Latent Dirichlet Allocation in R

Martin Ponweiser

Diploma Thesis
Thesis 2, May 2012

Institute for Statistics and Mathematics
http://statmath.wu.ac.at/

WIRTSCHAFTS
UNIVERSITÄT
WIEN VIENNA
UNIVERSITY OF
ECONOMICS
AND BUSINESS

Diplomarbeit

Latent Dirichlet Allocation in R

Zur Erlangung des akademischen Grades
eines Magisters der Sozial- und Wirtschaftswissenschaften
an der Wirtschaftsuniversität Wien

Unter Anleitung von

Assist.-Prof. Dipl.-Ing. Dr. Bettina Grün
Department of Applied Statistics
Johannes Kepler Universität Linz

Beurteiler

Univ.-Prof. Dipl.-Ing. Dr. Kurt Hornik
Institute for Statistics and Mathematics
Wirtschaftsuniversität Wien

von

Martin Ponweiser

Wien, Mai 2012

Topic Modelle sind ein neues Forschungsfeld innerhalb der Informationswissenschaften Information Retrieval und Text Mining. Mit Techniken aus dem maschinellen Lernen werden Dokumentenkorpora in statistische Modelle überführt und sind dadurch besser durchsuch- und erforschbar. Das bekannteste Topic Modell ist Latent Dirichlet Allocation (LDA), das sich seit seiner Einführung im Jahre 2003 durch Blei et al. als nützliches Werkzeug in verschiedenen Disziplinen etabliert hat und außerdem Grundlage für die Entwicklung von komplexeren Topic Modellen gewesen ist. Diese Diplomarbeit widmet sich dem Anwendungsaspekt von LDA. Die Datenanalyse eine der ersten Veröffentlichungen zu LDA, des Artikels "Finding scientific models" von Thomas Griffiths und Mark Steyvers (2004), wird in der statistischen Programmiersprache R unter Zuhilfenahme des neuen R Paketes "topicmodels" von Bettina Grün und Kurt Hornik nachvollzogen. Der komplette Ablauf, vom Extrahieren eines Textkorpus von der Webseite der Zeitschrift PNAS, über die Vorbereitung der Daten und das Überführen in eine Dokumenten-Term-Matrix, die Modellselektion, das Modellschätzen und die Aufbereitung und Visualisierung der Resultate, wird vollständig dokumentiert und kommentiert. Das Ergebnis entspricht weitestgehend der Analyse des ursprünglichen Artikels; es wird somit einerseits das Vorgehen von Griffiths/Steyvers reproduziert, andererseits kann die Eignung der quelloffenen Werkzeuge der R-Umgebung für das Text Mining mittels LDA attestiert werden.

Schlagworte: Latent Dirichlet Allocation, LDA, R, Topic Models, Text Mining, Information Retrieval, Statistik

Diploma Thesis

Latent Dirichlet Allocation in R

Martin Ponweiser

Institute for Statistics and Mathematics
Vienna University of Business and Economics

Supervisors: Bettina Grün, Kurt Hornik

Vienna, May 2012

Topic models are a new research field within the computer sciences information retrieval and text mining. They are generative probabilistic models of text corpora inferred by machine learning and they can be used for retrieval and text mining tasks. The most prominent topic model is latent Dirichlet allocation (LDA), which was introduced in 2003 by Blei et al. and has since then sparked off the development of other topic models for domain-specific purposes. This thesis focuses on LDA's practical application. Its main goal is the replication of the data analyses from the 2004 LDA paper "Finding scientific topics" by Thomas Griffiths and Mark Steyvers within the framework of the R statistical programming language and the R package topicmodels by Bettina Grün and Kurt Hornik. The complete process, including extraction of a text corpus from the PNAS journal's website, data preprocessing, transformation into a document-term matrix, model selection, model estimation, as well as presentation of the results, is fully documented and commented. The outcome closely matches the analyses of the original paper, therefore the research by Griffiths/Steyvers can be reproduced. Furthermore, this thesis proves the suitability of the R environment for text mining with LDA.

Keywords: latent Dirichlet allocation, LDA, R, topic models, text mining, information retrieval, statistics

Acknowledgements

First and foremost, I want to thank Regina, Helga, Josef and Simon for their unconditional love and faith in me. I owe you more than I can say.

I offer my sincerest gratitude to my thesis supervisor Bettina Grün, who during the last three years supported me with her profound knowledge and a seemingly never-ending patience whilst allowing me to work in my own way. Bettina's positive outlook and methodic approach to solving problems has been a huge inspiration to me. Thank you.

Ingo Feinerer helped me in setting up my development environment and by providing his dissertations's LaTeX source as a template, as well as by giving me general and tm-specific advice many times.

I also want to thank my sisters Rita and Barbara for their motivation. Extra hugs for Rita for proofreading the complete thesis.

Last, but not least, I am thankful to the whole staff of the Institute for Statistics and Mathematics, especially Kurt Hornik and Stefan Theußl, for giving me the opportunity to be part of the team last year as **wu.cloud** administrator.

Contents

1. Introduction and Overview **2**
1.1. Motivation and Scope of Thesis . . . 2
1.2. Organisation of Thesis . . . 3

2. Background in Computer Sciences **5**
2.1. Information Retrieval . . . 5
2.1.1. Common Information Retrieval Models . . . 7
2.2. Natural Language Processing . . . 9
2.3. Text Mining . . . 12

3. Latent Dirichlet Allocation: a Topic Model **13**
3.1. Topic Models . . . 13
3.2. Literature and Impact . . . 13
3.3. Observed Data / Model Input . . . 14
3.4. The Dirichlet Distribution . . . 14
3.5. Generative Process . . . 15
3.6. Model Estimation . . . 21
3.6.1. Collapsed Gibbs Sampling . . . 22
3.7. Model Evaluation and Selection . . . 24
3.7.1. Performance Measurement on Data . . . 24
3.7.2. Performance Measurement on Secondary Tasks . . . 26
3.7.3. Performance Measurement by Human Judgement . . . 26
3.8. Use of Fitted Topic Models . . . 26
3.8.1. Finding Similar Documents Through Querying and Browsing . . . 27
3.8.2. Exploring the Relation between Topics and Corpus Variables . . . 27
3.9. Extending LDA . . . 28

4. LDA in the R Environment **29**
4.1. Overview of LDA Implementations . . . 29
4.2. The R Programming Language and Environment . . . 31
4.3. Topic Modeling with the R Packages **tm** and **topicmodels** . . . 31
4.3.1. The **topicmodels** Package . . . 31
4.3.2. Preprocessing with the **tm** Package . . . 32
4.3.3. Model Selection by Harmonic Mean in **topicmodels** . . . 33
4.4. The **lda** Package . . . 36

5. Analyzing the 1991 to 2001 Corpus of PNAS Journal Abstracts **37**
5.1. Retrieving and Preprocessing the Corpus . . . 38
5.1.1. Legal Clearance . . . 38
5.1.2. Web Scraping . . . 38
5.1.3. Importing the Corpus and Generating a Document-Term Matrix in R . . . 39
5.2. Model Selection . . . 40
5.2.1. Result of Model Selection . . . 43
5.3. Model Fitting . . . 44
5.4. Relations between Topics and Categories (Year 2001) . . . 44
5.4.1. Preparations and Showing the Category Distribution for 2001 . . . 45
5.4.2. Finding the Diagnostic Topics . . . 47
5.4.3. Visualising Topics and Categories . . . 48
5.4.4. Comparison with Original Paper and Interpretation . . . 52
5.5. Hot and Cold Topics . . . 54
5.5.1. Comparison with Original Paper and Interpretation . . . 60
5.6. Document Tagging and Highlighting . . . 62
5.6.1. Finding the Example Abstract from Griffiths/Steyvers (2004) . . . 62
5.6.2. Tagging and Highlighting of an Abstract . . . 63
5.6.3. Comparison with Original Paper . . . 63

6. Conclusion and Future Work **64**
6.1. Future Work . . . 64

A. Appendix: Software Used **65**

B. Appendix: Model Fitting on the Sun Grid Engine **66**
B.1. Submitting Sun Grid Engine Jobs . . . 66

C. Appendix: Source Code Listings **67**
C.1. `beta-distribution.R` . . . 67
C.2. `dirichlet-3d.R` . . . 67
C.3. `dirichlet-samples.R` . . . 68
C.4. `sge-modelselection-chains.R` . . . 68
C.5. `sge-modelselection-likelihoods.R` . . . 69
C.6. `modelselection-chain-sizes.R` . . . 70
C.7. `classifications-fig-category-frequency.R` . . . 70
C.8. `classifications-fig-levelplot-most-diagnostic.R` . . . 71
C.9. `classifications-fig-levelplot-most-diagnostic-by-prevalence.R` 72
C.10. `classifications-table-most-diagnostic-five-terms.R` . . . 73
C.11. `classifications-fig-levelplot-five-most-diagnostic.R` . . . 73
C.12. `classifications-original-topics-table.R` . . . 74
C.13. `trends-table-year-frequencies.R` . . . 75
C.14. `trends-table-significance.R` . . . 75

C.15. trends-fig-all-300.R . . . 76
C.16. trends-fig-five-hot-and-cold.R . . . 76
C.17. trends-table-terms.R . . . 77
C.18. trends-original-terms-table.R . . . 77
C.19. tagging-document-tag-latex.R . . . 77
C.20. tagging-table-topics-most-prevalent.R . . . 80
C.21. appendix-model-300-terms-tables.R . . . 80

D. Appendix: Web Scraping a Corpus of PNAS Journal Abstracts 82
D.1. Site Structure and Content . . . 82
D.2. Implementation . . . 82
D.2.1. Preparing Scrapy . . . 83
D.2.2. Calling Scrapy . . . 83
D.2.3. Selecting All Further Downloads . . . 84
D.2.4. Downloading the Selected Abstracts and Full Texts . . . 84
D.2.5. Cleaning the Downloaded Files . . . 85
D.2.6. Merging and Inspecting the Available Metadata . . . 86
D.2.7. Importing Abstracts and Metadata as a **tm** Corpus in R . . . 87
D.3. Listings . . . 87
D.3.1. settings.py . . . 87
D.3.2. items.py . . . 88
D.3.3. pnas_spider.py . . . 88
D.3.4. pipelines.py . . . 91
D.3.5. 1-scrape.sh . . . 91
D.3.6. 2-select.py . . . 91
D.3.7. 3-get.py . . . 95
D.3.8. 4-scrub.py . . . 99
D.3.9. 5-zip.py . . . 103
D.3.10. csv_unicode.py . . . 108
D.3.11. opt-categories.py . . . 109
D.3.12. tm-corpus-make.R . . . 110

E. Appendix: PNAS Journal 1991-2001 Corpus: 300 Topics and Their Twelve Most Probable Terms from the LDA Model Sample 112

Bibliography 133

1. Introduction and Overview

Information overload is one of the problems that have arrived with the Information Age. Huge collections of data, be it literature, emails, web pages or content in other digital media have grown to sizes that make it hard for humans to handle them easily. It is the computer sciences of information retrieval, text mining and natural language processing that deal with this problem. Researchers from these fields have developed the techniques that help us transform our search queries (on Google or Bing) into meaningful search results, recommend to us new books based on our previous purchases (Amazon) or automatically translate documents between languages.

A new approach to extracting information from digital data are so-called **topic models**. As the name suggests, they are about modeling which data entities are likely to be from the same topic or subject. Topic model algorithms enable us to feed a computer program with data and have documents or other content (images or videos) assigned to topics that are meaningful to humans. The topics are the results of an unsupervised learning process which can then be used for searching or browsing the original data collection.

In this thesis, I focus on the topic model **latent Dirichlet allocation** (LDA), which was first proposed by Blei et al. in 2003. In comparison to other topic models, LDA has the advantage of being a probabilistic model that firstly performs better than alternatives such as probabilistic latent semantic indexing (PLSI) (Blei et al., 2003) and that secondly, as a Bayesian network, is easier to extend to more specific purposes. Variations on the original LDA have led to topic models such as correlated topic models (CTM), author-topic models (ATM) and hierarchical topic models (HTM), all of which make different assumptions about the data and with each being suited for specific analyses.

1.1. Motivation and Scope of Thesis

LDA and similar topic models are still relatively new and therefore offer many directions for further research. Most available papers focus on improvements to the theorical foundation, i.e., extensions to the basic LDA model and also better learning or performance evaluation techniques. This thesis explores a less prominent aspect of topic modeling with LDA, namely the practical side.

The initial and also many later papers on LDA give examples of applications in which one or more models are built on document collections. The resulting distributions are then presented in word tables or "refined" in tools using more advanced visualization and/or analysis. Due to the limited space in journals it is common practice that authors only publish the results of their analyses and omit most of the steps that were taken in

the process. I have found that authors choose to publish the modeling software they have written but not the source code for their papers. It is therefore hardly possible to test claims made by the authors, understand the complete context or improve the papers. Instead of trying to solve this general problem of scientific methodology, I have decided to replicate an existing LDA paper and explicitly document all the steps and difficulties involved to see how much information is actually missing in this specific example and draw conclusions on the general state of publications on topic models.

My thesis uses the framework of the statistical programming language R. The choice of R is an obvious one, as it is the *de facto* standard for open-source statistical analyses. It is widely used by academics and businesses, and a large variety of additional packages for solving user tasks are available.

In 2009, Bettina Grün and Kurt Hornik published the R package **topicmodels**, which includes three open-source implementations of topic models from other authors. The R language itself and an interface to the powerful R text mining package **tm** make this combination of tools ideal for both topic model research and application.

Consequently, the goals of this thesis are:

- to give an introduction to the sciences behind topic models,
- to review the literature on LDA and summarize the model,
- to demonstrate the use of the R package **topicmodels**,
- to replicate the results of an often cited paper (Griffiths and Steyvers, 2004),
- to extend and improve the **topicmodels** package where needed, and
- to identify potential research gaps.

1.2. Organisation of Thesis

In Chapter 2 I introduce the main computer sciences which apply topic models: information retrieval, natural language processing and text mining.

Chapter 3 describes topic models and their use in general. It explains LDA's theory and the Gibbs estimation algorithm.

Chapter 4 lists available topic model software implementations and then discusses how LDA topic modeling can be accomplished within the environment of the R programming language.

Chapter 5 is the main application part, in which I aim at replicating the results of an often cited paper on LDA (Griffiths and Steyvers, 2004). This involves downloading a large corpus of abstracts from the PNAS journal's[1] web page, preprocessing the data, performing model selection and fitting, and text mining the corpus via the model data. Chapter 6 concludes this thesis with a summary, the insights gained and the research gaps identified.

[1] PNAS... Proceedings of the National Academy of Sciences

The appendix contains source code that is too long to be printed in the main chapters and lists all other resources used for the thesis: a list of the software that was used, a description of the Sun Grid Engine, R source code listings and documentation of the corpus download.

2. Background in Computer Sciences

In this chapter I give short introductions to the computer sciences of information retrieval, natural language processing and text mining, which are closely related and sometimes overlap in their goals and tools. The first probabilistic topic model, latent Dirichlet allocation (LDA), was presented as an information retrieval model (Blei et al., 2003), solveable by machine learning (a subdiscipline of the computer science artificial intelligence) techniques. Topic models can also be seen as classical text mining or natural language processing tools, therefore I have opted to shortly describe these fields as well. It should be noted that research in these domains is highly cross-disciplinary and also touches on linguistics, statistics and psychology.

2.1. Information Retrieval

Consider a collection of hundred thousand or more digital text documents, for example scanned books. You now want to make these data accessible to a normal person through a computer interface. Ideally, the user should be able to enter a few words that describe what he or she is looking for and immediately after submitting a query, the relevant results should be displayed (in a ranked order), serving as an entry point to the full texts. Also, the user should be notified when new documents that are similar to his or her previous results are added to the collection, and furthermore they should also be able to browse the data guided by a meaningful structure instead of just filtering it.

If you were to design and implement a system according to the specifications above, you would invariably face the following problems:

- how to match the meaning of a query to that of a relevant text document; consider the case of synonyms (different words with the same meaning) and homographs (same spelling, different meaning),
- how to rank search results,
- how to find similar documents,
- how to to process terabytes of data when the user expects instant results, etc.

These and related questions are addressed by the computer science of **information retrieval** (commonly abbreviated as IR), which deals with "searching for documents, for information within documents, and for metadata about documents, as well as that of searching relational databases and the World Wide Web" (Wikipedia, 2011a). The history of information retrieval started in the 1950s, when it was one of the first applications

of early computers (Stock, 2007). Since then it has become an important aspect of the tasks we expect computers to carry out for us, most prominently shown by our everyday use of web search engines.

The main goals of IR may thus be reduced to enabling (Baeza-Yates and Ribeiro-Neto, 1999, Chapters 1 and 2):

- **adhoc retrieval**,
- **filtering** (of new documents in a collection, based on a user profile), and
- **browsing**.

As explained in Manning et al. (2009), the simplest way to match terms of an adhoc retrieval query to documents is a linear scan within the full texts (also called **grep**ping, after the standard Unix program that "**g**ets **r**egular **exp**ressions"), which returns the passages where the search terms occur. Naturally, this would require the system to scan through the entire collection from beginning to end for each new query, making this approach unfeasible for large databases. A general solution to this problem is to prepare a document **index** in advance.

An index is a sorted term dictionary that points to where these terms occur in a document collection. Therefore, to each term in a collection's vocabulary the index maps in which document the term was posted and optionally other information as well, such as how often the term occurs in general or in a specific document (see Figure 2.1).

Such indices are also called **inverted indices** or lists (Weiss et al., 2005) because they are built by sorting a forward index (a list of words for each document) by words (instead of by documents). The inverted index is "essentially without rivals [...] the most efficient structure for supporting ad hoc text search" (Manning et al., 2009), because:

- It can be **treated like a matrix**, but occupies less space than a regular matrix due to the sparsity of the information, meaning there are relatively few non-zero entries in the matrix. Adding and removing documents can be translated into intuitive operations on the matrix.
- It allows us to **separate dictionary lists and posting lists,** by keeping the terms in the working memory and referencing (the larger) term postings (that are stored on non-volatile storage devices like hard disks) by pointers, thereby optimizing the use of system resources.

One side effect of the use of inverted indices is that the word order in original documents is lost. This can be fixed by storing word positions in the posting lists, at the cost of additionally occupied memory. Most topic models (introduced in Chapter 3.1) rely on input from a so-called document-term matrix, which is essentially an index in the form of one sparse matrix that lists term-per-document frequencies.

Note that indices usually are built after **preprocessing** the original data. This step might include natural language processing techniques (see Section 2.2) such as tokenization, that is, the decision which sequence of letters is treated as a single term, or the removal of certain terms (so called "stop words").

Document 1	When Chuck Norris does division, there are no remainders.
Document 2	Chuck Norris can access private methods.

Term	Doc. ID
When	1
Chuck	1
Norris	1
does	1
division	1
there	1
are	1
no	1
remainders	1
Chuck	2
Norris	2
can	2
access	2
private	2
methods	2

Term	Doc. ID
access	2
are	1
can	2
Chuck	1
Chuck	2
division	1
does	1
methods	2
no	1
Norris	1
Norris	2
private	2
remainders	1
there	1
When	1

Term	Document frequency	→	Postings lists
access	1	→	2
are	1	→	1
can	1	→	2
Chuck	2	→	**1,2**
division	1	→	1
does	1	→	1
methods	1	→	2
no	1	→	1
Norris	2	→	**1,2**
private	1	→	2
remainders	1	→	1
there	1	→	1
When	1	→	1

Figure 2.1.: Construction of an inverted index from two short documents, adapted from Manning et al. (2009, Figure 1.4).

2.1.1. Common Information Retrieval Models

Having established inverted indices as the most basic representation of documents in information retrieval models we can now look at what types of models are available to accomplish the main goals of IR.

Figure 2.2 on page 8 is a taxonomy of IR models that was adapted from Wikipedia (2011a), similar categorizations can be found in Baeza-Yates and Ribeiro-Neto (1999), Stock (2007) and Manning et al. (2009).

The figure arranges IR models along two dimensions: first by the models' mathematical basis and, secondly, by the way term-interdependencies are treated. (The following two subsections were quoted verbatim from Wikipedia (2011a) and extended with references where seen as appropriate.)

Categorization by Mathematical Basis

"**Set-theoretic models** represent documents as sets of words or phrases. Similarities are usually derived from set-theoretic operations on those sets. Common models are:

- **Standard Boolean model**: [Baeza-Yates and Ribeiro-Neto (1999, Ch. 2.5.2)],
- **Extended Boolean model**: [Salton et al. (1983), Baeza-Yates and Ribeiro-

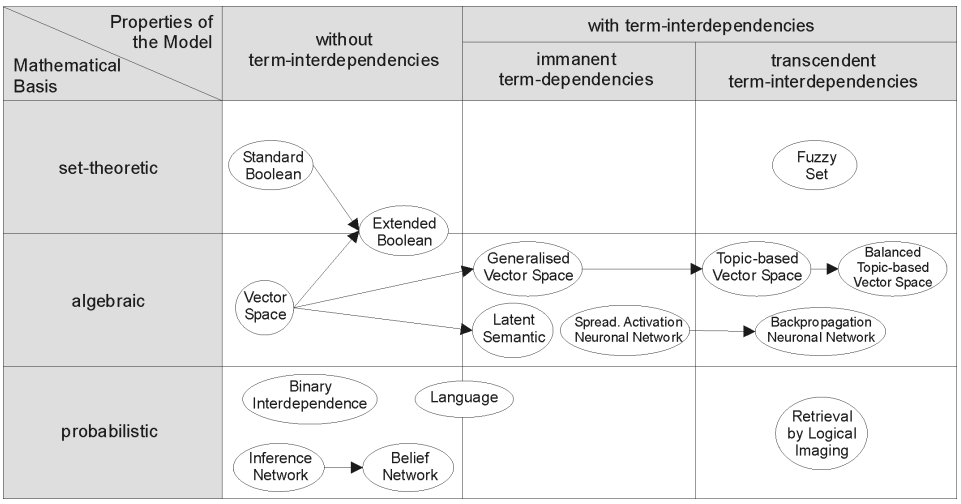


Figure 2.2.: Categorization of IR models, taken from the Wikipedia entry for IR, the original figure was published in Kuropka (2004).

Neto (1999, Ch. 2.6.2)]

- **Fuzzy retrieval**: [Fox and Sharat (1986), Baeza-Yates and Ribeiro-Neto (1999, Ch. 2.6.1)]

Algebraic models represent documents and queries usually as vectors, matrices, or tuples. The similarity of the query vector and document vector is represented as a scalar value.

- **Vector space model**: [Salton et al. (1975), Manning et al. (2009, Ch. 6.3),]
- **Generalized vector space model**: [Wong et al. (1985), Baeza-Yates and Ribeiro-Neto (1999, Ch. 2.5.1),]
- **Extended Boolean model**: [Salton et al. (1983), also a set-theoretic model.]
- **Latent semantic indexing** (LSI), [also known as] Latent semantic analysis (LSA): [see Deerwester et al. (1990), Baeza-Yates and Ribeiro-Neto (1999, Ch. 2.7.2). LSI can be considered a non-probabilistic topic model.]

Probabilistic models treat the process of document retrieval as a probabilistic inference [problem]. Similarities are computed as probabilities that a document is relevant for a given query. Probabilistic theorems like the Bayes' theorem are often used in these models.

- **Binary independence model**: [Robertson and Sparck Jones (1976), Baeza-Yates and Ribeiro-Neto (1999, Ch. 2.5.4).]
- **Probabilistic relevance model** [and its derivative, Okapi BM25: Robertson and Sparck Jones (1976).]
- **Uncertain inference**: [van Rijsbergen (1986).]

[...]

- **Language models**, [probability distributions over word sequences, like n-gram models: Manning et al. (2009, Ch. 12), Zhai (2009).]
- [**Probabilistic topic models**, probabilistic latent semantic indexing (PLSI, also called PLSA, Hofmann (1999)), latent Dirichlet allocation (LDA, Blei et al. (2003)).]

[...]" (Wikipedia, 2011a)

Categorization by Modelling of Term-Interdependencies

"**Models without term-interdependencies** treat different terms/words as independent. This fact is usually represented in vector space models by the orthogonality assumption of term vectors or in probabilistic models by an independency assumption for term variables.

Models with immanent term interdependencies allow a representation of interdependencies between terms. However the degree of the interdependency between two terms is defined by the model itself. It is usually directly or indirectly derived (e.g., by dimensional reduction) from the co-occurrence of those terms in the whole set of documents.

Models with transcendent term interdependencies allow a representation of interdependencies between terms, but they do not allege how the interdependency between two terms is defined. They [rely on] an external source for the degree of interdependency between two terms. (For example a human or sophisticated algorithms.)

[...]" (Wikipedia, 2011a)

2.2. Natural Language Processing

As explained above, information retrieval intends to make document collections more accessible via search engines or interfaces for browsing. In contrast to that, the computer science of **natural language processing** (NLP) deals with tasks of processing human language. NLP has its roots in artificial intelligence (AI) research, but other major scientific domains have approached the same problems as well: linguistics ("computational linguistics"), electrical engineering ("speech recognition") and psychology ("computational psycholinguistics") (Jurafsky and Martin, 1999).

Major goals in NLP are (Jurafsky and Martin, 1999): speech recognition and synthesis, natural language understanding and machine translation.

NLP tools have often been incorporated into IR systems to enhance retrieval performance, however especially with large corpora (i.e., gigabytes of full text) only the most efficient and robust techniques are advised (Kruschwitz, 2005, p. 24).

Figure 2.3 on page 11 shows which NLP techniques are commonly used in the context of IR. Order and actual application of the steps in the diagram to both queries and documents depend on the respective goals of an analysis.

The workflow in the figure can be described as follows (Stock, 2007, p. 100):

- If unknown, the **character encoding** (e.g., Unicode, ASCII, ...) of a document or query must be determined.
- Next, the **language** is detected (English, Russian, ...), to which the determination of a writing system (Latin alphabet, Cyrillic alphabet, ...) is also related, including its respective directionality (left-to-right or right-to-left).
- If the document is in a markup format like HTML, then the text needs to be separated from **layout or navigational elements**.
- Instead of using words as the basis of an analysis, one can also use a text that is converted into **n-grams**. In general, an n-gram is a subsequence of n items of a sequence (Wikipedia, 2011d). If $n = 1$ then one speaks of a unigram, an n-gram with $n = 2$ is referred to as a bigram, one with $n = 3$ as a trigram. Here in this context the items are sequences of single letters (as opposed to word n-grams, which for example are used in certain language models).

