
Clustering and Topic Analysis

CS 5604 Final Presentation

December 12, 2017

Virginia Tech, Blacksburg VA 24061

Acknowledgments

- Global Event and Trend Archive Research (GETAR) project, supported by NSF IIS-1619028
 - Integrated Digital Event Archiving and Library (IDEAL) project supported by NSF IIS-1319578
 - CS5604 GTA Liuqing Li, Instructor Edward A. Fox
-

Team Members

Topic Analysis

Ashish Baghudana

Shruti Shetty

Aman Ahuja

Ashish Malpani

Clustering

Rammohan Chintha

Mo Yang

Pavan Bellam

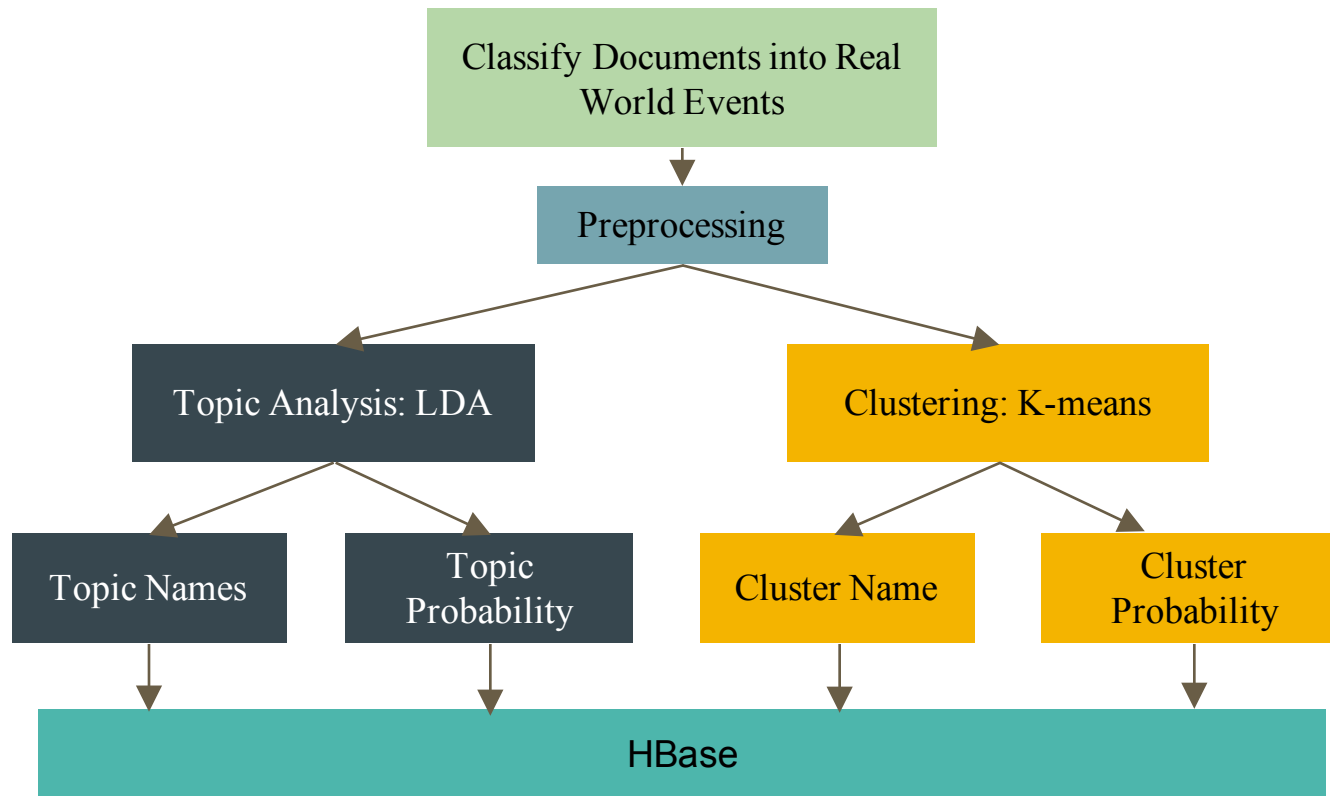
Prathyush Sambaturu

Overview

Use topic analysis and clustering algorithms to find sub-themes and similar patterns in collections of webpages and tweets about real world events

