HADOOP USE CASES

CKME 134 – BIG DATA ANALYTICS TOOLS

RYERSON UNIVERSITY

SPRING 2015

Instructor: Shaohua Zhang

Course Outline

1. Intro to Big Data
2. Distributed Computing and MapReduce
3. Hadoop Ecosystem
4. Programming Hive
5. Advanced Hive
6. Mid-Term Review
7. Programming Pig
8. Advanced Pig
9. Hadoop Use Cases
10. Building Data Product & Next-Gen Hadoop (Spark)
11. Beyond Hadoop: Graph Analytics

Lecture 4 - Outline

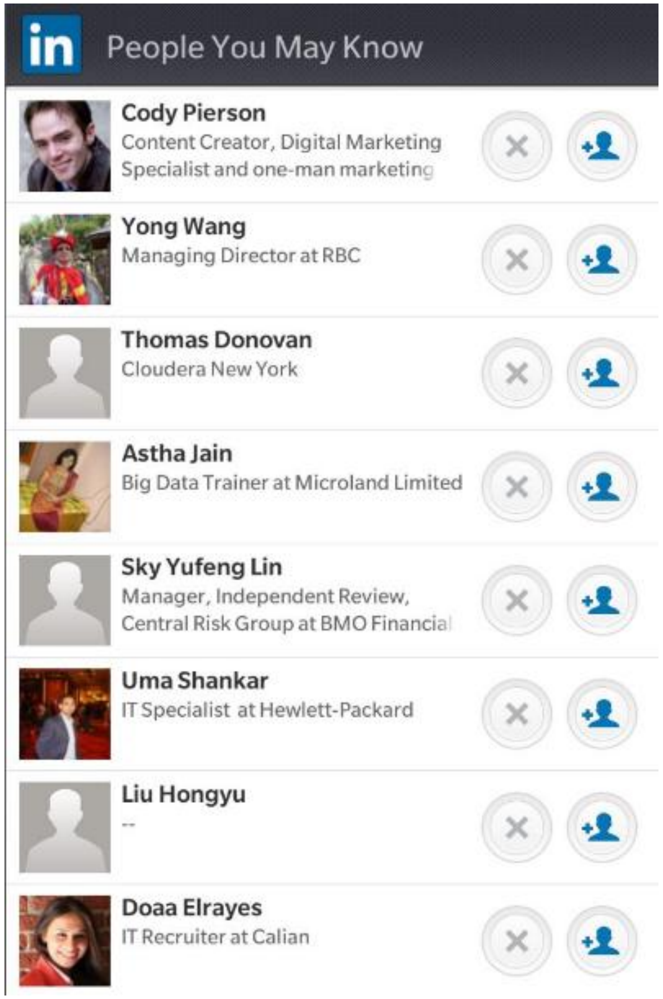
1. Big Data Use Case: PYMK
2. Big Data Use Case: TF-IDF
3. Big Data Use Case: Location Analytics
4. Big Data Use Case: Machine Learning

Hadoop Use Case

People You May Know Recommendation

Hadoop Use Cases: PYMK

- Why are they recommended?
  - Friend of friend
  - Same school?
  - Same company?
  - Same city?
  - Similar job title?
  - Similar skillsets?
  - Same LinkedIn groups?
  - etc.



