Distributed Text Mining with tm

Stefan Theußl[1] Ingo Feinerer[2] Kurt Hornik[1]

Institute for Statistics and Mathematics,
WU Vienna[1]

Institute of Information Systems, DBAI Group
Technische Universität Wien[2]

17.04.2010

Motivation

For illustration assume that a leader of a party is facing an upcoming election. He or she might be interessted in tracking the general sentiment about the parties in the target population via analyzing political media coverage in order to improve his or her campaign accordingly[1].

The New York Stock Exchange (NYSE) processes and stores a massive amount of data during trading days. Many news agencies like Reuters, Bloomberg, and further publishers provide news and stories partly in structured form through RSS feeds or grant paying customers access to their news databases.

Text mining might help to extract interesting patterns from news available on the Web.

[1] For the 2004 US Presidential Election, Scharl and Weichselbraun (2008) developed a webmining tool to analyze about half a million documents in weekly intervals.

Text Mining

Text Mining

- Highly interdisciplinary research field utilizing techniques from computer science, linguistics, and statistics
- Vast amount of textual data available in machine readable format:
  - Content of Websites (Google, Yahoo, etc.)
  - Scientific articles, abstracts, books, etc. (CiteSeerX Project, Epub Repositories, Gutenberg Project, etc.)
  - News feeds (Reuters and other news agencies)
  - Memos, letters, etc.
  - blogs, forums, mailing lists, Twitter etc.
- 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, etc.)
  - Surveys
- Available *transformations*: `stemDocument()`, `stripWhitespace()`, `tmTolower()`, etc.
- Methods for
  - Clustering
  - Classification
  - Visualization
- Feinerer (2010) and Feinerer et al. (2008)

Text Mining in R

Components of a text mining framework, in particular **tm**:

- Sources which abstract input locations (`DirSource()`, `VectorSource()`, etc.)
- Readers (`readPDF()`, `readPlain()`, `readXML()`, etc.)
- A (PlainText-) Document contains contents of the document and meta data
- A corpus contains one or several documents and corpus-level meta data (abstract class in R)

Pre-constructed corpora are available from `http://datacube.wu.ac.at`.

E.g., Reuters21578:

```
install.packages("tm.corpus.Reuters21578", repos =
"http://datacube.wu.ac.at")
```

Functions and Methods

Display
: The `print()` and `summary()` convert documents to a format so that R can display them. Additional meta information can be shown via `summary()`.

Length
: The `length()` function returns the number of documents in the corpus.

Subset
: The `[[` operator must be implemented so that individual documents can be extracted from a corpus.

Apply
: The `tm_map()` function which can be conceptually seen as an `lapply()` implements functionality to apply a function to a range of documents.

Example: Handling Corpora in R

```
> library("tm")

> corpus <- Corpus(DirSource("Data/reuters"), list(reader = readReut21578XML))

> library("tm.corpus.Reuters21578")
> data(Reuters21578)
> Reuters21578

A corpus with 21578 text documents

> length(Reuters21578)

[1] 21578

> Reuters21578[[3]]

Texas Commerce Bancshares Inc's Texas
Commerce Bank-Houston said it filed an application with the
Comptroller of the Currency in an effort to create the largest
banking network in Harris County.
    The bank said the network would link 31 banks having
13.5 billion dlrs in assets and 7.5 billion dlrs in deposits.

 Reuter
```

Preprocessing

Stemming:

- Erasing word suffixes to retrieve their radicals
- Reduces complexity almost without loss of information
- Stemmers provided in packages **Rstem**[1] and **Snowball**[2](preferred) based on Porter (1980)
- Function `stemDocument()`

Stopword removal:

- Words with a very low entropy
- Based on base function `gsub()`
- Function `removeWords()`
- Removal of whitespace (`stripWhitespace()`) and punctuation (`removePunctuation()`) work similar

[1] Duncan Temple Lang (version 0.3-1 on Omegahat)
[2] Kurt Hornik (version 0.0-7 on CRAN)

