WIRTSCHAFTS
UNIVERSITÄT
WIEN VIENNA
UNIVERSITY OF
ECONOMICS
AND BUSINESS

tm and plugins

Overview tm Plugins

- ***tm.plugin.dc***
- ***tm.plugin.tags***
- **tm.plugin.mail**
- **tm.plugin.OAI**

Managing Linguistic Data

Questions

Chapter 11 deals with the following questions:

- How do we design a new language resource and ensure that its coverage, balance, and documentation support a wide range of uses?
- When existing data is in the wrong format for some analysis tool, how can we convert it to a suitable format?
- What is a good way to document the existence of a resource we have created so that others can easily find it?

Corpus Structure

- NLTK includes a diverse set of corpora which can be read using the **nltk.corpus** package. Each corpus is accessed by means of a “corpus reader” object from **nltk.corpus**.
- Corpora consist of a set of files, each containing a document (or other pieces of text).
- Each corpus reader provides a variety of methods to read data from the corpus, depending on the format of the corpus.
- NLTK corpus types: Plaintext Corpora, Tagged Corpora, Chunked Corpora, Parsed Corpora, Word Lists and Lexicons, Categorized Corpora, Propbank Corpus, TIMIT, etc.
- `http://nltk.googlecode.com/svn/trunk/doc/howto/corpus.html`

Design Features

The TIMIT corpus (a speech corpus):

- contains two layers of annotation, at the phonetic and orthographic levels. In general, a text or speech corpus may be annotated at many different linguistic levels, including morphological, syntactic, and discourse levels.
- is balanced across multiple dimensions of variation, for coverage of dialect regions and diphones.
- division between the original linguistic event and the annotations of that event (text corpora, in the sense that the original text usually has an external source, and is considered to be an immutable artifact. Any transformations of that artifact which involve human judgment (e.g., even tokenization) are subject to later revision; thus it is important to retain the source material in a form that is as close to the original as possible).
- hierarchical structure of the corpus

Fundamental Data Types (1)

Corpora are basically collections of texts together with record-structured data, thus two fundamental data types:

- lexicons (record structure; e.g., conventional dictionary or comparative wordlist, see also Chapter 2)
- annotated texts (temporal organization, i.e., text is a representation of a real or fictional speech event, and the time-course of that event carries over into the text itself; word or sentence, or a complete narrative or dialogue; with/out annotation)

Two extremes: eg., Brown Corpus (text files and a table to relate the files to 15 different genres) vs. WordNet containing 117,659 synset records and many example sentences (mini-texts) to illustrate word usages.

Fundamental Data Types (2)

Lexicon

Abstraction: fielded records

key	field	field	field	field
key	field	field	field	field

Eg: dictionary

wake: weɪk, [v], cease to sleep...
walk: wɔːk, [v], progress by lifting and setting down each foot...

Eg: comparative wordlist

wake; aufwecken; acordar
walk; gehen; andar
write; schreiben; enscrever

Eg: verb paradigm

wake	woke	woken
write	wrote	written
wring	wrung	wrung

Text

Abstraction: time series

token attrs	token attrs	token attrs	...

time →

Eg: written text

A long time ago, Sun and Moon lived together. They were good brothers. ...

Eg: POS-tagged text

A/DT long/JJ time/NN ago/RB ,/,
Sun/NNP and/CC Moon/NNP
lived/VBD together/RB ./.

Eg: interlinear text

Ragaipa	irai		vateri
ragai -pa	ira	-i	vate -ri
PP.1.SG -BEN	RP.3.SG.M -ABS		give -2.SG