People You May Know

Cody Pierson
Content Creator, Digital Marketing Specialist and one-man marketing

Yong Wang
Managing Director at RBC

Thomas Donovan
Cloudera New York

Astha Jain
Big Data Trainer at Microland Limited

Sky Yufeng Lin
Manager, Independent Review, Central Risk Group at BMO Financial

Uma Shankar
IT Specialist at Hewlett-Packard

Liu Hongyu
--

Doaa Elrayes
IT Recruiter at Calian

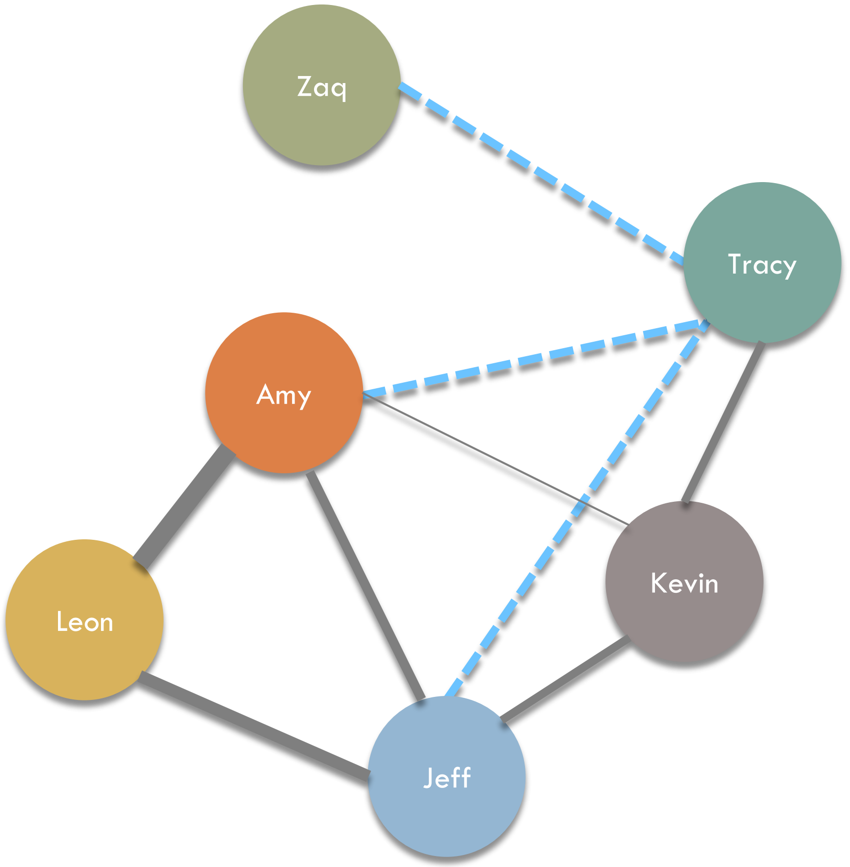
PYMK - Graph Representation

	Friend				
User	Amy	Kevin	Jeff	Tracy	Leon
Amy		0.8	1.7		4
Kevin	0.8		1.3	1	
Jeff	1.7	1.3			2.2
Tracy		1			
Leon	4		2.2		

Matrix

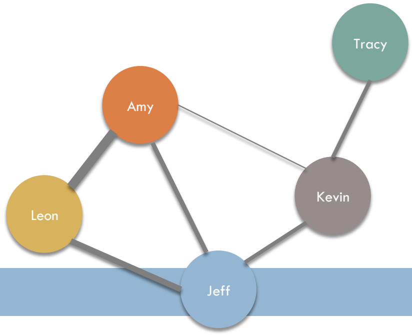
Key	*Value*
Amy	*(Kevin,0.8), (Jeff, 1.7) (Leon,4)*
Kevin	*(Amy,0.8), (Jeff,1.3), (Tracy,1)*
Jeff	*(Amy, 1.7), (Kevin,1.3), (Leon,2.2)*
Tracy	*(Kevin,1)*
Leon	*(Amy,4), (Jeff,2.2)*

Adjacency List



Graph

PYMK - Potential Friends



User	Friend
Amy	Kevin, Leon
Kevin	Amy, Jeff, Tracy
Jeff	Amy, Kevin, Leon
Tracy	Kevin
Leon	Amy, Jeff

→

Existing Friend Pairs
(Amy, Kevin), (Amy, Jeff), (Amy, Leon)
(Kevin, Amy), (Kevin, Jeff), (Kevin, Tracy)
(Jeff, Amy), (Jeff, Kevin), (Jeff, Leon)
(Tracy, Kevin)
(Leon, Amy), (Leon, Jeff)

→

Existing Friend Pairs
(Amy, Kevin), (Amy, Jeff), (Amy, Leon)
~~(Kevin, Amy), (Kevin, Jeff)~~, (Kevin, Tracy)
~~(Jeff, Amy)~~, (Jeff, Kevin), (Jeff, Leon)
~~(Tracy, Kevin)~~
~~(Leon, Amy), (Leon, Jeff)~~

User	Friend
Amy	Kevin, Leon
Kevin	Amy, Jeff, Tracy
Jeff	Amy, Kevin, Leon
Tracy	Kevin
Leon	Amy, Jeff

→

Potential Friend Pairs
~~(Kevin, Jeff)~~, (Kevin, Leon), (Jeff, Leon)
(Amy, Jeff), (Amy, Tracy), (Jeff, Tracy)
(Amy, Kevin), (Amy, Leon), ~~(Kevin, Leon)~~
~~(Amy, Jeff)~~

Exclude existing friend pairs →

Potential Friend Pairs
~~(Kevin, Jeff)~~, (Kevin, Leon), ~~(Jeff, Leon)~~
~~(Amy, Jeff)~~, (Amy, Tracy), (Jeff, Tracy)
~~(Amy, Kevin)~~, ~~(Amy, Leon)~~, ~~(Kevin, Leon)~~
~~(Amy, Jeff)~~

Potential Friend Pairs
(Kevin, Leon),
(Amy, Tracy), (Jeff, Tracy)

PYMK - Relevance Ranking

problem definition

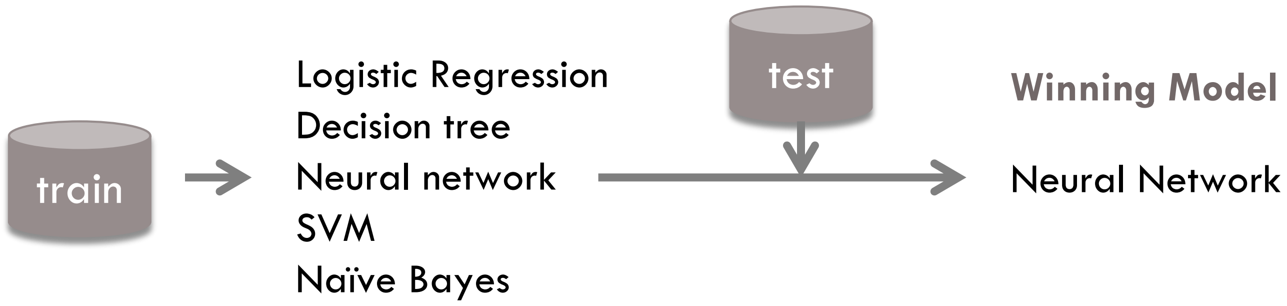
feature	feature description	example value
f1	number of common friends	15
f2	same school	1
f3	worked at same company?	0
f4	same city	1
f5	work experience similarity	0.8
f6	skill set similarity	0.65
f7	joined same linkedin group?	0.7

target	description	example value
action	click to add friend	1
no action	does not click to add fr	0

data preparation

User	Recommended Friend	f1	f2	f3	f4	f5	f6	f7	action Y/N
Jack	Fox	3	1	1	1	0.3	0.7	0.6	1
Hasan	Loran	45	0	0	0	0.4	0.1	0.2	0
Jennifer	Jason	2	0	1	1	0.5	0.2	0.1	0
---	---	---	---	---	---	---	---	---	---
Alex	Florence	1	1	0	1	0.2	0.7	0.3	0

model training and validation



scoring

User	Recommended Friend	f1	f2	f3	f4	f5	f6	f7	action Y/N
Tracy	Jeff	2	0	0	0	0.7	0.8	0.5	0
Tracy	Amy	15	1	0	1	0.8	0.65	0.7	1