Example: Preprocessing

```
> stemmed <- tm_map(Reuters21578[1:5], stemDocument)
> stemmed[[3]]

Texa Commerc Bancshar Inc Texas
Commerc Bank-Houston said it file an applic with the
Comptrol of the Currenc in an effort to creat the largest
bank network in Harri County.
    The bank said the network would link 31 bank having
13.5 billion dlrs in asset and 7.5 billion dlrs in deposits.

 Reuter

> removed <- tm_map(stemmed, removeWords, stopwords("english"))
> removed[[3]]

Texa Commerc Bancshar Inc Texas
Commerc Bank-Houston   file  applic
Comptrol   Currenc   effort  creat  largest
bank network  Harri County.
    The bank   network  link 31 bank
13.5 billion dlrs  asset  7.5 billion dlrs  deposits.

 Reuter
```

Document-Term Matrices

A very common approach in text mining for actual computation on texts is to build a so-called *document-term matrix* (DTM) holding frequencies of distinct terms $tf_{i,j}$, i.e., the *term frequency* (TF) of each term t_i for each document d_j. Its construction typically involves pre-processing and counting TFs for each document.

$$tf_{i,j} = n_{i,j}$$

where $n_{i,j}$ is the number of occurrences of the considered term i in document j.

DTMs are stored using a simple sparse (triplet) representation implemented in package **slam** by Hornik et al. (2010).

Document-Term Matrices

- Counting TFs is problematic regarding relevancy in the corpus
- E.g., (stemmed) terms like signatures occurs in almost all documents in the corpus
- Typically, the *inverse document frequency* (IDF) is used to suitably modify the TF weight by a factor that grows with the *document frequency df*

$$idf_i = \log \frac{N}{df_i}$$

- Combining both, the TF and IDF weighting we get the *term frequency - inverse document frequency* (*tf-idf*)

$$tf\text{-}idf_{i,j} = tf_{i,j} \times idf_i$$

Example: Document-Term Matrices

```
> Reuters21578_DTM <- DocumentTermMatrix(Reuters21578, list(stemming = TRUE,
+     removePunctuation = TRUE))
> data(Reuters21578_DTM)
> Reuters21578_DTM
A document-term matrix (21578 documents, 33090 terms)

Non-/sparse entries: 877918/713138102
Sparsity           : 100%
Maximal term length: 30
Weighting          : term frequency (tf)
> inspect(Reuters21578_DTM[51:54, 51:54])
A document-term matrix (4 documents, 4 terms)

Non-/sparse entries: 0/16
Sparsity           : 100%
Maximal term length: 10
Weighting          : term frequency (tf)

    Terms
Docs abdul abdulaziz abdulhadi abdulkadir
  51     0         0         0          0
  52     0         0         0          0
  53     0         0         0          0
  54     0         0         0          0
```

Challenges

- Data volumes (corpora) become bigger and bigger
- Many tasks, i.e. we produce output data via processing lots of input data
- Processing large data sets in a single machine is limited by the available main memory (i.e., RAM)
- 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**

Distributed Text Mining in R

Data sets:

- *Reuters-21578*: one of the most widely used test collection for text categorization research (news from 1987)
- *NSF Research Award Abstracts (NSF)*: consists of 129,000 plain text abstracts describing NSF awards for basic research submitted between 1990 and 2003. It is divided into three parts.
- *Reuters Corpus Volume 1 (RCV1)*: > 800.000 text documents
- *New York Times Annotated Corpus (NYT)*: > 1.8 million articles articles published by the New York Times between 1987-01-01 and 2007-06-19

	# documents	corpus size [MB][1]	size DTM [MB]
Reuters-21578	21,578	87	16.4
NSF (Part 1)	51,760	236	101.4
RCV1	806,791	3,800	1130.8
NYT	1,855,658	16,160	NA

[1] calculated with the Unix tool `du`

Opportunities

- Distributed computing environments are scalable in terms of CPUs and memory (disk space and RAM) employed.
- 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**, **RHIPE**, **nws**, **iterators**, **multicore**, **Rmpi**, **snow**, etc.

Employing such systems with the right tools we can significantly reduce runtime for processing large data sets.