The Life Cycle of a Corpus

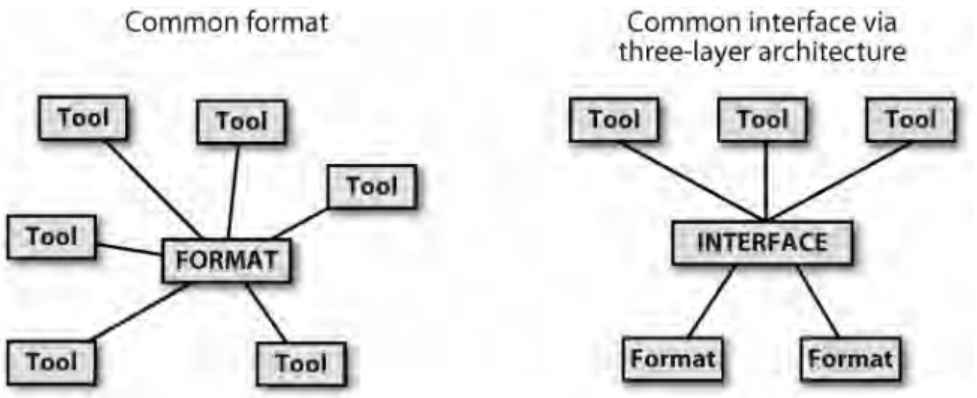
- Corpus Creation (three scenarios: field linguistics, common task, reference corpus)
  - collect raw data
  - cleaned up and document
  - stored in a systematic structure
- Quality control
  - find inconsistencies in the annotations
  - ensure the highest possible level of inter-annotator agreement (Kappa coefficient)

Acquiring Data

- Obtaining data from the web, word processors, spreadsheets or databases (discussed in Chapter 3)
- Converting data formats (encodings discussed in Chapter 3)
- Deciding which layers of annotation to include:
  - word tokenization
  - sentence segmentation
  - paragraph segmentation
  - part-of-speech (POS)
  - syntactic structure
  - shallow semantics
  - dialogue and discourse
- Inline vs. standoff annotation

Tools and Standards

- Adequate tools for creation, publication, and use of linguistic data are not widely available.
- No adequate, generally accepted standards for expressing the structure and content of corpora (adequate standards are unlikely to be developed, used, and accepted)
- Instead of focusing on a common format, we believe it is more promising to develop a common interface.
- A common corpus interface insulates application programs from data formats.



Meta Data

- Metadata is descriptive information about an object or resource, whether it be physical or electronic.
- Dublin Core (DC) Metadata Initiative
  - represent a broad, interdisciplinary consensus about the core set of elements that are likely to be widely useful to support resource discovery.
  - consists of 15 metadata elements, where each element is optional and repeatable (Title, Creator, Subject, etc.)
- Open Archives Initiative (OAI)
  - provides a common framework across digital re- positories of scholarly materials, regardless of their type, including documents, data, software, recordings, physical artifacts, digital surrogates
  - Repository consists of a network-accessible server offering public access to archived items
  - Each item has a unique identifier, and is associated with a Dublin Core metadata record

Meta Data

- Open Language Archives Community (OLAC)
  - OLAC Metadata is a standard for describing language resources
  - `http://www.language-archives.org/`
  - extends DC Meta data sets and uses OAI protocol

Summary Chapter 11

- Fundamental data types, present in most corpora, are annotated texts and lexicons. Texts have a temporal structure, whereas lexicons have a record structure.
- The life cycle of a corpus includes data collection, annotation, quality control, and publication. The life cycle continues after publication as the corpus is modified and enriched during the course of research.
- Corpus development involves a balance between capturing a representative sample of language usage, and capturing enough material from any one source or genre to be useful; multiplying out the dimensions of variability is usually not feasible be- cause of resource limitations.
- The Open Language Archives Community (OLAC) provides an infrastructure for documenting and discovering lang. resources.

tm Corpora

tm Corpora

- Pre-constructed **tm** corpus packages
- “free” corpora available from `http://datacube.wu.ac.at`.
- E.g., Reuters21578:
  `install.packages("tm.corpus.Reuters21578", repos = "http://datacube.wu.ac.at")`
- Usually includes a Corpus object and a DTM

Overview tm Corpora

- tm.corpus.20Newsgroups
- tm.corpus.afg
- tm.corpus.AmazonReviews
- tm.corpus.APA
- tm.corpus.Congress
- tm.corpus.JSM.2008.abstracts
- tm.corpus.LearnAtWU
- tm.corpus.MovieReviews
- tm.corpus.NSF
- tm.corpus.NYTimes
- tm.corpus.PNAS
- tm.corpus.RCV1
- tm.corpus.Reuters21578
- tm.corpus.SpamAssassin

Other Corpus Packages

- **corpus.CiteSeerX**
- **corpus.CRAN.meta**
- **corpus.ePubWU.theses**
- **corpus.ePubWU.workingpapers**
- **corpus.JSS.papers**
- **corpus.Project.Euclid**
- **corpus.useR.2008.abstracts**

tm.plugin.dc

tm.plugin.dc

- Objective: create a tm plugin package so that we can process large corpora
- We need to consider **tm**'s corpus class and corresponding methods
- Make use of large scale data processing tools

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)

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.