PYMK - *** Additional Lab ***

- How to do PYMK in Hadoop?
  - Create a social graph from the geo-tagged tweets dataset
    - hint: assume users and their mentioned users "@" know each other
  - Find friend of a friend in Hive
    - hint: for a particular user, can you use hive query to retrieve a list of potential friend based on friend of a friend?
  - Potential Friend List Generation in Pig
    - Do the PYMK main algorithm in Pig for all users
  - Recommended Friend Ranking in Pig
    - Instead of building a machine learning model, rank the potential friends by number of common friends (more shared friends meaning more likely to know or connect with each other)

Hadoop Use Case

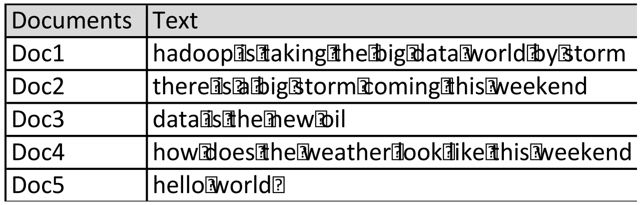
TF-IDF (Term-Frequency, Reverse Document Frequency)

Hadoop Use Cases: TF-IDF

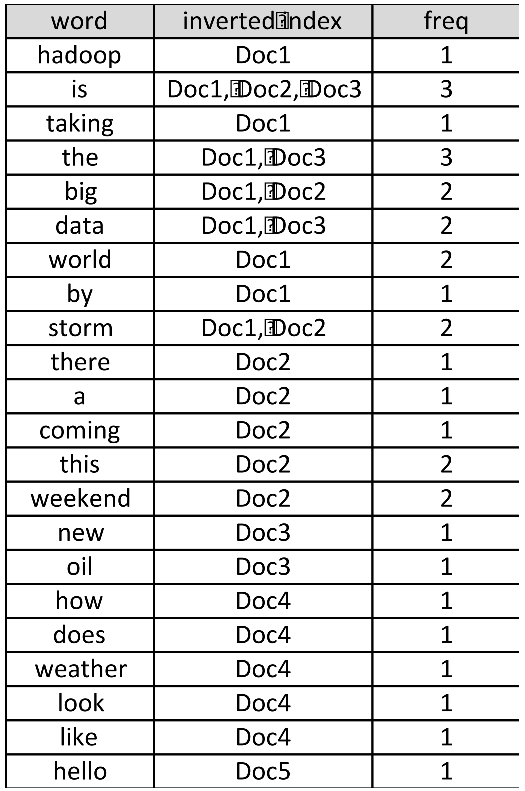
- Term Frequency – Inverse Document Frequency
- Weighting scheme for text clustering and document classification
  - puts more weight on relevant keywords
- Search engine key word weighting
  - Calculating IDF of the web using Google Search
- Stopword filtering in text summarization

TF-IDF - Search Engine In a Nutshell

- Inverted Index
  - Search engines like Google also use an inverted index for searching the web.
  - In fact, the need to build a web-scale inverted index led to the invention of MapReduce
- A list of documents that the term appear in