- Pull and clean documents
 - Topic Analysis
 - Clustering
 - Store results in HBase
-

Overview



Topic Analysis

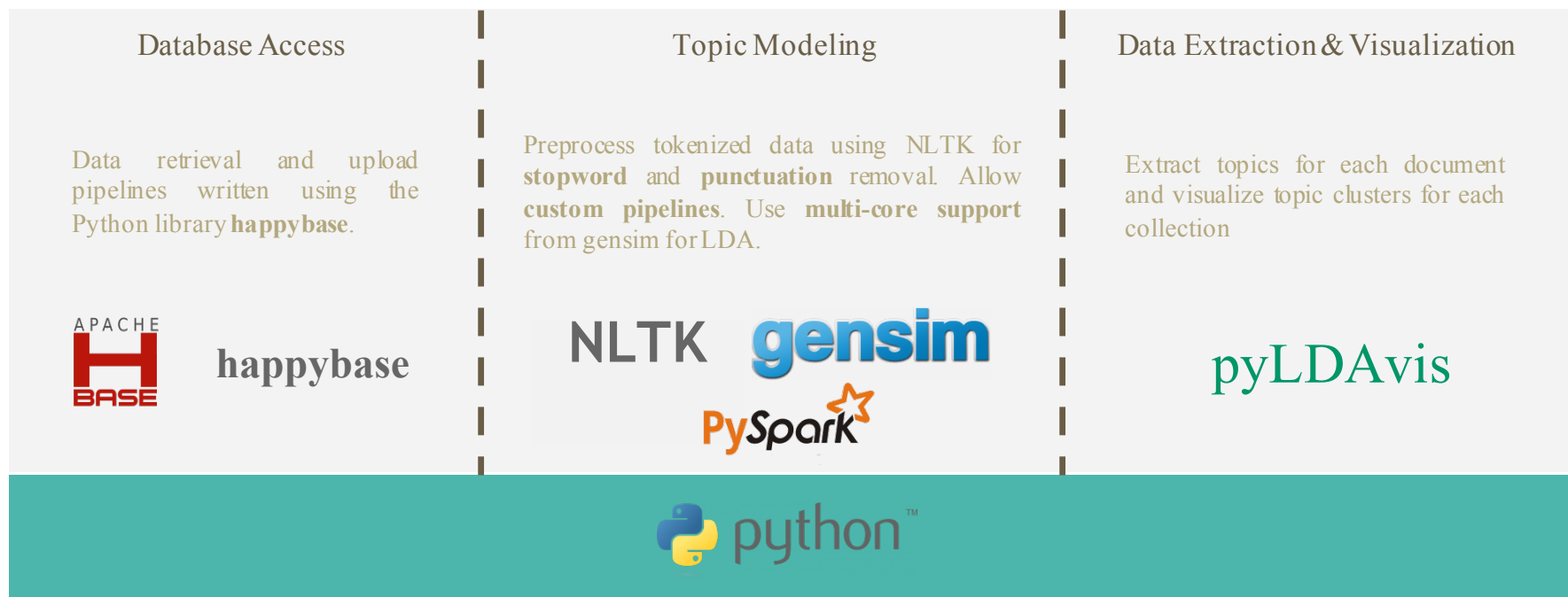


A (really) brief overview of topic models

Topic models discover “abstract” topics in a collection of documents where a topic is a group of “semantically similar” words.

