Authorship Identification with Amazon Review Data

Xiaohui Chen

I love this cellphone

This cellphone worked like a bomb and it totally ruined my life!!!

Whatever, I don't care

Buyer 1

Buyer 2

Buyer 3

Steps

- Feature Extraction
- Build ML Model
- Tune Model
- Prediction

An Anonymous Text comes in for another product

- “This laptop looks as if it is going to explode because it is always overheating. My life is destroyed by that thing!!!”
- Most likely from buyer 2

Dataset

- data in which all users and items have at least 5 reviews (41.13 million reviews)
- 30GB file size

Sample Review Text

- { "reviewerID": "A2SUAM1J3GNN3B", "asin": "0000013714", "reviewerName": "J. McDonald", "helpful": [2, 3], "reviewText": "I bought this for my husband who plays the piano. He is having a wonderful time playing these old hymns. The music is at times hard to read because we think the book was published for singing from more than playing from. Great purchase though!", "overall": 5.0, "summary": "Heavenly Highway Hymns", "unixReviewTime": 1252800000, "reviewTime": "09 13, 2009" }

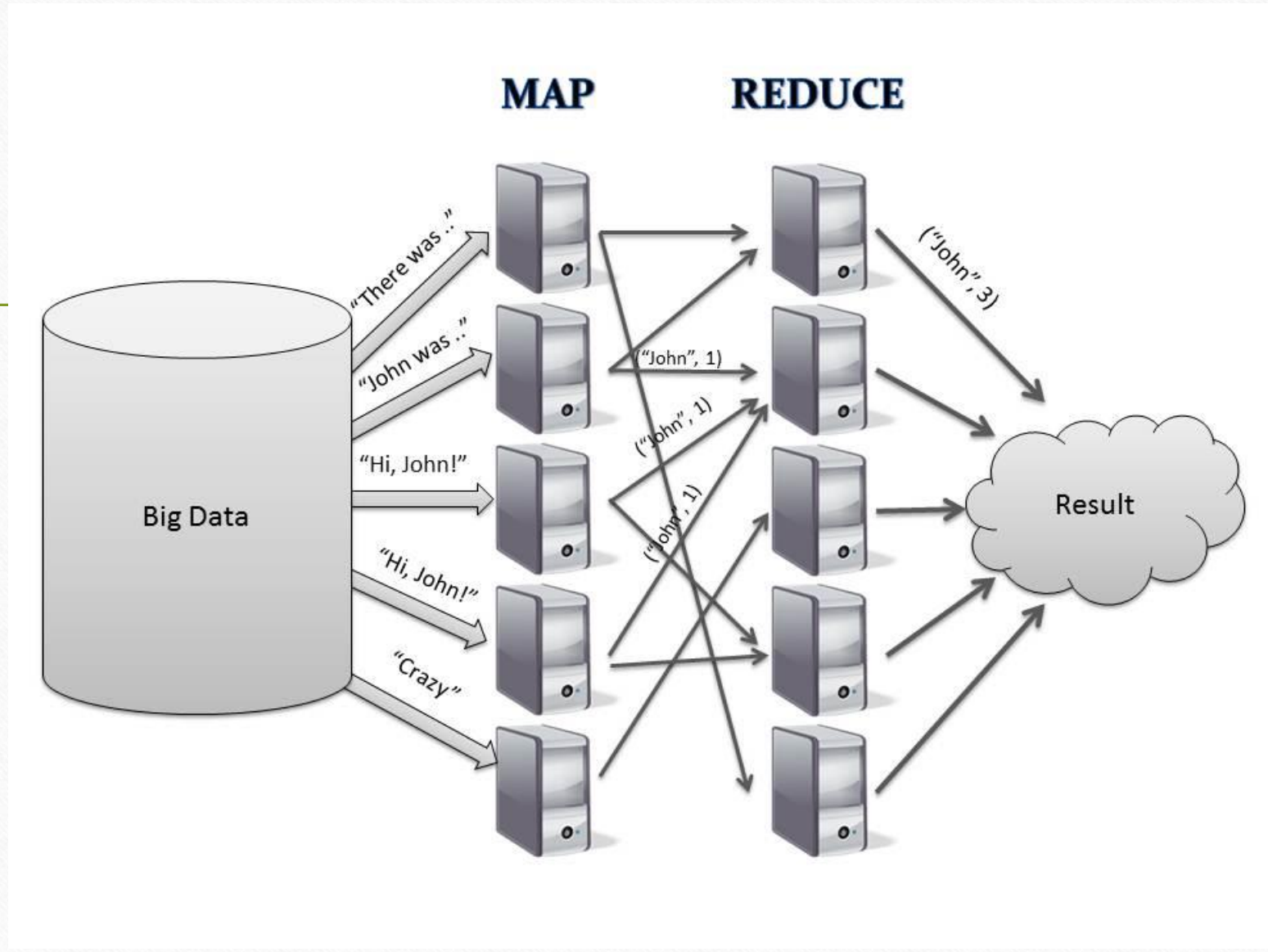
Steps for Feature Extraction

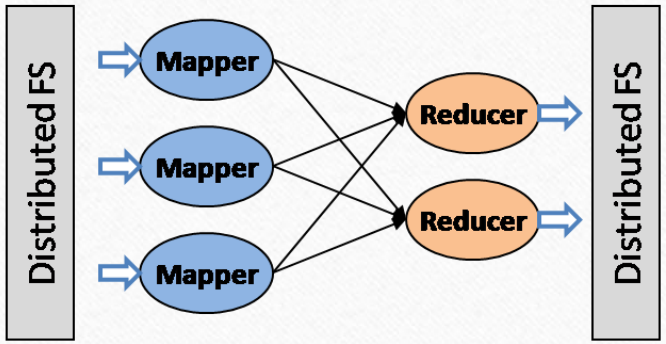
- Add review ID, remove unnecessary features, calculate helpful rate
- Sentiment Analysis (polarity, subjectivity)
- Use map reduce to calculate tfidf scores

TF-IDF

- Wikipedia: In information retrieval, tf–idf, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in information retrieval and text mining.