Challenges

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

Distributed Text Mining in R

Data sets:

- *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	$>$ 2000

[1] calculated with the Unix tool du

Distributed Text Mining in R

- 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 at WU
- 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.

Summary

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

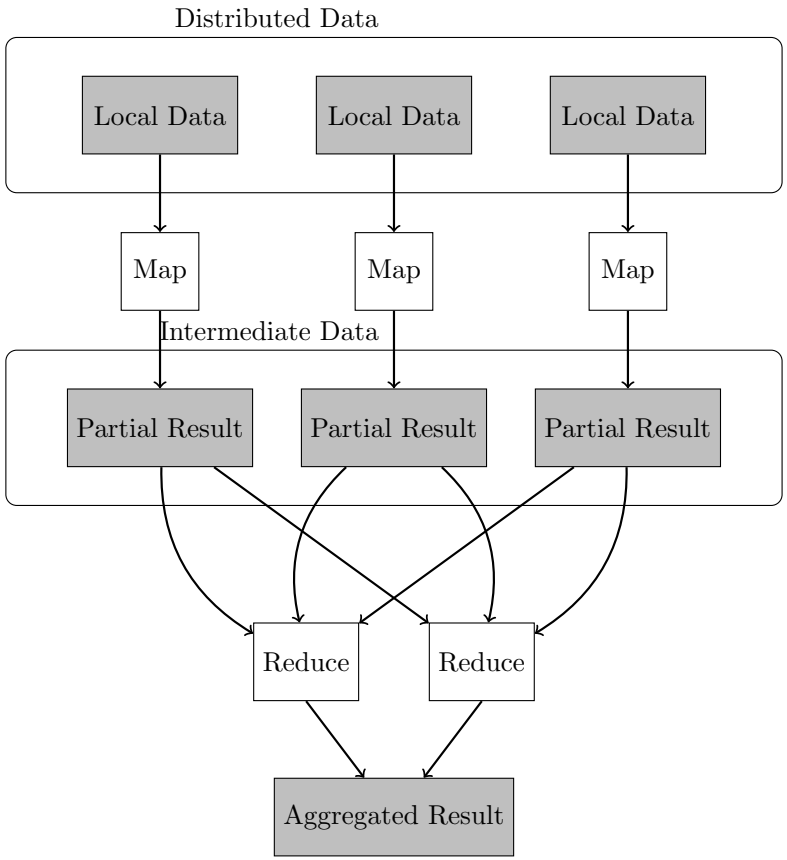


Figure: Conceptual Flow

The MapReduce Programming Model

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

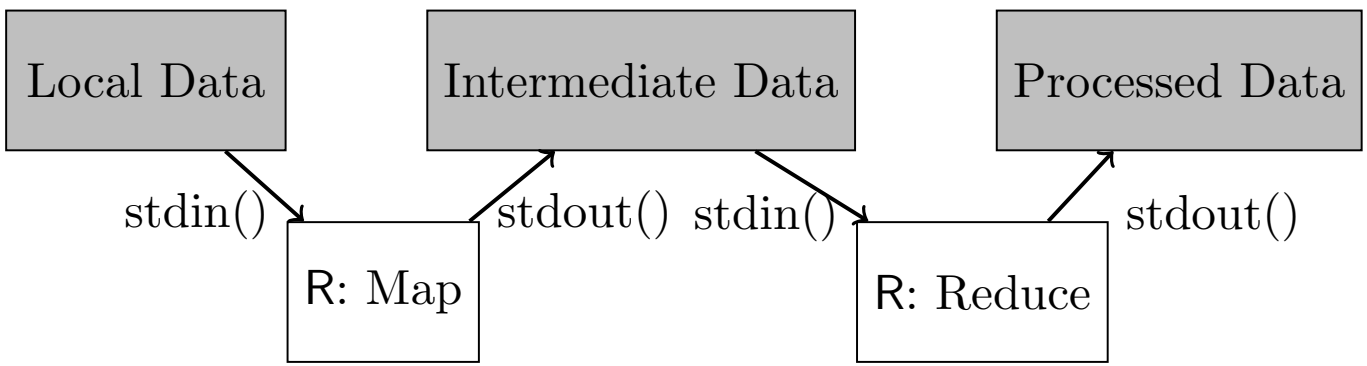
- Master/worker architecture (Namenode/Datanodes)
- Data locality
- Map tasks are applied to partitioned data
- Map tasks scheduled so that input blocks are on same machine
- Datanodes read input at local disk speed
- Data replication leads to fault tolerance
- Application does not care whether nodes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