- Mainly latent Dirichlet allocation (LDA) or its variants
 - Probabilistic algorithm
 - Optimized via Gibbs Sampling or Expectation Maximization
 - LDA assumes documents a mixture over topics and topics a mixture over words
-

Tech Stack - Python++

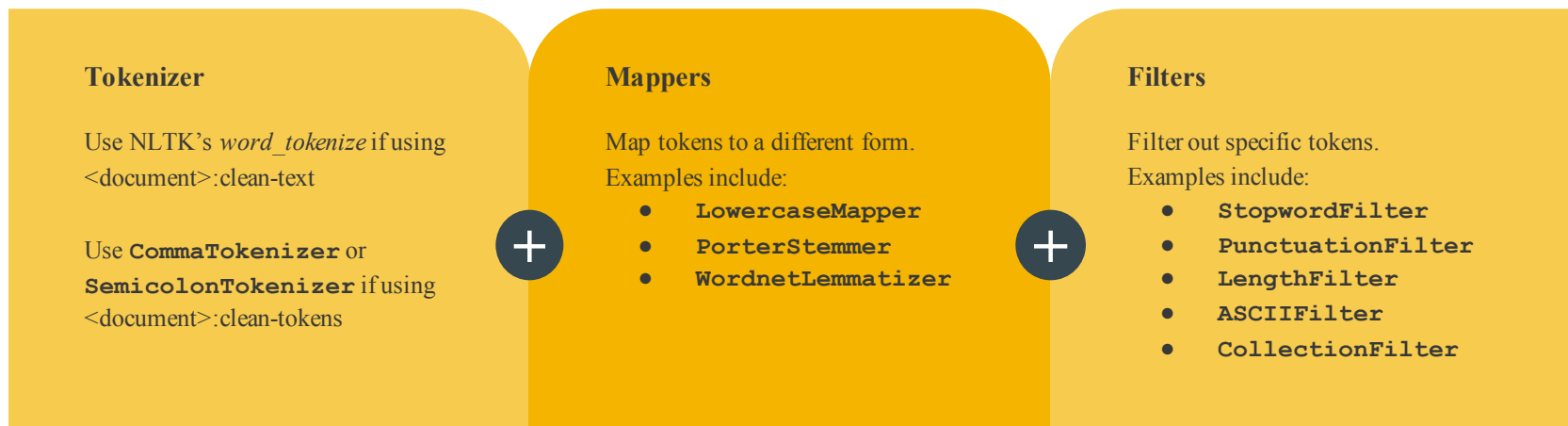


Alternate Tech Stacks

- **Scala + Spark + MLLib** is an alternate stack
 - Might scale better for larger datasets
 - However, lacks the visualization and evaluation mechanisms that are built into gensim and pyLDAvis
 - **Gotchas**
 - **PySpark + Spark + MLLib**
 - does not implement the class `DistributedLDAModel`
 - `LDAModel` does not implement several important methods such as: *topTopicsPerDocument()*, *logPerplexity()* and *topicCoherence()*
 - **Python + Gensim is best suited (multi-core support really helps!)**
-

Preprocessing Pipeline (where the magic happens)

Built a fast and customizable preprocessing pipeline.



Collection Filter - Customizing for each collection

Certain words occur repeatedly in each document and can be regarded as stop words for that collection

Solar Eclipse 2017			Hurricane Irma		
solar	eclipses	facebook	hurricane	sept	business
eclipse	subnav	twitter	irma	september	insider
totality	aug	published	national	csbn	guardian
tse	august	username	global	reuters	subscribe
eclipse2017	account	password	us	florida	bloomberg

Developer Manual

Topic Modeling is end-to-end integrated as a single script with several built-in options

```
usage: lda.py [-h] -c COLLECTION_NAME -t TOPICS [TOPICS ...] [-p] [--table_name TABLE_NAME] (-hb | -f FILE) [-l LOGS] [-a ALPHA] [-b BETA] [-i ITER] [--save_dir SAVE_DIR] [--tokenizer TOKENIZER] [--mappers MAPPERS [MAPPERS ...]] [--filters FILTERS [FILTERS ...]] [--filter_words FILTER_WORDS]
```