Distributed Text Mining in R

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 0.1-2)

[2] Luke Tierney (version 0.3-3)

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

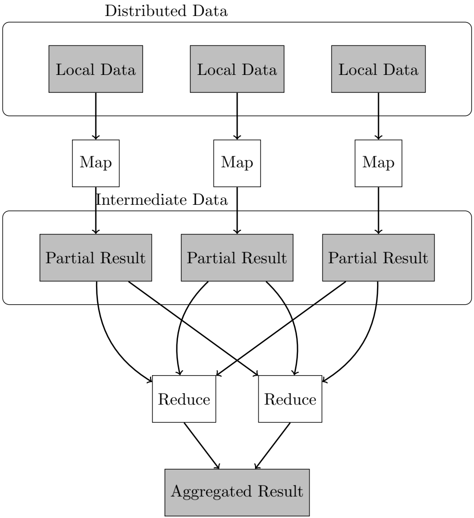


Figure: Conceptual Flow

The MapReduce Programming Model

A MapReduce implementation like Hadoop typically provides a distributed file system (DFS, Ghemawat et al., 2003):

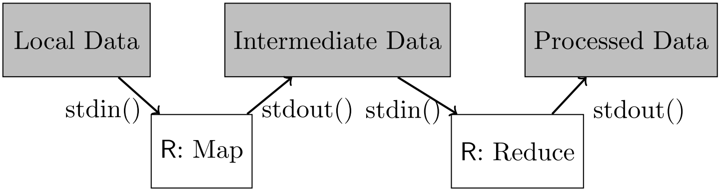
- Master/worker architecture (Namenode/Datanodes)
- Data locality
- Map tasks are applied to partitioned data
- Map tasks scheduled so that input blocks are on same machine
- Datanodes read input at local disk speed
- Data replication leads to fault tolerance
- Application does not care whether nodes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

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:

```
> 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\" [...]
[3] " <DATE>26-FEB-1987 15:03:27.51</DATE>"
[4] " <TOPICS/>"
[5] " <PLACES>"
[6] "  <D>usa</D>"
```

Distributed Text Mining in R

Solution:

1. Distributed storage
   - Data set copied to DFS (‘`DistributedCorpus`’)
   - Only meta information about the corpus remains in memory
2. Parallel computation
   - Computational operations (*Map*) on all elements in parallel
   - MapReduce paradigm
   - Work horses `tm_map()` and `TermDocumentMatrix()`

Processed documents (revisions) can be retrieved on demand.

Implemented in a “plugin” package to **tm**: **tm.plugin.dc**.

Distributed Text Mining in R

```
> library("tm.plugin.dc")

> dc <- DistributedCorpus(DirSource("Data/reuters"),
+                         list(reader = readReut21578XML) )

> dc <- as.DistributedCorpus(Reuters21578)
> summary(dc)

A corpus with 21578 text documents

The metadata consists of 2 tag-value pairs and a data frame
Available tags are:
  create_date creator
Available variables in the data frame are:
  MetaID

--- Distributed Corpus ---
Available revisions:
  20100417144823
Active revision: 20100417144823
DistributedCorpus: Storage
- Description: Local Disk Storage
- Base directory on storage: /tmp/RtmpuxX3W7/file5bd062c2
- Current chunk size [bytes]: 10485760

> dc <- tm_map(dc, stemDocument)
```

Distributed Text Mining in R

```
> print(object.size(Reuters21578), units = "Mb")

109.5 Mb

> dc

A corpus with 21578 text documents

> dc_storage(dc)

DistributedCorpus: Storage
- Description: Local Disk Storage
- Base directory on storage: /tmp/RtmpuxX3W7/file5bd062c2
- Current chunk size [bytes]: 10485760

> dc[[3]]

Texas Commerce Bancshares Inc's Texas
Commerce Bank-Houston said it filed an application with the
Comptroller of the Currency in an effort to create the largest
banking network in Harris County.
    The bank said the network would link 31 banks having
13.5 billion dlrs in assets and 7.5 billion dlrs in deposits.

 Reuter

> print(object.size(dc), units = "Mb")

0.6 Mb
```