$$\mathrm{tf}_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \qquad \mathrm{idf}_i = \log \frac{|D|}{|\{j : t_i \in d_j\}|} \qquad \mathrm{tfidf}_{i,j} = \mathrm{tf}_{i,j} \times \mathrm{idf}_i$$


MAP
REDUCE
Big Data
"There was .."
"John was .."
"Hi, John!"
"Hi, John!"
"Crazy"
("John", 1)
("John", 1)
("John", 1)
("John", 3)
Result



tf_{wd} = $wordCount_{wd} / wordsPerDoc_{d}$

idf_{wd} = $log(totalDocs / docsPerWord_{w})$

$tf\text{-}idf_{wd} = tf_{wd} * idf_{wd}$

Round 1
wordCount

{docId => [words]} → {[word, docId] => 1} → {[word, docId] => [1, 1 …]} → {[word, docId] => wordCount}

Round 2
wordCount per doc

{[word, docId] => wordCount} → {docId => [word, wordCount]} → {docId => [[word1, wordCount1], [word2, wordCount2] …]} → {[word, docId] => [wordCount, wordsPerDoc]}

{[word, docId] => [wordCount, wordsPerDoc]} → {word => [docId, wordCount, wordsPerDoc]} → {word => [[doc1, wc1, wpd1], [doc2, wc2, wpd2] …]} → {[word, docId] => [wordCount, wordsPerDoc, docsPerWord]} → {[word, docId] => tfIdf}

Round 3
docCount per word

$tfidf_{wd}$ = $(wordCount_{wd} / wordsPerDoc_{d}) * log(totalDocs / docsPerWord_{w})$

Map Reduce Using Amazon EMR

- EMR: Elastic Map Reduce
- Configure to support Map Reduce
- Currently stuck on bootstrapping the EC2 instances

After Feature Extraction

- Put Data into Amazon Redshift (simple, train-test split)
- Use AWS Machine Learning to build the model (muti-class classification, can train dataset up to 100GB)
- For multiclass classification, Amazon ML uses multinomial logistic regression (multinomial logistic loss + SGD).

Questions?