  If an IR model is n-gram based, the steps of single word based processing can be skipped and the n-grams are directly fed into models (Stock, 2007, p. 211).
- If the model is based on words, a necessary step is to split a document into single words. This method is called **text (or word) segmentation** or **tokenization** (Manning and Schütze, 1999, p. 124–129) and aided by the fact that most languages delimit words by whitespace.
- Words which are of little use to an analysis ("and", "is", ...) can be tagged and deleted as **stop words** that are often listed on a **stop list** (Manning and Schütze, 1999, p. 533).
- When a model relies on direct user input, **recognition and correction of spelling errors** may be warranted at this point (Stock, 2007, Ch. 18).
- To keep certain words from being mangled in a subsequent step, **named entity recognition** may be applied to tag these words (Stock, 2007, p. 255).

  Different word forms (for example "retrieving" and "retrieval") can then be normalized via **conflation** (Stock, 2007, p. 227). If just the word stem is needed for retrieval, then one employs **stemming** ("retriev"). Stemming is commonly implemented as a simple removal of suffixes (or prefixes). If the conflation requires a linguistically correct word form as an outcome ("retrieve"), **lemmatization** is used. In this case dictionaries might be needed to help map the different word forms to a base form.

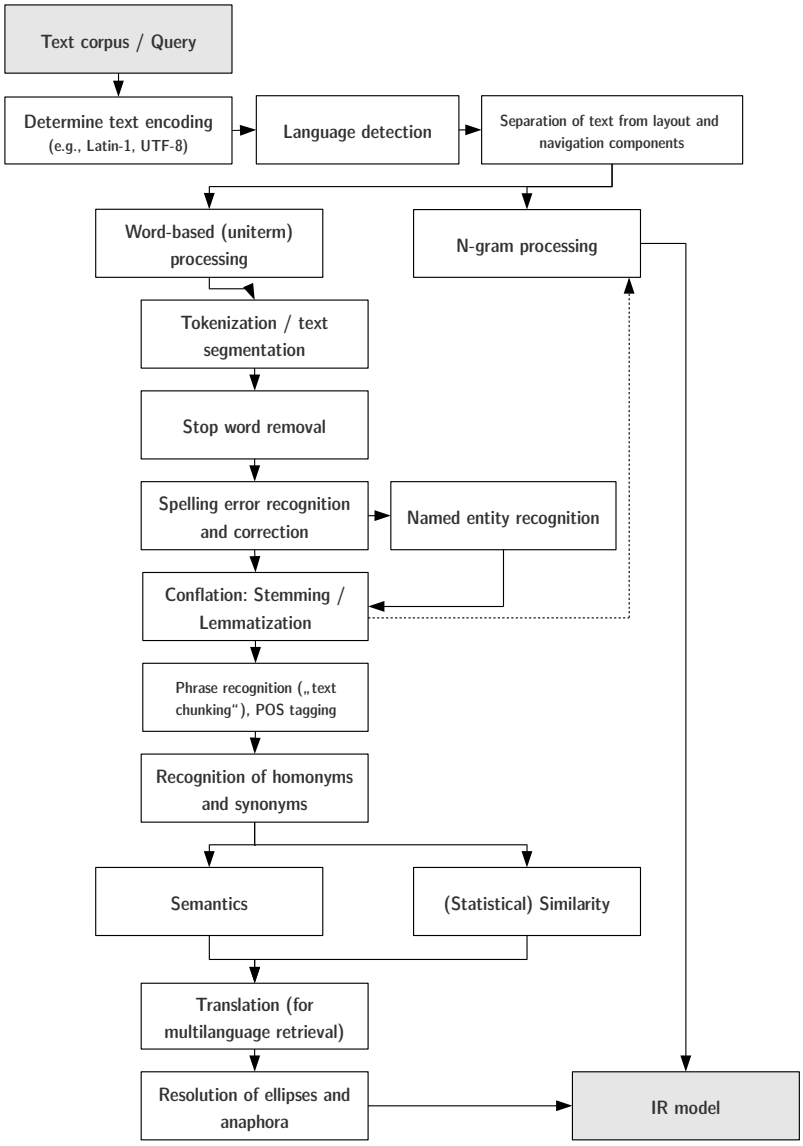


Figure 2.3.: Typical application of NLP techniques for information retrieval models (adapted and translated from Stock (2007, p. 101, Fig. 8.5)).

- **Part-of-speech tagging** is used to mark up word classes within sentences (noun, verb, ...; Manning and Schütze, 1999, Ch. 10). A more shallow form thereof is **text chunking**, which may be applied to identify and tag compound words (also called "phrases"; Stock, 2007, Ch. 15). Information obtained by these techniques may help to build a better inverted index and thus increase retrieval performance.
- Moving into the world of **semantics** (meaning) and **similarity**, words may be tagged as **homonyms** (words with the same spelling and pronounciation but different meanings) or **synonyms** (different words with the same meaning).
- If a query should return results in more than one language, then a **translation** module must be used (Stock, 2007, Ch. 28).
- Finally, **resolution of anaphora** (pronouns like "she" or "he") and **ellipses** (omissions in a sentence) may further normalize the meaning of a text (Stock, 2007, Ch. 17).

2.3. Text Mining

Text mining is the "process of deriving [(new)] high-quality information from text" (Wikipedia, 2011b). It is a highly cross-disciplinary field that can trace its roots to the theory and practice of **data mining** (also called "knowledge discovery in databases", or KDD; Alexander and Wolff, 2005; Weiss et al., 2005). Due to its relative novelty, text mining often overlaps with information retrieval or natural language processing (Hearst, 1999; Alexander and Wolff, 2005; Wikipedia, 2011c). (Semi-)automated tasks in text mining include:

- **Classification**, or **categorization**, i.e., "assign[ing] a priori known labels to text documents" (Feinerer, 2008), with applications like spam-classification (Berry and Kogan, 2010).
- **Clustering**, or "relationship identification, i.e., finding connections and similarities between distinct subsets of documents in the corpus" (Feinerer, 2008). Topic models are well suited to address this task (Srivastava and Sahami, 2009; Berry and Kogan, 2010).
- **Keyword extraction** (Berry and Kogan, 2010).
- **Summarization** (Fan et al., 2006).
- **Topic tracking** (Fan et al., 2006). Topic models have been proposed specifically for this task (Blei and Lafferty, 2006; Wang and McCallum, 2006; Wei et al., 2007).
- **Anomaly (novelty) and trend detection** and **prediction** (Weiss et al., 2005; Berry and Kogan, 2010), which are related to topic tracking.

3. Latent Dirichlet Allocation: a Topic Model

3.1. Topic Models

Topic models are "[probabilistic] latent variable models of documents that exploit the correlations among the words and latent semantic themes" (Blei and Lafferty, 2007). The name "topics" signifies the hidden, to be estimated, variable relations (=distributions) that link words in a vocabulary and their occurrence in documents. A document is seen as a mixture of topics. This intuitive explanation of how documents can be generated is modeled as a stochastic process which is then "reversed" (Blei and Lafferty, 2009) by machine learning techniques that return estimates of the latent variables. With these estimates it is possible to perform information retrieval or text mining tasks on a document corpus.

3.2. Literature and Impact

Latent Dirichlet Allocation (LDA) has been thoroughly explained in the original paper by Blei et al. (2003), and also in Griffiths and Steyvers (2004); Heinrich (2005); Blei and Lafferty (2009); Berry and Kogan (2010); Blei (2011) and others. A video lecture on the topic given by David Blei at the MLSS[1] 2009 in Cambridge is also available online (Blei, 2009).

LDA "[...] has made a big impact in the fields of natural language processing and statistical machine learning" (Wang and McCallum, 2005) and "[...] has quickly become one of the most popular probabilistic text modeling techniques in machine learning and has inspired a series of research works in this direction" (Wei and Croft, 2007).

Among the research that LDA has spawned are different estimation methods (e.g., Griffiths and Steyvers (2004), Asuncion et al. (2009), etc.) and also numerous extensions to the standard LDA model, e.g., hierarchical Dirichlet processes (HDP, Teh et al., 2006b), dynamic topic models (DTM, Blei and Lafferty, 2006), correlated topic models (CTM, Blei and Lafferty, 2007), etc.

What follows in the next sections is an overview of the LDA topic model, largely based on the original authors' work.

[1] MLSS... Machine Learning Summer School

3.3. Observed Data / Model Input

LDA was first applied to text corpora (Blei et al., 2003), although it later has been extended to images (Iwata et al., 2007) and videos (Wang et al., 2007) as well. In this thesis I will only cover analyses of texts. Words (= terms) are the basic unit of data in a document. The set of all words in a corpus is called a vocabulary. Splitting up a document into words is referred to as "tokenization" (also see Section 2.2).

LDA relies on the bag-of-words assumption (Blei et al., 2003), which means that the words in a document are exchangeable and their order therefore is not important. This leads to the representation of a document collection as a so-called document-term matrix (DTM, sometimes also a term-document matrix, which simply is the transposed DTM), in which the frequencies of words in documents are captured. Figure 3.1 shows an exemplary document-term matrix of a small corpus.

	Term 1	Term 2	Term 3	Term 4	Term 5=V
Document 1	2	1	0	0	1
Document 2	1	0	2	3	1
Document 3=D	0	2	4	0	0

Figure 3.1.: Exemplary document-term matrix with $D = 3$ documents and vocabulary of size $V = 5$ terms.

3.4. The Dirichlet Distribution

LDA's generative model posits that the characteristics of topics and documents are drawn from Dirichlet distributions.

The Dirichlet distribution is the multivariate generalization of the beta distribution which itself has been used in Bayesian statistics for modeling belief (see for example Neapolitan, 2003, Chapters 6 and 7; Balakrishnan and Nevzorov, 2003, Ch. 27). The Dirichlet distribution's probability density function is defined as (see for example Blei and Lafferty, 2009):

$$p(x|\alpha_1, \ldots, \alpha_K) = \frac{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)}{\prod_{i=1}^{K} \Gamma(\alpha_i)} \prod_{i=1}^{K} x_i^{\alpha_i - 1}, \tag{3.1}$$

with α being a positive K-vector and Γ denoting the Gamma function, which is a generalization of the factorial function to real values (see for example Neapolitan, 2003, p. 298):

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t}\, dt. \tag{3.2}$$

Figure 3.2 on page 16 shows the density of Dirichlet distributions with various α parameters over the two-simplex. A greater sum of α (sometimes called "scaling") leads to a peakier density (Blei, 2009). Values for α smaller than one lead to higher density at the extremes of the simplex.

Figure 3.3 on page 17 shows samples drawn from Dirichlet distributions of order five. The Dirichlets are symmetric, meaning that the vector α contains identical values. As these values approach zero, the draws get sparser, favoring more extreme outcomes on the five-simplex.

3.5. Generative Process

In order to infer topics from a corpus one has to have a certain model of how documents are generated, i.e. how a person with a limited set of tools and information (model parameters) would come up with a new document, word by word (not taking into account correct word order). LDA's generative model can be summarized as:

1. for each topic: decide what words are likely.
2. for each document,
   a) decide what proportions of topics should be in the document,
   b) for each word,
      i. choose a topic,
      ii. given this topic, choose a likely word (generated in step 1).

The LDA model involves drawing samples from Dirichlet distributions (see previous section) and from multinomial distributions. A generalization of the binomial distribution, the multinomial distribution tells us the probability of observing a count of two or more independent events, given the number of draws and fixed probabilities per outcome that sum to one. In our case the outcomes are terms and topics.

A geometric interpretation of the model is shown in Figure 3.4 on page 18, where a simple generative model of three terms, three topics and the resulting four documents is represented on the two-simplex.

To formally describe the model, some symbols are introduced in Table 3.1 on page 18.

The probabilistic generative process is defined (Blei and Lafferty, 2009) as:

1. For each topic k, draw a distribution over words $\phi_k \sim Dir(\alpha)$.
2. For each document d,
   a) Draw a vector of topic proportions $\theta_d \sim Dir(\beta)$.
   b) For each word i,
      i. Draw a topic assignment $z_{d,i} \sim Mult(\theta_d), z_{d,n} \in \{1, \ldots, K\}$,
      ii. Draw a word $w_{d,i} \sim Mult(\phi_{z_{d,i}}), w_{d,i} \in \{1, \ldots, V\}$.

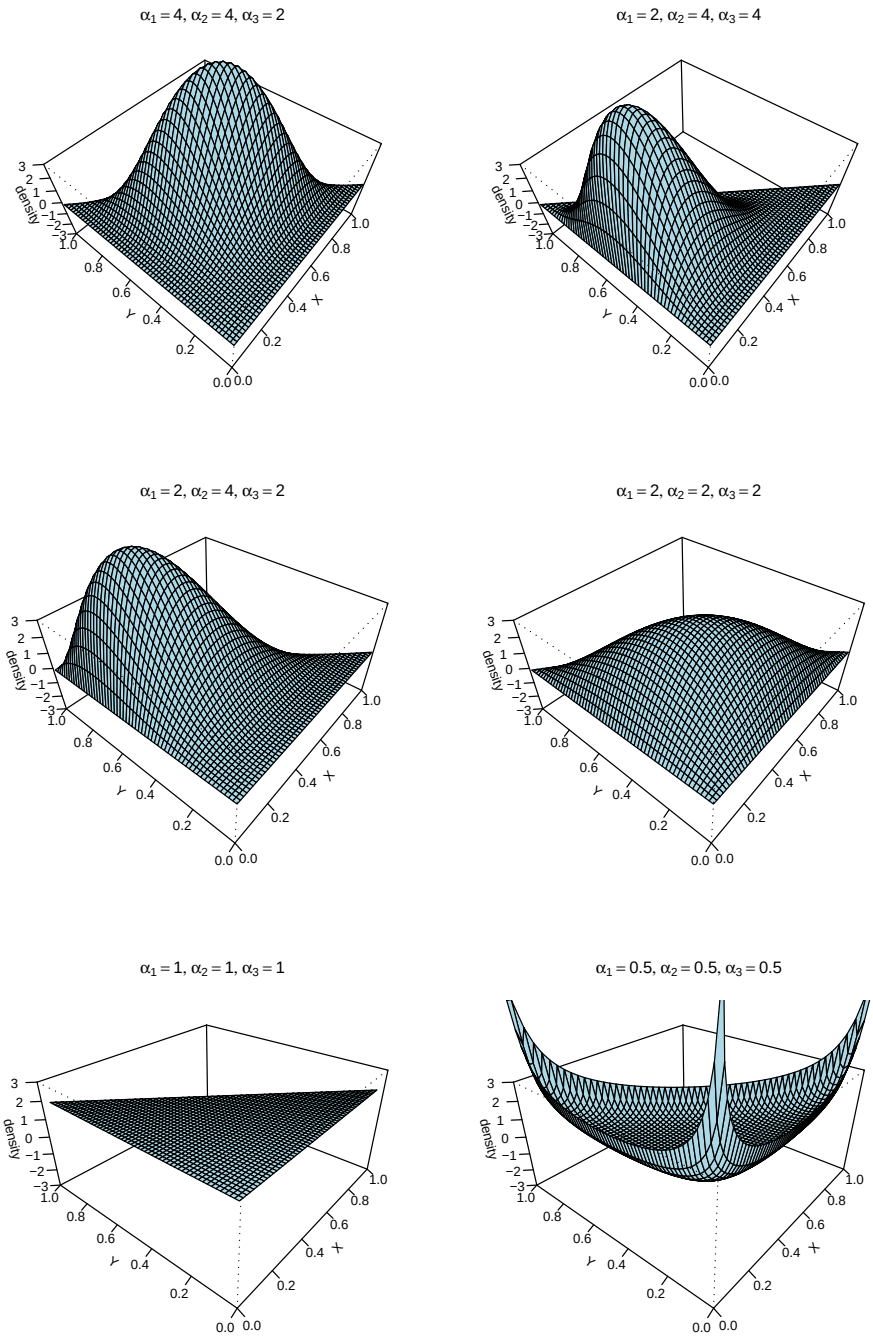


Figure 3.2.: Order $K = 3$ Dirichlet density plots (R code: Appendix C.2).

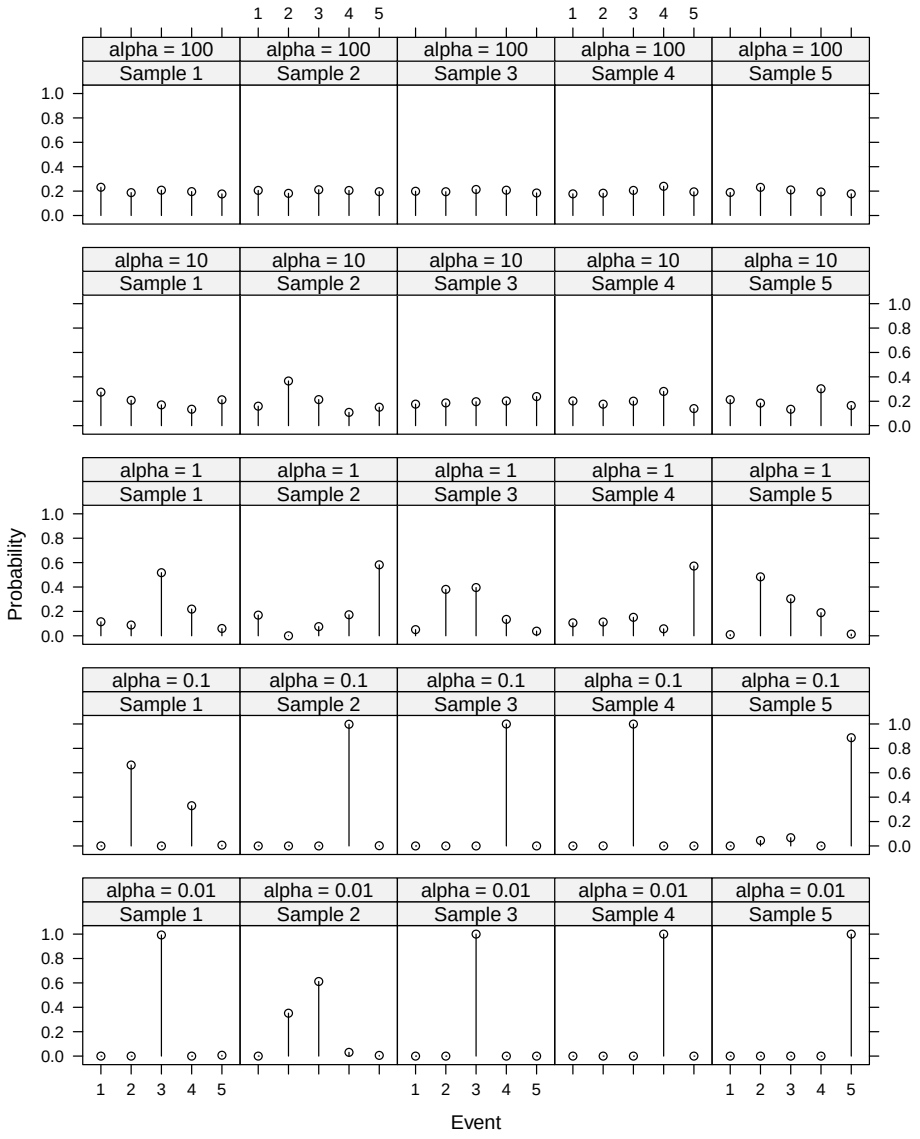
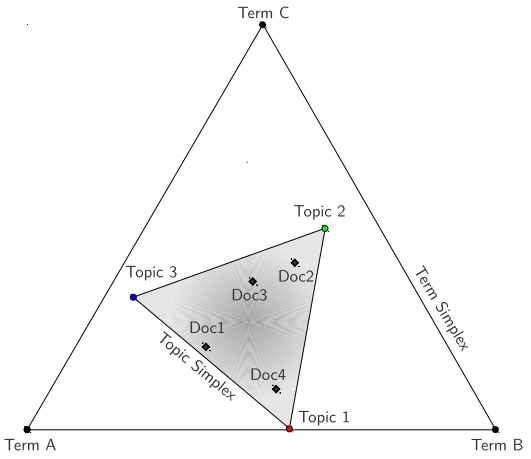


Figure 3.3.: $K = 5$ symmetric Dirichlet random samples (R code: Appendix C.3).



	Term A	Term B	Term C
Topic 1	0.45	0.56	0.00
Topic 2	0.12	0.39	0.49
Topic 3	0.62	0.06	0.32

	Topic 1	Topic 2	Topic 3
Document 1	0.41	0.05	0.55
Document 2	0.13	0.74	0.13
Document 3	0.15	0.50	0.35
Document 4	0.77	0.12	0.11

	Term A	Term B	Term C
Document 1	0.52	0.28	0.20
Document 2	0.23	0.36	0.41
Document 3	0.34	0.30	0.36
Document 4	0.42	0.48	0.10

Figure 3.4.: Geometric interpretation of LDA (adapted from Blei et al., 2003, Figure 4).

K	specified number of topics
k	auxiliary index over topics
V	number of words in vocabulary
v	auxiliary index over topics
d	auxiliary index over documents
N_d	document length (number of words)
i	auxiliary index over words in a document
α	positive K-vector
β	positive V-vector
$Dir(\alpha)$	a K-dimensional Dirichlet
$Dir(\beta)$	a V-dimensional Dirichlet
z	Topic indices: $z_{d,i} = k$ means that the i-th word in the d-th document is assigned to topic k

Table 3.1.: List of symbols in this thesis.

As an example of a hierarchical Bayesian model (see for example Gelman et al., 2004) LDA can also be denoted as a graphical model in plates notation. Graphical models are a standard way of representing statistical models of "random variables that are linked in complex ways" (Jordan, 2004). Figure 3.5 shows LDA in plates notation, where rectangles signify a repetition over enclosed nodes.

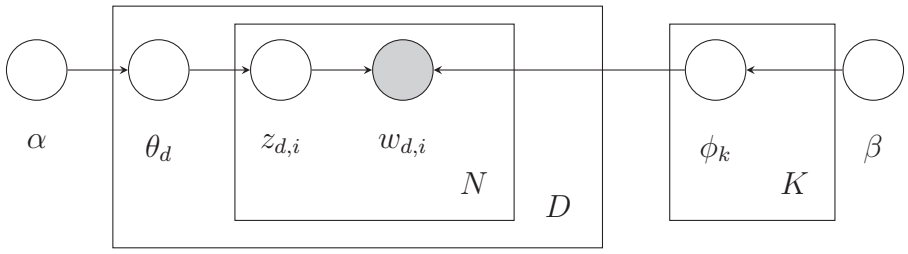


Figure 3.5.: LDA in plates notation (adapted from Blei and Lafferty (2009)).

A translation of this directed acyclical graph (DAG) into a joint probability distribution of independent variables is possible when the Markov condition holds true, which is the case (see for example Neapolitan, 2003). From Figure 3.5 we therefore deduce the following joint distribution:

$$p(w, z, \theta, \phi|\alpha, \beta) = p(\theta|\alpha)p(z|\theta)p(\phi|\beta)p(w|z, \phi). \tag{3.3}$$

Marginalizing over the joint distribution will allow us to get the looked after probabilities that are contained in the model.

I will now explain in detail the four factors on the right hand side of the equation above.

First, the **topics distribution per document**, which is drawn from the Dirichlet distribution, given the Dirichlet parameter α, which is a K-vector with components $\alpha_k > 0$:

$$p(\theta|\alpha) = \frac{\Gamma(\sum_{k=1}^{K} \alpha_k)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \theta_1^{\alpha_k - 1} \cdots \theta_K^{\alpha_K - 1} = \frac{\Gamma(\alpha_\cdot)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} \theta_k^{\alpha_k - 1}.$$

The second, shorter version of this equation introduces the use of the dot operator $(\cdot)$ in the variable index as a shorthand for summation over all the variables' values. The following table is an example of three such per-document distributions:

		Topic 1	Topic 2	Topic 3	Topic 4
Document 1	$\theta_{d=1}$	0.5	0.1	0.3	0.1
Document 2	$\theta_{d=2}$	0.0	0.9	0.1	0.0
Document 3	$\theta_{d=3=D}$	0.02	0.48	0.25	0.25

The next component of the joint distribution is the distribution of the **topic to words assignments** in the corpus, z, which depends on the aforementioned distribution θ. Each word w_i in a document of N words is therefore assigned a value from $1 \dots K$. The next table shows an example of this:

		Word w_1	Word w_2	Word w_3	Word w_4	Word w_5	Word w_6
Document 1	$z_{d=1}$	Topic k=2	Topic k=1	Topic k=1	Topic k=4	Topic k=3	Topic k=3
Document 2	$z_{d=2}$	Topic k=2	Topic k=3	Topic k=2	Topic k=2		
Document 3	$z_{d=3=D}$	Topic k=4	Topic k=2	Topic k=2	Topic k=4	Topic k=3	

In the joint distribution we express the probability of z for all documents and topics in terms of the word count $n_{d,k,\cdot}$ which is the number of times topic k has been assigned to any word in document d:

$$p(z|\theta) = \prod_{d=1}^{D} \prod_{k=1}^{K} \theta_{d,k}^{n_{d,k,\cdot}}.$$

The **term distributions per topic** of the whole corpus ϕ_k are (again) drawn from a Dirichlet distribution with parameter β. This is an example of some possible topic distributions for four topics:

		Term 1	Term 2	Term 3	Term 4	Term 5=V
Topic 1	$\phi_{k=1}$	0.1	0.1	0	0.7	0.1
Topic 2	$\phi_{k=2}$	0.2	0.1	0.2	0.2	0.3
Topic 3	$\phi_{k=3}$	0.01	0.2	0.39	0.3	0.1
Topic 4	$\phi_{k=4=K}$	0.0	0.0	0.5	0.3	0.2

$\phi_{k,v}$ gives us the probability that term v is drawn when the topic was chosen to be k. We express the probability of ϕ for all topics and all words of the vocabulary as:

$$p(\phi|\beta) = \prod_{k=1}^{K} \frac{\Gamma(\beta_{k,\cdot})}{\prod_{v=1}^{V} \Gamma(\beta_{k,v})} \prod_{v=1}^{V} \phi_{k,v}^{\beta_{k,v}-1}.$$

Finally, the **probability of a corpus** w given its parents z and ϕ in the graphical model is:

$$p(w|z,\phi) = \prod_{k=1}^{K} \prod_{v=1}^{V} \phi_{k,v}^{n_{\cdot,k,v}},$$

where $n_{\cdot,k,v}$ is the count how many times topic k was assigned to vocabulary term v in the whole corpus.

