WIRTSCHAFTS
UNIVERSITÄT
WIEN VIENNA
UNIVERSITY OF
ECONOMICS
AND BUSINESS

Do Media Sentiments Reflect Economic Indices

Outline

Motivation I

Sentiment analysis attempts to extract opinions from written text in an automatic manner. Most of the effort has been in mining *human generated* text, like reviews, blogs, etc.
Here we use a simple algorithm to extract **sentiment scores** from newspaper articles and relate these scores to an economic index.
Our objective is to use Support Vector Machines (SVM) to study the relationship between the sentiments in the newspapers and the current economic welfare expressed by the index in order to forecast future directions of the economy index.

Motivation II

Media news affect the expectations about the economy in several ways:

- Media covers the economic data and the outlook of the economy.
- Decisions of agents may be influenced by the media.
- Through the tone of articles, individuals get a picture about the significance of a topic.
- News coverage may influence people in their consumption behavior.

New York Times

Documents of the New York Times (NYT).

- *New York Times Annotated Corpus* contains over 1.8 million short- to medium length articles (on average $\sim$ 3303 characters).
- Corpus is drawn from historical archive of the New York Times and includes metadata provided by The New York Times Newsroom, NYT Indexing Service and the online NYTimes.com.
- It contains nearly every article published in the NYT between 1987-01-01 and 2007-06-19.
- NYT Documents are provided in a XML like format – News Industry Text Format (NITF).
- Size of the corpus (appr. 16GB)

CFNAI I

NYT sentiments are used to “forecast” CFNAI index

- Chicago Fed National Activity Index (CFNAI) is a weighted average of 85 monthly indicators of national economic activity.
- The CFNAI provides a single summary measure of a common factor in these national economic data.
- CFNAI are drawn from four broad categories of data: 1) production and income (23 series), 2) employment and hours (24 series), 3) personal consumption and housing (15 series) and 4) sales, orders, and inventories (23 series).
- The CFNAI is a weighted average of the 85 economic indicators.

CFNAI II

Interpretation of CFNAI:

- A zero value for the index indicates that the national economy is expanding at its historical trend rate of growth.
- Negative values indicate below-average growth; and positive values indicate above-average growth.
- CFNAI< -0.7: Increasing likelihood that a recession has begun.
- CFNA> 0.2: Significant likelihood that a recession has ended.
- CFNAI> 1: Substantial likelihood that a period of sustained increasing inflation has begun.

