Data-Intensive Distributed Computing
CS 431/631 451/651 (Fall 2019)

Part 3: Analyzing Text (1/2)
September 26, 2019

Ali Abedi

These slides are available at https://www.student.cs.uwaterloo.ca/~cs451

This work is licensed under a Creative Commons Attribution-Noncommercial-Share Alike 3.0 United States
See http://creativecommons.org/licenses/by-nc-sa/3.0/us/ for details
Structure of the Course

“Core” framework features and algorithm design
Structure of the Course

“Core” framework features and algorithm design

Analyzing Text
Analyzing Graphs
Analyzing Relational Data
Data Mining
Count.
Count (Efficiently)

class Mapper {
    def map(key: Long, value: String) = {
        for (word <- tokenize(value)) {
            emit(word, 1)
        }
    }
}

class Reducer {
    def reduce(key: String, values: Iterable[Int]) = {
        for (value <- values) {
            sum += value
        }
        emit(key, sum)
    }
}
Pairs. Stripes.
Seems pretty trivial...

More than a “toy problem”? Answer: language models
Language Models

\[ P(w_1, w_2, \ldots, w_T) \]

Assigning a probability to a sentence

Why?

- **Machine translation**
  - \( P(\text{High winds tonight}) > P(\text{Large winds tonight}) \)

- **Spell Correction**
  - \( P(\text{Waterloo is a great city}) > P(\text{Waterloo is a grate city}) \)

- **Speech recognition**
  - \( P(\text{I saw a van}) > P(\text{eyes awe of an}) \)
Language Models

\[ P(w_1, w_2, \ldots, w_T) \]

\[ = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2) \ldots P(w_T|w_1, \ldots, w_{T-1}) \]

[chain rule]

\[ P(\text{“Waterloo is a great city”}) = \]
\[ P(\text{Waterloo}) \times P(\text{is | Waterloo}) \times P(\text{a | Waterloo is}) \]
\[ \times P(\text{great | Waterloo is a}) \]
\[ \times P(\text{city | Waterloo is a great}) \]

Is this tractable?
Approximating Probabilities: $N$-Grams

Basic idea: limit history to fixed number of $(N - 1)$ words
(Markov Assumption)

$$P(w_k|w_1, \ldots, w_{k-1}) \approx P(w_k|w_{k-N+1}, \ldots, w_{k-1})$$

$N=1$: Unigram Language Model

$$P(w_k|w_1, \ldots, w_{k-1}) \approx P(w_k)$$

$$\Rightarrow P(w_1, w_2, \ldots, w_T) \approx P(w_1)P(w_2)\ldots P(w_T)$$
Approximating Probabilities: $N$-grams

Basic idea: limit history to fixed number of $(N - 1)$ words
(Markov Assumption)

\[ P(w_k|w_1, \ldots, w_{k-1}) \approx P(w_k|w_{k-N+1}, \ldots, w_{k-1}) \]

$N=2$: Bigram Language Model

\[ P(w_k|w_1, \ldots, w_{k-1}) \approx P(w_k|w_{k-1}) \]

\[ \Rightarrow P(w_1, w_2, \ldots, w_T) \approx P(w_1|<S>)P(w_2|w_1)\ldots P(w_T|w_{T-1}) \]
Approximating Probabilities: $N$-Grams

Basic idea: limit history to fixed number of $(N - 1)$ words
(Markov Assumption)

$$P(w_k|w_1, \ldots, w_{k-1}) \approx P(w_k|w_{k-N+1}, \ldots, w_{k-1})$$

$N=3$: Trigram Language Model

$$P(w_k|w_1, \ldots, w_{k-1}) \approx P(w_k|w_{k-2}, w_{k-1})$$

$$\Rightarrow P(w_1, w_2, \ldots, w_T) \approx P(w_1|<S><S>) \ldots P(w_T|w_{T-2}w_{T-1})$$
Building $N$-Gram Language Models