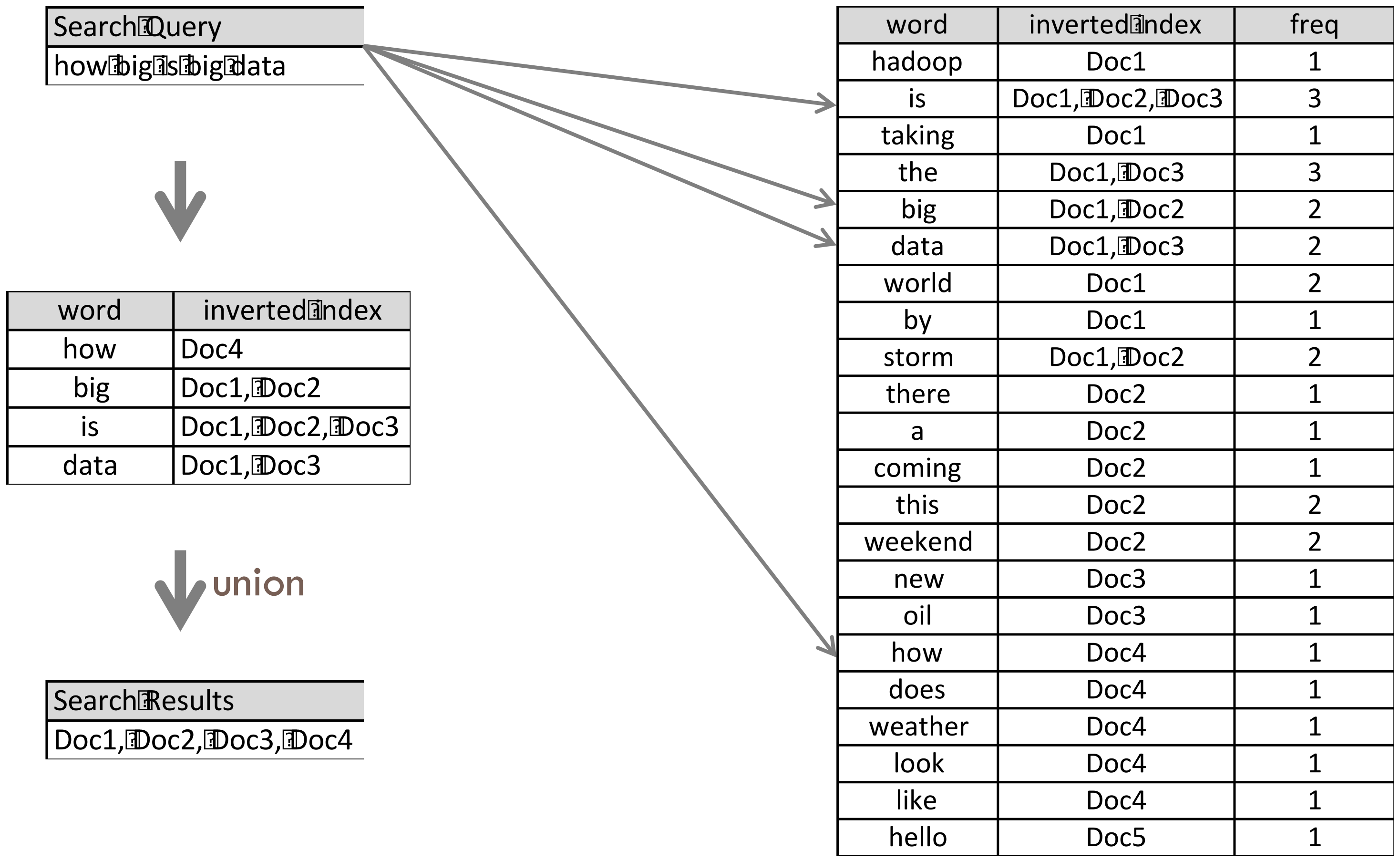
Documents	Text
Doc1	hadoop is taking the big data world by storm
Doc2	there is a big storm coming this weekend
Doc3	data is the new oil
Doc4	how does the weather look like this weekend
Doc5	hello world



word	inverted index	freq
hadoop	Doc1	1
is	Doc1, Doc2, Doc3	3
taking	Doc1	1
the	Doc1, Doc3	3
big	Doc1, Doc2	2
data	Doc1, Doc3	2
world	Doc1	2
by	Doc1	1
storm	Doc1, Doc2	2
there	Doc2	1
a	Doc2	1
coming	Doc2	1
this	Doc2	2
weekend	Doc2	2
new	Doc3	1
oil	Doc3	1
how	Doc4	1
does	Doc4	1
weather	Doc4	1
look	Doc4	1
like	Doc4	1
hello	Doc5	1

TF-IDF - Search Engine In a Nutshell

Inverted Index Document Retrieval



Search Query
how big is big data

word	inverted index
how	Doc4
big	Doc1, Doc2
is	Doc1, Doc2, Doc3
data	Doc1, Doc3

Search Results
Doc1, Doc2, Doc3, Doc4

word	inverted index	freq
hadoop	Doc1	1
is	Doc1, Doc2, Doc3	3
taking	Doc1	1
the	Doc1, Doc3	3
big	Doc1, Doc2	2
data	Doc1, Doc3	2
world	Doc1	2
by	Doc1	1
storm	Doc1, Doc2	2
there	Doc2	1
a	Doc2	1
coming	Doc2	1
this	Doc2	2
weekend	Doc2	2
new	Doc3	1
oil	Doc3	1
how	Doc4	1
does	Doc4	1
weather	Doc4	1
look	Doc4	1
like	Doc4	1
hello	Doc5	1

TF-IDF - Search Engine Basics

Relevance Scoring

Search Query
how big is big data

Query	TF	hits	IDF	TF-IDF
how	1	1	0.699	0.699
big	2	2	0.398	0.796
is	1	3	0.222	0.222
data	1	2	0.398	0.398

TF_t ➔ Frequency of *t* in a document

$IDF_t = \log(N/DF_t)$

TF-IDF for the word 'how':

$TF_{how} = 1$

$N = 5$

$TF_{how} = 1$

$IDF_{how} = \log(5/1) = 0.699$

$TF\text{-}IDF_{how} = TF_{how} \times IDF_{how} = 0.699$

Doc1	hits	TF	IDF	TF-IDF
hadoop	1	1	0.699	0.699
is	3	1	0.222	0.222
taking	1	1	0.699	0.699
the	3	1	0.222	0.222
big	2	1	0.398	0.398
data	2	1	0.398	0.398
world	2	1	0.398	0.398
by	1	1	0.699	0.699
storm	2	1	0.398	0.398

Doc2	hits	TF	IDF	TF-IDF
there	1	1	0.699	0.699
is	3	1	0.222	0.222
a	1	1	0.699	0.699
big	2	1	0.398	0.398
storm	2	1	0.398	0.398
coming	1	1	0.699	0.699
this	2	1	0.398	0.398
weekend	2	1	0.398	0.398