CFNAI III

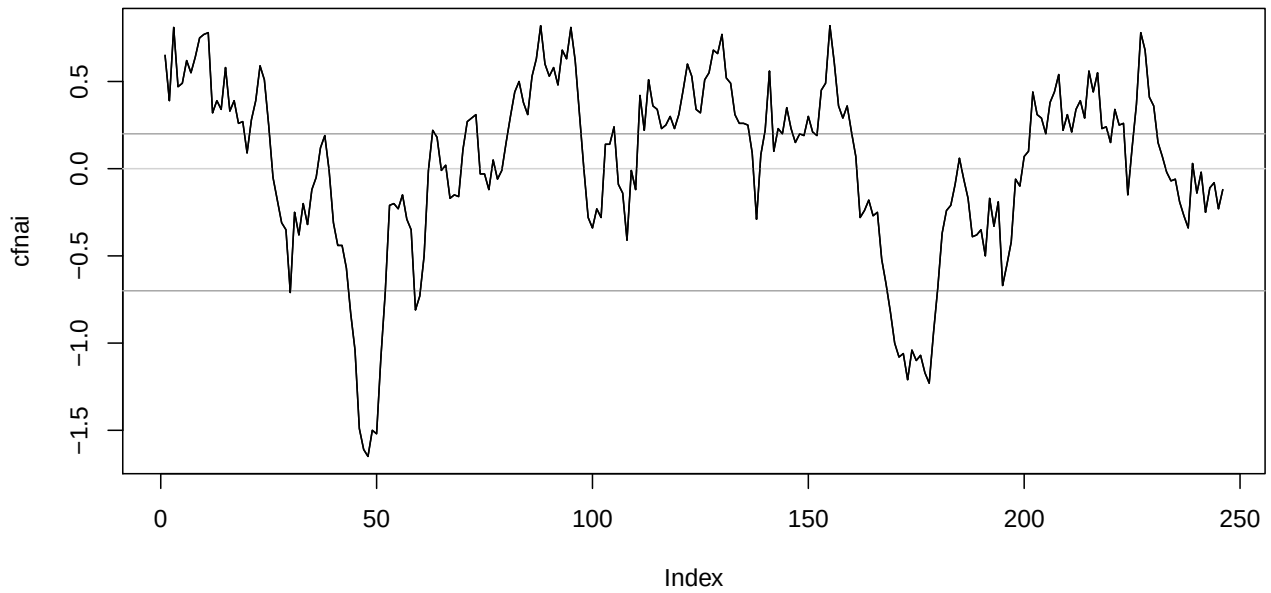

Figure: The CFNAI index

But how to derive sentiments from electronically published newspaper articles?

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)
  - blogs, forums, mailing lists, Twitter etc.
- Steady increase of text mining methods (both in academia as in industry) within the last decade.
- Tools available for convenient text handling (we use R and the **tm** package).

Note also the seminar "Statistical Natural Language Processing" this fall.

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)

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
```

Large Datasets?

Distributed text mining in R.

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**.

Example: Distributed Text Mining

```
> library("tm.plugin.dc")
> dc <- DistributedCorpus(DirSource("Data/reuters"),
+                         list(reader = readReut21578XML) )
> 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:
  20100909104609-0-c
Active revision: 20100909104609-0-c
DistributedCorpus: Storage
- Type: local_disk
- Base directory on storage: /tmp/RtmpzWC3Wn/file4db127f8
- Current chunk size [bytes]: 10485760
```

Example: Corpora

Pre-constructed corpora are available from `http://datacube.wu.ac.at`. E.g., “free” corpora like Reuters21578 directly installable via `install.packages("tm.corpus.Reuters21578", repos = "http://datacube.wu.ac.at")` and “non-free” corpora like RCV1 or NYT upon request.

```
> library("tm.corpus.NYTimes")
> data("NYT")
> length(NYT)
[1] 1855658

> dc_storage(NYT)
DistributedCorpus: Storage
- Type: local_disk
- Base directory on storage: /home/theussl/lib/R/2.11/tm.corpus.NYTimes/dc
- Current chunk size [bytes]: 52428800
```

Sentiment Construction

Extraction of “opinions” in order to derive sentiment scores for each NYT article is done in a three-step approach:

1. calculating term frequencies (TF) and store it in a so-called Document-Term-Matrix (DTM).
2. tagging terms according to its sentiments
3. aggregating TF based on tag and date to derive a sentiment score.

(1) 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 $f_{t,d}$, i.e., the *term frequency* (TF) of each term t for each document d. Its construction typically involves pre-processing and counting TFs for each document. $f_{t,d}$ denotes the number of occurrences of term t in document d, $\rightarrow$ *bag of words model*. Since data volumes (corpora) are very large, and

preprocessing is rather time consuming we use Hadoop (`http://hadoop.apache.org/core/`) and the R package **hive** in order to efficiently construct the DTM .

Example: NYT

DTM construction using the NYT distributed corpus:

```
> termfreq_control <- list(removePunctuation = TRUE, removeNumbers = TRUE,
+     stemming = TRUE, stopwords = TRUE)
> NYT_DTM <- DocumentTermMatrix(NYT, control = termfreq_control)
```

A pre-constructed DTM will also be supplied with **tm.corpus.NYTimes**.