Compute maximum likelihood estimates (MLE) for Individual $n$-gram probabilities

Unigram

$$P(w_i) = \frac{C(w_i)}{N}$$

Bigram

$$P(w_i, w_j) = \frac{C(w_i, w_j)}{N}$$

$$P(w_j | w_i) = \frac{P(w_i, w_j)}{P(w_i)} = \frac{C(w_i, w_j)}{\sum_w C(w_i, w)} = \frac{C(w_i, w_j)}{C(w_i)}$$

Generalizes to higher-order $n$-grams

State of the art models use $\sim$5-grams

We already know how to do this in MapReduce!
The two commandments of estimating probability distributions...

Source: Wikipedia (Moses)
Probabilities must sum up to one
Thou shalt smooth

What? Why?
Example: Bigram Language Model

Training Corpus

<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

P( I | <s> ) = 2/3 = 0.67
P( am | I ) = 2/3 = 0.67
P( </s> | Sam )= 1/2 = 0.50

P( Sam | <s> ) = 1/3 = 0.33
P( do | I ) = 1/3 = 0.33
P( Sam | am) = 1/2 = 0.50

Bigram Probability Estimates
Note: We don’t ever cross sentence boundaries
Data Sparsity

P(I like ham) = P(I | <s>) P(like | I) P(ham | like) P(</s> | ham) = 0

Why is this bad?

Issue: Sparsity!
Thou shalt smooth!

Zeros are bad for any statistical estimator
Need better estimators because MLEs give us a lot of zeros
A distribution without zeros is “smoother”

The Robin Hood Philosophy: Take from the rich (seen $n$-grams)
and give to the poor (unseen $n$-grams)
Need better estimators because MLEs give us a lot of zeros
A distribution without zeros is “smoother”

Lots of techniques:
Laplace, Good-Turing, Katz backoff, Jelinek-Mercer
Kneser-Ney represents best practice
Laplace Smoothing

Simplest and oldest smoothing technique
Just add 1 to all $n$-gram counts including the unseen ones
So, what do the revised estimates look like?
Laplace Smoothing

Unigrams

$$P_{MLE}(w_i) = \frac{C(w_i)}{N} \quad \text{→} \quad P_{LAP}(w_i) = \frac{C(w_i) + 1}{N + V}$$

Bigrams

$$P_{MLE}(w_i, w_j) = \frac{C(w_i, w_j)}{N} \quad \text{→} \quad P_{LAP}(w_i, w_j) = \frac{C(w_i, w_j) + 1}{N + V^2}$$

What if we don’t know $V$?
Jelinek-Mercer Smoothing: Interpolation

Mix higher-order with lower-order models to defeat sparsity

\[ \text{Mix} = \text{Weighted Linear Combination} \]

\[
P(w_k|w_{k-2}w_{k-1}) = \lambda_1 P(w_k|w_{k-2}w_{k-1}) + \lambda_2 P(w_k|w_{k-1}) + \lambda_3 P(w_k)
\]

\[
0 \leq \lambda_i \leq 1 \quad \sum_{i} \lambda_i = 1
\]
Kneser-Ney Smoothing

Interpolate discounted model with a special "continuation" \( n \)-gram model

Based on appearance of \( n \)-grams in different contexts

Excellent performance, state of the art

\[
P_{Kn}(w_k|w_{k-1}) = \frac{C(w_{k-1}w_k) - D}{C(w_{k-1})} + \beta(w_k)P_{CONT}(w_k)
\]

\[
P_{CONT}(w_i) = \frac{N(\cdot w_i)}{\sum_{w'} N(\cdot w')}
\]

\( N(\cdot w_i) \) = number of different contexts \( w_i \) has appeared in
Kneser-Ney Smoothing: Intuition

I can’t see without my __________
“San Francisco” occurs a lot
I can’t see without my Francisco?
Stupid Backoff

Let’s break all the rules:

\[
S'(w_i | w_{i-k+1}^{i-1}) = \begin{cases} 
\frac{f(w_{i-k+1}^i)}{f(w_{i-k+1}^{i-1})} & \text{if } f(w_{i-k+1}^i) > 0 \\
\alpha S(w_i | w_{i-k+1}^{i-1}) & \text{otherwise}
\end{cases}
\]

\[
S(w_i) = \frac{f(w_i)}{N}
\]

But throw *lots* of data at the problem!
What the...
Stupid Backoff Implementation: Pairs!

