaded with publications or receive too few publications
- Impossible to compare different matching publications as ranking functions are not defined,
- Partial matching between subscriptions and publications is not supported.

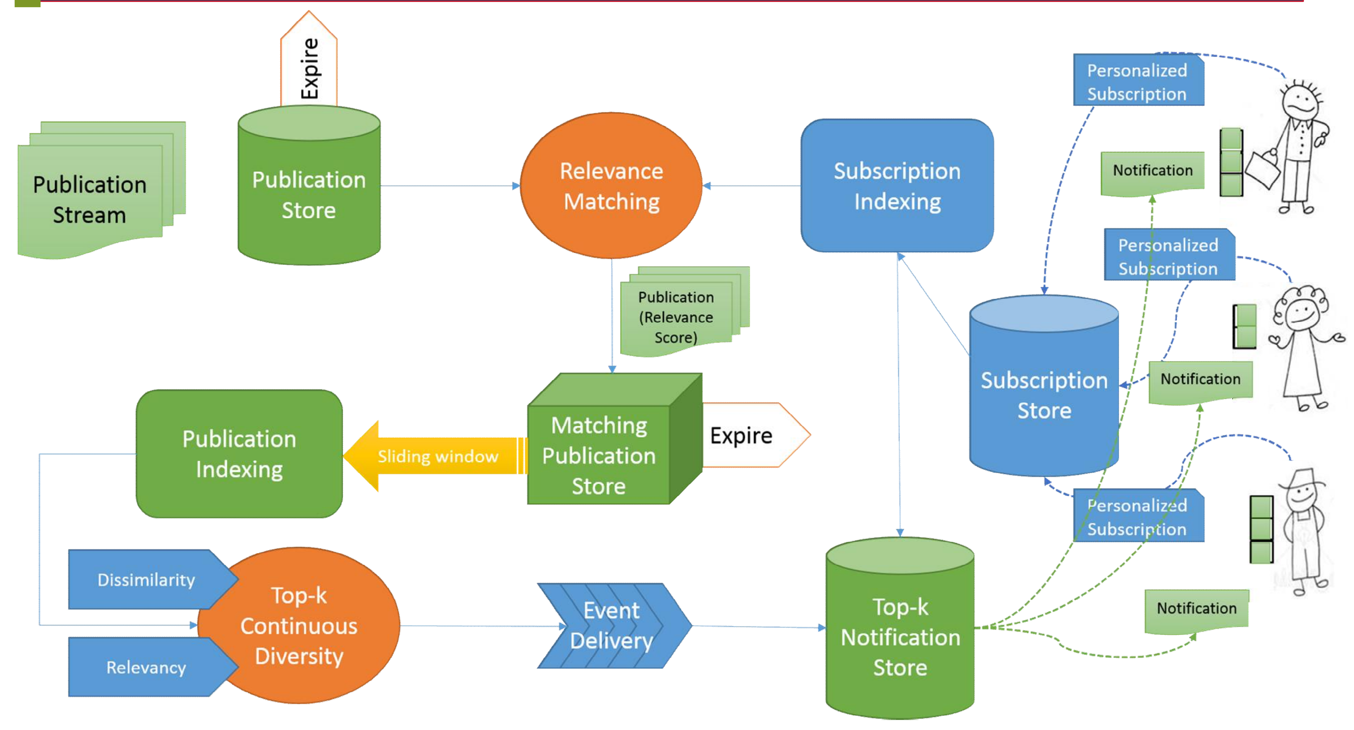


Top-k publish/subscribe

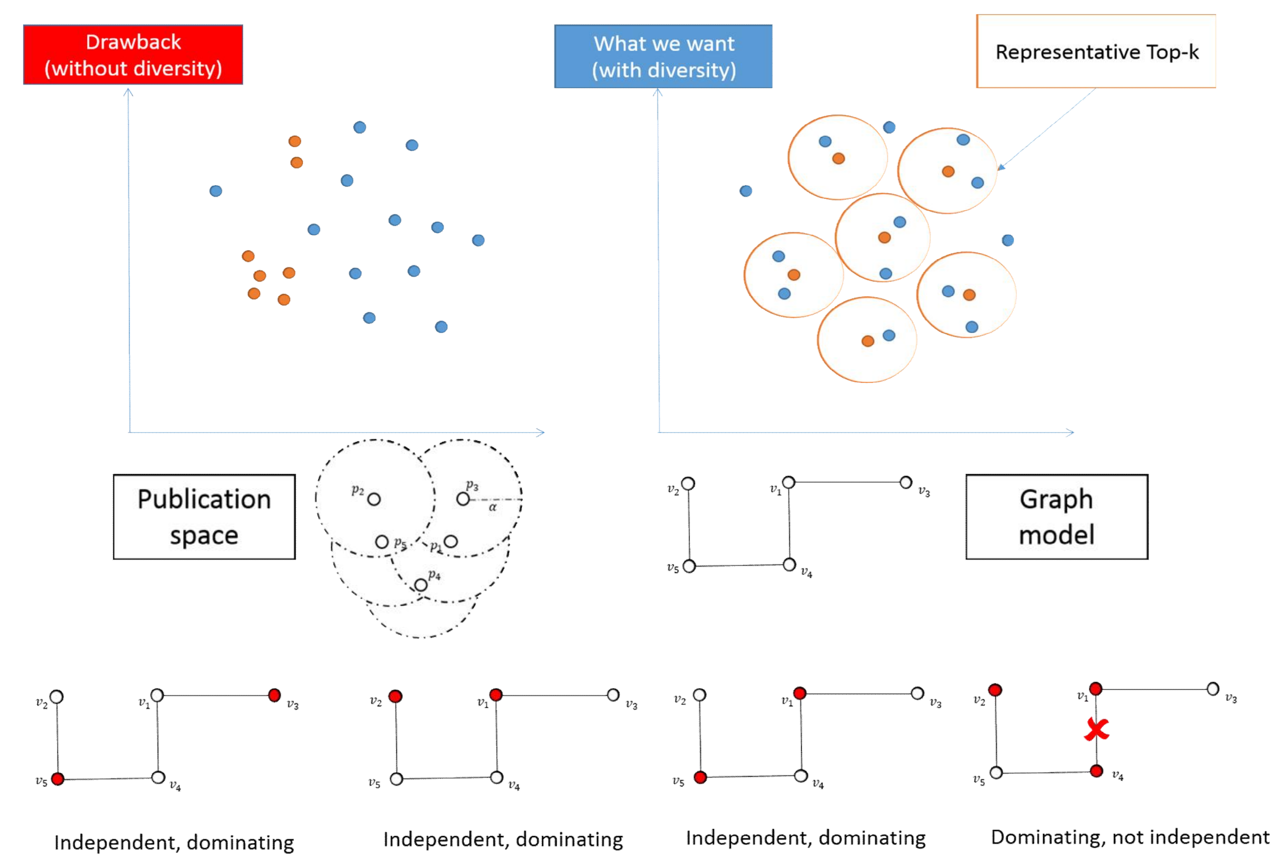
- Expressive stateful query processing systems
- User defined parameter ***k*** restricts the delivered publications at a given instance
