Distributed Text Mining with tm

Stefan Theußl$^1$  Ingo Feinerer$^2$  Kurt Hornik$^1$

Department of Statistics and Mathematics,  
WU Vienna University of Economics and Business$^1$

Institute of Information Systems, DBAI Group  
Technische Universität Wien$^2$

08.07.2009
Text Mining in R

▶ Highly interdisciplinary research field utilizing techniques from computer science, linguistics, and statistics
▶ Vast amount of textual data available in machine readable format:
  ▶ scientific articles, abstracts, books, . . .
  ▶ memos, letters, . . .
  ▶ online forums, mailing lists, blogs, . . .
▶ Steady increase of text mining methods (both in academia as in industry) within the last decade
Text Mining in R

- **tm Package**
- Tailored for
  - Plain texts, articles and papers
  - Web documents (XML, SGML, ...)
  - Surveys
- Available *transformations*: `stemDoc()`, `stripWhitespace()`, `tmTolower()`, ...
- Methods for
  - Clustering
  - Classification
  - Visualization
- Feinerer (2009) and Feinerer et al. (2008)
Motivation

▶ Data volumes (corpora) become bigger and bigger
▶ Many tasks, i.e. we produce output data via processing lots of input data
▶ Text mining methods are becoming more complex and hence computer intensive
▶ Want to make use of many CPUs
▶ Typically this is not easy (parallelization, synchronization, I/O, debugging, etc.)
▶ Need for an integrated framework
▶ preferably usable on large scale distributed systems

→ Main motivation: large scale data processing
Motivation

- Multi-processor environments and large scale compute clusters/clouds available for a reasonable price
- Integrated frameworks for parallel/distributed computing available (e.g., Hadoop)
- Thus, parallel/distributed computing is now easier than ever
- R already offers extensions to use this software (e.g. via hive, nws, Rmpi, snow, etc.)
Distributed Text Mining in R

Example: Stemming

- Erasing word suffixes to retrieve their radicals
- Reduces complexity
- Stemmers provided in packages Rstem\textsuperscript{1} and Snowball\textsuperscript{2}

Data:

- *Reuters-21578*: one of the most widely used test collection for text categorization research
- *New York Times Annotated Corpus*: > 1.6 million text files

\textsuperscript{1}Duncan Temple Lang (version 0.3-1 on Omegahat)
\textsuperscript{2}Kurt Hornik (version 0.0-6 on CRAN)
Distributed Text Mining in R

Difficulties:

- Large data sets
- Corpus typically loaded into memory
- Operations on all elements of the corpus (so-called *transformations*)

Strategies:

- Text mining using tm and MapReduce/hive\(^1\)
- Text mining using tm and MPI/snow\(^2\)

\(^{1}\)Stefan Theußl (version 0.1-1)
\(^{2}\)Luke Tierney (version 0.3-3)
The MapReduce Programming Model
The MapReduce Programming Model

- Programming model inspired by functional language primitives
- Automatic parallelization and distribution
- Fault tolerance
- I/O scheduling
- Examples: document clustering, web access log analysis, search index construction, ...
- Dean and Ghemawat (2004)

Hadoop (http://hadoop.apache.org/core/) developed by the Apache project is an open source implementation of MapReduce.
The MapReduce Programming Model

Figure: Conceptual Flow
The MapReduce Programming Model

A MapReduce implementation like Hadoop typically provides a distributed file system (DFS):

- Master/worker architecture (Namenode/Datanodes)
- Data locality
- Map tasks are applied to partitioned data
- Map tasks scheduled so that input blocks are on same machine
- Datanodes read input at local disk speed
- Data replication leads to fault tolerance
- Application does not care whether nodes are OK or not
Hadoop Streaming

- Utility allowing to create and run MapReduce jobs with any executable or script as the mapper and/or the reducer

```
$HADOOP_HOME/bin/hadoop jar $HADOOP_HOME/hadoop-streaming.jar
  -input inputdir
  -output outputdir
  -mapper ./mapper
  -reducer ./reducer
```
Hadoop InteractiVE (hive)
Hadoop InteractiVE (hive)

hive provides:

- Easy-to-use interface to Hadoop
- Currently, only Hadoop core (http://hadoop.apache.org/core/) supported
- High-level functions for handling Hadoop framework (hive_start(), hive_create(), hive_is_available(), etc.)
- DFS accessor functions in R (DFS_put(), DFS_list(), DFS_cat(), etc.)
- Streaming via Hadoop (hive_stream())
- Available on R-Forge in project RHadoop
Example: Word Count

Data preparation:

```r
> library("hive")
Loading required package: rJava
Loading required package: XML
> hive_start()
> hive_is_available()
[1] TRUE
> DFS_put("~/Data/Reuters/minimal", "/tmp/Reuters")
> DFS_list("/tmp/Reuters")
[1] "reut-00001.xml" "reut-00002.xml" "reut-00003.xml"
[4] "reut-00004.xml" "reut-00005.xml" "reut-00006.xml"
[7] "reut-00007.xml" "reut-00008.xml" "reut-00009.xml"
> head(DFS_read_lines("/tmp/Reuters/reut-00002.xml"))
[1] "<?xml version="1.0"?>"
[2] "<REUTERS TOPICS="NO" LEWISSPLIT="TRAIN" [...]"
[4] " <TOPICS/>
[5] " <PLACES>
[6] "  <D>usa</D>"
```
Example: Word Count

```r
mapper <- function(){
  mapred_write_output <- function(key, value)
    cat(sprintf("%s\t%s\n", key, value), sep = "")

  trim_white_space <- function(line)
    gsub("(^ +)|( +$)", "", line)
  split_into_words <- function(line)
    unlist(strsplit(line, "[[[:space:]]+"))

  con <- file("stdin", open = "r")
  while (length(line <- readLines(con, n = 1,
        warn = FALSE)) > 0) {
    line <- trim_white_space(line)
    words <- split_into_words(line)
    if(length(words))
      mapred_write_output(words, 1)
  }
  close(con)
}
```
Example: Word Count

```r
reducer <- function(){
  [...]  
env <- new.env(hash = TRUE)
con <- file("stdin", open = "r")
while (length(line <- readLines(con, n = 1,
    warn = FALSE)) > 0) {
  split <- split_line(line)
  word <- split$word
  count <- split$count
  if(nchar(word) > 0){
    if(exists(word, envir = env, inherits = FALSE)) {
      oldcount <- get(word, envir = env)
      assign(word, oldcount + count, envir = env)
    }
    else assign(word, count, envir = env)
  }
  close(con)
  for (w in ls(env, all = TRUE))
    cat(w, "\t", get(w, envir = env), "\n", sep = "")
}
```
Example: Word Count

```r
> hive_stream(mapper = mapper,
  reducer = reducer,
  input = "/tmp/Reuters",
  output = "/tmp/Reuters_out")
> DFS_list("/tmp/Reuters_out")
[1] "_logs" "part-00000"
> results <- DFS_read_lines(
  "/tmp/Reuters_out/part-00000")
> head(results)
[1] "-\t2" "--\t7"
[3] ":\t1" ".\t1"
[5] "0064</UNKNOWN>\t1" "0066</UNKNOWN>\t1"
> tmp <- strsplit(results, "\t")
> counts <- as.integer(unlist(lapply(tmp, function(x)
  x[[2]])))
> names(counts) <- unlist(lapply(tmp, function(x)
  x[[1]]))
> head(sort(counts, decreasing = TRUE))
the to and of at said
58 44 41 30 25 22
Distributed Text Mining in R

Solution (Hadoop):

- Data set copied to DFS (‘DistributedCorpus’)
- Only meta information about the corpus in memory
- Computational operations (Map) on all elements in parallel
- Work horse tmMap()
- Processed documents (revisions) can be retrieved on demand
Distributed Text Mining in R - Listing