Straightforward approach: count each order separately

- \( A \ B \)  
  \[ S(C|A \ B) = \frac{f(A \ B \ C)}{f(A \ B)} \]
- \( A \ B \ C \)  
  \[ S(D|A \ B) = \frac{f(A \ B \ D)}{f(A \ B)} \]
- \( A \ B \ E \)  
  \[ S(E|A \ B) = \frac{f(A \ B \ E)}{f(A \ B)} \]
  ...

More clever approach: count all orders together

- \( A \ B \)  
  remember this value
- \( A \ B \ C \)  
  remember this value
- \( A \ B \ C \ P \)  
  remember this value
- \( A \ B \ C \ Q \)
- \( A \ B \ D \)  
  remember this value
- \( A \ B \ D \ X \)
- \( A \ B \ D \ Y \)
  ...

Note: The equations are simplified for demonstration.
Stupid Backoff: Additional Optimizations

Replace strings with integers
Assign ids based on frequency (better compression using vbyte)

Partition by bigram for better load balancing
Replicate all unigram counts
State of the art smoothing (less data) vs. Count and divide (more data)

Source: Wikipedia (Boxing)
Statistical Machine Translation

Training Data
- "i saw the small table"
- "vi la mesa pequeña"
- "he sat at the table"
- "the service was good"
- "Target-Language Text"

Word Alignment
- (vi, i saw)
- (la mesa pequeña, the small table)

Phrase Extraction
- "vi la mesa pequeña"
- "Language Model"
- "Translation Model"
- "..."

Decoder
- "maria no daba una bofetada a la bruja verde"
- "mary did not slap the green witch"
- "English Output Sentence"

Training Data
- "maria"
- "no"
- "daba"
- "una"
- "bofetada"
- "a"
- "la"
- "bruja"
- "verde"

English Output Sentence
- "mary did not slap the green witch"

\[ \hat{e}_i^l = \arg\max_{e_i^l} P(e_i^l | f_i^l) = \arg\max_{e_i^l} P(e_i^l)P(f_i^l | e_i^l) \]
Translation as a Tiling Problem

\[
\hat{e}_1^l = \arg \max_{e_1^l} P(e_1^l | f_1^l) = \arg \max_{e_1^l} P(e_1^l)P(f_1^l | e_1^l)
\]
## Results: Running Time

<table>
<thead>
<tr>
<th></th>
<th>target</th>
<th>webnews</th>
<th>web</th>
</tr>
</thead>
<tbody>
<tr>
<td># tokens</td>
<td>237M</td>
<td>31G</td>
<td>1.8T</td>
</tr>
<tr>
<td>vocab size</td>
<td>200k</td>
<td>5M</td>
<td>16M</td>
</tr>
<tr>
<td># n-grams</td>
<td>257M</td>
<td>21G</td>
<td>300G</td>
</tr>
<tr>
<td>LM size (SB)</td>
<td>2G</td>
<td>89G</td>
<td>1.8T</td>
</tr>
<tr>
<td>time (SB)</td>
<td>20 min</td>
<td>8 hours</td>
<td>1 day</td>
</tr>
<tr>
<td>time (KN)</td>
<td>2.5 hours</td>
<td>2 days</td>
<td>–</td>
</tr>
<tr>
<td># machines</td>
<td>100</td>
<td>400</td>
<td>1500</td>
</tr>
</tbody>
</table>

Source: Brants et al. (EMNLP 2007)
Results: Translation Quality

Source: Brants et al. (EMNLP 2007)
What’s actually going on?