This is the complete and regrouped joint distribution:

$$p(w,z,\theta,\phi|\alpha,\eta) = p(\theta|\alpha)p(z|\theta)p(\phi|\beta)p(w|z,\phi) =$$
$$= \left(\prod_{d=1}^{D}\frac{\Gamma(\alpha_.)}{\prod_{k=1}^{K}\Gamma(\alpha_k)}\prod_{k=1}^{K}\theta_{d,k}^{\alpha_k-1}\right)\left(\prod_{d=1}^{D}\prod_{k=1}^{K}\theta_{d,k}^{n_{d,k,\cdot}}\right)\times$$
$$\left(\prod_{k=1}^{K}\frac{\Gamma(\beta_{k,\cdot})}{\prod_{k=1}^{V}\Gamma(\beta_{k,v})}\prod_{v=1}^{V}\phi_{k,v}^{\beta_{k,v}-1}\right)\left(\prod_{k=1}^{K}\prod_{v=1}^{V}\phi_{k,v}^{n_{\cdot,k,v}}\right) =$$
$$= \left(\prod_{d=1}^{D}\frac{\Gamma(\alpha_.)}{\prod_{k=1}^{K}\Gamma(\alpha_k)}\prod_{k=1}^{K}\theta_{d,k}^{\alpha_k+n_{d,k,\cdot}-1}\right)\times$$
$$\left(\prod_{k=1}^{K}\frac{\Gamma(\beta_{k\cdot})}{\prod_{v=1}^{V}\Gamma(\beta_{k,v})}\prod_{v=1}^{V}\phi_{k,v}^{\beta_{k,v}+n_{\cdot,k,v}-1}\right). \quad (3.4)$$

We now marginalize the latent variables out in order to be able to write down a model's probability when a corpus w and the hyperparameters (α and β) are given. This probability is needed for a "maximum-likelihood estimation of the model parameters and to infer the distribution of latent variables" (Chang and Yu, 2007):

$$p(w|\alpha,\beta) = \int_\phi\int_\theta\sum_z\left(\prod_{d=1}^{D}\frac{\Gamma(\alpha_.)}{\prod_{k=1}^{K}\Gamma(\alpha_k)}\prod_{k=1}^{K}\theta_{d,k}^{\alpha_k+n_{d,k,\cdot}-1}\right)\times$$
$$\left(\prod_{k=1}^{K}\frac{\Gamma(\beta_{k\cdot})}{\prod_{v=1}^{V}\Gamma(\beta_{k,v})}\prod_{v=1}^{V}\phi_{k,v}^{\beta_{k,v}+n_{\cdot,k,v}-1}\right)\mathrm{d}\theta\mathrm{d}\phi. \quad (3.5)$$

The sum over all possible combinations of topic assignments z ($n_{d,k,\cdot}$, $n_{\cdot,k,v}$) makes this probability computationally intractable (Chang and Yu, 2007; Blei et al., 2003) and therefore prohibits the use of the conventional expectation maximization algorithm. Instead, we have to resort to machine learning techniques to find good approximations of the marginal probability (see next section).

3.6. Model Estimation

The following learning algorithms have been used to infer topic models' (latent) variables (Asuncion et al., 2009; Blei and Lafferty, 2009):

- **Maximum likelihood** (ML) estimation (Hofmann, 1999) for the PLSA model (LDA's predecessor without Dirichlet priors), where α and β are ignored, and ϕ and θ are treated as parameters.
- **Maximum a posteriori** (MAP) estimation (Chien and Wu, 2008), an extension to Hofmann's PLSA model.

- **Variational Bayesian inference** (VB, Blei et al., 2003), where in the smoothed model version z, ϕ and θ are hidden variables and the posterior distribution is approximated using a variational distribution.
- **Collapsed variational Bayesian inference** (Teh et al., 2006a), where the variables ϕ and θ are marginalized ("collapsed").
- **Collapsed Gibbs sampling** (Griffiths and Steyvers, 2004), a Markov Chain Monte Carlo (MCMC)-based method. (Described in more detail in the next subsection.)
- **Expectation Propagation** (EP) (Minka and Lafferty, 2002; Griffiths and Steyvers, 2004).

Having reviewed previous work in Teh et al. (2006a); Mukherjee and Blei (2009); Welling et al. (2008); Girolami and Kabán (2003) and having compared these algorithms, Asuncion et al. (2009) come to the following conclusions:

- "Update equations in these algorithms are closely connected", and
- "[...] using the appropriate hyperparameters causes the performance difference between these algorithms to largely disappear". Hyperparameters can be learned during model training through various approaches (listed in Asuncion et al. (2009)).
- Parallelizing the algorithms is possible and allows close to real-time analysis of small corpora ($D = 3000$ documents).

In the same paper the authors propose an optimized version of collapsed variational Bayesian inference that offers a slight performance advantage compared to the other algorithms ("CVB0").

In this thesis I focus on one of the inference algorithms, namely Gibbs sampling, which I describe in the next subsection.

3.6.1. Collapsed Gibbs Sampling

Computing the marginal distribution (Equation 3.5 on page 21) "involves evaluating a probability distribution on a large discrete state space" (Steyvers et al., 2004). The paper by Steyvers et al. was the first to show how Gibbs sampling could be employed to estimate LDA's latent variables. The Gibbs sampling algorithm is a typical Markov Chain Monte Carlo (MCMC) method and was originally proposed for image restoration (Geman and Geman, 1984). MCMC methods solve the problem of obtaining samples from complex probability distributions by the use of random numbers (MacKay, 2005, Ch. 29).

"Gibbs sampling (also known as alternating conditional sampling) [...] simulates a high-dimensional distribution by sampling on lower-dimensional subsets of variables where each subset is conditioned on the value of all others. The sampling is done

sequentially and proceeds until the sampled values approximate the target distribution." (Steyvers and Griffiths, 2007). Applied to the LDA model, we need the "probability of topic $z_{a,b}$ being assigned to $w_{a,b}$, the b-th word of the a-th document, given $z_{-(a,b)}$, all the other topic assignments to all the other words" (Carpenter, 2010):

$$p(z_{a,b}|z_{-(a,b)}, w, \alpha, \beta), \tag{3.6}$$

which is proportional to (Carpenter, 2010; Chang and Yu, 2007):

$$\propto p(w, z|\alpha, \beta) = \int_{\theta}\int_{\phi} p(w, z, \theta, \phi|\alpha, \beta)\,\mathrm{d}\theta\mathrm{d}\phi. \tag{3.7}$$

A series of expansions and equation transformations (Carpenter, 2010; Elkan, 2010) results in the unnormalized conditional probability (Griffiths and Steyvers, 2004; Steyvers and Griffiths, 2007) (a more general derivation can be found in Heinrich (2005)):

$$p(z_{a,b}|z_{-(a,b)}, w, \alpha, \beta) \propto \frac{n^{-(a,b)}_{z_{a,b},a,\cdot} + \alpha}{n^{-(a,\cdot)}_{z_{a,b},a,\cdot} + K\alpha} \times \frac{n^{-(a,b)}_{z_{a,b},\cdot,w_{a,b}} + \beta}{n^{-(a,b)}_{z_{a,b},\cdot,\cdot} + J\beta}. \tag{3.8}$$

If asymmetric priors α and β are allowed and the topic-related denominator is omitted (which is the same for every word in the vocabulary), we arrive at the following conditional probability (Carpenter, 2010):

$$p(z_{a,b}|z_{-(a,b)}, w, \alpha, \beta) \propto \frac{\left(n^{-(a,b)}_{z_{a,b},a,\cdot} + \alpha_{z_{a,b}}\right) \times \left(n^{-(a,b)}_{z_{a,b},\cdot,w_{a,b}} + \beta_{w_{a,b}}\right)}{n^{-(a,b)}_{z_{a,b},\cdot,\cdot} + \sum_{j=1}^{J}\beta_j}. \tag{3.9}$$

"The first multiplicand in the numerator, $n^{-(a,b)}_{z_{a,b},a,\cdot} + \alpha_{z_{a,b}}$, is just the number of other word in document a that have been assigned to topic $z_{a,b}$ plus the topic prior. The second multiplicand in the numerator, $n^{-(a,b)}_{z_{a,b},\cdot,w_{a,b}} + \beta_{w_{a,b}}$, is the number of times the current word $w_{a,b}$ has been assigned to topic $z_{a,b}$ plus the word prior. The denominator just normalizes the second term to a probability." (Carpenter, 2010). The normalized conditional probability is:

$$p(z_{a,b}|z_{-(a,b)}, w, \alpha, \beta) = \frac{\left(\frac{\left(n^{-(a,b)}_{z_{a,b},a,\cdot} + \alpha_{z_{a,b}}\right) \times \left(n^{-(a,b)}_{z_{a,b},\cdot,w_{a,b}} + \beta_{w_{a,b}}\right)}{n^{-(a,b)}_{z_{a,b},\cdot,\cdot} + \sum_{j=1}^{J}\beta_j}\right)}{\left(\sum_{k=1}^{K} \frac{\left(n^{-(a,b)}_{z_{a,b},a,\cdot} + \alpha_{z_{a,b}}\right) \times \left(n^{-(a,b)}_{z_{a,b},\cdot,w_{a,b}} + \beta_{w_{a,b}}\right)}{n^{-(a,b)}_{z_{a,b},\cdot,\cdot} + \sum_{j=1}^{J}\beta_j}\right)}. \tag{3.10}$$

The learning algorithm can be summarized (Steyvers and Griffiths, 2007) as:

1. Initialize the topic to word assignments z randomly from $\{1,\ldots,K\}$.
2. For each Gibbs sample:
   a) "For each word token, the count matrices $n^{-(a,b)}$ are first decremented by one for the entries that correspond to the current topic assignment." (Steyvers and Griffiths, 2007)

b) A new topic is sampled from the distribution in Equation 3.10.

c) The count matrices are updated by incrementing by one at the new topic assignment.

3. Discard samples during the initial burn-in period.

4. After the Markov chain has reached a stationary distribution, i.e., the posterior distribution over topic assignments, samples can be taken at a fixed lag (averaging over Gibbs samples is recommended for statistics that are invariant to the ordering of topics (Griffiths and Steyvers, 2004)).

The other latent variables θ and ϕ, which are the interest of most analyses, can be estimated from a single Gibbs sample of z (Griffiths and Steyvers, 2004):

$$\hat{\theta}_{d,k} = \frac{\alpha_k + n_{d,k,\cdot}}{\alpha_\cdot + n_{d,\cdot,\cdot}} \tag{3.11}$$

$$\hat{\phi}_{k,v} = \frac{\beta_{k,v} + n_{\cdot,k,v}}{\beta_{k,\cdot} + n_{\cdot,k,\cdot}} \tag{3.12}$$

3.7. Model Evaluation and Selection

Model evaluation, i.e., measuring a model's performance, is needed to ensure that the (topic) model is able to generalize from the training data in a useful way. One also speaks of **model selection** when the decision is taken of "what model(s) to use for inference", given data x and potential fitted models (Burnham and Anderson, 2002, Ch. 1). For instance, a common problem in topic modeling is to choose the number of topics if this parameter is not specified *a priori* (Blei and Lafferty, 2009). Depending on the goals and available means, a researcher can employ a variety of performance metrics (Wallach et al., 2009; Buntine, 2009; Chang and Blei, 2009), which I list in the next subsections.

3.7.1. Performance Measurement on Data

Most topic model papers measure the performance of the learned model on held-out (i.e., non-training, or test) data (with the PNAS analysis in Griffiths and Steyvers (2004) being an exception, where model selection is performed on the complete corpus). "Estimating the probability of held-out documents provides a clear, interpretable metric for evaluating the performance of topic models relative to other topic-based models as well as to other non-topic-based generative models" (Wallach et al., 2009). These metrics are well suited for qualitative goals, "such as corpus exploration [and] choosing the number of topics that provides the best language model" (Blei and Lafferty, 2009).

A related approach is "document completion", namely to divide documents (as opposed to dividing corpora) into training and test sets (Rosen-Zvi et al., 2004; Wallach et al., 2009). "While this approach is reasonable, it [...] does not address the whole problem, the quality of individual documents" (Buntine, 2009).

So far the following metrics have been used in topic modeling:

- **Perplexity** "[...], used by convention in language modeling, is monotonically decreasing in the likelihood of the test data, and is algebraically equivalent to the inverse of the geometric mean per-word likelihood. A lower perplexity score indicates better generalization performance" (Blei et al., 2003), see also Rosen-Zvi et al. (2004).
- **Empirical likelihood,** see Li and McCallum (2006).
- **Marginal likelihood,** which can be approximated by:
  - **Harmonic mean method,** see Newton and Raftery (1994); Griffiths and Steyvers (2004); Buntine and Jakulin (2004); Griffiths et al. (2005); Wallach (2006) and next subsection.
  - **(Annealed/Mean field) importance sampling,** see Neal (2001); McCallum (2002); Li and McCallum (2006); Buntine (2009).
  - **Chib-style estimation,** see Chib (1995); Murray and Salakhutdinov (2009).
  - **Left-to-right samplers,** see Del Moral et al. (2006); Wallach (2008); Buntine (2009).

In Wallach et al. (2009) it was found that "the Chib-style estimator and 'left-to-right' algorithm presented in this paper provide a clear methodology for accurately assessing and selecting topic models". Further research in Buntine (2009) shows that a sequential version of Wallach's left-to-right sampler and the even more efficient mean field importance sampler are optimal for measuring marginal model likelihood on held-out data.

Harmonic Mean Method

This method has first been applied by Griffiths and Steyvers in their 2004 Bayesian approach to finding the optimal number of topics. The method has since then been used in a variety of papers (Griffiths et al., 2005; Zheng et al., 2006; Wallach, 2006) because of its simplicity and relative computational efficiency, but has also been proven to be unstable (Wallach et al., 2009; Buntine, 2009).

In its effort to replicate the findings of Griffiths and Steyvers, this thesis also makes use of the harmonic mean method for model selection. The method is best summarized by a direct quote from their paper (symbol notation, literature and equation references have been adapted accordingly):

> "In our case, the data are the words in the corpus, w, and the model is specified by the number of topics, K, so we wish to compute the likelihood $p(w|K)$. The complication is that this requires summing over all possible assignments of words to topics z [, i.e., $p(w|K) = \int p(w|z,K)p(z)dz$]. However, we can approximate $p(w|K)$ by taking the harmonic mean of a set of values of $p(w|z,K)$ when z is sampled from the posterior $p(z|w,K)$ [(Kass and Raftery, 1995, Section 4.3)]. Our Gibbs sampling algorithm provides such samples, and the value of $p(w|z,K)$ can be computed from Eq. 3.13[:]"

$$P(w|z) = \left(\frac{\Gamma(V\beta)}{\Gamma(\beta)^V}\right)^K \prod_{k=1}^{K} \frac{\prod_V \Gamma(n_k^{(w)} + \beta)}{\Gamma(n_k^{(\cdot)} + V\beta)}, \tag{3.13}$$

"[...] in which $n_k^{(w)}$ is the number of times word w has been assigned to topic k in the vector of assignments z, and $\Gamma(\cdot)$ is the standard Gamma function" (Griffiths and Steyvers, 2004).

3.7.2. Performance Measurement on Secondary Tasks

In some scenarios the model can be evaluated by cross validation on the error of an external task at hand, such as document classification or information retrieval (Wei and Croft, 2006; Titov and McDonald, 2008).

3.7.3. Performance Measurement by Human Judgement

Steyvers and Griffiths (2007) "showed that the number of topics a word appears in correlates with how many distinct senses it has and reproduced many of the metrics used in the psychological community based on human performance" (Chang and Blei, 2009). Following up on this line of thought, Chang and Blei (2009) presented two metrics that allow for a better evaluation of a topic model's latent structure by using input from humans:

- **Word intrusion** "measures how semantically 'cohesive' the topics inferred by a model are and tests whether topics correspond to natural groupings for humans" (Chang and Blei, 2009).
- **Topic intrusion** "measures how well a topic model's decomposition of a document as a mixture of topics agrees with human associations of topics with a document" (Chang and Blei, 2009).

The result of Chang and Blei (2009) was that for three topic models (PLSI, LDA, CTM) traditional measures were "indeed, negatively correlated with the measures of topic quality developed in [their] paper". The authors therefore suggest that topic model developers should "focus on evaluations that depend on real-world task performance rather than optimizing likelihood-based measures" – a view which is re-iterated by Blei (2011).

3.8. Use of Fitted Topic Models

"Representing the content of words and documents with probabilistic topics has one distinct advantage over a purely spatial representation. Each topic is individually interpretable, providing a probability distribution over words that picks out a coherent cluster of correlated terms" (Steyvers and Griffiths, 2007). Topic models thus can be seen as

"soft clusters" (Heinrich, 2005; Newman et al., 2006), which makes them candidates for a multitude of applications.

3.8.1. Finding Similar Documents Through Querying and Browsing

LDA and topic models in general can be applied for the classical tasks of information retrieval, i.e., ad-hoc retrieval, browsing and filtering (see Section 2.1). "Querying can be considered topic estimation of unknown documents (queries) and comparing these topic distributions to those of the known documents. Appropriate similarity measures permit ranking." (Heinrich, 2005).

Enabling the exploring of corpora through **browsing** is LDA's major strength (Blei et al., 2003; Blei and Lafferty, 2009). Links to online examples can be found on David Blei's homepage `http://www.cs.princeton.edu/~blei/topicmodeling.html`.

Examples of LDA's use in **ad-hoc retrieval** have been shown in Azzopardi et al. (2004); Wei and Croft (2006, 2007); Yi and Allan (2008). Filtering new documents in databases and recommender systems are other obvious, though so far unexplored, applications.

3.8.2. Exploring the Relation between Topics and Corpus Variables

Related to browsing and text mining, it is possible to combine topics with secondary corpus variables that were not part of the training process. For the LDA model this was first demonstrated in Griffiths and Steyvers (2004), where two document metavariables, namely year of publication and category designated by the authors, were used to gain additional insight into the corpus structure and model semantics. For example, to find out topic trends over the years, the per-document topic distributions were aggregated by averaging over all documents from the same year. This concept is illustrated in Figure 3.6.

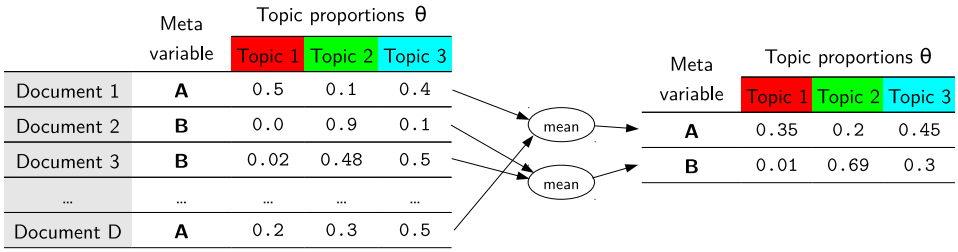


Figure 3.6.: Aggregating topic distributions by a document meta-variable.

3.9. Extending LDA

Since the publication of the LDA topic model (Blei et al., 2003) hundreds of papers have been written on the subject. Among these have been improvements to the model estimation (e.g., distributed inference algorithms, see Newman et al. (2007); Wang (2008); Newman et al. (2009)) and dozens of new model specifications that take into account not only documents but corresponding metadata like date (dynamic topic models, see Blei and Lafferty (2006)) or authors (author-topic model, see Rosen-Zvi et al. (2004, 2005); Shen et al. (2006)). A current bibliography of topic modeling by David Mimno is available at: `http://www.cs.princeton.edu/~mimno/topics.html`. Discussions and announcements on topic modeling are carried out on David Blei's mailing list: `https://lists.cs.princeton.edu/mailman/listinfo/topic-models`.

4. LDA in the R Environment

This chapter gives an overview over current software implementations of the LDA topic model and describes the R packages **tm** and **topicmodels** which I use for the data analysis in more detail.

4.1. Overview of LDA Implementations

The first LDA software was published by David Blei in 2004 as **lda-c** and implements the variational inference described in his earlier paper (Blei et al., 2003). Other researchers have implemented LDA and many other topic models so that today there exists a variety of both open and closed source topic modeling software. Table 4.1 on page 30 lists packages that allow model fitting by LDA. This list is not exhaustive and omits stand-alone implementations of other topic models.

It should be noted that LDA can also be implemented in general machine learning frameworks, examples thereof are:

- **HBC (Hierarchical Bayes Compiler)**, see `http://www.cs.utah.edu/~hal/HBC/`.
- **WinBUGS,** see `http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/contents.shtml` and `http://www.arbylon.net/projects/` for an LDA script by Gregor Heinrich.
- **Apache Mahout,** see `https://cwiki.apache.org/confluence/display/MAHOUT/Latent+Dirichlet+Allocation`.

Package	Models	Language	Author(s)	License
Gensim	LSA, LDA	Python	Radim Řehůřek	GNU LGPL
GibbsLDA++ / JGibbLDA	LDA	C++ / Java	Xuan-Hieu Phan and Cam-Tu Nguyen	GNU LGPL 2
Infer.NET	LDA and others	.NET languages	Microsoft Research Cambridge	non-commercial use only
lda	LDA and others	R	Jonathan Chang	GNU LGPL
lda	LDA	Matlab or C	Daichi Mochihashi	n.a.
lda-c	LDA	C	David Blei	GNU LGPL 2
lda-j	LDA	Java	Gregor Heinrich	GNU LGPL 2
LDA, PLSA	LDA and PLSA	Matlab	Jakob Verbeek	n.a.
lda1pfs	LDA (1PFS inference)	Java	Anthony DiFranco	mostly GPL 3
LingPipe	LDA and others	Java	Alias-i, Inc	proprietary license
LDA with Gibbs sampler	LDA	Python	Mathieu Blondel	n.a.
lsa-lda	LSA, LDA	C	n.a.	GNU GPL 2
MALLET	LDA and others	Java	MALLET team	Common Public License Version 1.0 (CPL)
Matlab Topic Modeling Toolbox	LDA, AT, HMM-LDA, LDA-COL	Matlab	Mark Steyvers, Tom Griffiths	free of charge for research and education
Nonparametric Bayesian Mixture Models	LDA and others	Matlab, C	Yee Whye Teh	free for research or education
online lda	Online inference for LDA	Python	Matt Hoffman	GNU GPL 3
plda	Parallel LDA	C++	Zhiyuan Liu, Yuzhou Zhang, Edward Y. Chang, Maosong Sun	Apache License 2.0
streamLDA	LDA (online/streamed)	Python	Matthew D. Hoffman, Jessy Cowan-Sharp, Jordan Boyd-Grader	GNU GPL 3
tmve	Topic Model Visualization Engine (for lda-c)	Python	Allison Chaney	MIT license
topicmodels	LDA, CTM	R (C, C++)	Bettina Grün, Kurt Hornik	GNU GPL 2
Yahoo! LDA	Parallel LDA (Hadoop)	C++	Shravan Narayanamurthy	Apache License 2.0

Table 4.1.: LDA implementations (2011).

4.2. The R Programming Language and Environment

The practical part of this thesis is topic modeling and exploring by use of the R programming language. The following description is a direct quote from the R homepage (R Development Core Team, 2011):

> "R is a language and environment for statistical computing and graphics. It is a GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R. R provides a wide variety of statistical (linear and nonlinear modeling, classical statistical tests, time-series analysis, classification, clustering, ...) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity."

R is available on Unix-like platforms (Unix, FreeBSD, Linux), Windows and MacOS and its source is openly available under the GNU General Public License 2 (GPL) (R Development Core Team, 2011). The program can be invoked either from the command line in **batch mode,** where complete R scripts are read and evaluated by the interpreter, or run as an **interactive session,** where typed-in expressions are read, evaluated and the results printed immediately. There also exists a variety of graphical user interfaces (GUIs) which wrap the base program for users who prefer integrated development environments instead of a command line, e.g. **RStudio**. Furthermore, text editors like **Emacs** or **vim** offer modes which make writing and testing R scripts easier.

The standard capabilities of R are extensible by so-called **packages** which can be installed by the user and can be loaded from within an R session. A central repository of such additional packages by third parties (and the R code and documentation itself) is the **Comprehensive R Archive Network** (CRAN, `http://cran.r-project.org/`). CRAN mirrors are available around the world and currently host more than 3,400 packages from a wide range of categories, such as econometrics, graphics or machine learning.

4.3. Topic Modeling with the R Packages tm and topicmodels

4.3.1. The topicmodels Package

The R package **topicmodels** (Grün and Hornik, 2011) was written by Bettina Grün and Kurt Hornik and is available from CRAN servers. It includes and provides interfaces to David Blei's open source implementations of LDA (**lda-c**) and CTM (**ctm-c**), both written in the C programming language and using the VEM algorithm for inference.

Also included as an alternative inference method for LDA models is the open source Gibbs sampler **GibbsLDA++** by Xuan-Hieu Phan and Cam-Tu Nguyen.

Introductions and examples to **topicmodels** are available from:

- the package reference documentation which can be viewed via the R help system, or as a single PDF file (`topicmodels.pdf` at `http://cran.r-project.org/package=topicmodels`), or
- the authors' paper in the Journal of Statistical Software (Grün and Hornik, 2011).

For data input the **topicmodels** package depends on the package **tm** (text mining) which is described in the next subsection.

4.3.2. Preprocessing with the tm Package

tm is a framework for handling text mining tasks within R by Ingo Feinerer (Feinerer, 2008). The **tm** framework is designed as middleware between the R system environment and the application layer, which in the case of this paper is the R package **topicmodels**. It enables the user to import data from various formats via so-called "readers" and offers standard tools for preprocessing and further transformation and filtering of texts and text collections. Furthermore it handles the creation of document-term matrices, a feature which **topicmodels** makes use of.