Constructing DTMs via MapReduce

- Parallelization of transformations via `tm_map()`
- Parallelization of DTM construction by appropriate methods
- Via Hadoop streaming utility (R interface `hive_stream()`)
- Key / Value pairs: docID / tmDoc (document ID, serialized **tm** document)
- Differs from MPI/snow approach where an `lapply()` gets replaced by a `parLapply()`

Constructing DTMs via MapReduce

1. Input: <docID, tmDoc>
2. Preprocess (Map): <docID, tmDoc> $\rightarrow$ <term, docID, tf>
3. Partial combine (Reduce): <term, docID, tf> $\rightarrow$ <term, list(docID, tf)>
4. Collection: <term, list(docID, tf)> $\rightarrow$ DTM

Distributed Text Mining in R

Infrastructure:

bignode.q – 4 nodes	
2	Dual Core Intel XEON 5140 @ 2.33 GHz
16	GB RAM
node.q – 68 nodes	
1	Intel Core 2 Duo 6600 @ 2.4 GHz
4	GB RAM

This is a total of 152 64-bit computation nodes and a total of 336 gigabytes of RAM.

MapReduce framework:

- Hadoop 0.20.1 (implements MapReduce + DFS)
- R (2.10.1) with **tm** (0.5-2) and **hive** (0.1-2)
- Code implementing '`DistributedCorpus`' in (tm.plugin.dc)

Benchmark

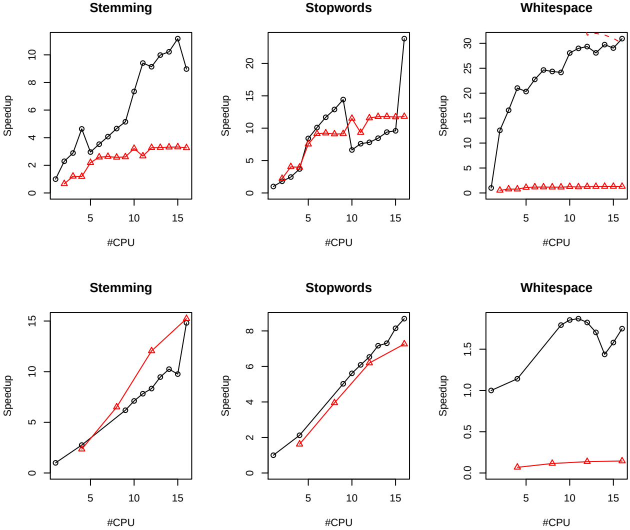


Figure: Runtime in seconds for stemming, stopword removal, and whitespace removal on the full Reuters-21578 data set (above) and on

Outlook and Conclusion

Outlook - Computing on Texts

```
> library("slam")
> cs <- col_sums(Reuters21578_DTM)
> (top20 <- head(sort(cs, decreasing = TRUE), n = 20))

    mln    dlrs  reuter     pct compani    bank billion   share     cts  market
  25513   20528   19973   17013   11396   11170   10240    9627    8847    7869
  price   trade     inc     net   stock    corp    loss    rate    sale    oper
   6944    6865    6695    6070    6050    6005    5719    5406    5138    4699

> DTM_tfidf <- weightTfIdf(Reuters21578_DTM)
> DTM_tfidf

A document-term matrix (21578 documents, 33090 terms)

Non-/sparse entries: 877918/713138102
Sparsity           : 100%
Maximal term length: 30
Weighting          : term frequency - inverse document frequency (normalized) (

> cs_tfidf <- col_sums(DTM_tfidf)
> cs_tfidf[names(top20)]

      mln      dlrs    reuter       pct   compani      bank   billion     share
 780.2629  544.7084  103.8984  455.8033  347.4688  355.8040  388.7671  421.8325
      cts    market     price     trade       inc       net     stock      corp
1119.2683  217.8757  235.4840  216.8312  341.8051  572.6022  290.0943  292.2116
     loss      rate      sale      oper
 552.9568  197.7876  320.2611  253.8356
```