\[
P(e|f) = \frac{P(e) \cdot P(f|e)}{P(f)}
\]

\[
\hat{e} = \arg \max_e P(e)P(f|e)
\]

Source: http://www.flickr.com/photos/johnmueller/3814846567/in/pool-56226199@N00/
It's hard to recognize speech
It's hard to wreck a nice beach

$$P(e|f) = \frac{P(e) \cdot P(f|e)}{P(f)}$$

$$\hat{e} = \arg \max_e P(e)P(f|e)$$
Neural Networks
Have taken over...
The Central Problem in Search

Do these represent the same concepts?
Abstract IR Architecture

- **Query**: Representation Function → Query Representation → Comparison Function → Hits
- **Documents**: Representation Function → Document Representation → Index

Document acquisition (e.g., web crawling)
How do we represent text?
Remember: computers don’t “understand” anything!

“Bag of words”
Treat all the words in a document as index terms
Assign a “weight” to each term based on “importance”
(or, in simplest case, presence/absence of word)
Disregard order, structure, meaning, etc. of the words
Simple, yet effective!

Assumptions
Term occurrence is independent
Document relevance is independent
“Words” are well-defined
What’s a word?

天主教教宗若望保祿二世因感冒再度住進醫院。這是他今年第二度因同様的病因住院。

الناطق باسم - وقال مارك ريجيف إن شارون قبل - الخارجية الإسرائيلية الدعوة وسيقوم للمرة الأولى بزيارة تونس، التي كانت لفترة طويلة المقر الرسمي لمنظمة التحرير الفلسطينية بعد خروجها من لبنان عام 1982.

 Выступая в Мещанском суде Москвы экс-глава ЮКОСа заявил не совершал ничего противозаконного, в чем обвиняет его генпрокуратура России.

भारत सरकार ने आर्थिक सर्वेक्षण में वित्तीय वर्ष 2005-06 में सात फीसदी विकास दर हासिल करने का आकलन किया है और कर सुधार पर ज़ोर दिया है

日米連合で台頭中国に対処…アーミテージ前副長官提言

조재영 기자= 서울시는 25일 이명박 시장이 `행정중심복합도시" 건설안에 대해 `군대라도 동원해 막고싶은 심정"이라고 말했다는 일부 언론의 보도를 부인했다.
McDonald's slims down spuds

Fast-food chain to reduce certain types of fat in its french fries with new cooking oil.

NEW YORK (CNN/Money) - McDonald's Corp. is cutting the amount of "bad" fat in its french fries nearly in half, the fast-food chain said Tuesday as it moves to make all its fried menu items healthier.

But does that mean the popular shoestring fries won't taste the same? The company says no. "It's a win-win for our customers because they are getting the same great french-fry taste along with an even healthier nutrition profile," said Mike Roberts, president of McDonald's USA.

But others are not so sure. McDonald's will not specifically discuss the kind of oil it plans to use, but at least one nutrition expert says playing with the formula could mean a different taste.

Shares of Oak Brook, Ill.-based McDonald's (MCD: down $0.54 to $23.22, Research, Estimates) were lower Tuesday afternoon. It was unclear Tuesday whether competitors Burger King and Wendy's International (WEN: down $0.80 to $34.91, Research, Estimates) would follow suit. Neither company could immediately be reached for comment.

...
Counting Words...

documents \rightarrow \text{Bag of Words} \rightarrow \text{Inverted Index}

case folding, tokenization, stopword removal, stemming

syntax, semantics, word knowledge, etc.

\text{Documents}

\text{Bag of Words}

\text{Inverted Index}
What goes in each cell?