- Pub/Sub **Matching**
  - Top-k pub/sub scoring or ranking
- Pub/Sub **Indexing**
  - Indexing to support personalized subscriptions
  - Indexing to support continuous Top-k publications retrieval



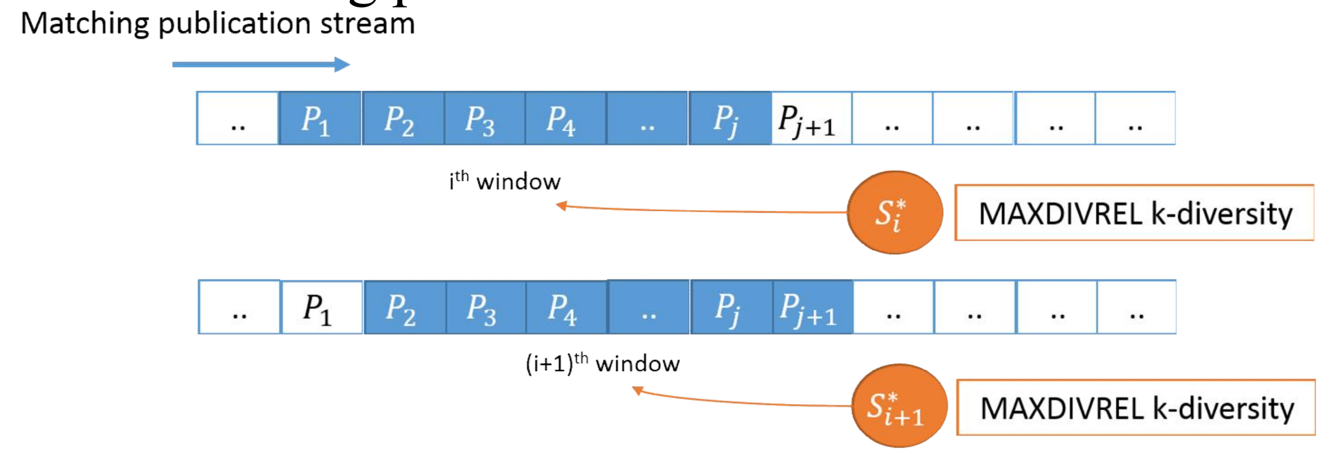
Design & Architecture



Top-k representative set



- **MAXDIVREL continuous k-diversity**
  - Sliding window Top-k computation to handle streaming publications