A list of all **tm** readers is shown in Table 4.2. If available, metadata will be stored with the documents and the corpus.

Reader	Description
`readPlain()`	Read in files as plain text ignoring metadata
`readRCV1()`	Read in files in Reuters Corpus Volume 1 XML format
`readReut21578XML()`	Read in files in Reuters 21578 XML format
`readGmane()`	Read in Gmane RSS feeds
`readNewsgroup()`	Read in newsgroup postings in UCI KDD archive format
`readPDF()`	Read in PDF documents
`readDOC()`	Read in MS Word documents
`readHTML()`	Read in simply structured HTML documents

Table 4.2.: **tm** readers (Feinerer, 2008, Table 4.1).

After reading, corpora can be transformed and filtered (=searched) with the functions `tm_map()` and `tm_filter()`. Available text and corpus transformations include:

- **removal of punctuation, numbers, whitespace and stop words**,
- **conversion to lower case**, and
- **stemming** (see Section 2.2).

Linguistic models that are based on the bag-of-words assumption (like LDA), rely on document-term matrices (see Section 3.3 on page 14). In **tm** these matrices are constructed from the corpus as an instance of the class `DocumentTermMatrix` and the following options can be specified:

- different **tokenizers** (see Section 2.2),
- matching against a **dictionary**,
- **minimum term frequencies**,
- **minimum word lengths**, and
- most of the **standard transformations** (see above).

The current functionality of **tm** is documented in detail in the package's online help and in its reference manual (`tm.pdf` at `http://cran.r-project.org/package=tm`).

4.3.3. Model Selection by Harmonic Mean in topicmodels

As explained in Section 3.7.1, one way of selecting the model parameter K, which is the number of topics, is to approximate the marginal corpus likelihood (that depends on K) by taking the harmonic mean of a set of samples generated by the Gibbs sampler (Griffiths and Steyvers, 2004). Version 0.7 of **topicmodels** did not provide this method of model selection[1], and since I wanted to replicate the Griffiths and Steyvers (2004) paper on exploring the PNAS corpus via LDA, I had to implement this functionality by myself. This required a small addition to the **topicmodels** source code and some lines of R code.

When an LDA model is fitted by Gibbs sampling, the function `LDA()` returns the model as an object of class `LDA_Gibbs` which contains nearly all data of interest for further processing. However, the following R lines:

```
> library("topicmodels")
> getSlots("LDA_Gibbs")
```

and a cross check with the package's documentation reveal that one sample attribute needed for calculating the formula in Section 3.7.1, namely the number of instances of word i assigned to topic j, is missing from the returned object's slots. This variable `nw` is internally used by the **GibbsLDA++** Gibbs sampler as a substitute for the explicit topic to word assignments z, therefore it can be added to the C code that constructs the returned R object.

[1] The latest versions of **topicmodels** have already implemented this feature.

Changes to the topicmodels Source Code

The R interface is implemented by the package's source file `rGibbslda.cpp`. Using `diff` to compare the original file with my version shows the updated lines:

```
79,86d78
<
< // MP: nw: number of instances of word i assigned to topic j, size VxK
<   tp = PROTECT(allocMatrix(INTSXP, model->K, model->V));
<   for (i = 0; i < model->K; i++)
<     for (j = 0; j < model->V; j++)
<       INTEGER(tp)[i + model->K * j] = model->nw[j][i];
<   SET_SLOT(ans, install("nw"), tp);
<   UNPROTECT(1);
```

A second change was then made to `allclasses.R`, which contains all class declarations (again, as `diff` output):

```
131,132c131
<                           delta = "numeric",
<                           nw = "matrix"), # MP: additional slot
---
>                           delta = "numeric"),
```

These changes are only available from within R after compiling and installing this custom build of **topicmodels**. Instructions on how to install R packages (in user space) are given in the manual "R Installation and Administration" (R Development Core Team, 2010).

Functions in lda-gibbs-tools.R

The file `lda-gibbs-tools.R` is a collection of all further functions that implement the model selection as described in Section 3.7.1 on page 25.

The wrapper function `ldaGibbsSamples()` is needed for the repeated call of the LDA model estimation in order to produce a chain of model samples. Samples are taken after a burn-in interval and at a specified lag.

```
ldaGibbsSamples <- function(dtm, k, burniniter, sampleinterval, nsamples,
      control=NULL, model=NULL, ...) {
  # Value: returns a list of objects of class "LDA"
  # Note: The "call" slot of samples contains just the
  # wrapper's variable names, is therefore useless.

  # check control and remove iter parameter, if present:
  if (!is.null(control)) {
    stopifnot(is.list(control))
```

```
    control <- control[names(control) != "iter"]
  }

  chain <- list(LDA(dtm, k, method = "Gibbs", c(control,
    list(iter = burniniter), model, ...)))

  for (jj in 1:nsamples) {
    lastSampleIndex <- jj-1
    if(jj == 1) lastSampleIndex <- 1

    chain[[jj]] <- LDA(dtm, k, method="Gibbs", c(control,
        list(iter = sampleinterval)),
        model = chain[[lastSampleIndex]], ...)
  }
  return(chain)
}
```

Function `logPwzT()` is an implementation of Equation 2 from Griffiths and Steyvers (2004, p. 5229). It calculates the log-likelihood $p(w|z, K)$ of one model sample. The conversion to logarithms is necessary because calculation of `gamma(V)` and others would result in underflows caused by the small numbers.

```
logPwzT <- function(model) {
  BETA <- model@delta
  M <- model@Dim[1]  # number of documents
  V <- model@Dim[2]  # number of unique terms...W, or V
  K <- model@k       # number of topics (T)

  nwsum <- rowSums(model@nw)
  lPwz1 <- K * (lgamma(V * BETA)-V * lgamma(BETA))
  # second factor
  lPwz2 <- 0
  for(j in 1:K) {
    subsum <- 0
    for(w in 1:V) subsum <- subsum + lgamma(model@nw[j,w] + BETA)
    lPwz2 <- lPwz2 + (subsum - lgamma(nwsum[j] + V*BETA))
  }
  lPwz <- lPwz1 + lPwz2
}
```

The function `harmonicMeanPwT()` takes one chain of samples as argument to first collect all sample log-likelihoods and then to calculate the harmonic mean of these likelihoods, which is an approximation of $p(w|K)$, i.e., the likelihood of the corpus given the number of topics. This function requires the high-precision-arithmetic package **Rmpfr**

to handle the very small numbers. Also – thanks to a suggestion by Bettina Grün – a trick from the R package **bayesm** is employed to shift the sample values into a range of more precise numbers by adding and substracting the median at appropriate positions.

```
harmonicMeanPwT <- function(chain, precision=2000L) {
  library("Rmpfr")

  logLikelihoods <- sapply(chain, logPwzT)

  llMed <- median(logLikelihoods)
  as.double(llMed - log(mean(exp(-mpfr(logLikelihoods,
    prec = precision) + llMed))))
}
```

4.4. The lda Package

In this thesis I only focus on the **topicmodels** package, however another R package for topic modeling that is available from CRAN is **lda** by Jonathan Chang. The following description is a direct quote with minor adaptations from Chang (2011):

> "This package implements latent Dirichlet allocation (LDA) and related models. This includes (but is not limited to) sLDA [(supervised LDA, see Blei and McAuliffe (2007))], [related topic models (RTM, see Chang and Blei (2009))], and the mixed-membership stochastic blockmodel [(MMSB, see Airoldi et al. (2008))]. Inference for all of these models is implemented via a fast collapsed Gibbs sampler written in C. Utility functions for reading/writing data typically used in topic models, as well as tools for examining posterior distributions are also included."

5. Analyzing the 1991 to 2001 Corpus of PNAS Journal Abstracts

The *Proceedings of the National Academy of Sciences* (Pnas) is a multi-disciplinary, peer-reviewed scientific journal with a high impact factor (PNAS, 2010). First published in 1915, it is the official journal of the U.S. National Academy of Sciences. Its coverage includes scientific articles from physical, social and biological sciences, with the latter category accounting for the largest part of papers (Figure 5.2 on p. 46 shows the distribution of categories for the year 2001, similarly in Erosheva et al. 2004).

In May 2003 the National Academy of Sciences sponsored the colloquium "Mapping Knowledge Domains" to present and discuss modern information retrieval and text mining techniques. Participants of the colloquium were given access to a collection of Pnas full text documents ranging from 1997 to 2002, which the researchers could use as showcase data in their presentations (Shiffrin and Bo, 2004). Two papers on probabilistic topic models were published as part of the colloquium's proceedings in Pnas: "Mixed-membership models of scientific publications" (Erosheva et al., 2004) and "Finding scientific topics" (Griffiths and Steyvers, 2004) (also referred to hereinafter as GS04). The paper by Griffiths and Steyvers gives a thorough introduction to Latent Dirichlet Allocation and is the first to demonstrate the use of Gibbs sampling to fit these models. It also shows different applications of fitted models that were later refined by more complex model specifications like dynamic topic models.

In this chapter I aim at replicating the experiments in GS04. The reasons for this are manifold: Firstly, the results in their paper are significant for information retrieval as well as highly interpretable and interesting and therefore merit a second look and detailed notes on replicating these results. Secondly, the replication in R should correspond to the paper's findings and – as a side effect – test the relatively new package **topicmodels** on a large dataset. Thirdly, access to the online Pnas articles, abstracts and metadata is free and thus relatively easy to obtain via web-scraping (see below).

The following section deals with retrieving and preprocessing of the Pnas corpus. Later sections are dedicated to documenting the processes of model selection, examining the relationship between category designations and topics, as well as trends in topic frequency as previously reported by GS04. The last section shows how documents can be tagged and highlighted by using the model data.

5.1. Retrieving and Preprocessing the Corpus

The PNAS web page gives full and free access to all published content (including full text) from the first issue in 1915 to the most recent journal issues (available for free only six months after print publication). There is no direct access to the data in structured form such as CSV or XML which could easily be imported via the R package **tm**, therefore I had to prepare scripts for the automated downloading of all abstracts from 1991 to 2001. This process is called **web scraping** and there are a variety of frameworks for web scraping as stand-alone applications or in high-level programming languages like Perl or Python. For many projects, such as this one, a combination of standard Unix text processing tools and `wget` or `curl` would usually suffice, and my first attempt at writing a scraper was implemented in these standard Unix tools. Driven by the desire to have a more robust and maintainable code base, I later reimplemented the scraper in Python, which is discussed in this subsection and in Appendix D. It should also be mentioned that R has support for web scraping via the built-in function `readLines()` or packages like **RCurl** (Temple Lang, 2012).

5.1.1. Legal Clearance

Web scraping often touches on the subject of legality and morality of downloading copyrighted material. In this case this was not an issue because the "Proprietary Rights Notice" (`http://www.pnas.org/site/misc/terms.shtml`) allows the viewing and storing of journal copies for personal, noncommercial use. Even mass downloading of articles is easily possible because the usual protection against automated access by web crawlers, the file `robots.txt`, was not active. To be sure however, I contacted PNAS' Kat Rodenhizer (production staff) via email in April 2010 to state my intentions. Part of her answer was:

> "[...] Because you will not be directly copying any material from our papers, no additional permission is required from us, however if you decide later on to include any figures, tables, graphs, etc. from any of our papers into your thesis, please contact us back to secure permission. [...]"

5.1.2. Web Scraping

The complete process of making the 1991 to 2001 corpus of PNAS abstracts available as a **tm** `Corpus` object is documented in Appendix D.

I assume that the authors of "Finding scientific topics" had at least a partially web-scraped corpus of abstracts, because the official corpus provided by the National Academy of Sciences only covered the full texts of the years 1997 to 2001 (Shiffrin and Bo, 2004). There is no mention of this in their article, they only thank Kevin Boyack (member of the organizing committee) for providing the abstract categories (which were part of the official corpus in Microsoft Access 97 format).

5.1.3. Importing the Corpus and Generating a Document-Term Matrix in R

The result of the web scraping in Appendix D is a **tm** corpus that still needs to be converted into a document-term matrix, which is the standard input for fitting topic models that rely on the bag-of-words assumption (i.e., word order is not important). The conversion to a document-term matrix (DTM) is also handled by the **tm** package and is usually a simple call of the function `DocumentTermMatrix()` with the corpus as an argument and a list of parameters. These parameters determine the data available to the topic model and therefore influence how the model is fitted. Common steps of preprocessing have been listed in Chapter 4.3.2. For this analysis I followed the parameters of GS04, which can be mostly found on page 5231 of their article. Table 5.1 on page 41 summarizes these parameters. Values that are in parentheses were not specified explicitly in their paper and therefore were assumed or deduced by me. These cases are:

1. the **minimum word length** – which can be derived from one of the examples of topic terms which also include two-letter words;
2. **abstract headings** – it is unsure whether Griffiths and Steyvers included the abstract titles with the main document. Given the additional information they present for model learning, I decided to merge them with the abstracts. If headings are not part of a document, the document term matrix's vocabulary size decreases by 518 words and the total word occurrence is reduced by 246,107. The average document length would drop to 98.7 terms.

The discrepancies of vocabulary size and total word occurence between the original article and my re-analysis can be explained by one or more of the following reasons:

- Different approaches to extracting data from the PNAS online articles, resulted in a **lower number of documents** produced by my web-scraper. Griffiths and Steyvers possibly included (uncategorized) commentaries, corrections and/or retractions in their document collection, which I omitted.
- The **tokenizer** in the original paper used any white-space or other delimiting character (including hyphens) to separate words, whereas the **tm** tokenizer separates by white-space (space or tab characters) only and removes punctuation in a later step. In my replication this leads to potentially longer words by keeping hyphenated compound terms like "hydrogen-oxygen" as "hydrogenoxygen".
- As mentioned above, it is unclear whether Griffiths and Steyvers considered **abstract headings** as part of a whole document.
- The exact list of **stop words** is unknown.
- The **order of text preprocessing steps** could have slightly influenced the final outcome.

As a first step, we load the **tm** package and the corpus of abstracts.

```
> library("tm")
> load("pnas-abstracts-tm-corpus.RData")
> load("pnas-abstracts-meta-dataframe.RData")
```

The following lines show how the headings, which until now were saved as a separate metavariable, are merged with the abstracts.

```
> heading_prepend <- function(x) PlainTextDocument(
+    paste(Heading(x), x, sep = " "), origin = Origin(x),
+    id = ID(x), language = Language(x))
> abstracts_corpus <- tm_map(abstracts_corpus, heading_prepend)
```

The document term matrix is constructed by:

```
> dtm <- DocumentTermMatrix(abstracts_corpus,
+    control = list(
+        tolower = TRUE,
+        removePunctuation = TRUE,
+        ## (standard tm tokenizer, see termFreq)
+        removeNumbers= TRUE,
+        stemming = FALSE,
+        stopwords = TRUE,
+        minWordLength = 2))
```

Now we reduce the matrix to words which occur in at least five documents:

```
> dtm <- dtm[ , which(table(dtm$j) >= 5)]
```

and save the final document-term matrix:

```
> save(dtm, file = "abstracts-dtm.RData")
```

I would like to emphasize that the resulting corpus and the document-term matrix are likely to be very similar to the ones used by Griffiths and Steyvers. It will be shown in later sections that all analyses that are based on this data result in outcomes that are nearly identical to the original paper.

5.2. Model Selection

In Section "How many topics?" in Griffiths and Steyvers (2004) the problem of determining the number of topics that best match a corpus was approached by Bayesian model selection (introduced in Section 3.7).

Repeated model samples from the Gibbs sampler with different parameters were used to calculate marginal likelihoods of the corpus given the number of topics, $p(w|K)$.

	Griffiths and Steyvers (2004)	**Replication in this thesis**
Corpus source	provided by PNAS, as part of the Arthur M. Sackler colloquium *Mapping Knowledge Domains* (or unknown source)	downloaded from the PNAS webpage
Number of abstracts 1991-2001	28,154	27,292
Number of abstracts 2001	2,620	2,456
Paper title is considered part of abstract	(unknown)	yes
Letter case	(all upper case)	all lower case
Remove punctuation	(yes, part of tokenizer)	yes
Tokenization	"any delimiting character, including hyphens[...]"	default **tm** tokenizer
Remove numbers	yes, part of stop list	yes
Stemming	(no)	no
Stop words	"standard 'stoplist' used in computational linguistics, including numbers, individual characters, and some function words"	**tm**'s default `stopwords()` (488 words)
Minimum word length	(2)	2
Minimum number of documents where term appears	5	5
*Vocabulary size (*DTM*)*	20,551	19,480
*Total occurrence of words (*DTM*)*	3,026,970	3,128,218
*Average document length (*DTM*)*	107.51 terms	114.62 terms

Table 5.1.: Summary of PNAS document-term matrix and its construction from the corpus (abstracts from years 1991 to 2001).

Table 5.2 shows all combinations of parameters and sample chains. The Dirichlet hyperparameters were fixed at $\beta = 0.1$ (topics) and $\alpha = 50/K$ (documents). In their paper Griffiths and Steyvers summarize the outcome of their model selection in a figure that shows the calculated marginal log-likelihoods per number of topics. A maximum at 300 topics is clearly visible, which they describe as a typical outcome of "varying the dimensionality of a statistical model, with the optimal model being rich enough to fit the information available in the data, yet not so complex as to begin fitting noise" (GS04, p. 5231).

Number of topics (k)	Chains	Samples	Burn-in interval	Sample lag
50; 100; 200; 300; 400; 500; 600	8	10 per chain	1,100	100
1,000	6	2 per chain	800	100
Total	62	572	-	-

Table 5.2.: Overview of chains for model selection, as in GS04.

The basis for replicating aforementioned results is a modification to the **topicmodels** source and a collection of wrapper functions in the file `lda-gibbs-tools.R` (see Section 4.3.3). Calculating the model samples was delegated to the **Cluster@WU,** which is the high-performance computing infrastructure at the Vienna University of Economics and Business (Wu). An introduction to the Sun Grid Engine (Sge) and a short example job are given in Appendix B.

The scripts for carrying out the model selection are described below:

sge-modelselection-chains.R (listing: Appendix C.4 on page 68): This script makes use of the parallelizing capabilities of the Sge to generate six to eight sample chains per topic number K (see Table 5.2), each with a different seed to the random number generator. For each sub-job, one sample chain is generated by a call of `ldaGibbsSamples()` and later saved as a unique `.RData` file. Also saved are the parameters, which are identical for all jobs and therefore repeatedly overwritten. I submitted the script twice, each time with different random seed parameters that still lead to similar results. Completion of the job took approximately three to four days (running time for multiple runs of identical jobs varies because of changing cluster workloads).

modelselection-chain-sizes.R (optional) : The generated data took up a total of 15.4 GB as compressed R data files and 110 GB in memory, as reported by R's `object.size()` function. This script extracts these sizes and can be found in Appendix C.6.

sge-modelselection-likelihoods.R (listing: Appendix C.5): Reads each of the previously generated chains, calculates the estimated log-likelihood by applying `harmonicMeanPwT()` and saves the result as a `data.frame` object in a `.RData` file.

5.2.1. Result of Model Selection

The following R code loads the previously saved dataframe and plots a figure similar to Figure 3 in Griffiths and Steyvers (2004).

```
> load("modelselection-result.RData")
> library("lattice")
> print(xyplot(logLikelihood ~ topics, data = result,
+     type = c("p","a","g"), fun = mean,
+     ylab = "Log-likelihood", xlab = "Topics",
+     scales = list(x = list(relation="free", at =
+     c(50,100,200,300,400,500,600,1000)))))
```

Figure 5.1 on page 43 shows the estimated marginal log-likelihood of each sample chain, grouped by number of topics. Additionally there are lines connecting the average value of each group of chain likelihoods. As noted in Griffiths and Steyvers (2004), there is little deviation within each group, even though the sample chains were produced with different random number generator seeds.

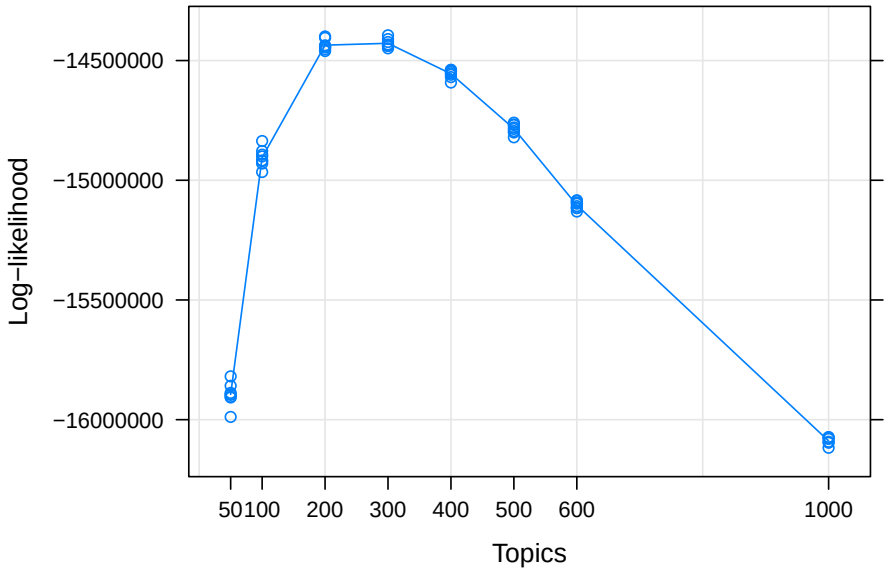


Figure 5.1.: Estimated marginal log-likelihoods per number of topics (circles), average likelihoods are connected by lines.

Given the fixed hyperparameters and a choice of number of topics K ranging from 50 to 1,000 topics, we see that a number of 300 topics accounts best for the corpus. The same result was reached by GS04, therefore I consider this part of the replication a success.

5.3. Model Fitting

The following sections deal with exploring the PNAS corpus via a fitted LDA model, as presented in Griffiths and Steyvers (2004, page 5232).

Common elements in these sections are:

- All code listings require the package **topicmodels** with its package dependencies (**tm** and others) to be loaded.
- An LDA model of the 1991 to 2001 PNAS corpus, returned after 2,000 iterations of Gibbs sampling, with $K = 300$ topics, and Dirichlet hyperparameters $\beta = 0.1$ (topics) and $\alpha = 50/K$ (documents). In **topicmodels** these are the default hyperparameters, therefore they need not be specified. The LDA model is fitted by running the following R code:

```
> library("topicmodels")
> load("abstracts-dtm.RData")
> system.time(model_lda <- LDA(dtm, 300, method = "Gibbs",
+     control = list(iter = 2000, seed = 33)))
> save(model_lda, file = "model-lda-gibbs-300topics.RData")
```

  Tables with the twelve most probable terms of all 300 topics are available in Appendix E.

- Figures were plotted with the **lattice** package. Source code for these figures was omitted in order to improve readability and can be found in Appendix C. An introduction to **lattice** is given in Sarkar (2008).

5.4. Relations between Topics and Categories (Year 2001)

As explained in Section 3.8.2, one possible use of a fitted model is to closely examine the relation between the estimated topics and another (meta)variable in the corpus. In GS04 the section "Scientific Topics and Classes" is dedicated to analysing the correspondence of a model's topics and PNAS article categories.

The PNAS journal requires authors to classify their submissions according to the schema laid out in the Information for Authors, which is published in the first printed issue of the year and online at `http://www.pnas.org/site/misc/iforc.shtml`. There are three major categories (Physical, Social and Biological Sciences) and around 30 to 40 minor categories available, depending on the submission year. In 2001 there were 33 minor categories (shown in Table 5.3), whereas the most recent (2010) number of possible designations has increased to 42 and includes additions such as Sustainability Science or substitutions (from Ecology to Environmental Sciences).

A minor category might also appear in two different major categories. For example, in 2001 the minor categories Anthropology and Psychology were available in both Social

and Biological Sciences. In GS04 and in this section such categories were treated as two distinct categories.

Major category	Minor categories
Physical Sciences	Applied Mathematics, Applied Physical Sciences, Astronomy, Chemistry, Engineering, Geology, Geophysics, Mathematics, Physics and Statistics
Social Sciences	Anthropology, Economic Sciences, Psychology and Social Sciences
Biological Sciences	Agricultural Sciences, Anthropology, Applied Biological Sciences, Biochemistry, Biophysics, Cell Biology, Developmental Biology, Ecology, Evolution, Genetics, Immunology, Medical Sciences, Microbiology, Neurobiology, Pharmacology, Physiology, Plant Biology, Population Biology and Psychology

Table 5.3.: PNAS major and minor article categories (2001).

Also mentioned in the Information for Authors is the possibility for submissions of "dual classifications [...] between major categories and in exceptional cases, subject to Editorial Board approval, within a major category". These additional metadata were taken into account by the web scraping scripts but were ignored in Griffiths and Steyvers (2004) and this thesis. This means that if two categories were specified on an issue web page, the first one was used.

The goal of this section is to visualize the differences and similarities between the categories and the topics estimated by the model.

5.4.1. Preparations and Showing the Category Distribution for 2001

First we load the corpus metadata which is in `data.frame` format:

```
> load("pnas-abstracts-meta-dataframe.RData")
```

In the next code chunk the categories of year 2001 are prepared for a figure showing their frequencies and also later analyses. The result is a data frame with 33 rows containing names of major, minor, abbreviated major categories and combined names in a shortened format for the figure to be printed. The rows are sorted first by major and then minor category.

```
> categories_2001_major <-
+     abstracts_meta$category[abstracts_meta$year==2001]
> categories_2001_minor <-
+     abstracts_meta$category_minor[abstracts_meta$year==2001]
> categories_2001 <- data.frame(major=categories_2001_major,
+     minor=categories_2001_minor, stringsAsFactors = FALSE)
> categories_2001$major_abbr <- factor(categories_2001_major)
> levels(categories_2001$major_abbr) <- c("(BS)","(PS)","(SS)")
> categories_2001$pretty <-
+     paste(categories_2001$minor, categories_2001$major_abbr)
```

For two figures in this section we also want to differentiate major categories by color, therefore we assign color codes:

```
> category_levels <- unique(categories_2001)
> category_levels <- category_levels[order(category_levels$major,
+     category_levels$minor),]
> rownames(category_levels) <- category_levels$pretty
> category_levels$color <- factor(category_levels$major)
> levels(category_levels$color) <- c("#00CC33","#0033CC", "#CC0099")
> category_levels$color <- paste(category_levels$color)
```

In order to get an impression of how the categories are distributed in the 2001 corpus, we use **lattice** to print a barplot of the class frequencies (Figure 5.2 on page 46, listing: Appendix C.7). One can clearly see the dominance of biological sciences in the diagram. As will be shown, our analyses are robust enough to cope with this skewed input.