- boolean
- count
- positions

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>blue</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>cat</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>egg</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>fish</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>green</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>ham</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>hat</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>one</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>red</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>two</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Abstract IR Architecture
<table>
<thead>
<tr>
<th>Indexing: building this structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieval: manipulating this structure</td>
</tr>
<tr>
<td>Where have we seen this before?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Doc 1</th>
<th>Doc 2</th>
<th>Doc 3</th>
<th>Doc 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>one fish, two fish</td>
<td>red fish, blue fish</td>
<td>cat in the hat</td>
<td>green eggs and ham</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>blue</th>
<th>red</th>
<th>one</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cat</th>
<th>two</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>egg</th>
<th>green</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>fish</th>
<th>hat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>green eggs and ham</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>one</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>red</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>two</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
</tbody>
</table>
Doc 1
one fish, two fish

Doc 2
red fish, blue fish

Doc 3
cat in the hat

Doc 4
green eggs and ham

postings lists
Indexing: Performance Analysis

Fundamentally, a large sorting problem
Terms usually fit in memory
Postings usually don’t

How is it done on a single machine?
How can it be done with MapReduce?

First, let’s characterize the problem size:
Size of vocabulary
Size of postings
Vocabulary Size: Heaps’ Law

\[ M = kT^b \]

- \( M \) is vocabulary size
- \( T \) is collection size (number of documents)
- \( k \) and \( b \) are constants

Typically, \( k \) is between 30 and 100, \( b \) is between 0.4 and 0.6

Heaps’ Law: linear in log-log space

Surprise: Vocabulary size grows unbounded!
Heaps’ Law for RCV1

First 1,000,020 terms:
Predicted = 38,323
Actual = 38,365

k = 44
b = 0.49

Postings Size: Zipf’s Law

\[ f(k; s, N) = \frac{1/k^s}{\sum_{n=1}^{N} (1/n^s)} \]

N  number of elements  
k  rank  
s  characteristic exponent

Zipf’s Law: (also) linear in log-log space

Specific case of Power Law distributions

In other words:

A few elements occur very frequently
Many elements occur very infrequently
Zipf’s Law for RCV1

Fit isn’t that good... but good enough!


Manning, Raghavan, Schütze, Introduction to Information Retrieval (2008)
Zipf’s Law for Wikipedia

Rank versus frequency for the first 10m words in 30 Wikipedias (dumps from October 2015)
MapReduce: Index Construction

Map over all documents
Emit *term* as key, *(docid, tf)* as value
Emit other information as necessary (e.g., term position)

Sort/shuffle: group postings by term

Reduce
Gather and sort the postings (typically by *docid*)
Write postings to disk

MapReduce does all the heavy lifting!
# Inverted Indexing with MapReduce

**Map**

<table>
<thead>
<tr>
<th>Doc 1</th>
<th>one fish, two fish</th>
<th>Doc 2</th>
<th>red fish, blue fish</th>
<th>Doc 3</th>
<th>cat in the hat</th>
</tr>
</thead>
<tbody>
<tr>
<td>one</td>
<td></td>
<td>red</td>
<td></td>
<td>cat</td>
<td></td>
</tr>
<tr>
<td>two</td>
<td></td>
<td>blue</td>
<td></td>
<td>hat</td>
<td></td>
</tr>
<tr>
<td>fish</td>
<td></td>
<td>fish</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 1 1</td>
<td>2 1</td>
<td>3 1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Reduce**

<table>
<thead>
<tr>
<th></th>
<th>cat</th>
<th>blue</th>
<th>hat</th>
<th>two</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>3 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fish</td>
<td>1 2</td>
<td>2 2</td>
<td>3 1</td>
<td>1 1</td>
</tr>
<tr>
<td>one</td>
<td>1 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>red</td>
<td>2 1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Inverted Indexing: Pseudo-Code

class Mapper {
  def map(docid: Long, doc: String) = {
    val counts = new Map()
    for (term <- tokenize(doc)) {
      counts(term) += 1
    }
    for ((term, tf) <- counts) {
      emit(term, (docid, tf))
    }
  }
}

class Reducer {
  def reduce(term: String, postings: Iterable[(docid, tf)]) = {
    val p = new List()
    for ((docid, tf) <- postings) {
      p.append((docid, tf))
    }
    p.sort()
    emit(term, p)
  }
}

What’s the problem?

Stay tuned...