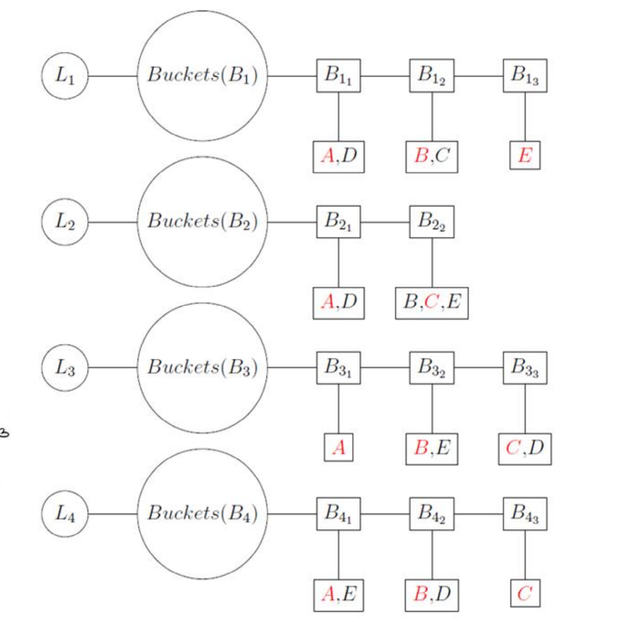
- NP-Hard : *MAXDIVREL is mapped to **minimum independent domination set problem** in graph theory*

- **Indexing streaming publications**
  - Avoid the curse of re-calculating neighborhood
  - Based on Locality Sensitive Hashing (LSH)
  - ***Fast Min-Hashing***
    - Minhash signatures to represent publications

$minhash_i$	publication X	publication Y	publication Z
h_1	$h_1(X)$	$h_1(Y)$	$h_1(Z)$
h_2	$h_2(X)$	$h_2(Y)$	$h_2(Z)$
…	..	..	..
h_m	$h_m(X)$	$h_m(Y)$	$h_m(Z)$

Signature Matrix

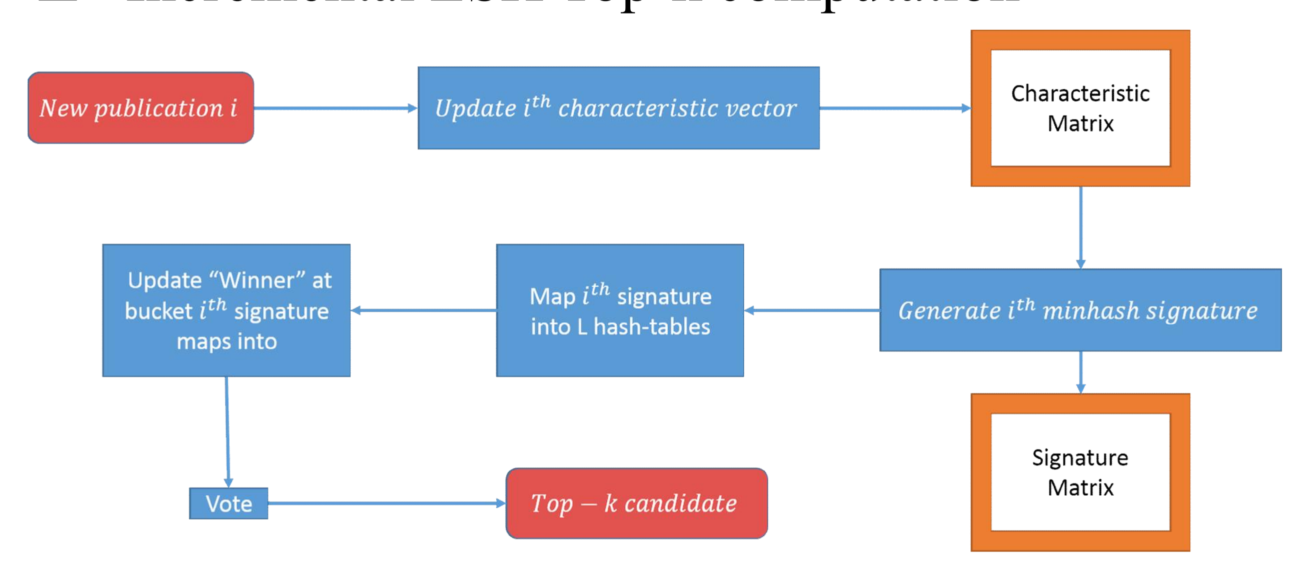
- Map the signatures into,
  - **L** Hash-Tables
  - With arbitrary **b** number of buckets
- *Voting mechanism to retrieve Top-k publications*