```
> data("NYT_DTM")
```

Dimensionality Reduction

- DTM of NYT is very sparse
- it contains over $2,000,000$ different terms
- a lot of them are useless since they are usually “created” when applying several preprocessing steps, especially `removePunctuation`
- a suitable set of terms is derived through the tf-idf weighted DTM
- the *reduced* DTM is derived by subsetting the original DTM using a number of important terms

```
> NYT_DTM_tfidf <- weightTfIdf(NYT_DTM)
> important_terms <- Terms(NYT_DTM_tfidf)[cm > 1e-04]
> NYT_DTM_reduced <- NYT_DTM[, important_terms]
```

(2) Tagging

- $f_{t,d}$ are now contained in a DTM.
- Terms have to be tagged according to their conotated sentiment via the *General Inquirer* (GI).
- GI tag categories is a collection of the following sources:
  - Harvard IV-4 dictionary
  - Lasswell value dictionary
  - other categories (see http://www.wjh.harvard.edu/ inquirer/)
- GI is said to be appropriate for developing scoring procedures that enlist multiple categories and analyze the tag patterns.

Example: Tagging I

```
> require("tm.plugin.tags")
> control <- list(stemming = TRUE)
> sample(tm_get_tags("Negativ", control = control), 10)
 [1] "gloomi"      "muddi"       "betray"      "disprov"     "substitut"
 [6] "unnecessari" "cannib"      "hazi"        "undon"       "burn"

> sample(tm_get_tags("Positiv", control = control), 10)
 [1] "humbl"       "fortun"      "spectacular" "respons"     "promin"
 [6] "hilari"      "glad"        "versatil"    "pleasant"    "golden"
```

Example: Tagging II

After data cleansing we have NYT_DTM_reduced and NYT_meta:

```
> load("NYT_DTM_reduced.rda")
> load("NYT_meta.rda")
> head(NYT_meta, n = 2)
     ID       Date nChar
1 857784 1996-06-15  1480
2 857785 1996-06-15  5859
> which_pos <- Terms(NYT_DTM_reduced) %in% tm_get_tags("Positiv")
> which_neg <- Terms(NYT_DTM_reduced) %in% tm_get_tags("Negativ")
> score_pos <- row_sums(NYT_DTM_reduced[, which_pos])
> score_neg <- row_sums(NYT_DTM_reduced[, which_neg])
> NYT_meta <- cbind(NYT_meta, NYT_DTM_reduced_pos, NYT_DTM_reduced_neg)
> colnames(NYT_meta) <- c(colnames(NYT_meta)[-c(4, 5)], "nTerms_pos",
+     "nTerms_neg")
> dim(NYT_meta)
[1] 1855657       5
> head(NYT_meta, n = 2)
       ID       Date nChar nTerms_pos nTerms_neg
874423  0 1987-01-01   806         12          1
874424  1 1987-01-01   423          7          2
```

(3) Sentiment Score

Sentiment score is calculated using the ratio of the positive and negative connotations from GI tag categories for all documents within e.g., a given month. For a given time t, it is derived by an obvious classifier based on $f_{t,d}$

$$score(t) = \frac{pos(t)}{neg(t)} - \frac{1}{T}\frac{\sum_{t=1}^{T} pos(t)}{\sum_{t=1}^{T} neg(t)}$$

Example: Scoring

```
> library("xts")
> get_score <- function(meta, by = c("day", "month")) {
+     by <- match.arg(by)
+     pos_neg_ratio <- with(meta, sum(nTerms_pos)/sum(nTerms_neg))
+     sents_docs <- with(meta, nTerms_pos/nTerms_neg - pos_neg_ratio)
+     meta.xts <- xts(meta[, c(3:5)], order.by = meta$Date)
+     aggr <- switch(by, day = format(time(meta.xts), "%Y-%m-%d"),
+         month = as.yearmon)
+     out <- aggregate(meta.xts, aggr, function(x) sum(x, na.rm = TRUE))
+     if (by %in% c("day"))
+         attr(out, "index") = as.Date(attr(out, "index"))
+     to_remove <- which(is.na(time(out)))
+     out <- out[-to_remove, ]
+     sentiments <- with(out, nTerms_pos/nTerms_neg - pos_neg_ratio)
+     out <- cbind(out, sentiments)
+ }
> NYT_sentiments <- get_score(NYT_meta, "month")
> head(NYT_sentiments)
            nChar nTerms_pos nTerms_neg sentiments
Jan 1987 26742896     199829     131397 0.06384989
Feb 1987 24751386     186543     120027 0.09722184
```

Support Vector Machines I

Support Vector Machines (SVM) perform classification by constructing hyperplanes that optimally separate the data.

- Learning takes place in the feature space and the data points only appear inside dot products, $\rightarrow$ "kernel trick".
- If $\Phi : X \rightarrow H$ is used, the dot product $\langle \Phi(x), \Phi(y) \rangle$ can be represented by a kernel function

$$k(x, y) = \langle \Phi(x), \Phi(y) \rangle.$$

- R package **kernlab** provides several kernel implementations, e.g., linear kernel,Gaussian radial, etc.
- kernel with the best "hit-ratio" is chosen, i.e., the success of forecasting the sign of the index development.

Support Vector Machines II

We used the R package **kernlab** for the SVM calculations. **kernlab** provides several kernel implementations, e.g.,

- linear kernel,
- Gaussian radial,
- polynomial kernels.

To ealuate the forecasting ability of the SVM, we used different kernels and evaluated the “hit-ratio”, i.e., the sucess of forecasting the sign of the index development.

Example: SVM in R