- Use Hadoop via **hive** package
- Access and modify DFS
- Count Words

Distributed Text Mining in R

Our 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**.

Example: Distributed Text Mining in R

- ‘`DistributedCorpus`’ usage
- Parallel computation

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

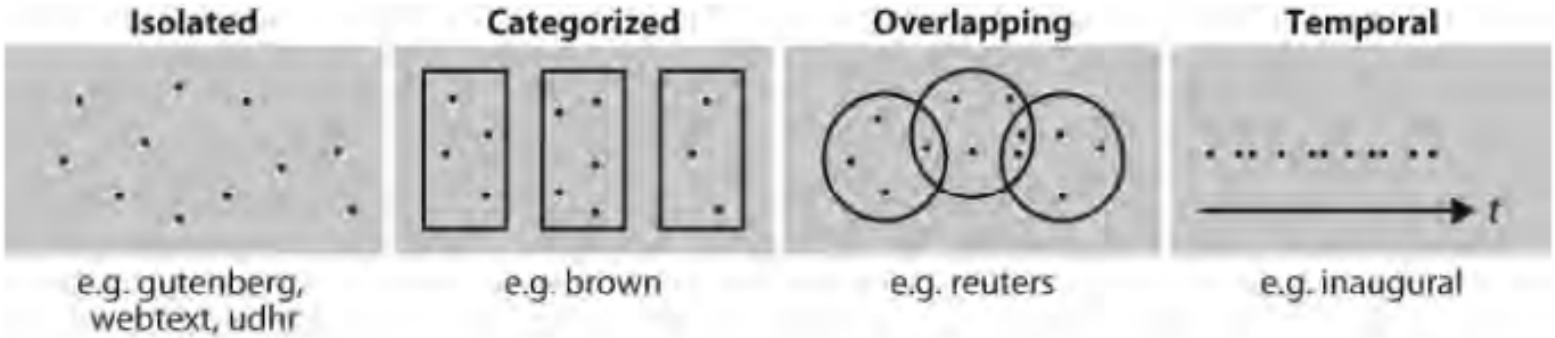
Text Corpora and Lexical Resources

Questions

Chapter 2 deals with the following questions:

- What are some useful text corpora and lexical resources, and how can we access them with Python?
- Which Python constructs are most helpful for this work?
- How do we avoid repeating ourselves when writing Python code?

Text Corpus Structure



- a collection of isolated texts with no particular organization
- structured into categories, such as genre (e.g., Brown Corpus)
- categorizations overlap, such as topic categories (e.g., Reuters Corpus)
- represent language use over time (e.g., Inaugural Address Corpus, news collections)

Basic NLTK Corpus functionality

- `fileids()` The files of the corpus
- `fileids([categories])` The files of the corpus corresponding to these categories
- `categories()` The categories of the corpus
- `categories([fileids])` The categories of the corpus corresponding to these files
- `raw()` The raw content of the corpus
- `raw(fileids=[f1,f2,f3])` The raw content of the specified files
- `raw(categories=[c1,c2])` The raw content of the specified categories
- `words()` The words of the whole corpus
- `words(fileids=[f1,f2,f3])` The words of the specified fileids
- `words(categories=[c1,c2])` The words of the specified categories
- `sents()` The sentences of the specified categories
- `sents(fileids=[f1,f2,f3])` The sentences of the specified fileids
- `sents(categories=[c1,c2])` The sentences of the specified categories
- `abspath(fileid)` The location of the given file on disk
- `encoding(fileid)` The encoding of the file (if known)
- `open(fileid)` Open a stream for reading the given corpus file
- `root()` The path to the root of locally installed corpus
- `readme()` The contents of the README file of the corpus