Run LDA on a given collection

required arguments:

```
-c COLLECTION_NAME, --collection_name COLLECTION_NAME
                        Collection name
-t TOPICS [TOPICS ...], --topics TOPICS [TOPICS ...]
                        Number of topics to run the model on
-p, --preprocess        Preprocess data
--table_name TABLE_NAME
                        Table name for HBase
-hb, --hbase            Get collection from HBase
-f FILE, --file FILE   File name for tokens
```

preprocessing arguments to be added when using -p flag:

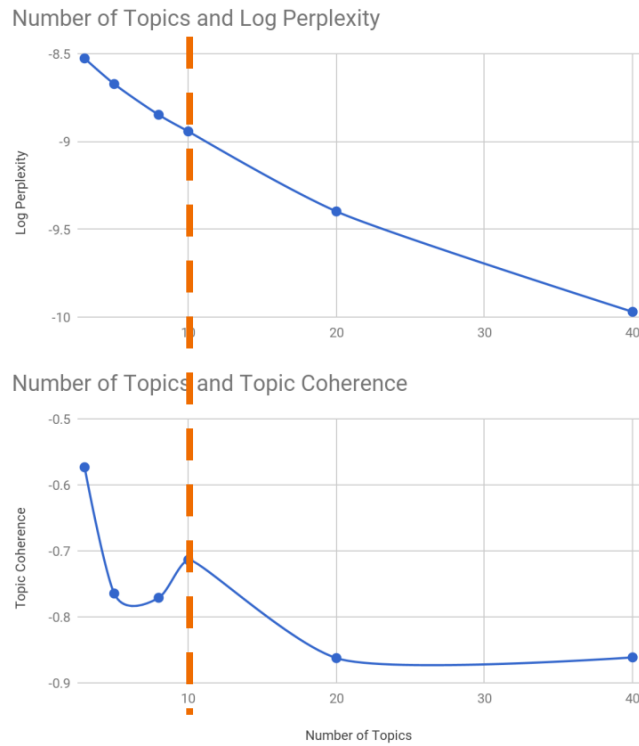
```
--tokenizer TOKENIZER
                        Tokenizer to use
--mappers MAPPERS [MAPPERS ...]
                        Mappers to use
--filters FILTERS [FILTERS ...]
                        Filters to use
--filter_words FILTER_WORDS
                        Filename with words to filter out
```

optional arguments:

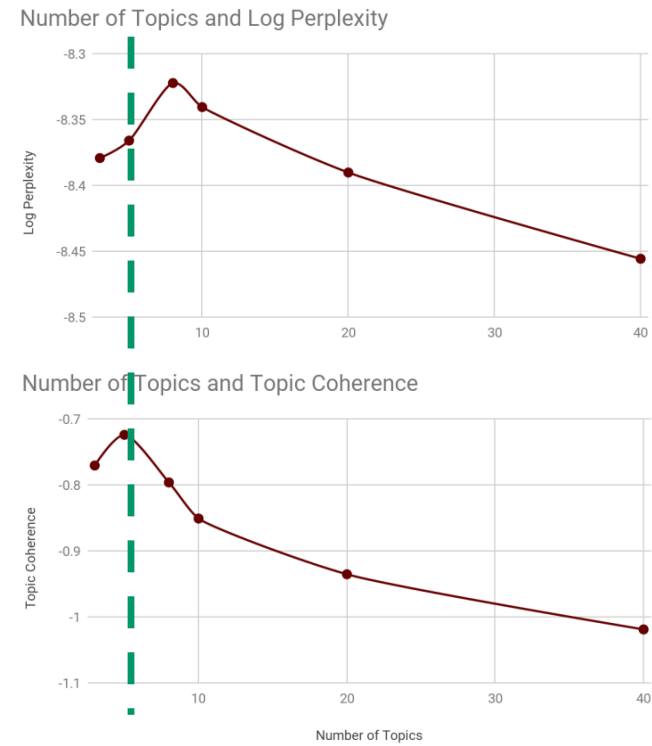
```
-h, --help              show this help message and exit
-l LOGS, --logs LOGS   Log directory
-a ALPHA, --alpha ALPHA
                        Alpha hyperparameter
-b BETA, --beta BETA   Beta hyperparameter
-i ITER, --iter ITER   Number of iterations
--save_dir SAVE_DIR    Save directory for topic models
```