publication (P)	A	B	C	E
Votes	4	3	3	1

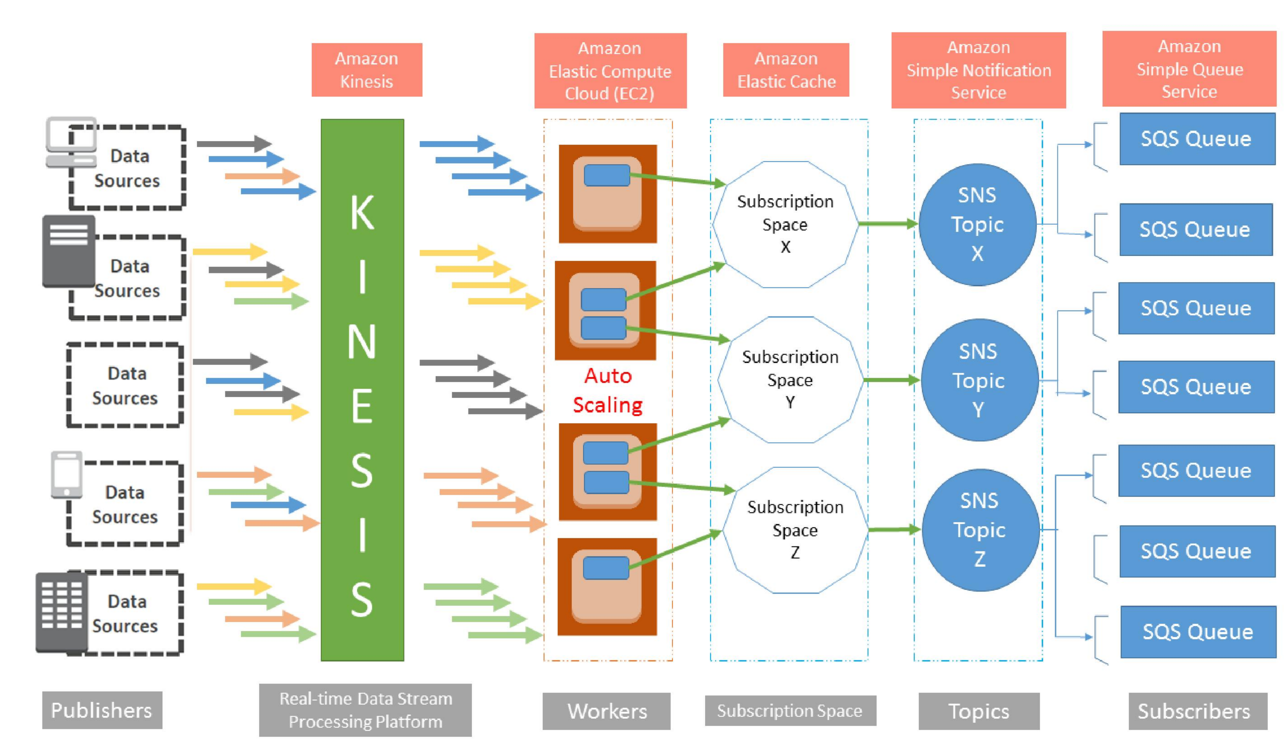
- Bucket "Winner" – a publication which has the highest relevancy score
  - ✓ Winner is ***dominant*** to represent it's bucket neighborhood
- Top-k "winners" that have a majority of votes
  - ✓ ***k winners are independent***