Doc3	hits	TF	IDF	TF-IDF
data	2	1	0.398	0.398
is	3	1	0.222	0.222
the	3	1	0.222	0.222
new	1	1	0.699	0.699
oil	1	1	0.699	0.699

Doc4	hits	TF	IDF	TF-IDF
how	1	1	0.699	0.699
does	1	2	0.699	1.398
the	3	3	0.222	0.666
weather	1	4	0.699	2.796
look	1	5	0.699	3.495
like	1	6	0.699	4.194
this	2	7	0.398	2.786
weekend	2	8	0.398	3.184

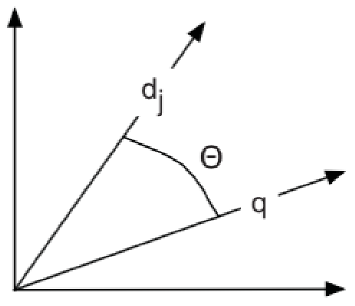
Doc5	hits	TF	IDF	TF-IDF
hello	1	1	0.699	0.699
world	2	2	0.398	0.796

TF-IDF - Search Engine Basics

Relevance Scoring

	hadoop	is	taking	the	big	data	world	by	storm	there	a	coming	this	weekend	new	oil	how	does	weather	like	hello
Doc1	0.699	0.222	0.699	0.222	0.398	0.398	0.398	0.699	0.398	0	0	0	0	0	0	0	0	0	0	0	0
Doc2	0	0	0	0	0.398	0	0	0	0.398	0.699	0.699	0.699	0.398	0.398	0	0	0	0	0	0	0
Doc3	0	0.222	0	0.222	0	0.398	0	0	0	0	0	0	0	0	0.699	0.699	0	0	0	0	0
Doc4	0	0	0	0.222	0	0	0	0	0	0	0	0	0.398	0.398	0	0	0.699	0.699	0.699	0.699	0

Query	0	0.398	0	0	0.796	0.398	0	0	0	0	0	0	0	0	0	0	0.699	0	0	0	0



$d_j = \langle w_{1,j}, w_{2,j}, \ldots, w_{n,j} \rangle$

$q = \langle w_{1,q}, w_{2,q}, \ldots, w_{n,q} \rangle$

w = weight assigned to term

Cosine similarity happens to be the dot product of two vectors

Relevancy	Score	Ranking
Query vs Doc1	0.56335	1
Query vs Doc4	0.48856	2
Query vs Doc2	0.31671	3
Query vs Doc3	0.24664	4

TF-IDF - Use Cases

- Application Categorization
  - Treat app store app descriptions as documents
  - Collect a corpus of apps and descriptions
  - Calculate TF-IDF and select significant keywords for each app
  - Send keywords to Amazon Mechanical Turks for human categorization
  - Build your machine learning classification models with the human lables
- Text Clustering/Classification
- Search Engine