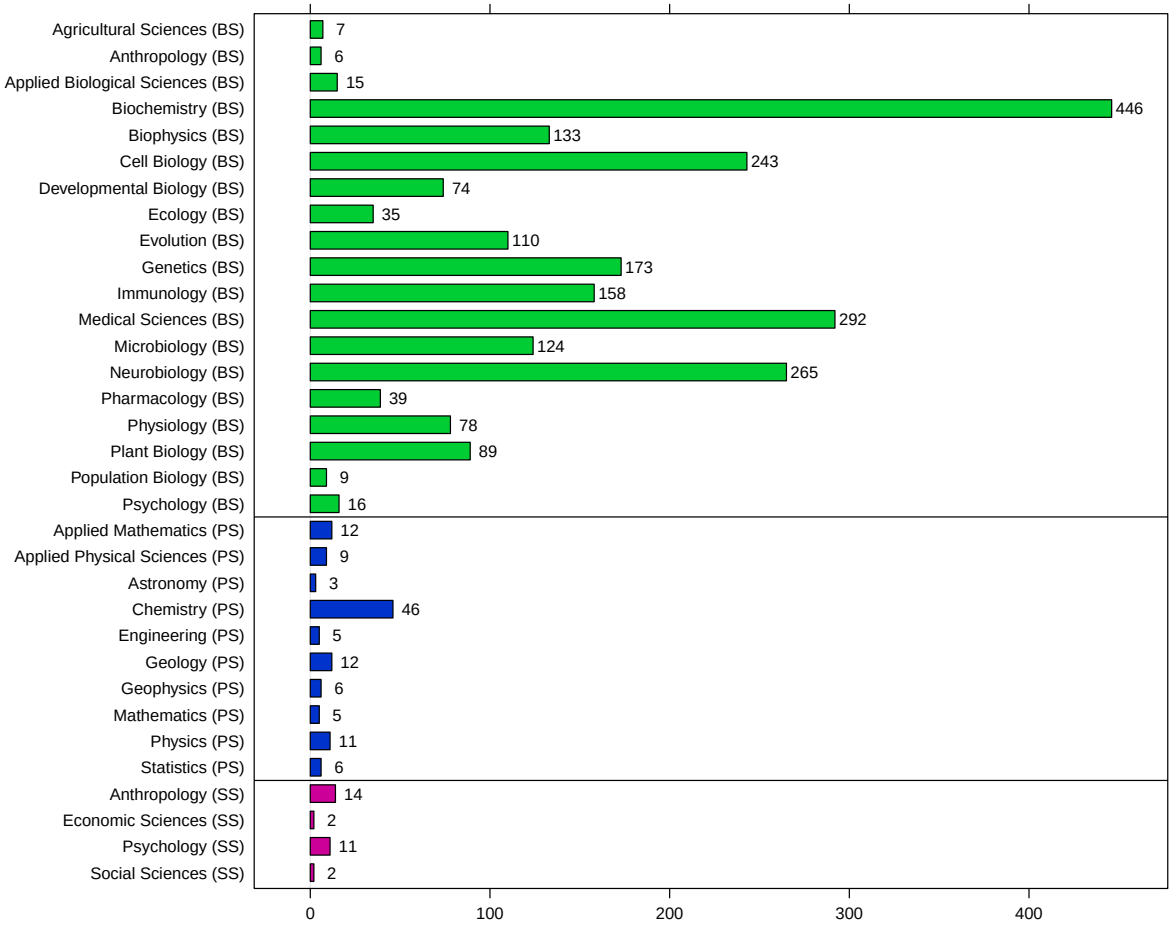


Figure 5.2.: Minor category frequency counts for 2001 PNAS abstracts.

5.4.2. Finding the Diagnostic Topics

In order to match topics and author-assigned categories, a series of steps is necessary. First we load the fitted LDA model (1991 to 2001):

```
> load("model-lda-gibbs-300topics.RData")
```

and then we extract θ (the per-document probabilities for topics) for the year 2001 from the fitted model's topic posteriors:

```
> theta <- posterior(model_lda)$topics[abstracts_meta$year == 2001,]
```

Afterwards we reduce θ, which is a 2,456 (abstracts) by 300 (topics) matrix, by taking the mean by categories which serve as an index to this "ragged array". This leaves us with the `matrix` `theta_mean`:

```
> theta_mean_by <- by(theta, categories_2001$pretty, colMeans)
> theta_mean <- do.call("rbind", theta_mean_by)
```

With all categories' mean θ available, it is now possible to select the most diagnostic (i.e., representative) topics of each of these categories. Griffiths and Steyvers defined a diagnostic topic of a category d_c as the topic j which has the highest ratio of θ to the sum of all other categories' θ. This criterion can be expressed as a formula (with C and J denoting the set of all categories and all topics, respectively; and r denoting categories different from c):

$$\text{Diagnostic topic of category } c\text{: } d_c = \arg\max_{j \in J} \frac{\theta_j^{(c)}}{\sum\limits_{\substack{r \in C \\ r \neq c}} \theta_j^{(r)}}$$

In R the ratios are first calculated in a nested `for` loop and then sorted by size in a separate index matrix. This allows for easy selection of one or more of the topics (and their corresponding θ values) with the highest and therefore most representative ratios.

```
> theta_mean_ratios <- theta_mean
> for (ii in 1:nrow(theta_mean)) {
+     for (jj in 1:ncol(theta_mean)) {
+         theta_mean_ratios[ii,jj] <-
+             theta_mean[ii,jj] / sum(theta_mean[ii,-jj])
+     }
+ }
> topics_by_ratio <- apply(theta_mean_ratios, 1,
+     function(x) sort(x, decreasing = TRUE, index.return = TRUE)$ix)
```

The most diagnostic topics per category are found in the first row of the index matrix:

```
> topics_most_diagnostic <- topics_by_ratio[1,]
```

5.4.3. Visualising Topics and Categories

We can now reduce the mean θ matrix to just the most diagnostic topics per category:

```
> theta_diagnostic <-
+     theta_mean_ratios[,topics_most_diagnostic]
> colnames(theta_diagnostic) <- topics_most_diagnostic
```

Figure 5.3 on page 48 shows the **lattice** `levelplot` of this reduced matrix. The source code for generating this plot can be found in Appendix C.8.

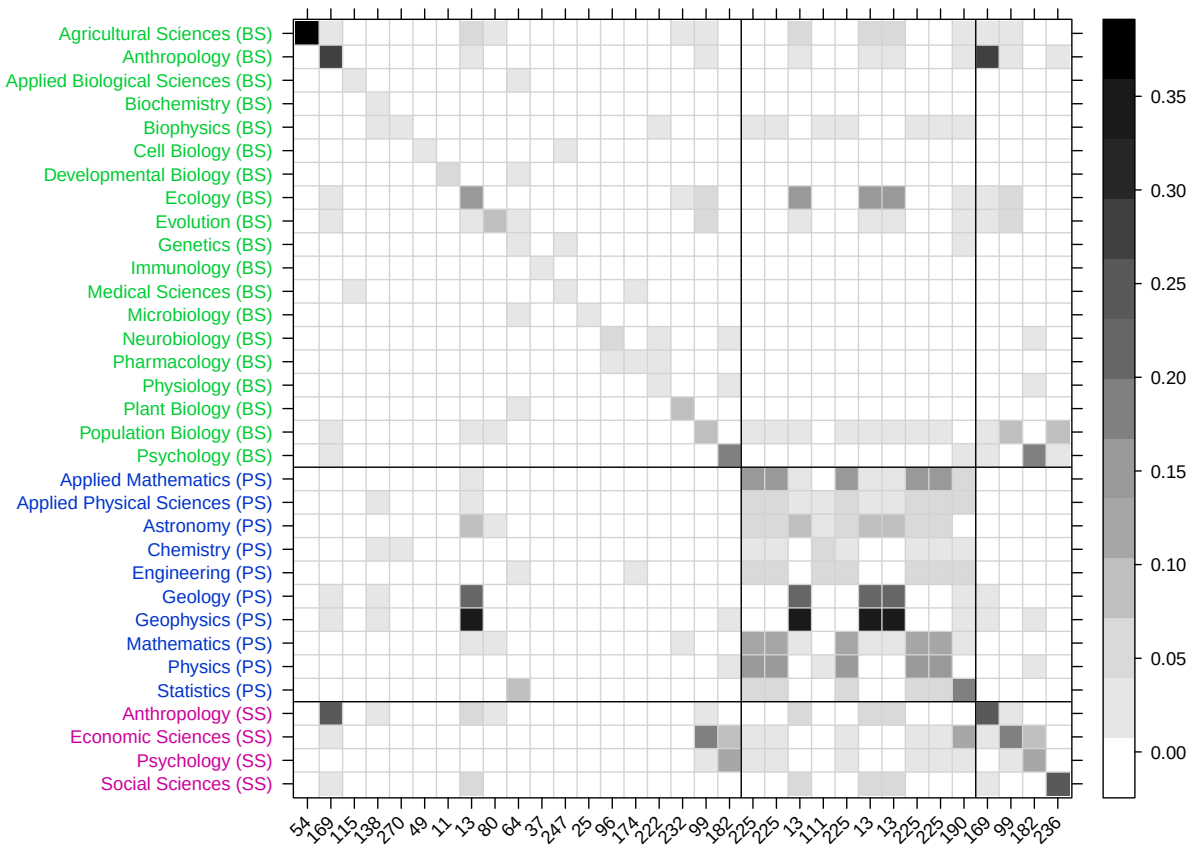


Figure 5.3.: Mean θ of 2001 categories by most diagnostic topics.

Complementing Figure 5.3, Table 5.4 on page 49 shows the five most probable terms per unique diagnostic topic, with rank 1 denoting the most probable term. The table was generated by the `xtable()` function from the package of the same name (Dahl, 2009). The function call is documented in Appendix C.10.

A different version of the previous levelplot is shown in Figure 5.4 on page 50. The columns and rows were reordered to show the most to least frequent categories in the

	Topic 54	Topic 169	Topic 115	Topic 225	Topic 13	Topic 138
1	fragment	monkeys	gene	theory	species	structure
2	fragments	humans	transfer	time	diversity	crystal
3	restriction	primates	vector	space	global	resolution
4	length	primate	vectors	random	marine	structural
5	endonuclease	world	expression	shape	ecological	xray

	Topic 270	Topic 49	Topic 111	Topic 11	Topic 99	Topic 80
1	folding	nuclear	water	development	size	evolution
2	protein	localization	energy	neural	selection	species
3	native	nucleus	free	expression	evolution	phylogenetic
4	unfolding	cytoplasm	hydrogen	embryos	fitness	evolutionary
5	transition	protein	transition	embryonic	social	sequences

	Topic 64	Topic 37	Topic 247	Topic 25	Topic 96	Topic 174
1	genes	cd	cancer	virulence	synaptic	endothelial
2	gene	cells	tumor	salmonella	glutamate	vascular
3	expression	cell	tumors	pseudomonas	neurons	cells
4	encoding	tcell	human	bacterial	hippocampal	hypoxia
5	identified	lymphocytes	breast	aeruginosa	postsynaptic	vegf

	Topic 222	Topic 232	Topic 182	Topic 236	Topic 190
1	channel	plants	cortex	age	data
2	channels	plant	brain	aging	method
3	current	arabidopsis	subjects	life	using
4	na	tobacco	functional	months	methods
5	currents	thaliana	regions	decline	set

Table 5.4.: Unique diagnostic topics of 2001 and their five most probable terms.

2001 corpus, with the most frequent at the top. This version is useful when the category frequencies are skewed (which they are in this case) and one wants to get an impression of the model's quality.

The source for Figure 5.4 can be found in Appendix C.9.

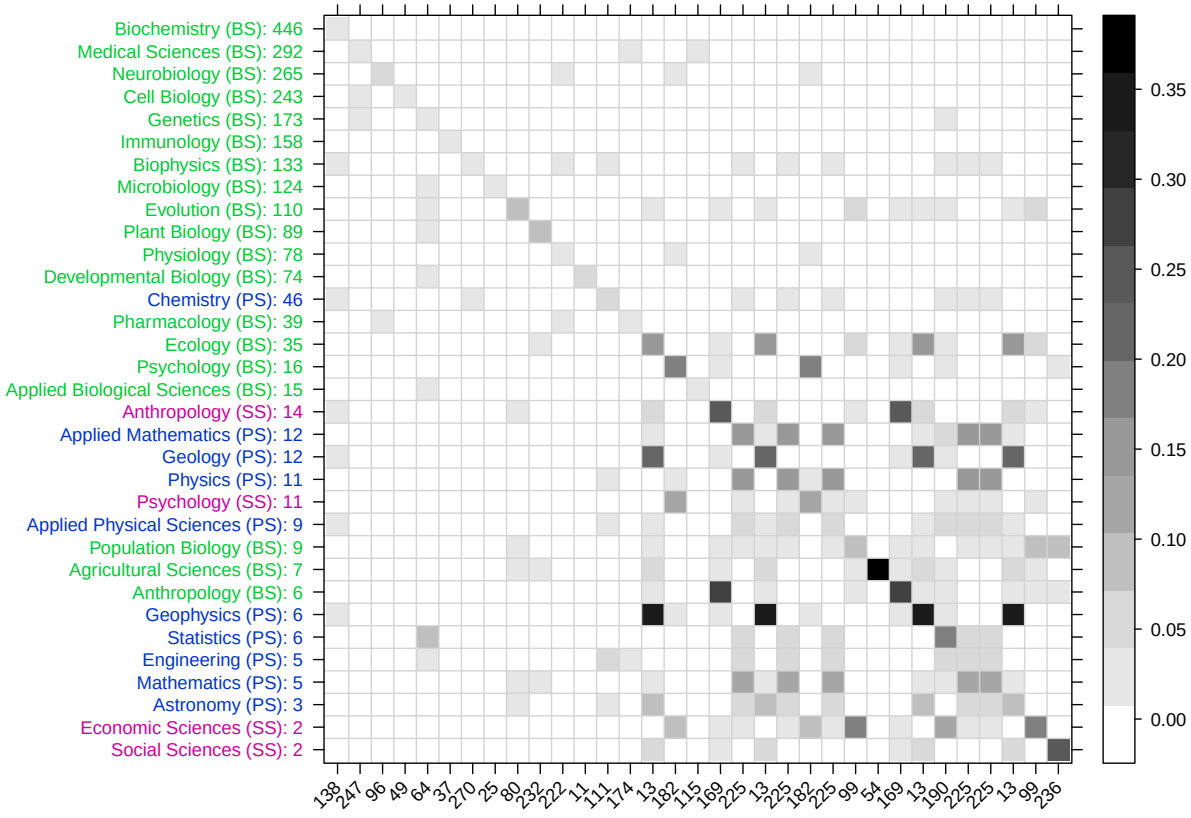


Figure 5.4.: Mean θ of 2001 categories by diagnostic topics, sorted by prevalence of categories. Frequencies are listed with category names.

A third option of visualizing the model's mean θ categories is given by Figure 5.5 on page 51. The main difference to the previous levelplots is that not only a single topic per category was chosen but the five most probable topics. Also, duplicate topics were removed going from left to right. This explains why the matrix diagonal is not a straight line anymore but concave. I have decided to include this third version of the levelplot because it gives an impression of how "unique" certain topics are. The source code for this figure is printed in Appendix C.11.

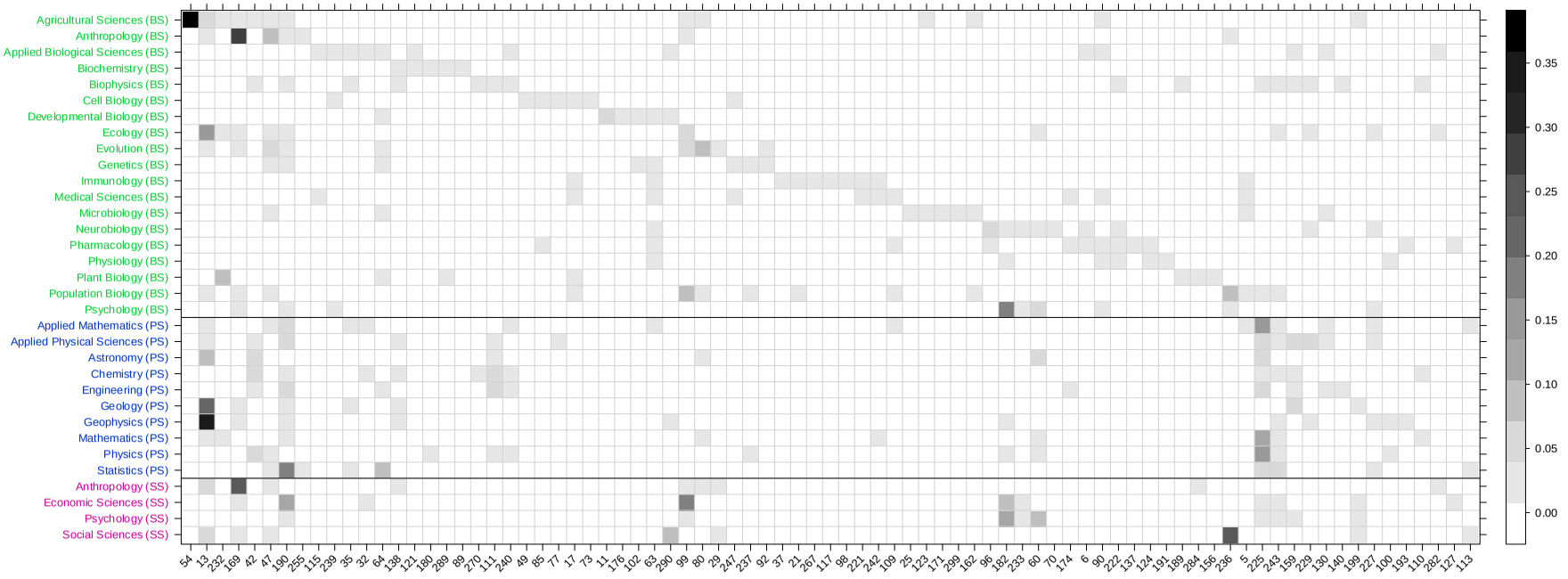


Figure 5.5.: Mean θ of categories with five most probable topics per category, duplicate topics removed (2001).

5.4.4. Comparison with Original Paper and Interpretation

Minor differences between GS04 and my results are different topic numbers and term distributions within the topics, caused by the randomization in the algorithm. Also of note is the (accidental?) omission of the category "Statistics" in GS04.

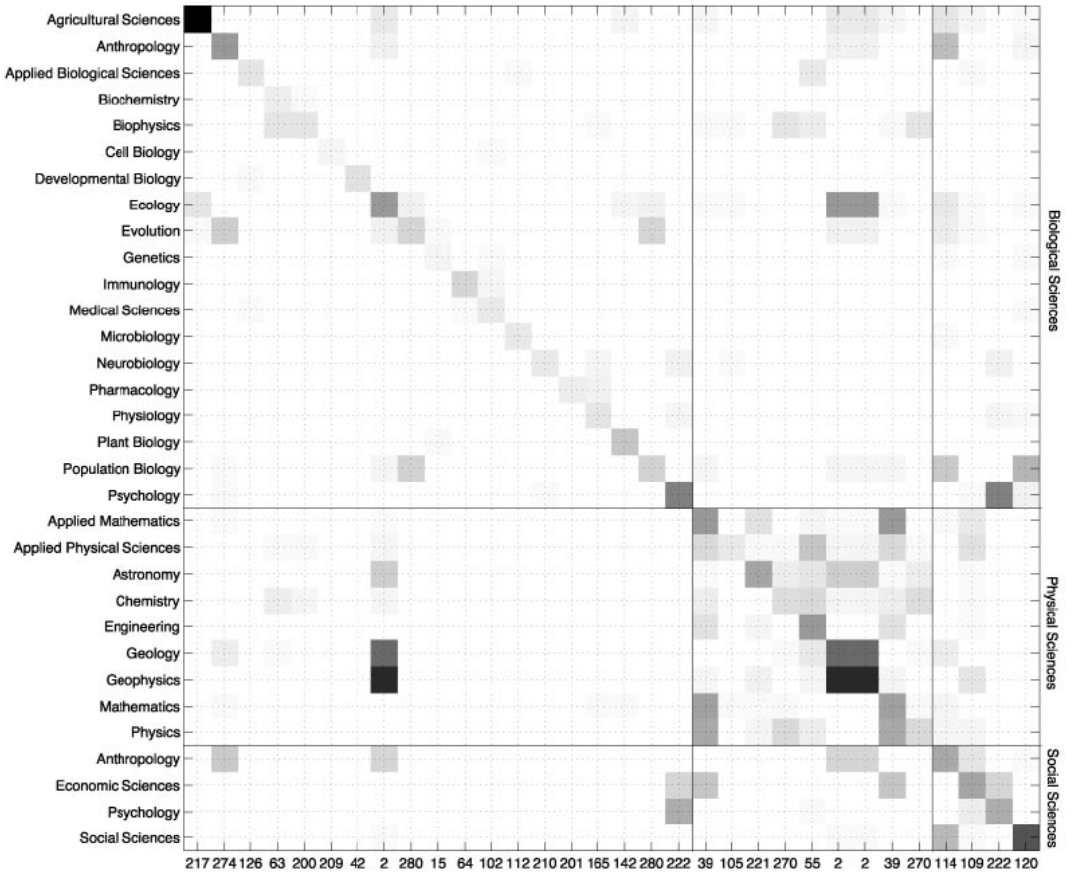


Figure 5.6.: Figure 4 (*Upper*) from GS04: "Mean values of θ at each of the diagnostic topics for all 33 PNAS minor categories, computed by using all abstracts published in 2001."

Figure 5.3 on page 48 closely resembles Figure 4 of the original paper (shown in Figure 5.6 on page 52), which is explained by the authors in the following way:

"The strong diagonal is a consequence of our selection procedure, with diagnostic topics having high probability within the classes for which they are diagnostic, but low probability in other classes. The off-diagonal elements illustrate the relationships between classes, with similar classes showing similar distributions across topics" (Griffiths and Steyvers, 2004, p. 5232).

For example, easily recognisable in both GS04's and my version is Topic 13 (Topic 2 in the original paper) as diagnostic for the categories "Geology", "Geophysics" and "Ecol-

ogy", where it seems to relate to global climate change. In my version it was also chosen by the model as diagnostic for "Astronomy" whereas in GS04 it is not the most diagnostic topic but still ranks among the most probable topics. Also, the diagnostic topic (Topic 236) for category "Social Sciences" with only two abstracts, still managed to capture some of the semantic differences. The second version of the levelplot in Figure 5.4 on page 5.4 helps in estimating how well the topic model works even when category frequencies are skewed.

	Topic 217	Topic 274	Topic 126	Topic 63	Topic 200	Topic 209
1	insect	species	gene	structure	folding	nuclear
2	myb	phylogenetic	vector	angstrom	native	nucleus
3	pheromone	evolution	vectors	crystal	protein	localization
4	lens	evolutionary	expression	residues	state	cytoplasm
5	larvae	sequences	transfer	structures	energy	export

	Topic 42	Topic 2	Topic 280	Topic 15	Topic 64	Topic 102
1	neural	species	species	chromosome	cells	tumor
2	development	global	selection	region	cell	cancer
3	dorsal	climate	evolution	chromosomes	antigen	tumors
4	embryos	co2	genetic	kb	lymphocytes	human
5	ventral	water	populations	map	cd4	cells

	Topic 112	Topic 210	Topic 201	Topic 165	Topic 142	Topic 222
1	host	synaptic	resistance	channel	plants	cortex
2	bacterial	neurons	resistant	channels	plant	brain
3	bacteria	postsynaptic	drug	voltage	arabidopsis	subjects
4	strains	hippocampal	drugs	current	tobacco	task
5	salmonella	synapses	sensitive	currents	leaves	areas

	Topic 39	Topic 105	Topic 221	Topic 270	Topic 55	Topic 114
1	theory	hair	large	time	force	population
2	time	mechanical	scale	spectroscopy	surface	populations
3	space	mb	density	nmr	molecules	genetic
4	given	sensory	observed	spectra	solution	diversity
5	problem	ear	observations	transfer	surfaces	isolates

	Topic 109	Topic 120
1	research	age
2	new	old
3	information	aging
4	understanding	life
5	paper	young

Table 5.5.: Figure 4 (*Lower*) from GS04: "The five most probable words in the topics themselves listed in the same order as on the horizontal axis in *Upper* [Figure 5.6]."

As to the term distribution within topics, a rough visual comparison of my Table 5.4

on page 49 to GS04's Table 5.5 on page 53 shows that many topics are similar in their composition, even though the corpus, the preprocessing and the learning steps were not the same.

I therefore consider the replication of this part of GS04's article a success and note that LDA is a robust tool that produces similar outcomes even in applications where starting conditions are not identical.

5.5. Hot and Cold Topics

Section "Hot and cold topics" in Griffiths and Steyvers (2004) describes how another metavariable, namely year of publication, can be used to explore the corpus. Instead of averaging over categories we now are interested in the mean θ by years.

We begin by loading the model sample and corpus and generate a `factor` of years from the `Origin` corpus metadata. This enables us to select abstracts in the model and the corpus by years.

```
> load("model-lda-gibbs-300topics.RData")
> load("pnas-abstracts-meta-dataframe.RData")
> years <- levels(factor(abstracts_meta$year))
> topics_n <- model_lda@k
> theta <- posterior(model_lda)$topics
```

The following table reveals that the number of documents per year is roughly equally distributed, which was different in last section, where we had skewed meta variables. The R code listing for this table is available in Appendix C.13.

	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001
Frequency	2382	2477	2364	2568	2382	2617	2516	2628	2498	2404	2456

Table 5.6.: Abstracts frequency per year.