```r
> library("tm")
Loading required package: slam
> input <- "~/Data/Reuters/reuters_xml"
> co <- Corpus(DirSource(input), [...])
> co
A corpus with 21578 text documents
> print(object.size(co), units = "Mb")
65.5 Mb

> source("corpus.R")
> source("reader.R")
> dc <- DistributedCorpus(DirSource(input), [...])
> dc
A corpus with 21578 text documents
> dc[[1]]
Showers continued throughout the week in [...]
> print(object.size(dc), units = "Mb")
1.9 Mb
```
Mapper, i.e. input to `hive_stream()` (called by `tmMap()`):

```r
mapper <- function(){
  require("tm")
  fun <- some_tm_method
  [
  con <- file("stdin", open = "r")
  while(length(line <- readLines(con, n = 1L, warn = FALSE)) > 0) {
    input <- split_line(line)
    result <- fun(input$value)
    if(length(result))
      mapred_write_output(input$key, result)
  }
  close(con)
}
```
Distributed Text Mining in R

Infrastructure:

- Development platform: 8-core Power 6 shared memory system
  - IBM System p 550
    | 4  | 128 |
    |----|-----|
    | 2-core IBM POWER6 @ 3.5 GHz | GB RAM |
- Computers of PC Lab used as worker nodes
  - 8 PCs with an Intel Pentium 4 CPU @ 3.2 GHz and 1 GB of RAM
  - Each PC has > 20 GB reserved for DFS

MapReduce framework:

- Hadoop (implements MapReduce + DFS)
- R (2.9.0) with tm (0.4) and hive (0.1-1)
- Code implementing ‘DistributedCorpus’
- Cluster installation coming soon (loose integration with SGE)
Benchmark

Wizard of Oz data set (PC Lab cluster):

- **Runtime**
  - Graph showing runtime in seconds (s) vs. number of CPUs (1 to 8).
  - The runtime decreases as the number of CPUs increases.

- **Speedup**
  - Graph showing speedup vs. number of CPUs (1 to 8).
  - The speedup increases linearly as the number of CPUs increases.
Benchmark

Reuters-21578:

▶ Single processor runtime (\texttt{lapply()}): > 30 min.
▶ \texttt{tm/hive} on 8-core SMP (\texttt{hive\_stream()}) : 4 min.
▶ \texttt{tm/snow} on 8 nodes of cluster\texttt{@WU} (\texttt{parLapply()}) : 2.13 min.
Lessons Learned

- Problem size has to be sufficiently large
- Requirement: access text documents in R via \[ \text{file path} \], thus location of texts in DFS important (currently: \( \text{ID} = \text{file path} \))
- Serialization difficult: how updating text IDs? Currently via meta information to each chunk (chunk name, position in the chunk)
- Remote file operation on DFS around 2.5 sec. (significantly reduced with Java implementation)
Conclusion

- Use of Hadoop in particular the DFS enhances handling of large corpora
- Significant speedup in text mining applications
- Thus, MapReduce has proven to be a useful abstraction
- Greatly simplifies distributed computing
- Developer focus on problem
- Implementations like Hadoop deal with messy details
  - different approaches to facilitate Hadoop’s infrastructure
  - language- and use case dependent
Thank You for Your Attention!

Stefan Theußl
Department of Statistics and Mathematics
e-mail: Stefan.Theussl@wu.ac.at
URL: http://statmath.wu.ac.at/~theussl

WU Vienna University of Economics and Business
Augasse 2–6, A-1090 Wien
Jeffrey Dean and Sanjay Ghemawat.  
MapReduce: Simplified data processing on large clusters.  

I. Feinerer  
*tm: Text Mining Package*, 2009  
URL http://CRAN.R-project.org/package=tm  
R package version 0.3-3

I. Feinerer, K. Hornik, and D. Meyer  
Text mining infrastructure in R  
ISSN 1548-7660  
URL http://www.jstatsoft.org/v25/i05