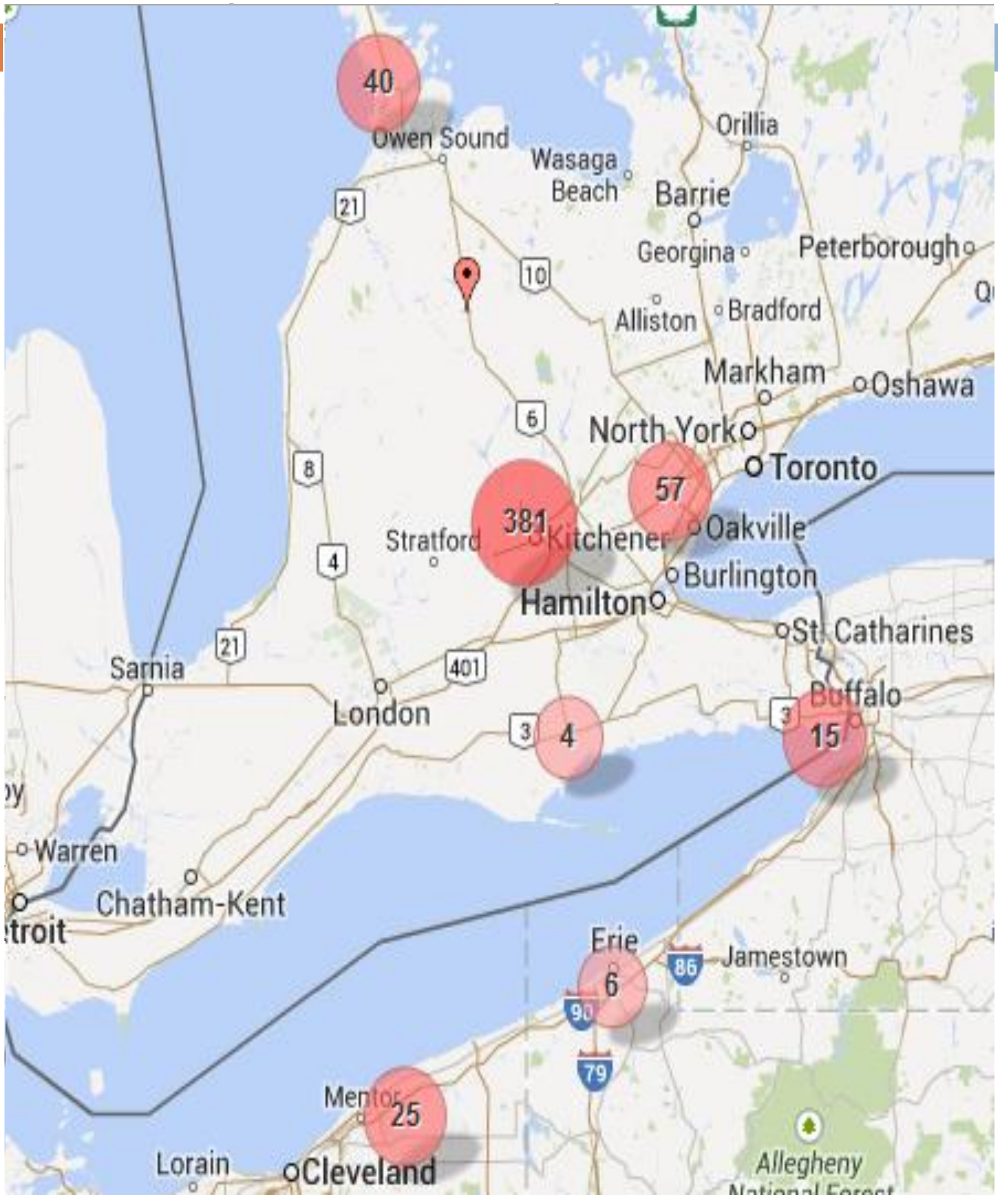
TF-IDF - *** Additional Lab ***

- TF-IDF
  - http://horicky.blogspot.ca/2009/01/solving-tf-idf-using-map-reduce.html
  - Dataset: shakespeare
    - all-shakespeare (text file)

Hadoop Use Case

Location Analytics

Hadoop Use Cases: Location Analytics



- Location data can be used to infer home/work location
- User activities tend to cluster around home/work locations
- Understanding use mobility
- Find traveling patterns
- Location clustering (K-means)

Location Analytics – Census Data

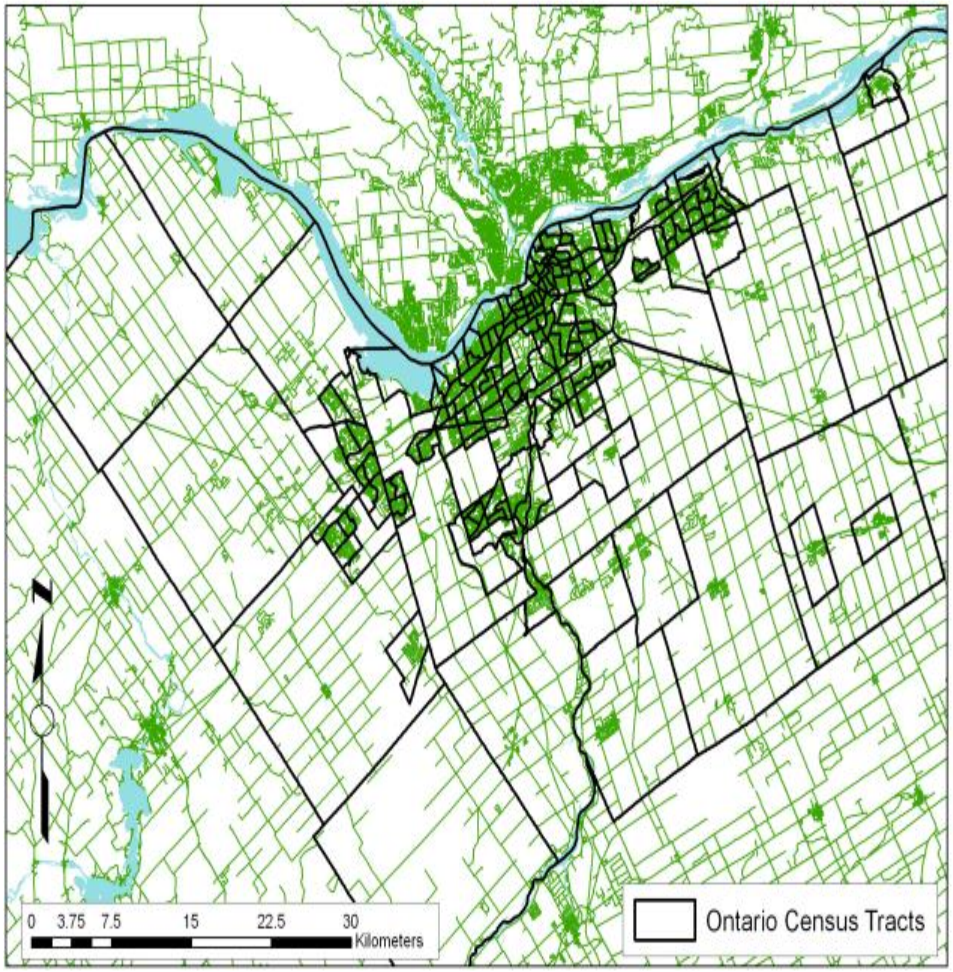
- Census data has a wealth of information
- You can repackage it, layering your insight and sell them
- Census Tract Level Detail
  - CT is a small geographic region with population size 5,000 ~ 8,000

Canada - National Household Survey Data (NHS)

- http://www12.statcan.gc.ca/nhs-enm/2011/dp-pd/prof/details/page.cfm?Lang=E&amp;Geo1=CT&amp;Code1=2709&amp;Data=Count&amp;SearchText=L6H7N3&amp;SearchType=Begins&amp;SearchPR=01&amp;A1=All&amp;B1=All&amp;Custom=&amp;TABID=2