Choosing the Right Number of Topics



Solar Eclipse Webpages



Hurricane Irma Webpages

Visualizing Topics

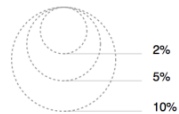
Selected Topic:

Slide to adjust relevance metric:⁽²⁾
 $\lambda = 1$ 0.0 0.2 0.4 0.6 0.8 1.0

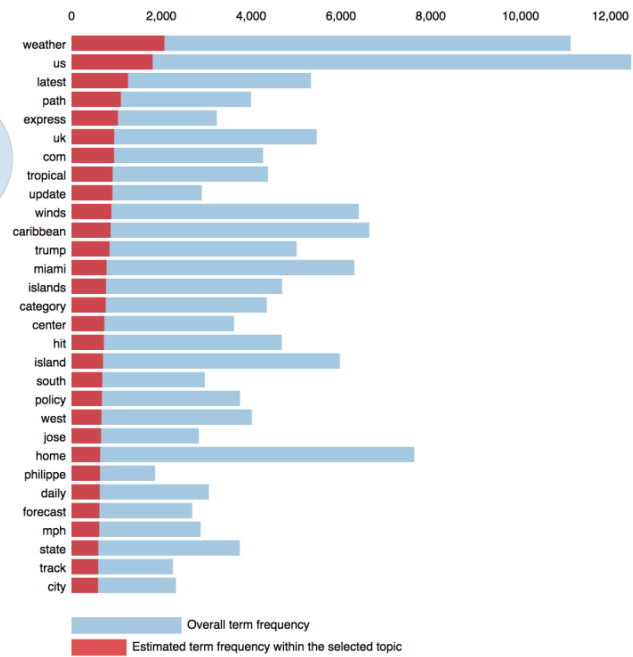
Intertopic Distance Map (via multidimensional scaling)



Marginal topic distribution



Top-30 Most Relevant Terms for Topic 4 (14.2% of tokens)



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
 2. relevance(term w | topic t) = $\lambda * p(w | t) + (1 - \lambda) * p(w | t)/p(w)$; see Sievert & Shirley (2014)

Naming Topics

Once we finalize on the number of topics, each “topic” has to be named. We follow two strategies:

- Automatic Naming
 - Choose the first non-repeating word in the topic-word distribution (sorted by probability)
 - Suitable for large number of topics
 - Manual Naming
 - Look at top words manually to decide topic name
 - Suitable for small number of topics
-

Solar Eclipse: Tweets

eclipse	safety	pictures	experience	midflight	forecast	exo
sun	view	photos	truly	catch	cloud	exo
moon	look	pictures	remarkable	breathtaking	shadow	totaleclipse
block	glass	photobomb	beautiful	mid	weather	thepowerofmusic
watch	don't	timelapse	breathtaking	flight	path	planet
cover	eye	space	great	international	rain	message
totality	watch	lifetime	pretty	space	outside	verexo
circle	safely	live	happy	wow	forecast	que

These results aren't perfect however. Despite strict filtering, we see words like *trumpresign* and *president* in the list.



Computational Complexity

	Collection Size	Preprocessing	Model Creation	Log Perplexity	Topic Coherence
Hurricane Irma (Webpages)	2714	56s	1m:07s	1m:01s	3s
Solar Eclipse (Webpages)	722	16s	41s	19s	2s
Solar Eclipse (Tweets)	2667726	11m28s	16m14s	24m13s	55s

* Models trained with 500 iterations and 10 topics each
+ Preprocessing is faster on local machines because of SSDs

HBase Fields

We populate:

- `topic:topic-list`
- `topic:probability-list`
- `topic:display-topicnames`

How these are used:

- The front-end team uses `display-topicnames` as a facet in their interface
 - It is also possible to use probability scores for recommending *similar* documents
-