Outlook - Sentiment Analysis

- Compute sentiment scores based on e.g., daily news
- Use (normalized) TF-IDF scores
- Currently *General Inquirer* tag categories are employed (provided in package **tm.plugin.tags**)
- Construct time series of tagged documents

```
> library("tm.plugin.tags")
> head(tm_get_tags("Positiv", collection = "general_inquirer"))

[1] "abide"     "ability"   "able"      "abound"    "absolve"   "absorbent
```

- Compare with time series of interest (e.g., of a financial instrument)

Outlook - Sentiment Analysis

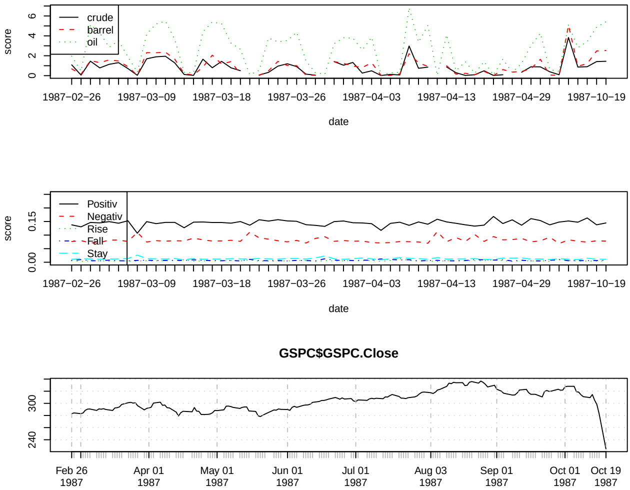


Figure: Time series of scores (normalized TF) for barrel-crude-oil, sentiment scores, and the stock price of CBT.

Conclusion - tm and Hadoop

- 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

Conclusion - Text Mining in R

The complete text mining infrastructure consists of many components:

- **tm**, text mining package (0.5-3.2, Feinerer, 2010)
- **slam**, sparse lightweigt arrays and matrices (0.1-11, Hornik et.al., 2010)
- **tm.plugin.dc**, distributed corpus plugin (0.1-2, Theussl and Feinerer, 2010)
- **tm.plugin.tags**, tag category database (0.0-1, Theussl, 2010)
- **hive**, Hadoop/MapReduce interface (0.1-5, Theussl and Feinerer, 2010)

Two of them are are released on CRAN (**tm**, **slam**), two are currently in an advanced development stage on R-Forge in project *RHadoop* (**hive**, **tm.plugin.dc**), and one will be released shortly (**tm.plugin.tags**).

- Eventually, combine everything in a news mining package

Thank You for Your Attention!

Stefan Theußl
Institute for Statistics and Mathematics
email: `Stefan.Theussl@wu.ac.at`
URL: `http://statmath.wu.ac.at/~theussl`

WU Vienna
Augasse 2–6, A-1090 Wien

References

J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. In *Proceedings of the Sixth Symposium on Operating System Design and Implementation*, pages 137–150, 2004. URL http://labs.google.com/papers/mapreduce.html.

I. Feinerer. **tm**: *Text Mining Package*, 2010. URL http://tm.r-forge.r-project.org/. R package version 0.5-3.

I. Feinerer, K. Hornik, and D. Meyer. Text mining infrastructure in R. *Journal of Statistical Software*, 25(5):1–54, March 2008. ISSN 1548-7660. URL http://www.jstatsoft.org/v25/i05.

S. Ghemawat, H. Gobioff, and S. Leung. The Google File System. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, pages 29–43, New York, NY, USA, October 2003. ACM Press. doi: http://doi.acm.org/10.1145/1165389.945450.

K. Hornik, D. Meyer, and C. Buchta. *slam: Sparse Lightweight Arrays and Matrices*, 2010. URL http://CRAN.R-project.org/package=slam. R package version 0.1-9.

M. Porter. An algorithm for suffix stripping. *Program*, 3:130–137, 1980.

A. Scharl and A. Weichselbraun. An automated approach to investigating the online media coverage of US presidential elections. *Journal of Information Technology and Politics*, 5(1):121–132, 2008.