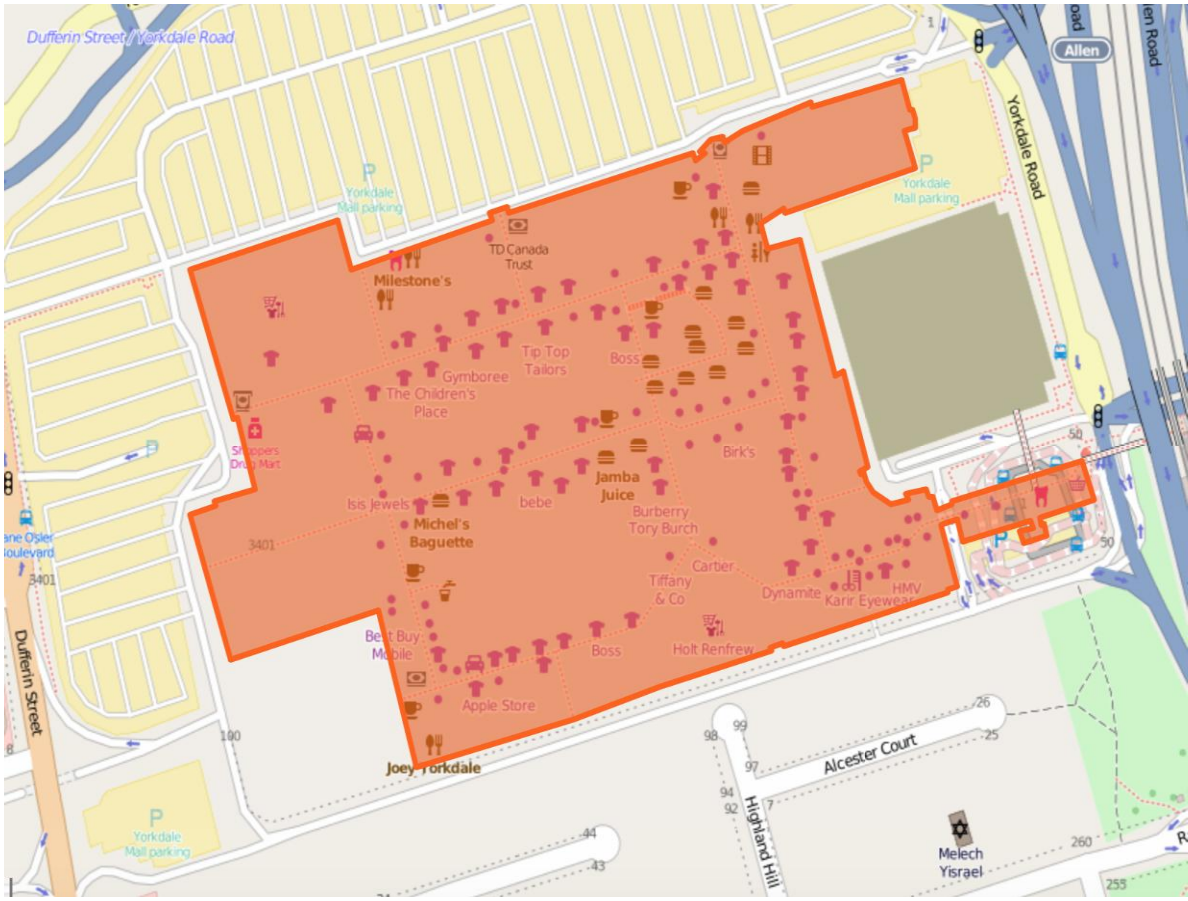
Canada - NHS Census Tract Level Data Dump

- http://www12.statcan.gc.ca/nhs-enm/2011/dp-pd/prof/details/download-telecharger/comprehensive/comp-ivt-xml-nhs-enm.cfm?Lang=E



Location Analytics – OpenStreetMap

- Point Of Interest Data APIs
  - Foursquare
  - Factual
  - Infogroup
  - OpenStreetMap

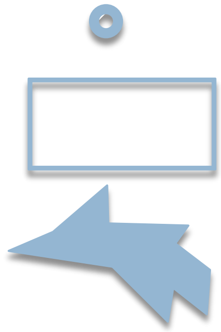


http://www.openstreetmap.org/way/19887459#map=17/43.72557/-79.45214

Location Analytics

Pigeon – A Pig UDF Library

- Spatial Data Types
  - Point
  - Rectangle
  - Polygon



- Lat-lon Accuracy

1	1	.	1	1	1	1	1	1	1	1	1
1000 km	111 km		11.1km	1.1km	110 m	11m	1.1m	0.11m	11mm	1.1mm	110 microns