Shortcomings and Improvements

- Automatic elimination of collection-specific stop-words
 - Currently requires manual curation
 - Crowd experiments to decide topic names and coherence
 - Topic names decided either naively or based on the experimenter's judgement
 - Better to use the class strength to crowdsource annotations
 - Topic modeling visualizations can be part of front end
 - pyLDAvis provides visualizations of the documents in a cluster via a MDS algorithm
 - Could be part of the faceted search
 - Joint models for Tweets and Webpages
-

Clustering

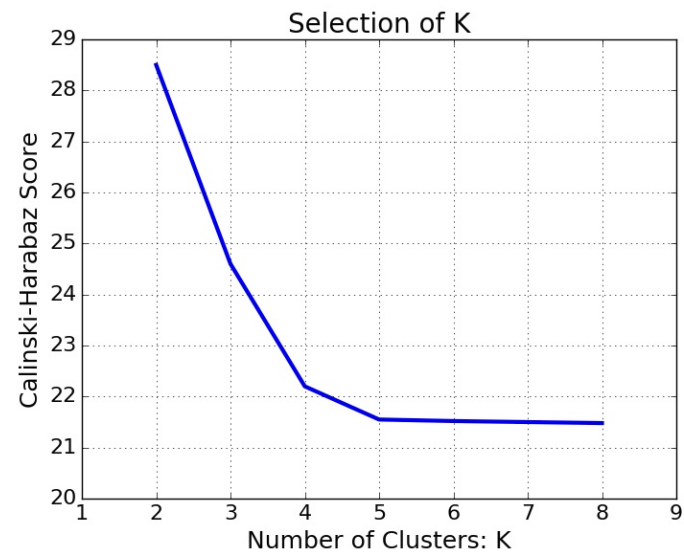


A brief overview of Clustering

Clustering categorize data into clusters such that objects grouped in same cluster are similar to each other according to specific metrics

- K-means Algorithm
 - Elbow method to find number of K
 - Clustering based on cosine similarity

$$\text{sim}(d_1, d_2) = \frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{|\vec{V}(d_1)| |\vec{V}(d_2)|}$$



Tech Stack

- **DataBase Access:**

Data retrieval and upload pipelines written using the Python library **happybase**.

- **Clustering:**

Preprocess tokenized data using NLTK for **stopword** and **punctuation** removal.
Scala and **Spark** for used for Clustering

- **Result Evaluation & Visualization:**

Python and **matplotlib** are used for calculating inter-/intra-cluster similarity and plotting the results



happybase



matplotlib



Developer Manual

Create a Scala Package using

```
sbt-package
```

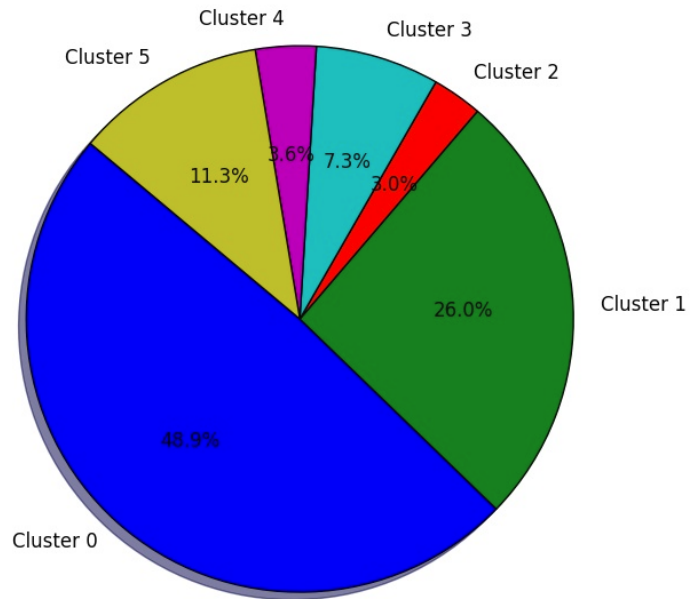
And submit the generated jar to Spark

```
Spark-submit <jar file> <input file>
```