- Incremental LSH Top-k computation

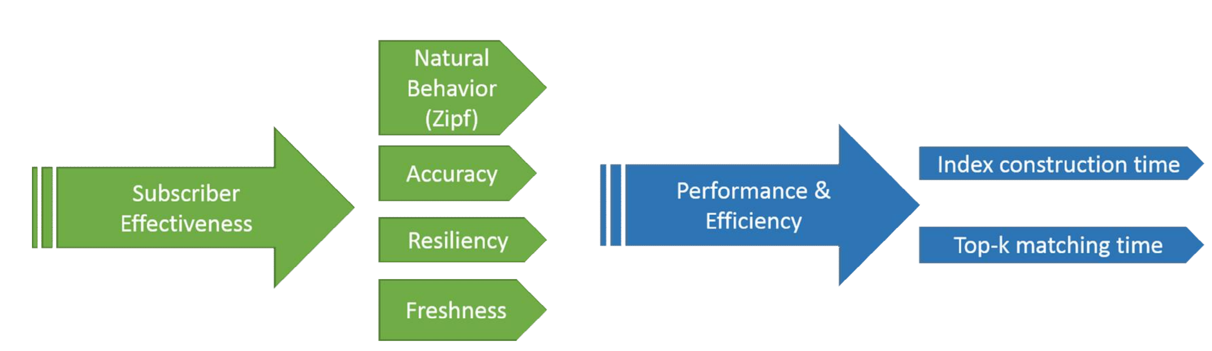


Implementation & Evaluation

- Implementation on top of Amazon Web Services

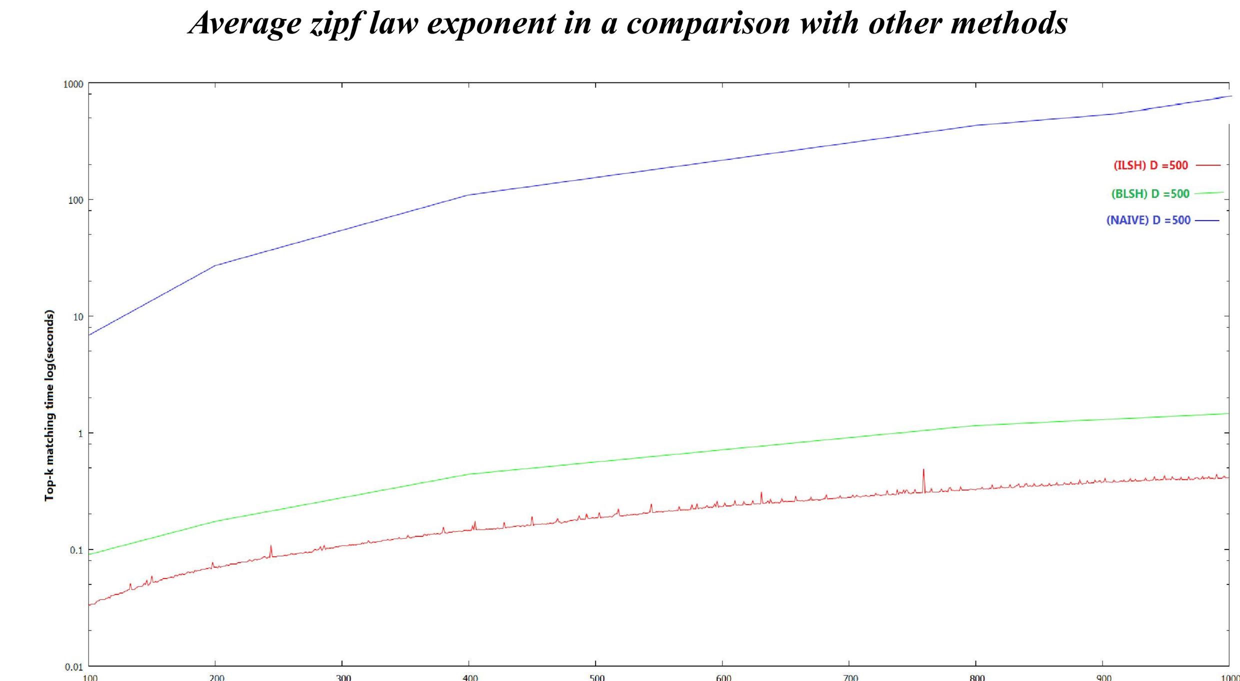


- Evaluation Methodology



k \ Diversity method	MAXMIN	DisC	MAXDIVREL
10	4.6123	3.4632	2.4883
50	12.2535	2.7392	2.4851
250	46.1347	2.5381	2.1956
500	50.3878	2.1023	1.9420
1000	62.5921	2.2003	1.9591

Average zipf law exponent in a comparison with other methods



A comparison of incremental LSH indexing with naïve & batch method: ranking time on number of publications with dimension D=500