Pigeon Functions

```
DEFINE ST_Area edu.umn.cs.pigeon.Area;
DEFINE ST_AsHex edu.umn.cs.pigeon.AsHex;
DEFINE ST_AsText edu.umn.cs.pigeon.AsText;
DEFINE ST_Buffer edu.umn.cs.pigeon.Buffer;
DEFINE ST_Connect edu.umn.cs.pigeon.Connect;
DEFINE ST_Contains edu.umn.cs.pigeon.Contains;
DEFINE ST_ConvexHull edu.umn.cs.pigeon.ConvexHull;
DEFINE ST_Crosses edu.umn.cs.pigeon.Crosses;
DEFINE ST_Envelope edu.umn.cs.pigeon.Envelope;
DEFINE ST_Extent edu.umn.cs.pigeon.Extent;
DEFINE ST_Intersection edu.umn.cs.pigeon.Intersection;
DEFINE ST_IsEmpty edu.umn.cs.pigeon.IsEmpty;
DEFINE ST_MakeLine edu.umn.cs.pigeon.MakeLine;
DEFINE ST_MakePoint edu.umn.cs.pigeon.MakePoint;
DEFINE ST_MakePolygon edu.umn.cs.pigeon.MakePolygon;
DEFINE ST_MakeLinePolygon edu.umn.cs.pigeon.MakeLinePolygon;
DEFINE ST_MakeBox edu.umn.cs.pigeon.MakeBox;
DEFINE ST_Overlaps edu.umn.cs.pigeon.Overlaps;
DEFINE ST_Size edu.umn.cs.pigeon.Size;
DEFINE ST_Union edu.umn.cs.pigeon.Union;
```

```
-- register and define functions
register /home/lab/pigeon-1.0-SNAPSHOT.jar;
register /home/lab/esri-geometry-api-1.2.jar;
DEFINE ST_Contains edu.umn.cs.pigeon.Contains;
DEFINE ST_MakePoint edu.umn.cs.pigeon.MakePoint;
DEFINE ST_MakePolygon edu.umn.cs.pigeon.MakePolygon;

-- load full_text data
data = load '/user/lab/pig/full_text.txt' AS (id:chararray, ts:chararray, location:chararray, lat:double, lon:double,
tweet:chararray);
state_polygon = load '/user/lab/pig/US_state_boundary.txt' as (state:chararray, seq:int, lat:double, lon:double);

-- make geometry point
data1 = FOREACH data GENERATE id, ts, lat, lon, ST_MakePoint(lat, lon) AS geom_point, tweet;
state_polygon_geom = FOREACH state_polygon GENERATE state, seq, ST_MakePoint(lat, lon) as state_polygon_geom;
state_polygon_geom_sort = order state_polygon_geom by state, seq;
grp = GROUP state_polygon_geom_sort by state;

-- make geometry polygon
state_polygon_geom = FOREACH grp GENERATE group as state,
ST_MakePolygon(state_polygon_geom_sort.state_polygon_geom) as geom_polygon;

-- cross join lat-lon geometry points with state geometry polygon
polygon_cross = cross data1, state_polygon_geom;

-- filter results by point-in-polygon
results = FILTER polygon_cross BY ST_Contains(state_polygon_geom::geom_polygon, data1::geom_point);
results_text = FOREACH results GENERATE data1::id, data1::ts, data1::lat, data1::lon, data1::tweet,
state_polygon_geom::state;

-- store data
STORE results_text INTO '/user/lab/pig/full_text_state';
```

Location Analytics - ** Additional lab **

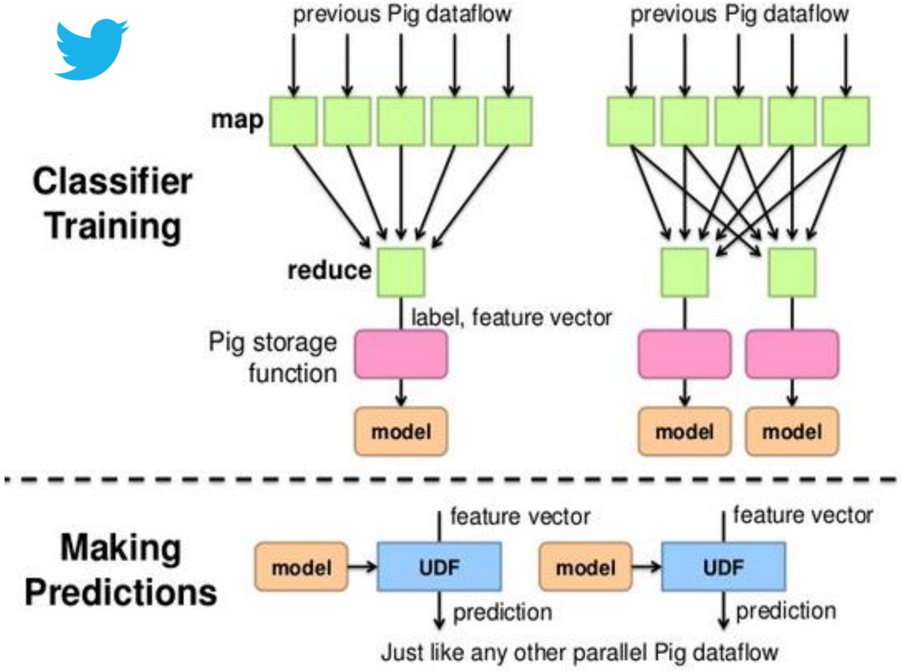
- In the geo-tagged tweet data, find top-5 tweeters per U.S. state
  - Top-5 tweeters: tweeters who had the most number of tweets in the dataset (distinct at timestamp level)
  - Leverage Pigeon pig UDFs for point-in-polygon operations
    - ST_Contains function in Pigeon
  - Need to use ST_MakePoint and ST_MakePolygon functions to turn lat-lons into geometry points and polygons first
  - Hint: cross-join is one potential solution

Hadoop Use Case

Machine Learning

Hadoop Use Cases: Machine Learning

- Use Pig for feature preparation
- Use Mappers to create random samples
- Leverage Reducers to train separate models
- Save models via PigStorage function
- Write UDF functions for model scoring
- Make predictions in pig



- Scaling Big Data Mining Infrastructure Twitter Experience: http://slidesha.re/1x0VLGX
- Recommendation at Netflix Scale: http://slidesha.re/1urTiEt