Taking the mean of θ by years is accomplished by the `by()` function. Afterwards it is coerced into a `ts` object, which is the R standard container for time series (Cowpertwait and Metcalfe, 2009). A second variable – the corresponding time axis – is produced by `time()`.

```
> theta_mean_by_year_by <- by(theta, abstracts_meta$year, colMeans)
> theta_mean_by_year <- do.call("rbind",theta_mean_by_year_by)
> colnames(theta_mean_by_year) = paste(1:topics_n)
> theta_mean_by_year_ts <- ts(theta_mean_by_year,
+     start = as.integer(years[1]))
> theta_mean_by_year_time <- time(theta_mean_by_year)
```

Now we apply the fitting of a linear model to each of the 300 topics with the `lm()` function. The model coefficients, significance levels and trend slopes are each stored in separate variables.

```
> theta_mean_lm <- apply(theta_mean_by_year, 2,
+     function(x) lm(x ~ theta_mean_by_year_time))
> theta_mean_lm_coef <- lapply(theta_mean_lm,
+     function(x) coef(summary(x)))
> theta_mean_lm_coef_sign <- sapply(theta_mean_lm_coef,
+     '[',"theta_mean_by_year_time","Pr(>|t|)")
> theta_mean_lm_coef_slope <- sapply(theta_mean_lm_coef,
+     '[',"theta_mean_by_year_time","Estimate")
```

Then we divide the trend slopes into positive and negative slopes:

```
> theta_mean_lm_coef_slope_pos <-
+     theta_mean_lm_coef_slope[theta_mean_lm_coef_slope >= 0]
> theta_mean_lm_coef_slope_neg <-
+     theta_mean_lm_coef_slope[theta_mean_lm_coef_slope < 0]
```

At a p-level of 0.0001, GS04 reported 54 topics with a significant increasing trend and 50 with a significant decreasing trend. These results do not match the numbers in this replication, as will be shown. In order to generate a table that lists the number of topics with positive or negative trends at all significance levels, we have to prepare the data:

```
> p_level <- c(0.05, 0.01, 0.001, 0.0001)
> significance_total <- sapply(p_level,
+     function(x) (theta_mean_lm_coef_sign[theta_mean_lm_coef_sign < x]))
> significance_neg <- sapply(1:length(p_level),
+     function(x) intersect(names(theta_mean_lm_coef_slope_neg),
+     names(significance_total[[x]])))
> significance_pos <- sapply(1:length(p_level),
+     function(x) intersect(names(theta_mean_lm_coef_slope_pos),
+     names(significance_total[[x]])))
```

The following table summarizes the number of topics, split by positive and negative slope, and p-level:

	p-level=0.05	p-level=0.01	p-level=0.001	p-level=0.0001
Negative trend	90	66	46	29
Positive trend	99	76	46	21
Total	189	142	92	50

Table 5.7.: Topics with significant trends (listing: Appendix C.14).

I assume that the large differences between the amount of topics with a significance level of 0.0001 here and in the original paper can be attributed to either a typo (0.0001 instead of 0.001) or a different test statistic used in GS04.

A first impression of how the 300 topics change over time is shown in Figure 5.7 by 30 panels with ten trend lines each. The topics are sorted by slope, therefore "cold" topics appear in the first panels, "hot" topics in the last panels.

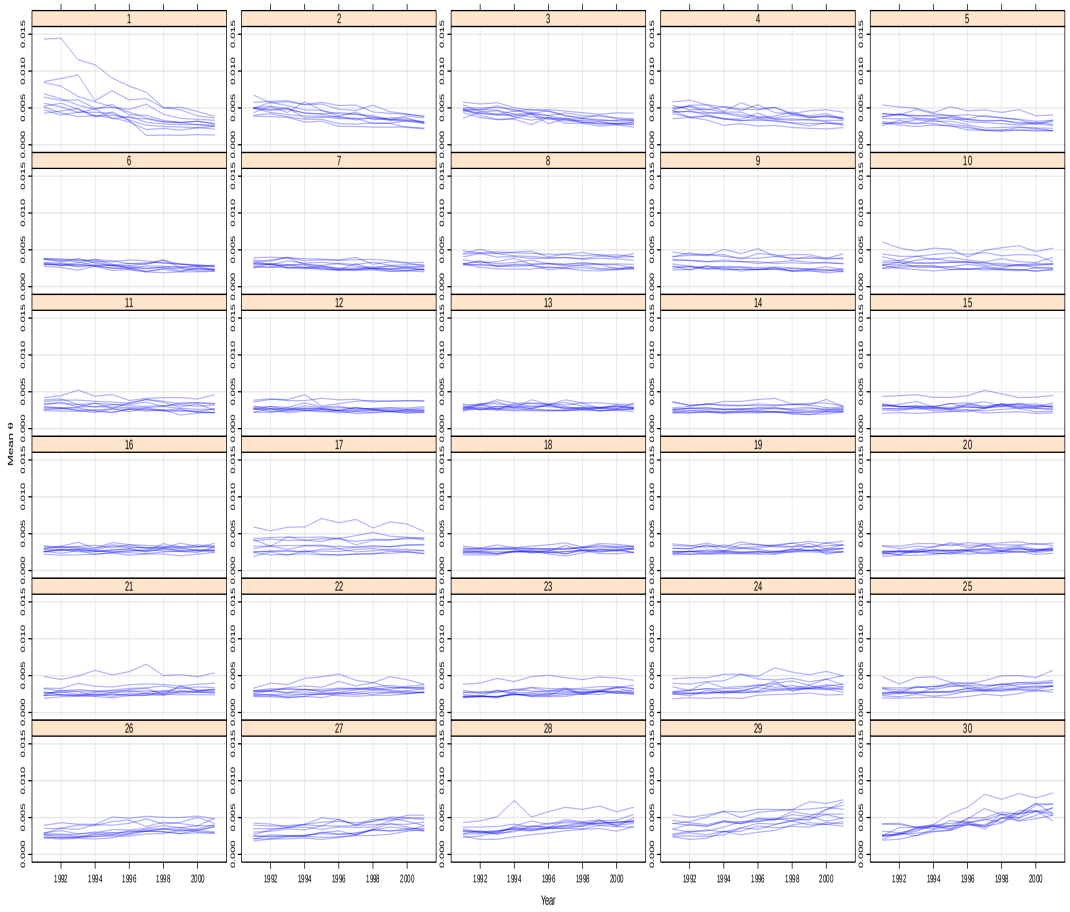


Figure 5.7.: All 300 topics sorted by slope of linear models, in panels of ten topics each. (Listing: Appendix C.15)

In the next step we will reproduce Figure 5 from Griffiths and Steyvers (2004). This involves plotting the five (three in the original paper) coldest and hottest topics in two panels and providing the most probable terms for these topics. To find the topics with the most extreme linear trends at a p-level of 0.0001 we can re-use data from the previous table.

```
> topics_hot <- as.numeric(names(sort(theta_mean_lm_coef_slope[
+     significance_pos[[4]]], decreasing=TRUE)))
> topics_cold <- as.numeric(names(sort(theta_mean_lm_coef_slope[
+     significance_neg[[4]]], decreasing=FALSE)))
```

Figure 5.8 on page 58 is printed by calling the R source code which can be found in Appendix C.16.

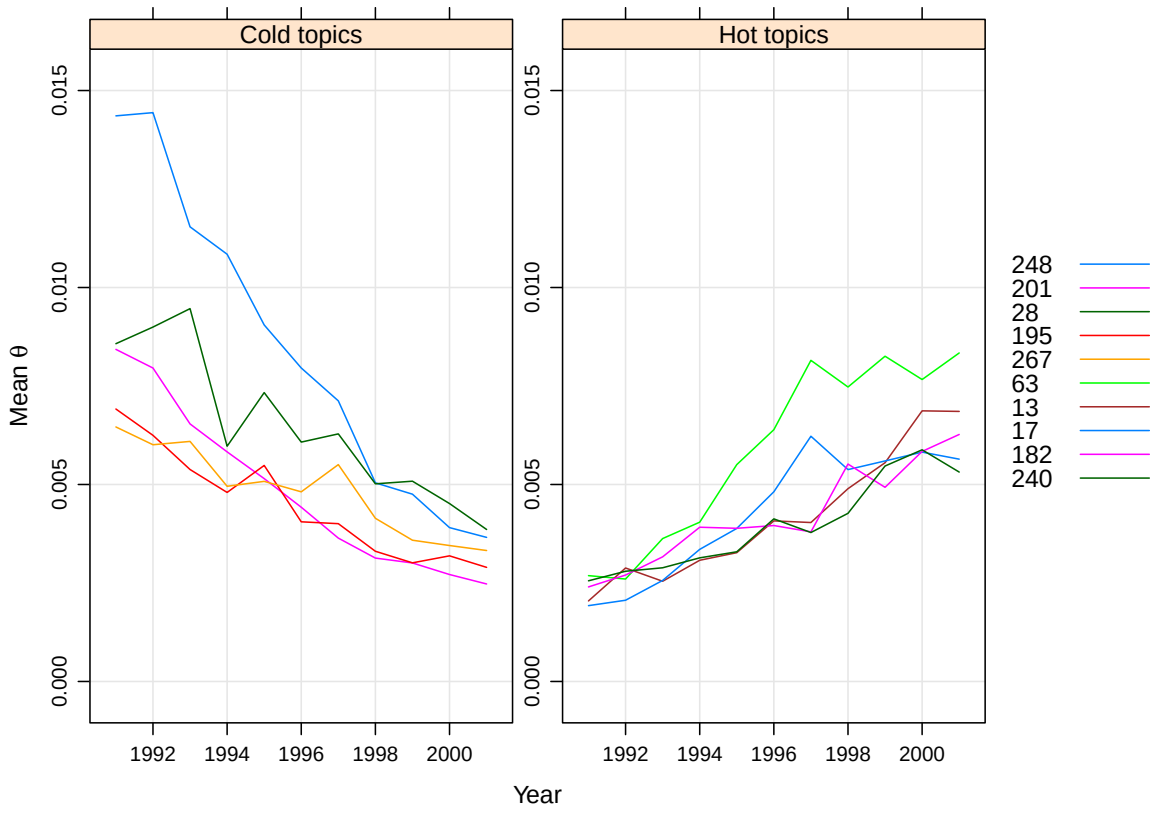


Figure 5.8.: Five coldest and hottest topics from 1991 to 2001, compare to Fig. 5 in Griffiths and Steyvers (2004).

The twelve most probable terms for our five cold and hot topics are collected in Table 5.8 on page 59 (source code: see Appendix C.17).

	Topic 248	Topic 201	Topic 28	Topic 195	Topic 267
1	cdna	kda	promoter	antibodies	class
2	sequence	protein	gene	antibody	major
3	amino	purified	transcription	monoclonal	mhc
4	encoding	molecular	element	antigen	molecules
5	cloning	polypeptide	region	mab	ii
6	protein	mass	elements	epitope	complex
7	isolated	purification	upstream	human	histocompatibility
8	acid	chromatography	enhancer	mabs	peptide
9	cloned	approximately	site	specific	antigen
10	clone	identified	expression	igg	peptides
11	library	antibodies	regulatory	epitopes	molecule
12	encodes	apparent	sequence	directed	presentation

	Topic 63	Topic 13	Topic 17	Topic 182	Topic 240
1	mice	species	apoptosis	cortex	fluorescent
2	normal	diversity	death	brain	fluorescence
3	gene	global	cell	subjects	microscopy
4	wildtype	marine	bcl	functional	green
5	development	ecological	apoptotic	regions	force
6	lacking	abundance	caspase	task	imaging
7	homozygous	community	induced	imaging	single
8	knockout	carbon	cells	flow	using
9	disruption	climate	survival	prefrontal	molecules
10	targeted	forest	fas	temporal	living
11	role	ecosystem	programmed	left	gfp
12	mouse	islands	bax	tomography	dye

Table 5.8.: Twelve most probable terms of five coldest (top) and five hottest topics (bottom).

5.5.1. Comparison with Original Paper and Interpretation

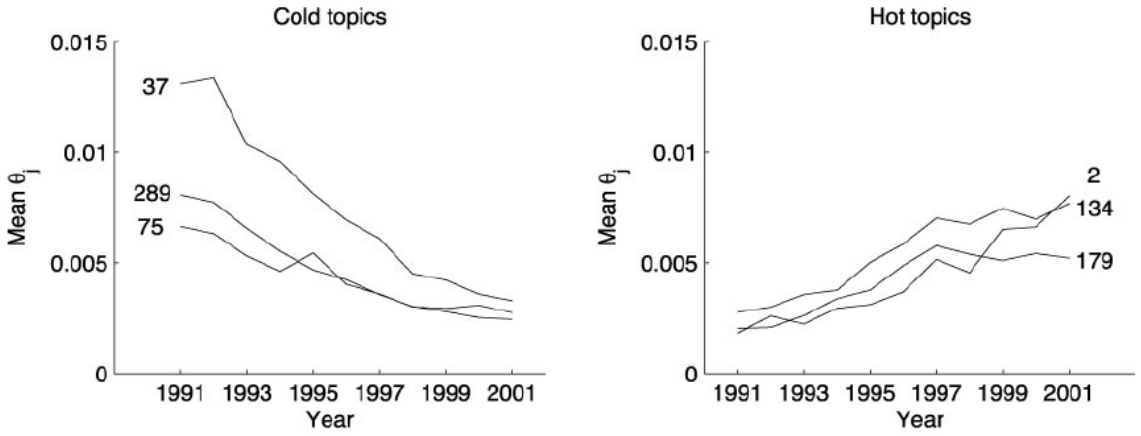


Figure 5.9.: Figure 5 (*Upper*) from GS04: "[. . .] the dynamics of the three hottest and three coldest topics from 1991 to 2001, defined as those topics that showed the strongest positive and negative linear trends."

Figure 5.9 (page 60) and Table 5.9 (page 61) show the results in GS04. A comparison with my results in Figure 5.8 (page 58) and Table 5.8 (page 59), respectively, indicates that the replication is very close to the original. The original three coldest topics can easily be recognised among the four coldest topics from the replication. Also, the three hottest topics can be matched, although with different ranks.

For the interpretation I provide an adapted full quote from Griffiths and Steyvers (2004, pp. 5233–5234) (my additions in brackets):

> "The three hottest and coldest topics, assessed by the size of the linear trend test statistic, are shown in Fig. 5 [here: Figure 5.9]. The hottest topics discovered through this analysis were topics 2 [here: 13], 134 [63], and 179 [17], corresponding to global warming and climate change, gene knockout techniques, and apoptosis (programmed cell death), the subject of the 2002 Nobel Prize in Physiology. The cold topics were not topics that lacked prevalence in the corpus but those that showed a strong decrease in popularity over time. The coldest topics were 37 [here: 248], 289 [201], and 75 [195], corresponding to sequencing and cloning, structural biology, and immunology. All these topics were very popular in about 1991 and fell in popularity over the period of analysis. The Nobel Prizes again provide a good means of validating these trends, with prizes being awarded for work on sequencing in 1993 and immunology in 1989."

This concludes the replication of topics and their trends over the period 1991 to 2001.

	Topic 37	Topic 289	Topic 75
1	cdna	kda	antibody
2	amino	protein	antibodies
3	sequence	purified	monoclonal
4	acid	molecular	antigen
5	protein	mass	igg
6	isolated	chromatography	mab
7	encoding	polypeptide	specific
8	cloned	gel	epitope
9	acids	sds	human
10	identity	band	mabs
11	clone	apparent	recognized
12	expressed	labeled	sera

	Topic 2	Topic 134	Topic 179
1	species	mice	apoptosis
2	global	deficient	death
3	climate	normal	cell
4	co2	gene	induced
5	water	null	bcl
6	environmental	mouse	cells
7	years	type	apoptotic
8	marine	homozygous	caspase
9	carbon	role	fas
10	diversity	knockout	survival
11	ocean	development	programmed
12	extinction	generated	mediated

Table 5.9.: Fig. 5 from GS04: "Twelve most probable terms [in the three coldest (top) and three hottest (bottom) topics from 1991 to 2001]".

Aside from a rather different number of topics with a trend at $p = 0.0001$, I can confirm the findings of Griffiths and Steyvers.

5.6. Document Tagging and Highlighting

Both Blei et al. (2003) and Griffiths and Steyvers (2004) give examples of how to tag and highlight a document with the help of a fitted topic model. The main reason for this is to give users novel ways to browse and read through corpora or search results.

Tagging Because our LDA model provides word-to-topic assignments, we can use these data to tag each word with topic superscripts. Words that were omitted during preprocessing will remain untagged.

Highlighting Given one reference topic per document (here: the most prevalent topic in an abstract), one can use the probability that a word belongs to this topic to set the word's color or contrast. This visualizes that topic's influence on the whole document.

We start by loading the model and our corpus of PNAS abstracts:

```
> load("model-lda-gibbs-300topics.RData")
> load("pnas-abstracts-tm-corpus.RData")
```

5.6.1. Finding the Example Abstract from Griffiths/Steyvers (2004)

In order to find out which abstract was used in GS04, we have to search for words that are part of this abstract, for this thesis I chose "generalized" and "fundamental" which come as second and third words in the example given by Griffiths and Steyvers (2004, Fig. 6, p. 5234). A simple lookup by abstract tokens is performed by the following R code:

```
> corpus_tokens <- sapply(abstracts_corpus, function(x) strsplit(x,
+     " "))
> for (dd in 1:length(corpus_tokens)) {
+     if (isTRUE(all.equal(corpus_tokens[[dd]][2:3], c("generalized",
+         "fundamental")))) {
+         print(dd)
+         example_abstract <- dd
+     }
+ }
[1] 24044
```

At these positions the words appear in only one abstract, which is number 24044, with the heading "Fundamental theorem of natural selection under gene-culture transmission".

5.6.2. Tagging and Highlighting of an Abstract

We can now call function `document_tag_latex()` (see Appendix C.19) to highlight the example abstract by the most prevalent topic and tag words with assigned topics as superscripts.

A generalized[225] fundamental[225] theorem[225] of natural[99] selection[99] is derived[237] for populations[47] incorporating[225] both genetic[47] and cultural[99] transmission[47]. The phenotype[237] is determined[65] by an arbitrary[225] number of multiallelic[237] loci[237] with two-factor epistasis[237] and an arbitrary[225] linkage[237] map[237], as well as by cultural[99] transmission[47] from the parents[99]. Generations[99] are discrete[225] but partially[187] overlapping[99], and mating[29] may be nonrandom[47] at either the genotypic[47] or the phenotypic[99] level[47] (or both). I show that cultural[99] transmission[47] has several important implications[99] for the evolution[99] of population[47] fitness[99], most notably[243] that there is a time[199] lag[199] in the response[131] to selection[99] such that the future[99] evolution[99] depends[225] on the past[225] selection[99] history[99] of the population[47].

To put this abstract into context, we provide a table with its five most probable topics and their twelve most probable terms:

	Topic 99	Topic 47	Topic 225	Topic 237	Topic 29
Probability	0.16349	0.08730	0.06825	0.05873	0.03016
1	size	populations	theory	locus	male
2	selection	population	time	genetic	female
3	evolution	genetic	space	loci	males
4	fitness	variation	random	linkage	females
5	social	alleles	shape	markers	sexual
6	species	diversity	local	susceptibility	sex
7	behavior	polymorphism	size	analysis	mating
8	competition	loci	linear	linked	pheromone
9	evolutionary	selection	systems	chromosome	germ
10	traits	polymorphisms	distribution	alleles	offspring
11	reproductive	individuals	probability	genes	reproductive
12	population	natural	terms	mapping	behavior

Table 5.10.: Abstract 24044: Five most probable topics and their twelve most probable terms (listing: Appendix C.20).

5.6.3. Comparison with Original Paper

The generated output closely matches the one printed in Griffiths and Steyvers (2004).

6. Conclusion and Future Work

I have given an introduction to the probabilistic topic model LDA and its foundations in computer science. The main goal of this thesis was to give a complete account of how text mining tasks can be solved within the R environment by using the package **topicmodels**. To this end I have successfully reproduced the findings of one of the first LDA papers (Griffiths and Steyvers, 2004) and thus confirmed 1.) the suitability of the used tools and 2.) the findings of the template paper. Initial concerns that my results would differ greatly from that paper have proven unfounded. Instead I have found that the information provided by the authors was sufficient for replicating all steps.

I note however, that, mirroring the recent trend for releasing primary (i.e, raw) data in addition to publishing studies in scientific journals (Alsheikh-Ali et al., 2011), it would generally be advisable to introduce similar requirements for open access to code. This point was also argued in Ince et al. (2012), where examples are given of journals (*Geoscientific Model Development*, *Biostatistics*) that already require code publication and reproducibility.

6.1. Future Work

In the process of completing the main goals of this thesis I have identified several questions which possibly merit further research and experiments.

- The PNAS journal corpus that was mined in this thesis consisted of (short) abstracts. How would the results change if the full article texts were used?
- What consequences would variations in the model hyperparameters α and β have?
- In this thesis as well as in Griffiths and Steyvers (2004), Gibbs sampling was used for estimating the topic model. How would variational inference methods affect performance and results? More specifically, what would be the differences between variational methods with fixed versus trained α hyperparameter? Would model performance benefit from using a trained α from variational inference as a fixed hyperparameter in Gibbs sampling?
- In my literature analysis I have not found any information on the importance of corpus preprocessing in topic modeling. I suggest a more detailed examination of this aspect.

A. Appendix: Software Used

The following software was used for this thesis:

Operating system Debian 6.0 Squeeze 64bit, Ubuntu 10.04 Lucid Lynx 64bit

Data analysis R 2.10.0./2.10.1

R packages topicmodels 0.0-3/0.0-5, tm 0.5-1, xtable 1.7-0, lattice 0.18-3

Document preparation Texlive 2009-07, R's Sweave, Kile 2.8.85, Vim 7.2

Figures OpenOffice Draw 3.2.0, Inkscape 0.47

Bibliography management Mendeley 0.9.8.2–1.5.1

B. Appendix: Model Fitting on the Sun Grid Engine

Model fitting is a CPU-intensive process which I partly delegated to the **Cluster@WU**, the high-performance computing infrastructure at the Vienna University of Economics and Business. The cluster's job queuing system is the Sun Grid Engine (SGE, now available as Oracle Grid Engine), the job submission is documented below.

B.1. Submitting Sun Grid Engine Jobs

Jobs to the cluster are submitted as batch files that are processed in an isolated runtime environment whose complete output is saved to the login server account's working directory. From there it can be copied to the researcher's main workstation. One of the SGE's capabilities is to pseudo-parallelize jobs by spawning multiple copies of a job with different job identification numbers that serve as an index to script parameters.

SGE jobs are like regular `bash` scripts (minus the hashbang `#!`) with parameters (denoted by a leading `#$`) that determine the job name (`-N`) and, optionally, the spawning of multiple job versions of the script (`-t`) as well as email notifications (`-M`, `-m`). A short example is given below:

```
#$ -N pnas-modelselection-chain
#$ -t 1:62
## email on end and abort
#$ -M m.ponweiser@gmail.com
#$ -m ea
R-g --vanilla <<-EOF
library("topicmodels")
## more R code ...
EOF
```

In this example, R is invoked as a call of `R-g`, which is a local version built with the GNU toolchain. The `--vanilla` parameter is a shortcut for other parameters that cause R to start a clean session (Venables et al., 2011). Also found on the same line is a shell input redirection that feeds all following lines (the main R script) to the program (`<<`), removing all leading tab characters (`-`) until the next appearance of the flag word (`EOF`).

The example can be submitted by the shell command `qsub example.R`.