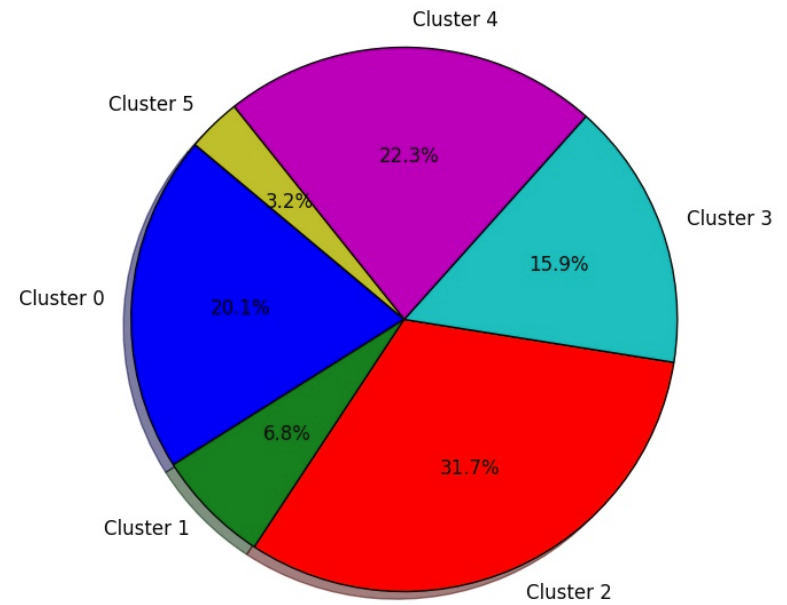
For large datasets, performance metrics like driver memory can be enhanced

Clustering Results

“Solar Eclipse” Tweets



“Solar Eclipse” Webpages



Similarity Analysis

“Solar Eclipse” Webpages

	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
Cluster 0	0.235892	0.057602	0.095294	0.077461	0.067748	0.087039
Cluster 1	0.057602	0.579325	0.092020	0.083038	0.081589	0.076596
Cluster 2	0.095294	0.092020	0.146127	0.105437	0.095530	0.099726
Cluster 3	0.077461	0.083038	0.105437	0.176871	0.077173	0.086996
Cluster 4	0.067748	0.081589	0.095530	0.077173	0.111608	0.084908
Cluster 5	0.087039	0.076596	0.099726	0.086996	0.084908	0.714758

Average intra-cluster similarity: 0.33

Average inter-cluster similarity: 0.08

Naming Clusters by Frequent words

- K-means just returns clusters of documents, but does not name the clusters
 - We can name each cluster by looking at a handful of documents in the cluster manually, which can be tedious in Big Data scenario
 - Therefore,
 - we determine the most frequent words in all documents within a cluster
 - name the cluster based on a few very frequent words in the cluster(Assumption: The frequent words in the cluster describe the information in the documents within the cluster)
-

Clusters in “Solar Eclipse” Tweets and Webpages

Cluster ID	Cluster Name for Tweets	Cluster Name for Webpages
0	DiamondRing	EclipseChasers
1	WatchEclipse	AjcEclipseNews
2	SafeEclipse	EclipseScience
3	ExoPlanetMusic	BusinessInsiderEclipseArticles
4	MidFlightEclipse	Eclipseville
5	NonEnglish	MuseumEclipse

HBase Fields

We populate:

- `cluster:cluster-list`
- `cluster:cluster-probability`
- `cluster:display-clusternames`

Script and Command:

- `Python hbase_write_cluster.py [Cluster_Result].csv`
 - `Cluster_Result: Document_ID, Cluster_Name, Cluster_Probability`
-

Shortcomings and Improvements

- Cluster Probability
 - We set it as “1” because we only did hard clustering
- Comparison with other clustering algorithm
 - Hierarchical Clustering
 - Density-based Clustering



Questions?

Code Repository: <https://github.com/ashishbaghudana/cs5604>
https://github.com/yuan9/CS5604_CTA