Parallel PLSI on Spark

Hongwei Wang and Bicheng Zhang
Department of Computer Science
{hwang172, zhang368} @ illinois.edu
Seeking Life’s Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—How many genes does an organism need to survive? Last week at the genome meeting here, two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today’s organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn’t be enough.

Although the numbers don’t match precisely, these predictions are not all that far apart, especially in comparison to the 75,000 genes in the human genome, notes Siv Anderson, a genomics analyst at Harvard University in Cambridge, MA. She arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game; particularly as more and more genomes are completely sequenced and analyzed. “It may be a way of organizing any newly sequenced genome,” explains Araceli Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an...
Probabilistic Latent Semantic Indexing (PLSI)

Generative Process of a \((d, w)\) pair:

- Choose a document \(d\) with \(P(d)\)
- Given \(d\), choose a topic \(z\) with \(P(z|d)\)
- Given \(z\), choose a word \(w\) with \(p(w|z)\)

Latent Variable

\[
P(d, w) = P(d)P(w|d) = P(d) \sum_z P(w|z)P(z|d)
\]

Joint probability of \((d, w)\)

\[
\prod_{d, w} P(d, w) = \prod_{d, w} P(d)P(w|d) = \prod_{d, w} P(d) \sum_z P(w|z)P(z|d)
\]

Likelihood of the whole corpus

Estimated Parameters
Expectation Maximization

$$\theta_t = \arg \max_{\theta} Q(\theta, \theta_{t-1})$$

$$P(z|d, w; \theta_{t-1}) = \frac{P_{t-1}(z|d)P_{t-1}(w|z)}{\sum_{z'} P_{t-1}(z'|d)P_{t-1}(w|z')}$$

E Step

$$P(w|z) = \frac{\sum_d n(d, w)P(z|d, w; \theta_{t-1})}{\sum_{d, w'} n(d, w')P(z|d, w'; \theta_{t-1})}$$

M Step

$$P(z|d) = \frac{\sum_w n(d, w)P(z|d, w; \theta_{t-1})}{\sum_w n(d, w)}$$

Iterative process to maximize likelihood to a local maxima
MapReduce Implementation

**Algorithm 1** Parallel PLSI

- initialize $P(z|d), P(w|z)$
- for each iteration do
  - carry out map reduce jobs
  - update $P(w|z), P(d|z)$
  - check likelihood
- end for
- output $P(w|z), P(d|z)$
Algorithm 2 Map(key, value)

Input: key = docIndex, value = n(d, w), P(z|d), P(w|z)
Output1: key = topicIndex-w-wordIndex, value = n(d, w) * P(z|d, w; θ_{t-1})
Output2: key = topicIndex-d-docIndex, value = P(d|z)
Output3: key = 1, value = likelihood

initialize an array sum_words to store \( \sum_w n(d, w) * P(z|d, w; \theta_{t-1}) \) for each topic
likelihood = 0
for Each word in vocabulary do
  // E-step
  for each topic do
    Compute \( P(z|d, w; \theta_{t-1}) = P(z|d) * P(w|z) \)
  end for
  // M-step
  for each topic do
    normalize \( P(z|d, w; \theta_{t-1}) \)
    compute \( n(d, w) * P(z|d, w; \theta_{t-1}) \)
    emit output1
    \( sum_words[z] = sum_words[z] + n(d, w) * P(z|d, w; \theta_{t-1}) \)
    likelihood = likelihood + n(d, w) * P(z|d, w; \theta_{t-1}) * [\log P(z|d) + \log P(w|z)]
  end for
  for each topic do
    compute \( P(z|d) = sum_words[z] / \sum_w n(d, w) \)
    emit output2
  end for
emit output3
Algorithm 3 Reduce(key, values)

Input: (key, values)
Output: (key, sum)

sum = 0
for each value in values do
    sum += value
end for
emit (key, sum)
Implement on Spark

Programming Model for Iterative MapReduce

Resilient Distributed Datasets (RDDs)
In Memory Computation
Data cache for efficient reuse
Support Python

Logistic regression in Hadoop and Spark
Experiment

5 EC2 instances
Spark-0.9.1
Hadoop-2.2.0
HDFS

10000 documents
58750 words
50 topics
### High frequent words within a topic

<table>
<thead>
<tr>
<th>Space</th>
<th>Country</th>
<th>Geophysics</th>
<th>Art</th>
<th>Australia</th>
</tr>
</thead>
<tbody>
<tr>
<td>space</td>
<td>countries</td>
<td>water</td>
<td>music</td>
<td>australia</td>
</tr>
<tr>
<td>stars</td>
<td>united</td>
<td>earth</td>
<td>people</td>
<td>south</td>
</tr>
<tr>
<td>light</td>
<td>country</td>
<td>mars</td>
<td>plays</td>
<td>land</td>
</tr>
<tr>
<td>pole</td>
<td>city</td>
<td>air</td>
<td>wrote</td>
<td>waves</td>
</tr>
<tr>
<td>universe</td>
<td>war</td>
<td>oxygen</td>
<td>written</td>
<td>food</td>
</tr>
<tr>
<td>red</td>
<td>world</td>
<td>planet</td>
<td>pop</td>
<td>large</td>
</tr>
<tr>
<td>north</td>
<td>union</td>
<td>things</td>
<td>composers</td>
<td>found</td>
</tr>
<tr>
<td>earth</td>
<td>queue</td>
<td>scientists</td>
<td>art</td>
<td>continent</td>
</tr>
<tr>
<td>energy</td>
<td>south</td>
<td>life</td>
<td>instrument</td>
<td>ocean</td>
</tr>
<tr>
<td>star</td>
<td>european</td>
<td>atmosphere</td>
<td>theatre</td>
<td>bird</td>
</tr>
<tr>
<td>time</td>
<td>parliament</td>
<td>liquid</td>
<td>shakespeare</td>
<td>australian</td>
</tr>
</tbody>
</table>
Conclusion & Future Work

• Implement local PLSI and distributed PLSI on Spark
• Optimize workflow and Spark features
• Compare with Hadoop