```
> library("kernlab")
> load("cfnai.rda")
> kernels <- c("rbfdot", "polydot", "vanilladot", "tanhdot", "laplacedot",
+     "besseldot", "anovadot", "splinedot")
> kernel.match <- function(x, mult = 10, t = 100, T = 246, kernels = "vanilladot")
+     training <- x[1:t, ]
+     prediction <- matrix(numeric(), nrow = nrow(x) - t, ncol = length(kernels))
+     colnames(prediction) <- kernels
+     for (j in (t + 1):T) {
+         for (i in 1:length(kernels)) {
+             model <- ksvm(CFNAI ~ ., data = x[1:j - 1, ], kernel = kernels[i])
+             prediction[j - t, i] <- predict(model, sentiments <- mult *
+                 x[j, 1])
+         }
+     }
+     prediction
+ }
> time <- index(cfnai) %in% index(NYT_sentiments)
> SVM_dat <- cbind(NYT_sentiments$sentiments, sign(cfnai[time]$CFNAI_MA3))
> colnames(SVM_dat) <- c("sentiments", "CFNAI")
> prediction <- kernel.match(SVM_dat, t = 100, T = 246, kernels = kernels)
```

Results I

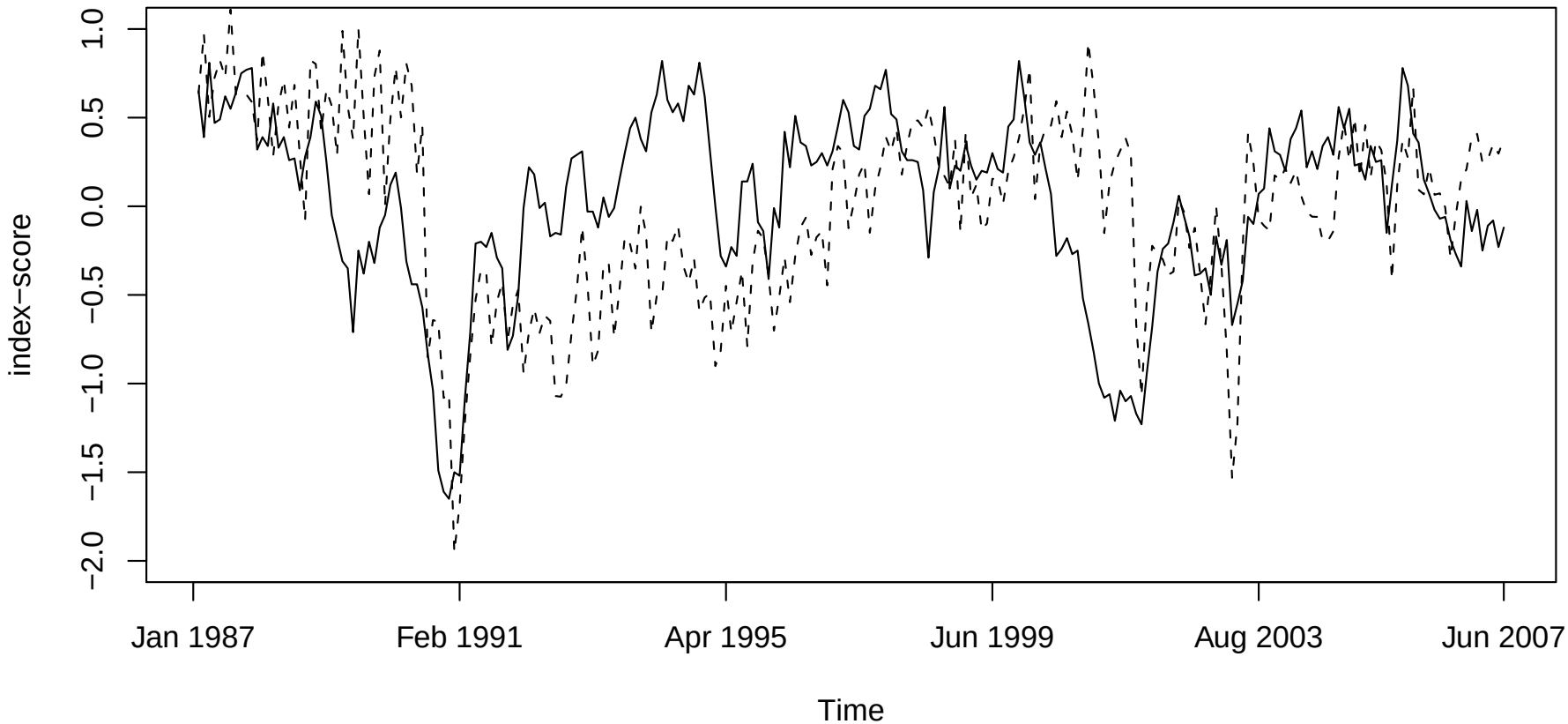

Figure: The CFNAI index and the estimated (scaled) sentiments of the NYT

Example: SVM in R