Example: Accessing Text Corpora

- Examine a variety of text corpora
- Select individual texts
- Work with text

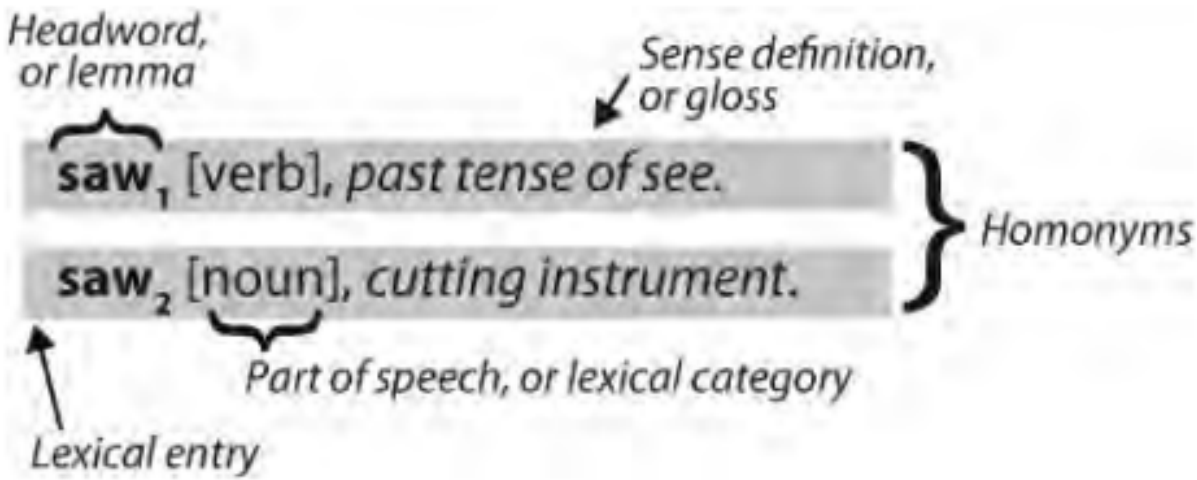
Python: Gutenberg Corpus, Web and Chat Text, Brown Corpus, Reuters Corpus, Inaugural Address Corpus (for further annotated text corpora see Bird et al., 2009, page 42, 43)

Example: Cond. Freq. Distributions

- Find the most frequent words of a text.
- Identify the words of a text that are most informative about the topic and genre of the text.
- How are the total number of word tokens in the text distributed across the vocabulary items? (frequency distribution)
- Compare separate frequency distributions for each category. (conditional frequency distribution)

Lexical Resources

- A lexicon, or lexical resource, is a collection of words and/or phrases along with associated information, such as part-of-speech and sense definitions.
- Lexical resources are secondary to texts, and are usually created and enriched with the help of texts (e.g., vocabulary, word frequencies).



Example: Wordlist Corpora

- Find unusual or misspelled
- Stopwords
- Comparative wordlists
- Shoebox/Toolbox

Summary Chapter 2

- A text corpus is a large, structured collection of texts. NLTK comes with many corpora.
- Some text corpora are categorized, e.g., by genre or topic; sometimes the categories of a corpus overlap each other.
- A conditional frequency distribution is a collection of frequency distributions, each one for a different condition. They can be used for counting word frequencies, given a context or a genre.

Skipped Sections

- Generating Random Text with Bigrams
- More Python: Reusing Code
- Wordnet
- Pronouncing Dictionary

Skipped Sections

- Special Considerations When Working with Endangered Languages
- Working with XML (discussed in Chapter 3?)
- Working with Toolbox Data (discussed in Chapter 2)

References

S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python*. O'Reilly Media, 2009. URL `http://www.nltk.org/book`.

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`.

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.

Thank you for your attention

Stefan Theußl
Department of Finance, Accounting and Statistics
Institute for Statistics and Mathematics
email: Stefan.Theussl@wu.ac.at
URL: http://statmath.wu.ac.at/~theussl

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