C. Appendix: Source Code Listings

C.1. beta-distribution.R

```
curve( dbeta(x,0.5,0.5), ylab="Density", xlim=c(0,1), ylim=c(0,5) )
curve( dbeta(x,1,1), add=TRUE, col='red' )
curve( dbeta(x,3,1), add=TRUE, col='green' )
curve( dbeta(x,1,5), add=TRUE, col='blue' )
curve( dbeta(x,2,2), add=TRUE, lty=2, lwd=1, col=par('fg') )
legend(par('usr')[2]/2, par('usr')[4], xjust=0.5,
        c('a=b=0.5', 'a=b=1', 'a=3, b=1', 'a=1, b=5', 'a=b=2' ),
        lty=c(1,1,1,1,2),
        col=c(par('fg'), 'red', 'green', 'blue', par('fg')))
```

C.2. dirichlet-3d.R

```
library("MCMCpack")
DiriPlotData <- function(gridsize=100, alpha) {
    # needs grid interval of at least 1/100, otherwise there will be
       nasty outliers on the Z side that corrupt the plot
    gridaxis <- seq(0, 1, length= gridsize)
    dens <- matrix(nr=gridsize,nc=gridsize)
    for(x in 1:gridsize) {
          for (y in 1:gridsize) {
              dens[x,y] <- NA # not really necessary, is initialized as
                  NA anyway
              # although there is a check in ddirichlet for the
                 gridaxis to be on the simplex, we have to put in a
                 check to avoid plotting of the saltus to the ground
                 plane (which does not look good)
              if ((gridaxis[x] + gridaxis[y]) < 1) dens[x,y] <-
                 ddirichlet(c(gridaxis[x],gridaxis[y],(1-gridaxis[x]-
                 gridaxis[y])) , alpha ) }
          }
    list(gridline=gridaxis, values=dens)
}

DiriPersp <- function(alph=c(2,3,2), ...) {
    # oldpar <- par(bg = "white")
    diridata <- DiriPlotData(gridsize=60,alpha=alph)
    persp(diridata$gridline, diridata$gridline, diridata$values, zlim=c
       (-3,3), theta = 310, expand = 0.5, col = "lightblue", ticktype="
       detailed", xlab="X", ylab="Y", zlab="density", ...)
    title(main = bquote(list(alpha[1]==.(alph[1]),
```

```
                              alpha[2]==.(alph[2]),
                              alpha[3]==.(alph[3]))))
}

par(mfrow=c(3,2))
DiriPersp(c(4,4,2),phi = 50)
DiriPersp(c(2,4,4),phi = 50)
DiriPersp(c(2,4,2),phi = 40)
DiriPersp(c(2,2,2),phi = 40)
DiriPersp(c(1,1,1),phi = 30)
DiriPersp(c(0.5,.5,.5),phi = 30)
```

C.3. dirichlet-samples.R

```
## http://stackoverflow.com/questions/3712402/r-how-to-change-lattice-levelplot-color-theme
lattice.options(default.theme = standard.theme(color = FALSE))
rd<-vector()
alpha = c(.01, .1, 1, 10, 100)
for (alp in alpha) rd <- c(rd, as.vector(t(rdirichlet(5,rep(alp,5)))))
rddf <- expand.grid(k=as.character(1:5), smpl=paste("Sample",
                 1:5), alp=paste("alpha =",alpha))
rddf$value <- rd
print(xyplot(value~ k | smpl*alp, data=rddf, type=c("p","h") , between=list(y=1), xlab="Event", ylab="Probability"))
#print(dotplot(k~ value | alp*smpl, data=rddf, type=c("p","h") , between=list(x=1), xlab="Probability", ylab="Event"))
```

C.4. sge-modelselection-chains.R

```
#$ -N pnas-modelselection-chain
#$ -t 1:62

# email on end and abort
#$ -M m.ponweiser@gmail.com
#$ -m ea
R-g --vanilla <<-EOF
library("topicmodels")

load("pnasDtm.Rda")
source("lda-gibbs-tools.R")

chains <- c(8, 8, 8, 8, 8, 8, 8, 6)
parameters <- list(
    chains = chains,
    seeds = 1:sum(chains),
    topics = rep(c(50, 100, 200, 300, 400, 500, 600, 1000),chains),
    topicsChainId = unlist(sapply(chains, function(x) seq(1,x))),
    samples = rep(c(10, 10, 10, 10, 10, 10, 10, 2), chains),
    burnIn = rep(c(1100, 1100, 1100, 1100, 1100, 1100, 1100, 800),
        chains),
```

```
21      sampleInterval = rep(c(100, 100, 100, 100, 100, 100, 100, 100),
            chains))
22
23  jobid <- as.integer(Sys.getenv("SGE_TASK_ID"))
24
25  chain <- ldaGibbsSamples(
26      pnasDtm,
27      k = parameters\$topics[jobid],
28      burniniter = parameters\$burnIn[jobid],
29      sampleinterval = parameters\$sampleInterval[jobid],
30      nsamples = parameters\$samples[jobid],
31      control = list(alpha = 50 / parameters\$topics[jobid],
32                     seed = parameters\$seeds[jobid] )
33      )
34
35  # concatenate filename, save file:
36  FILE <- paste("modelselection-chain-", jobid, "-", parameters\$topics[
        jobid],
37      ".Rda", sep = "")
38  save(chain, file = FILE)
39
40  save(parameters, file = "modelselection-parameters.Rda")
41
42  EOF
```

C.5. sge-modelselection-likelihoods.R

```
1   #$ -N pnas-modelselection-logl
2
3   # email on end and abort
4   #$ -M m.ponweiser@gmail.com
5   #$ -m ea
6   R-g --vanilla <<-EOF
7   library("topicmodels")
8
9   source("lda-gibbs-tools.R")
10
11  load("modelselection-parameters.Rda")
12
13  result <- data.frame(
14    topics = parameters\$topics,
15    ChainId = parameters\$topicsChainId,
16    logLikelihood = as.numeric(NA))
17
18  for(jj in 1:sum(parameters\$chains)) {
19    FILE <- paste("modelselection-chain-", jj, "-", parameters\$topics[[
          jj]],
20        ".Rda", sep = "")
21    print(FILE)
22    load(FILE)
23    result\$logLikelihood[jj] <- harmonicMeanPwT(chain, precision=2000)
24  }
```

```
25
26 save(result, file = "modelselection-result.Rda")
27
28 EOF
```

C.6. modelselection-chain-sizes.R

```
1  library("topicmodels")
2
3  load("modelselection-parameters.Rda")
4
5  sizes <- data.frame(
6    objectsize = as.numeric(NA)[1:length(parameters$samples)],
7    filesize = as.numeric(NA))
8
9  for(jj in 1:sum(parameters$chains)) {
10   FILE <- paste("modelselection-chain-", jj, "-", parameters$topics[[jj
       ]], ".Rda", sep = "")
11   load(FILE)
12   print(FILE)
13   sizes$objectsize[jj] <- object.size(chain)
14         sizes$filesize[jj] <- file.info(FILE)[["size"]]
15 }
16
17 save(sizes, file = "modelselection-chain-sizes.Rda")
```

C.7. classifications-fig-category-frequency.R

```
1  library(lattice)
2
3  theme.nopadding <- list(
4      layout.heights =
5      list(top.padding = 0,
6          main.key.padding = 0,
7          key.axis.padding = 0,
8          axis.xlab.padding = 0,
9          #xlab.key.padding = 0,
10         key.sub.padding = 0,
11         bottom.padding = 0),
12     layout.widths =
13     list(left.padding = 0,
14         key.ylab.padding = 0,
15         ylab.axis.padding = 0,
16         axis.key.padding = 0,
17         right.padding = 0))
18
19 categories_table <- table(categories_2001$pretty)
20 categories_table_barchart <-
21     rev(categories_table[category_levels$pretty])
22
```

```
23 #http://stackoverflow.com/questions/2147084/r-add-labels-to-lattice-
       barchart
24
25 print(
26     barchart(categories_table_barchart,
27     xlab = NULL,
28     par.settings = theme.nopadding,
29     panel = function(...) {
30         args <- list(...);
31         panel.barchart(..., col = rev(category_levels$color));
32         panel.abline(h = 4.5)
33         panel.abline(h = 14.5)
34         panel.text(args$x+10, args$y, args$x,fontsize = 10)
35     })
36     )
```

C.8. classifications-fig-levelplot-most-diagnostic.R

```
1  # we reorder both axes by major category
2  order_by_major_index <- 1:33
3  names(order_by_major_index) <- rownames(theta_diagnostic)
4  order_by_major_index <- order_by_major_index[category_levels$pretty]
5  theta_diagnostic_ordered_by_major <-
6      theta_diagnostic[order_by_major_index,order_by_major_index]
7
8  # transpose the matrix and reverse rows for orientation like in GS04
9  theta_diagnostic_plot <-
10     t(theta_diagnostic_ordered_by_major[nrow(theta_diagnostic_ordered_
           by_major):1,])
11
12 # problem: there is a bias for theta, meaning no values are actually
       zero
13 # drawing must therefore make sure to assign lower values the color "
       white".
14 # solution: cuts = 12
15 colors_plot <- rev(category_levels$color)
16
17 print(levelplot(theta_diagnostic_plot,
18   xlab = NULL,
19   ylab = NULL,
20   scales = list(
21     tck = 1,
22     x = list(rot = 45),
23     y = list(col = colors_plot)),
24   cuts = 12,
25   col.regions = gray(20:0/20),
26   colorkey = list(
27     space = "right",
28     tick.number = 10),
29   par.settings = theme.nopadding,
```

```
30    panel = function(...) {
31      panel.levelplot(...);
32      # the grid:
33      panel.abline(h = seq(1.5, 32.5), col = "lightgray");
34      panel.abline(v = seq(1.5, 32.5), col = "lightgray");
35      panel.abline(h = c(4.5, 14.5));
36      panel.abline(v = c(33-3.5, 33-13.5));
37      }
38    ))
```

C.9. classifications-fig-levelplot-most-diagnostic-by-prevalence.R

```
1  categories_prevalence <- table(categories_2001$pretty)
2
3  # make a copy our main theta matrix with added counts in rownames
4  theta_diagnostic_with_prevalence <- theta_diagnostic
5  rownames(theta_diagnostic_with_prevalence) <- paste(
6      rownames(theta_diagnostic),
7      categories_prevalence, sep=": ")
8
9  # prepare index to reorder theta matrix
10 categories_ordered_by_prevalence <-
11     names(sort(categories_prevalence, decreasing=TRUE))
12
13 theta_index <- 1:33
14 names(theta_index) <- rownames(theta_diagnostic)
15 theta_index <- theta_index[categories_ordered_by_prevalence]
16
17 # reorder theta matrix
18 theta_diagnostic_ordered_by_prevalence  <-
19     theta_diagnostic_with_prevalence[rev(theta_index),theta_index]
20
21 # plotting
22 colors_plot <- rev(category_levels[names(theta_index),]$color)
23
24 theta_plot <- t(theta_diagnostic_ordered_by_prevalence)
25
26 print(levelplot(theta_plot,
27   xlab = NULL,
28   ylab = NULL,
29   scales = list(
30     tck = 1,
31     x = list(rot = 45),
32     y = list(col = colors_plot)),
33   cuts = 12,
34   col.regions = gray(20:0/20),
35   colorkey = list(space = "right"),
36   par.settings = theme.nopadding,
37   panel = function(...) {
38     panel.levelplot(...);
```

```
39      # the grid:
40      panel.abline(h = seq(1.5, 32.5), col = "lightgray");
41      panel.abline(v = seq(1.5, 32.5), col = "lightgray");
42      }
43    ))
```

C.10. classifications-table-most-diagnostic-five-terms.R

```
1  library(xtable)
2  terms_most_diagnostic <- get_terms(model_lda,5)[,topics_most_diagnostic
       ]
3  terms_topics_unique <-
4      terms_most_diagnostic[,unique(colnames(terms_most_diagnostic))]
5  topics_per_tab <- 6
6  topics_total <- ncol(terms_topics_unique)
7  for(i in 1:(topics_total %/% topics_per_tab)) {
8      xtable_raw <- xtable(terms_topics_unique[,(topics_per_tab * (i-1)
9                       + 1):(topics_per_tab * i), drop = FALSE])
10     xtable_strings <-
11         unlist(strsplit(capture.output(print(xtable_raw)), "\n"))
12     xtable_strings <- xtable_strings[xtable_strings != "\\begin{table}[
           ht]"]
13     xtable_strings <- xtable_strings[xtable_strings != "\\end{table}"]
14     cat(xtable_strings, sep = "\n")
15 }
16 if((topics_total %% topics_per_tab) != 0) {
17     xtable_raw <- xtable(terms_topics_unique[,(topics_per_tab*i +
18                       1):topics_total, drop = FALSE])
19     xtable_strings <-
20         unlist(strsplit(capture.output(print(xtable_raw)), "\n"))
21     xtable_strings <- xtable_strings[xtable_strings != "\\begin{table}[
           ht]"]
22     xtable_strings <- xtable_strings[xtable_strings != "\\end{table}"]
23     cat(xtable_strings, sep = "\n")
24 }
```

C.11. classifications-fig-levelplot-five-most-diagnostic.R

```
1  topics_by_ratio_reordered <- topics_by_ratio[1:5,order_by_major_index]
2
3  topics_most_diagnostic_five <-
4      unique(as.vector(topics_by_ratio_reordered))
5
6  # because most plots' coordinate system starts at the bottom, we have
       to convert the data:
7  theta_plot <-
8      t(theta_mean_ratios[rev(order_by_major_index),topics_most_
           diagnostic_five])
```

```
rownames(theta_plot) <- paste(topics_most_diagnostic_five)

colors_plot <- rev(category_levels$color)

print(levelplot(theta_plot,
    xlab = NULL,
    ylab = NULL,
    scales = list(
        tck = 1,
        x = list(rot = 45),
        y = list(col = colors_plot)),
    cuts = 12,
    col.regions = gray(20:0/20),
    colorkey = list(space = "right"),
    #par.settings = theme.nopadding,
    panel = function(...) {
        panel.levelplot(...);
        panel.abline(v = seq(1.5, nrow(theta_plot)-0.5), col = "
           lightgray");
        panel.abline(h = seq(1.5, 32.5), col = "lightgray");
        panel.abline(h = 4.5);
        panel.abline(h = 14.5);
        #panel.abline(v = length(fiveMostDiagnosticTopics)-8.5);
        #panel.abline(v = length(fiveMostDiagnosticTopics)-31.5);
        }
    ))
```

C.12. classifications-original-topics-table.R

```
library(xtable)

terms_topics_unique <- matrix(data = c(
    c("insect","myb","pheromone","lens","larvae"),
    c("species","phylogenetic","evolution","evolutionary","sequences"),
    c("gene","vector","vectors","expression","transfer"),
    c("structure","angstrom","crystal","residues","structures"),
    c("folding","native","protein","state","energy"),
    c("nuclear","nucleus","localization","cytoplasm","export"),
    c("neural","development","dorsal","embryos","ventral"),
    c("species","global","climate","co2","water"),
    c("species","selection","evolution","genetic","populations"),
    c("chromosome","region","chromosomes","kb","map"),
    c("cells","cell","antigen","lymphocytes","cd4"),
    c("tumor","cancer","tumors","human","cells"),
    c("host","bacterial","bacteria","strains","salmonella"),
    c("synaptic","neurons","postsynaptic","hippocampal","synapses"),
    c("resistance","resistant","drug","drugs","sensitive"),
    c("channel","channels","voltage","current","currents"),
    c("plants","plant","arabidopsis","tobacco","leaves"),
    c("cortex","brain","subjects","task","areas"),
    c("theory","time","space","given","problem"),
    c("hair","mechanical","mb","sensory","ear"),
```

```
24      c("large","scale","density","observed","observations"),
25      c("time","spectroscopy","nmr","spectra","transfer"),
26      c("force","surface","molecules","solution","surfaces"),
27      c("population","populations","genetic","diversity","isolates"),
28      c("research","new","information","understanding","paper"),
29      c("age","old","aging","life","young")), nrow=5,
30      dimnames = list(1:5, paste("Topic",c(217,274,126,63,200,209,
            42,2,280,15,64,102,
31      112,210,201,165,142,222, 39,105,221,270,55,114, 109,120))))
32
33  topics_per_tab <- 6
34  topics_total <- ncol(terms_topics_unique)
35  for(i in 1:(topics_total %/% topics_per_tab)) {
36      xtable_raw <- xtable(terms_topics_unique[,(topics_per_tab * (i-1)
37                      + 1):(topics_per_tab * i), drop = FALSE])
38      xtable_strings <-
39          unlist(strsplit(capture.output(print(xtable_raw)), "\n"))
40      xtable_strings <- xtable_strings[xtable_strings != "\\begin{table}[
            ht]"]
41      xtable_strings <- xtable_strings[xtable_strings != "\\end{table}"]
42      cat(xtable_strings, sep = "\n")
43  }
44  if((topics_total %% topics_per_tab) != 0) {
45      xtable_raw <- xtable(terms_topics_unique[,(topics_per_tab*i +
46                      1):topics_total, drop = FALSE])
47      xtable_strings <-
48          unlist(strsplit(capture.output(print(xtable_raw)), "\n"))
49      xtable_strings <- xtable_strings[xtable_strings != "\\begin{table}[
            ht]"]
50      xtable_strings <- xtable_strings[xtable_strings != "\\end{table}"]
51      cat(xtable_strings, sep = "\n")
52  }
```

C.13. trends-table-year-frequencies.R

```
1  library(xtable)
2  frequencies <- t(table(abstracts_meta$year))
3  rownames(frequencies) <- "Frequency"
4  xTable <- xtable(frequencies)
5  xTableStrings <- unlist(strsplit(capture.output(print(xTable)), "\n"))
6  xTableStrings <- xTableStrings[xTableStrings != "\\begin{table}[ht]"]
7  xTableStrings <- xTableStrings[xTableStrings != "\\end{table}"]
8  cat(xTableStrings, sep = "\n")
```

C.14. trends-table-significance.R

```
1  outputmat <- rbind(
2      sapply(significance_neg, length),
3      sapply(significance_pos, length),
4      sapply(significance_total, length))
5  rownames(outputmat) <- c("Negative trend", "Positive trend", "Total")
```

```
colnames(outputmat) <- paste("p-level=",
    format(p_level, drop0trailing = TRUE, scientific = FALSE), sep="")
xTable <- xtable(outputmat)
xTableStrings <- unlist(strsplit(capture.output(print(xTable)), "\n"))
xTableStrings <- xTableStrings[xTableStrings != "\\begin{table}[ht]"]
xTableStrings <- xTableStrings[xTableStrings != "\\end{table}"]
xTableStrings <- append(xTableStrings, "\\hline", after = 9)
cat(xTableStrings, sep = "\n")
```

C.15. trends-fig-all-300.R

```
library(lattice)
print(xyplot(theta_mean_by_year_ts[,names(sort(theta_mean_lm_coef_slope
   ))],
  layout = c(5, 6),
  #screens = c(rep("cold topics", 5), rep("hot topics", 5)),
  screens = rep(1:30, each = 10),
  superpose = TRUE,
  col = "blue",
  alpha = 0.3,
  ylim = c(0, 0.015),
        #ylab = "Mean theta",
        ylab = expression(paste("Mean ",theta)),
  xlab = "Year",
  type = c("l", "g"),
  #aspect = "xy",
  #auto.key = list(space = "right"),
  auto.key = FALSE,
  scales = list(x = list(alternating = FALSE)),
  #par.settings = standard.theme(color = FALSE)
  ))
```

C.16. trends-fig-five-hot-and-cold.R

```
cold_and_hot_ts <- cbind(
    theta_mean_by_year_ts[,topics_cold[1:5]],
    theta_mean_by_year_ts[,topics_hot[1:5]], deparse.level=0)
colnames(cold_and_hot_ts) <-
    as.character(c(topics_cold[1:5], topics_hot[1:5]))

print(xyplot(cold_and_hot_ts,
  layout = c(2, 1),
        screens = c(rep("Cold topics", 5), rep("Hot topics", 5)),
  superpose = TRUE,
        ylim = c(0, 0.015),
  ylab = expression(paste("Mean ",theta)),
  xlab = "Year",
  type = c("l", "g"),
  auto.key = list(space = "right"),
  scales = list(x = list(alternating = FALSE))
  #par.settings = standard.theme(color = FALSE)
```

```
18         ))
```

C.17. trends-table-terms.R

```
1  terms_hot <- get_terms(model_lda,12)[,topics_hot[1:5]]
2  terms_cold <- get_terms(model_lda,12)[,topics_cold[1:5]]
3
4  for(i in list(terms_cold, terms_hot)) {
5    xTable <- xtable(i)
6    xTableStrings <- unlist(strsplit(capture.output(print(xTable)), "\n")
       )
7    xTableStrings <- xTableStrings[xTableStrings != "\\begin{table}[ht]"]
8    xTableStrings <- xTableStrings[xTableStrings != "\\end{table}"]
9    cat(xTableStrings, sep = "\n")
10 }
```

C.18. trends-original-terms-table.R

```
1  library("xtable")
2  terms_cold <- matrix(data=c(
3          c("cdna","amino","sequence","acid","protein","isolated","
              encoding","cloned","acids","identity","clone","expressed"),
4          c("kda","protein","purified","molecular","mass","chromatography
              ","polypeptide","gel","sds","band","apparent","labeled"),
5          c("antibody","antibodies","monoclonal","antigen","igg","mab","
              specific","epitope","human","mabs","recognized","sera")),
6          ncol = 3, dimnames = list(1:12,paste("Topic",c(37, 289,
7                                    75))))
8  terms_hot <- matrix(data=c(
9          c("species","global","climate","co2","water","environmental","
              years","marine","carbon","diversity","ocean","extinction"),
10         c("mice","deficient","normal","gene","null","mouse","type","
              homozygous","role","knockout","development","generated"),
11         c("apoptosis","death","cell","induced","bcl","cells","apoptotic
              ","caspase","fas","survival","programmed","mediated")),
12         ncol = 3, dimnames = list(1:12,paste("Topic",c(2, 134,
13                                   179))))
14 for(i in list(terms_cold,terms_hot)) {
15   xTable <- xtable(i)
16   xTableStrings <- unlist(strsplit(capture.output(print(xTable)), "\n")
       )
17   xTableStrings <- xTableStrings[xTableStrings != "\\begin{table}[ht]"]
18   xTableStrings <- xTableStrings[xTableStrings != "\\end{table}"]
19   cat(xTableStrings, sep = "\n")
20 }
```

C.19. tagging-document-tag-latex.R

```
1  document_tag_latex <- function(model_lda, corpus_tm, document_index) {
2      paste0 <- function( ..., sep="" ) paste( ..., sep = sep )
```

```
3       return_strings <- "
4       \\definecolor{tagging-base}{gray}{0.65}
5       \\definecolor{tagging-1}{gray}{0.60}
6       \\definecolor{tagging-2}{gray}{0.55}
7       \\definecolor{tagging-3}{gray}{0.55}
8       \\definecolor{tagging-4}{gray}{0.45}
9       \\definecolor{tagging-5}{gray}{0.37}
10      \\definecolor{tagging-6}{gray}{0.30}
11      \\definecolor{tagging-7}{gray}{0.22}
12      \\definecolor{tagging-8}{gray}{0.25}
13      \\definecolor{tagging-9}{gray}{0.17}
14      \\definecolor{tagging-10}{gray}{0.10}
15      "
16      # split abstract into tokens by whitespace
17      # (tm does tokenization only after punctuation removal)
18      words <- unlist(strsplit(corpus_tm[[document_index]], " "))
19      #words <- c("(,(twoparentheses","(test1--test2","te-st-test2","test
           ))")
20      # save punctuation separately
21      words_punctuation_index <- gregexpr("[[:punct:]]", words)
22      words_punctuation_index_leading <- gregexpr("^[[:punct:]]+", words)
23      words_punctuation_index_trailing <- gregexpr("[[:punct:]]+$", words
           )
24
25      # remove punctuation, to lower
26      terms <- gsub("[[:punct:]]","", words, perl=T)
27      terms <- gsub("[[:digit:]]","", terms, perl=T)
28      terms <- tolower(terms)
29
30      # most prevalent topic in abstract?
31      topic_prevalent <- which.max(model_lda@gamma[document_index,])
32      wordassignments_df <- data.frame(
33              topics = model_lda@wordassignments[document_index,]$v,
34              terms = model_lda@terms[model_lda@wordassignments[document_
                   index,]$j])
35
36      wordassignments_index <- list()
37      model_terms_index_list <- list()
38      for(jj in 1:length(words)) {
39              wordassignments_index[[jj]] <- which(wordassignments_df$
                   terms == terms[jj])
40              model_terms_index_list[[jj]] <- which(model_lda@terms ==
                   terms[jj])
41              }
42      model_terms_index <- as.numeric(model_terms_index_list)
43
44      phi_log <- model_lda@beta[topic_prevalent,(model_terms_index)]
45
46      word_phi_levels <- cut(phi_log,10,labels= F)
47      colors_latex <- paste0("tagging-",1:10, sep="")
48
49      color_base <- "tagging-base"
```

```
50
51      # cases:
52      # superscripts: word is in wordassignments of ldaModel, can, but
            need not have contrast
53      # contrast: word is a word of phi, can, but need not have
            superscript, create index in advance
54
55      for(jj in 1:length(words)) {
56          # leading punctuation in base color
57          word_split <- unlist(strsplit(words[[jj]] , ""))
58          leading <- sum(attr(words_punctuation_index_leading[[jj]],"
                match.length"))
59          if(leading<0) leading <- 0
60          trailing <- sum(attr(words_punctuation_index_trailing[[jj]],"
                match.length"))
61          if(trailing<0) trailing <- 0
62          if(leading>0) {
63              return_strings <- paste0(return_strings,"\n\\textcolor{",
                    color_base, "}{")
64              for(kk in 1:leading) return_strings <- paste0(return_
                    strings, word_split[kk])
65              return_strings <- paste0(return_strings,"}")
66          }
67          return_strings <- paste0(return_strings,"\\textcolor{")
68          # determine color for current word
69          if(!is.na(word_phi_levels[jj]))
70              return_strings <- paste0(return_strings,
71                   colors_latex[word_phi_levels[[jj]]])
72          else return_strings <- paste0(return_strings, color_base)
73          return_strings <- paste0(return_strings, "}{")
74          for(kk in ((leading+1):(length(word_split)-trailing)))
75              return_strings <- paste0(return_strings, word_split[kk])
76          return_strings <- paste0(return_strings, "}")
77          # topic superscript, if applicable
78          if(length(wordassignments_index[[jj]]) != 0) {
79              return_strings <- paste0(return_strings,"\\textcolor{")
80              if(!is.na(word_phi_levels[jj])) return_strings <- paste0(
                    return_strings, colors_latex[word_phi_levels[[jj]]])
81              else return_strings <- paste0(return_strings, color_base)
82              return_strings <- paste0(return_strings, "}{")
83              return_strings <- paste0(return_strings, "\\textsuperscript
                    {")
84              return_strings <- paste0(return_strings,
85                       wordassignments_df$topics[[wordassignments_index[[
                            jj]]]])
86              if(!is.na(word_phi_levels[jj])) return_strings <- paste0(
                    return_strings, "}")
87              return_strings <- paste0(return_strings, "}")
88          }
89          # trailing punctuation in base color
90          if(trailing>0) {
91              return_strings <-
```

```
92                  paste0(return_strings,"\\textcolor{", color_base, "}{")
93              for(kk in
94                      ((length(word_split)-trailing+1):length(word_split)
                           ))
95              return_strings <-
96                  paste0(return_strings, word_split[kk])
97              return_strings <- paste0(return_strings, "}")
98          }
99          return_strings <- paste0(return_strings, " ")
100     }
101     return(return_strings)
102 }
103
104 cat(document_tag_latex(model_lda, abstracts_corpus,
105                 example_abstract))
```

C.20. tagging-table-topics-most-prevalent.R

```
1  library(xtable)
2  topics_prevalent_five <- sort(model_lda@gamma[example_abstract,],index.
       return=T, decreasing=T)$ix[1:5]
3  #sort(ldaModel@gamma[abstract,],index.return=T, decreasing=T)
4  terms_prevalent <- get_terms(model_lda,12)[,topics_prevalent_five]
5  prevalence <- model_lda@gamma[example_abstract,topics_prevalent_five]
6  prevalence_table <- rbind(
7  format(model_lda@gamma[example_abstract,topics_prevalent_five],
8          digits=4),
9          terms_prevalent <-
10         get_terms(model_lda,12)[,topics_prevalent_five]
11 )
12 rownames(prevalence_table) <- c("Probability",paste(1:12))
13
14 xTable <-  xtable(prevalence_table)
15 xTableStrings <- unlist(strsplit(capture.output(print(xTable)), "\n"))
16 replaceString <- grep("Probability", xTableStrings)
17 #replaceString <- grep("  12", xTableStrings)
18 xTableStrings[[replaceString]] <- paste(xTableStrings[[replaceString]],
       "\\hline", sep="\n")
19 xTableStrings <- xTableStrings[xTableStrings != "\\begin{table}[ht]"]
20 xTableStrings <- xTableStrings[xTableStrings != "\\end{table}"]
21 cat(xTableStrings, sep = "\n")
```

C.21. appendix-model-300-terms-tables.R

```
1  library("topicmodels")
2  library("xtable")
3
4  load("model-lda-gibbs-300topics.RData")
5
6  topics_total <- model_lda@k
7  topics_per_line <- 5
```

```
lines_per_page <- 3

topics_index <- split(1:topics_total, rep(1:topics_total,
                each=topics_per_line*lines_per_page, len=topics_total))

page_cat <- function(topics, model_lda) {

    terms <- get_terms(model_lda,12)[,min(topics):max(topics)]

    # make output safe for plain pdflatex
    terms <- iconv(terms,from="UTF-8",to="ASCII",sub="")

    cat("\\begin{table}")
    for(i in 1:(length(topics) %/% topics_per_line)) {
        xtable_raw <- xtable(terms[,(topics_per_line * (i-1)
                        + 1):(topics_per_line * i), drop = FALSE])
        xtable_strings <-
            unlist(strsplit(capture.output(print(xtable_raw)), "\n"))
        xtable_strings <- xtable_strings[xtable_strings != "\\begin{
            table}[ht]"]
        xtable_strings <- xtable_strings[xtable_strings != "\\end{table
            }"]
        cat(xtable_strings, sep = "\n")
    }
    if((length(topics) %% topics_per_line) != 0) {
        xtable_raw <- xtable(terms[,(topics_per_line*i +
                        1):topics_total, drop = FALSE])
        xtable_strings <-
            unlist(strsplit(capture.output(print(xtable_raw)), "\n"))
        xtable_strings <- xtable_strings[xtable_strings != "\\begin{
            table}[ht]"]
        xtable_strings <- xtable_strings[xtable_strings != "\\end{table
            }"]
        cat(xtable_strings, sep = "\n")
    }
    # clearpage because otherwise there are too many open floats
    cat(paste("\\caption{Twelve most probable terms of Topics ",
                    min(topics)," to ",max(topics),".}
    \\end{table}
    \\clearpage",sep=""))
}
lapply(topics_index, page_cat, model_lda)
```

D. Appendix: Web Scraping a Corpus of PNAS Journal Abstracts

In order to reproduce and further expand upon the article "Finding scientific topics" (Griffiths and Steyvers, 2004) I had to download the corpus of PNAS abstracts and the corresponding metadata that were used. The general process is called **web scraping** or **screen scraping**, which is automated extraction of data from web pages. In this appendix I describe how the PNAS web site is structured and how I implemented the web scraping in the Python scripting language. Furthermore I show how the PNAS corpus that is subject to various analyses in this thesis was scraped and produced. All source code listings can be found in the final section of this appendix.