```
> t <- 100
> tplus <- t + 1
> classification <- apply(sign(prediction), 2, function(x) {
+     sum(x == SVM_dat[tplus:246, "CFNAI"])
+ })
> hitratio <- classification/(246 - t)
> hitratio
    rbfdot    polydot vanilladot    tanhdot laplacedot  besseldot   anovadot
 0.5136986  0.6027397  0.6027397  0.3972603  0.4657534  0.4452055  0.5342466
 splinedot
 0.6027397
```

Results II

Aggregating the NYT data to monthly sentiment scores and comparing the hit rate of the different SVM kernels:

Table: Forecasting performance of different SVM kernels

kernel	hit-ratio (HR) NYT	HR Bussiness
Gaussian radial basis	0.50	0.53
polynomial	0.61	0.51
linear	0.61	0.51
hyperbolic tangent	0.40	0.45
Laplace radial basis	0.48	0.53
ANOVA radial basis	0.54	0.47
Bessel kernel	0.45	0.42

Discussion - Further Research

- Improve sentiment estimation, like (1) Vector based classifiers or (2) Adjective-Adverb classifiers.
  1. Given classified test-vectors, the angle between the observed and the pre-classified vector can be chosen as a measure of closeness.
  2. This classifier is based on assumption, that adjectives and adverbs emphasize sentiments and require greater weight in the sentiment process.
- Extend this approach to other newspapers

Thank you for your attention

Paul Hofmarcher and Kurt Hornik and Stefan Theußl
Department of Finance, Accounting and Statistics
Institute for Statistics and Mathematics

email: Firstname.Lastname@wu.ac.at
URL: http://statmath.wu.ac.at/~lastname

WU Wirtschaftsuniversität Wien
Augasse 2–6, A-1090 Wien

Appendix: tm Infrastructure

The complete text mining infrastructure consists of many components:

- **tm**, text mining package (0.5-4.1, Feinerer, 2010)
- **slam**, sparse lightweigt arrays and matrices (0.1-11, Hornik et.al., 2010)
- **tm.plugin.dc**, distributed corpus plugin (0.1-5, Theussl and Feinerer, 2010)
- **tm.plugin.tags**, tag category database (0.0-1, Theussl, 2010)
- **hive**, Hadoop/MapReduce interface (0.1-8, 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**).

Appendix: 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