D.1. Site Structure and Content

The PNAS archive can be accessed via the URL `http://www.pnas.org/content/by/year`. All issues for a given year are listed one level deeper, for example `http://www.pnas.org/content/by/year/1991`. PNAS is published bi-weekly, in practice this equals approximately 24 to 26 issues per year. The table of content of a single issue can be accessed by URLs like: `http://www.pnas.org/content/88/15.toc`, which translates into volume 88 (=year 1991), issue 15. This page level finally lists all the issue's research articles, commentaries, retractions, corrections, etc. and the respective links to abstracts (in HTML format, from 1969 onwards) and full texts (in both PDF and HTML format, the latter only from 1996 onwards). From 1996's issue 23 onwards, this page is also divided into category sections, which I used to directly extract the major and minor categories (for a distinction between major and minor categories see Chapter 5.4) needed for the 2001-specific analysis. Where no categories are provided I also downloaded the full text in PDF format which contains a machine-readable category description in the page header.

D.2. Implementation

The following programs were used, all being part of official repositories of Debian 6.0:

- **Python** 2.6.6 – a high level scripting language (Python 2.5.5 was also tested successfully). See for example Ascher et al. (2005).
- **Scrapy** 0.10.3 – a web scraping framework implemented in Python.

- **pycurl** 7.19.0-3+b1 – Python bindings for **libcurl**, a multi-protocol file transfer library.
- **BeautifulSoup** 3.1.0.1-2 – an HTML parser for Python.
- **pdftotext** 0.12.4 – part of the **poppler** PDF utility package.
- **OpenSSH** 5.5p1 – the Secure shell, for proxy-tunnelling downloads through a remote host (Barrett and Silverman, 2001) (optional).
- **Gnumeric** – this spreadsheed is useful for manual checks of CSV files (optional).

Note: All script run times reported by me in later subsections were measured on Debian 6.0 running on a workstation equipped with an Intel E8400 Core 2 Duo processor and 4 GiB working memory. It is highly likely that the scripts will perform equally fast on other systems because their main bottleneck is file input and output.

D.2.1. Preparing Scrapy

A project folder and default settings can be generated by calling:

```
scrapy startproject pnas
```

At this point one would begin adapting the default scripts and settings to a specific webpage. **Scrapy** provides access to HTML data via XML Path Language (XPath) selectors, which allows add-ons to the **Mozilla Firefox** browser like **XPather** or **Firebug** to be used for determining the location of data containers in an HTML document.

For our purpose I had to modify the default **Scrapy** files `settings.py` (listing: Section D.3.1), `items.py` (Section D.3.2), `pnas-spider.py` (Section D.3.3) and `pipelines.py` (Section D.3.4)

A test of how the crawler performs can be made by calling:

```
scrapy shell http://www.pnas.org/content/by/year/1990
```

for the list of issues per year, or, for an issue's table of contents:

```
scrapy shell http://www.pnas.org/content/98/3.toc
```

D.2.2. Calling Scrapy

Once **Scrapy** is set up, the download of the corpus from 1991 to 2001 can begin:

```
scrapy crawl pnas --set FEED_FORMAT=csv --set FEED_URI=scraped.csv \
    --set DOWNLOAD_DELAY=10
```

As indicated by the arguments to the call, the scraped items will be saved in a Csv (comma separated values) file, which can be imported into most spreadsheets or databases. A download delay of ten seconds was specified to avoid getting blocked by the Pnas web site. This value was found by trial and error.

Depending on the current server load, the crawler will finish in approximately 30 minutes and produce a Csv file with approximately 30,000 rows, each row representing an entry in an issue's table of content. A row consists of fields such as category, authors, title, year and Urls to abstracts and full text versions.

D.2.3. Selecting All Further Downloads

The next Python script `2-select.py` (listing: see Section D.3.6) reads all items from `scraped.csv` and decides which data is relevant for the corpus and its metadata. A new file called `selected.csv` is written which extends `scraped.csv` by three more fields. These fields specify whether an item should be downloaded as abstract and full Pdf, and whether an item appears more than once (optional, not needed at this step because duplicates are simply overwritten).

`2-select.py` can be called with an additional switch (`-m`) to download only missing files, i.e. abstracts and Pdfs which are not found in the local folder structure.

Note: All Python scripts (except for `4-scrub.py`) depend on a module called `csv_unicode.py`, which provides a Csv reader for Utf-8 (Unicode) format (listing: see Section D.3.10).

D.2.4. Downloading the Selected Abstracts and Full Texts

`3-get.py` (listing in Section D.3.7) reads `selected.csv` and downloads – depending on the supplied arguments – abstracts in Html format (`abstracts`) and/or full Pdfs (`fullpdfs`). Downloading all abstracts takes nearly 120 hours, all full Pdfs another 60 hours. Again, this is caused by a necessary interval of ten seconds between downloads. In order to cut total the time by half by downloading from two different Ip addresses, I make use of a Ssh tunnel to a user account on a remote server. A local Socks5 web proxy is set up by calling `ssh`:

```
ssh -D 8080 -v -N h0053049@login.wu.ac.at
```

In one shell the script is executed by calling:

```
./3-get.py abstracts
```

for downloading all abstracts. In another shell, this time for downloading all Pdfs, the script is called with different arguments:

```
./3-get.py fullpdfs socks
```

Whenever the script (called with the `socks` argument) makes requests to the local port 8080, they are forwarded to the remote server and thus appear to the Pnas web

site as coming from a different location, thereby circumventing the download limit of one request per 10 seconds.

Failed downloads caused by server timeouts are caught by the script and fed into a download loop which starts after all regular downloads end. This loop then terminates either successfully (by finishing all downloads) or by being interrupted via keyboard (Ctrl-C), in which case the missing files' names are stored in a text file (for manual download).

Downloaded files are stored in the current working directory in folders by year. Filenames are similar to the download URLs, with the page number first (page numbering is continuous for each volume, i.e. year) and the extension is either `.abstract` for HTML abstracts or `.full.pdf` for full text PDFs. Example: `1991/8831.full.pdf`.

The total size of all relevant 27,664 `.abstract` files is approximately 1 GiB, all 14,681 `.full.pdf` files together amount to 19 GiB.

D.2.5. Cleaning the Downloaded Files

Further processing of the downloaded files is handled by the script `4-scrub.py` (listing in Section D.3.8). Available arguments which determine which file types need to be cleaned are `abstracts` and/or `fullpdfs`.

All **abstracts** are still in HTML format, therefore the script searches for all `.abstract` files and parses them via the BeautifulSoup library. It extracts the abstract texts to `.txt` files, keywords (if available) to `.keywords` files and category designations (if available) to `.categories` files, keeping the original UTF-8 encoding for all files.

```
./4-scrub.py abstracts
```

Conversion for all abstracts takes approximately 40 minutes. It is possible that there are errors caused by incomplete or empty web pages. These abstracts have to be re-downloaded manually and the script re-run.

In order to gain access to additional category designations that are not provided in abstracts or the issues' table of contents, the **full PDFs** are first converted to text files by the script. In a second step these text files are read in and the categories saved as `.full.category` files. The original PDFs are already in an OCR'd (optical character recognition) format, however in a rather low quality. Therefore the script also substitutes malformed categories like "Biochenmstry" for their correct strings. A run of the script:

```
./4-scrub.py fullpdfs
```

takes approximately 25 minutes. One PDF will produce an error: the file `1993/10295.full.pdf` apparently has not been published in a machine readable format. In this case the file `10295.full.category` needs to be created manually with the content `Biochemistry`.

D.2.6. Merging and Inspecting the Available Metadata

The last Python script `5-zip.py` (listing in Section D.3.9) handles the merging of all available additional metadata (see previous subsection) into the file `meta.csv`. It also performs additional sanity checks to ensure that all data is complete and puts out warnings if otherwise. The script depends upon a version of `selected.csv` in which all relevant abstracts have been tagged for download. Such a file is produced by calling `2-select.py` without the `-m` parameters (see previous Subsection D.2.3).

For debugging purposes the script can be called with the parameter `all`, resulting in all lines of `selected.csv` to be evaluated and saved in the output file. If the argument `skiperrors` is supplied instead, only items with complete metadata will be returned. In both cases, a field called `warning` can be inspected in the output file `meta.csv`, which lists any inconsistencies detected by the script.

Running:

```
./5-zip.py skiperrors
```

will therefore produce the file `meta.csv` with one header row and the same number of rows that were selected in `selected.csv` (given that downloads are complete). The following columns are now available:

Major and minor category : from the issue's table of content. Relevant only for the 2001 analysis of categories in Chapter 5.

Authors, title, volume, issue, pages : irrelevant.

Year : relevant.

URLs to HTML abstracts, extracts, full texts and full PDFs : needed by download script.

Path to local abstract .txt : Relevant for matching abstracts and metadata by serving as a unique identifier. See next subsection.

Duplicate, download abstract, download full PDF : binary (`yes` or empty), relevant only for previous scripts.

Category in full PDF, major and minor categories in abstract : Relevant for the final merged category.

Merged category : needed for the trend analysis of the whole corpus. As can be seen in the script's source code (Section D.3.9), all available category metadata is imported into a new "merged" category. Most minor categories are left intact, apart from a few substitutions to compensate for outliers:

Original category	Merged category	Comments
Botany	Plant Biology	only 1991, a few dozen in total
Molecular Biology	Biochemistry	only one abstract (1996)
Neurosciences	Neurobiology	only one abstract (1994)
Pharmacology	Physiology/Pharmacology	to account for category Phys./Pharm. from 1991
Physiology	Physiology/Pharmacology	see above
Plant Sciences	Plant Biology	only one abstract (1992)
Political Sciences	Social Sciences	only two abstracts (1999)

An additional script (`opt-categories.py`, see Section D.3.11) was used for listing the categories of each year in a Csv file, allowing me to compare the categories in a spreadsheet and manually produce the table above.

Keywords : these data are not available for all abstracts, therefore useless for our purposes.

Warnings : irrelevant when the file was produced with the `skiperrors` argument.

D.2.7. Importing Abstracts and Metadata as a tm Corpus in R

With the metadata complete and all abstracts saved as local text files we can now proceed to import the data in R. This is accomplished by executing:

```
R CMD BATCH --vanilla tm-corpus-make.R
```

The script `tm-corpus-make.R` (listing in Section D.3.12) reads in both metadata and abstracts, matches and merges them, and saves the resulting **tm** corpus to the file `tm-corpus-pnas-abstracts.RData`. The resulting file can then be loaded in R for further processing. It is worth noting that the initial 1 GiB of Html abstracts are converted to a compressed file of a mere 16 MiB which also includes metadata. A more detailed description of the corpus can be found in Chapter 5.

D.3. Listings

D.3.1. `settings.py`

```
1 # Scrapy settings for pnas project
2 #
3 # For simplicity, this file contains only the most important settings
    by
4 # default. All the other settings are documented here:
5 #
6 #     http://doc.scrapy.org/topics/settings.html
7 #
```

```
8  # Or you can copy and paste them from where they're defined in Scrapy:
9  #
10 #     scrapy/conf/default_settings.py
11 #
12
13 BOT_NAME = 'pnas'
14 BOT_VERSION = '1.0'
15
16 SPIDER_MODULES = ['pnas.spiders']
17 NEWSPIDER_MODULE = 'pnas.spiders'
18 DEFAULT_ITEM_CLASS = 'pnas.items.PnasItem'
19 USER_AGENT = '%s/%s' % (BOT_NAME, BOT_VERSION)
20
21 DOWNLOAD_DELAY = 10 #0.25    # 250 ms of delay
```

D.3.2. items.py

```
1  # Define here the models for your scraped items
2  #
3  # See documentation in:
4  # http://doc.scrapy.org/topics/items.html
5
6  from scrapy.item import Item, Field
7
8  class PnasItem(Item):
9      # define the fields for your item here like:
10     category = Field()
11     category_minor = Field()
12     authors = Field()
13     title = Field()
14     year = Field()
15     volume = Field()
16     issue = Field()
17     pages = Field()
18     url_abstract = Field()
19     url_extract = Field()
20     url_fulltext = Field()
21     url_fullpdf = Field()
```

D.3.3. pnas_spider.py

```
1  from scrapy.spider import BaseSpider
2  from scrapy.http import Request
3  from pnas.items import PnasItem
4  from scrapy.selector import HtmlXPathSelector
5  from BeautifulSoup import BeautifulSoup
6
7  def soup_flatten(soup):
8      # http://stackoverflow.com/questions/753052/strip-html-from-strings
           -in-python
9      soupstring = "".join(soup.findAll(text=True))
10     soupstring = soupstring.replace(u"\n",u" ")
```

```
11      # remove whitespace : http://bytes.com/topic/python/answers/590441-
            how-strip-mulitiple-white-spaces
12      soupstring = " ".join(soupstring.split())
13      return soupstring
14
15  def html_flatten(selection):
16      try:
17          soup = BeautifulSoup(selection)
18      except:
19          return ""
20      return soup_flatten(soup)
21
22  class PnasSpider(BaseSpider):
23      name = "pnas"
24      allowed_domains = ["www.pnas.org"]
25      url_base = "http://www.pnas.org"
26      start_urls = [url_base + "/content/by/year/" + str(year) for year
            in range(1991,2002)]
27
28      # default parser for start_urls
29      def parse(self, response):
30          hxs = HtmlXPathSelector(response)
31          issue_urls = hxs.select('//td[@class="proxy-archive-by-year-
                month"]/p/strong/a/@href').extract()
32          for issue_url in issue_urls:
33              yield Request(self.url_base + issue_url, callback = self.
                    parse_issue)
34
35      def parse_item(self, selection):
36          item = PnasItem()
37          if selection.select('.//h4') != []:
38              item['title'] = html_flatten(selection.select('.//h4')[0].
                    extract())
39              # "From the Cover: "... see 2-select.py
40              #titlestring = selection.select('.//h4/text()')[0].extract
                    ()
41              #titlestring = titlestring.replace(u"\n",u" ")
42              #titlestring = " ".join(titlestring.split())
43              #item['title'] = titlestring
44          authors = selection.select('.//ul[contains(@class,"cit-auth-
                list")]')
45          if authors != []:
46              item['authors'] = html_flatten(authors[0].extract())
47
48          if selection.select('.//a[@rel="full-text"]/@href') != []:
49              item['url_fulltext'] = self.url_base + selection.select('
                    .//a[@rel="full-text"]/@href')[0].extract()
50          if selection.select('.//a[@rel="full-text.pdf"]/@href') != []:
51              item['url_fullpdf'] = self.url_base + selection.select('.//
                    a[@rel="full-text.pdf"]/@href')[0].extract()
52          if selection.select('.//a[@rel="abstract"]/@href') != []:
```

```
53              item['url_abstract'] = self.url_base + selection.select('
                    .//a[@rel="abstract"]/@href')[0].extract()
54          if selection.select('.//a[@rel="extract"]/@href') != []:
55              item['url_extract'] = self.url_base + selection.select('.//
                    a[@rel="extract"]/@href')[0].extract()
56
57          if selection.select('.//span[@class="cit-print-date"]') != []:
58              item['year'] = selection.select('.//span[@class="cit-print-
                    date"]/text()')[0].extract().strip()
59          if selection.select('.//span[@class="cit-vol"]') != []:
60              item['volume'] = selection.select('.//span[@class="cit-vol
                    "]/text()')[0].extract().strip()
61          if selection.select('.//span[@class="cit-issue"]') != []:
62              item['issue'] = selection.select('.//span[@class="cit-issue
                    "]/text()')[0].extract().strip()
63          if selection.select('.//span[@class="cit-pages"]') != []:
64              item['pages'] = html_flatten(selection.select('.//span[
                    @class="cit-pages"]')[0].extract())
65          return item
66
67      def parse_issue(self, response):
68          hxs = HtmlXPathSelector(response)
69          div1s = hxs.select('//div[contains(@class,"toc-level level1")]'
                )
70
71          # For each section that is headed by H2 (Commentaries, Physical
                Sciences,...)
72          for div1 in div1s:
73              # Get H2
74              h2string = div1.select('h2/span/text()')[0].extract()
75
76              # If we encounter a list immediately after H2, we are in a
                    category without minor (Comments...)
77              # and can expect the articles (as list items li)
78              for li in div1.select('ul[@class="cit-list"]/li'):
79                  item = self.parse_item(li)
80                  item['category'] = h2string
81                  yield item
82
83              # If we encounter the level2 div, we can expect H3 and then
                    articles
84              for div2 in div1.select('div[contains(@class,"toc-level
                    level2")]'):
85                  if div2.select('h3/span/text()') != []:
86                      h3string = div2.select('h3/span/text()')[0].extract
                            ()
87
88                  for li in div2.select('ul[@class="cit-list"]/li'):
89                      item = self.parse_item(li)
90                      item['category'] = h2string
91                      item['category_minor'] = h3string
92                      yield item
```

D.3.4. pipelines.py

```
# Define your item pipelines here
#
# Don't forget to add your pipeline to the ITEM_PIPELINES setting
# See: http://doc.scrapy.org/topics/item-pipeline.html

class PnasPipeline(object):
    def process_item(self, item, spider):
        return item
```

D.3.5. 1-scrape.sh

```
#!/bin/bash
time scrapy crawl pnas --set FEED_FORMAT=csv --set FEED_URI=scraped.csv
    --set DOWNLOAD_DELAY=10
```

D.3.6. 2-select.py

```
#! /usr/bin/python
"""
"""
from csv_unicode import *
import os.path
from optparse import OptionParser

def main():
    parser = OptionParser("usage: %prog [options] arg")
    parser.add_option("-f", "--file", dest="filename", help="read from"
        )
    parser.add_option("-o", "--output", dest="output_filename", help="
        write to")
    parser.add_option("-d", "--check-dup", action="store_true", dest="
        check_duplicate_abstracts")
    parser.add_option("-s", "--skip-dup", action="store_true", dest="
        skip_duplicate_abstracts")
    parser.add_option("-m", "--missing-only", action="store_true", dest
        ="download_missing_only")
    parser.add_option("-t", "--clean-titles", action="store_true", dest
        ="clean_titles")
    parser.add_option("-c", "--clean-categories", action="store_true",
        dest="clean_categories")
    parser.add_option("-r", "--reorder-columns", action="store_true",
        dest="reorder_columns")
    parser.set_defaults(filename="scraped.csv",
                        output_filename="selected.csv",
                        check_duplicate_abstracts=False,
                        skip_duplicate_abstracts=False,
                        clean_titles=True,
                        clean_categories=True,
                        reorder_columns=True,
                        download_missing_only=False)
```

```
26      (options, args) = parser.parse_args()
27      if len(args) != 0: parser.error("incorrect number of arguments")
28
29      url_base = "http://www.pnas.org"
30      categories_major = [u"Biological Sciences", u"Physical Sciences", u
            "Social Sciences"]
31      reader = UnicodeReader(open(options.filename, "rb"))
32      firstline = next(reader)
33      field = dict(zip(firstline, range(len(firstline))))
34
35      writer = UnicodeWriter(open(options.output_filename, "wb")) #,
            delimiter="\t")
36
37      # Add header line to output, always the same extra columns,
            regardless of options
38      firstline = []
39      for value in range(len(field)):
40          for key in field:
41              if field[key] == value:
42                  firstline.append(key)
43      if options.reorder_columns==True:
44          firstline=["category","category_minor","authors","title","year"
                ,"volume","issue","pages","url_abstract","url_extract","
                url_fulltext","url_fullpdf"]
45      writer.writerow(firstline + ["url_abstract_is_duplicate", "
            download_abstract", "download_fullpdf"])
46
47      processed_abstract_fields = []
48      url_abstract_duplicates = 0
49      relevant_abstracts = 0
50      relevant_fullpdfs = 0
51      found_fullpdfs = 0
52      found_abstracts = 0
53      marked_abstracts = 0
54      marked_fullpdfs = 0
55      irrelevant_abstracts = 0
56      irrelevant_fullpdfs = 0
57      for row in reader:
58          url_abstract = row[field['url_abstract']]
59          url_abstract_is_duplicate = ""
60          download_abstract=""
61          download_fullpdf=""
62          # Clean manually some misformed major categories
63          if options.clean_categories==True:
64              if row[field['category']]==u"Biological Sciences:
                    Biochemistry":
65                  row[field['category']]=u"Biological Sciences"
66                  row[field['category_minor']]=u"Biochemistry"
67              if row[field['category']]==u"Biological Sciences:
                    Biophysics":
68                  row[field['category']]=u"Biological Sciences"
69                  row[field['category_minor']]=u"Biophysics"
```

```
70              if row[field['category']]==u"Biological Sciences: Evolution
                    ":
71                  row[field['category']]=u"Biological Sciences"
72                  row[field['category_minor']]=u"Evolution"
73              if row[field['category']]==u"Evolution":
74                  row[field['category']]=u"Biological Sciences"
75                  row[field['category_minor']]=u"Evolution"
76              if row[field['category']]==u"Immunology":
77                  row[field['category']]=u"Biological Sciences"
78                  row[field['category_minor']]=u"Immunology"
79              if row[field['category']]==u"Physical Sciences: Chemistry":
80                  row[field['category']]=u"Physical Sciences"
81                  row[field['category_minor']]=u"Chemistry"
82              if row[field['category']]==u"Physical Sciences: Geophysics"
                    :
83                  row[field['category']]=u"Physical Sciences"
84                  row[field['category_minor']]=u"Geophysics"
85
86          ## (Research articles are not affected by duplicates)
87          if options.check_duplicate_abstracts==True and url_abstract in
                processed_abstract_fields: #and row[field['category']] in
                categories_major :
88              url_abstract_is_duplicate="yes"
89              url_abstract_duplicates += 1
90
91          ## if a line has no category_minor it is possible it will
                appear later with category_minor from a different section.
                If not, it won't be needed anyways
92          if row[field['url_abstract']]!=u"" and row[field['
                category_minor']]!=u"": # could include major cat check here
93              ## if no duplicate checking is active, this will always
                    work
94              if options.skip_duplicate_abstracts==True and
                    url_abstract_is_duplicate=="yes":
95                  pass
96              else:
97                  download_abstract="yes"
98                  relevant_abstracts+=1
99
100         ### Full PDFs for abstracts that lack a category
101         if row[field['url_abstract']]!=u"" and row[field['category']]==
                u"Research Article" and row[field['url_fullpdf']]!=u"":
102             if options.skip_duplicate_abstracts==True and
                    url_abstract_is_duplicate=="yes":
103                 pass
104             else:
105                 download_abstract="yes"
106                 relevant_abstracts+=1
107                 download_fullpdf="yes"
108                 relevant_fullpdfs+=1
109
110 #         if url_abstract_is_duplicate=="":
```

```
111 #            if row[field['url_abstract']]!=u"" and row[field['
    category_minor']]!=u"": # could include major cat check here
112 #                download_abstract="yes"
113 #                relevant_abstracts+=1
114 #             What is with items that have no url_abstract? ignore, for
     categories_major these are only a few
115 #             irrelevant inaugural articles etc.
116 
117             ## Full PDFs for abstracts that lack a category
118 #            if row[field['category']]==u"Research Article" and row[
    field['url_abstract']]!=u"" and row[field['url_fullpdf']]!=u"":
119 #                download_abstract="yes"
120 #                relevant_abstracts+=1
121 #                download_fullpdf="yes"
122 #                relevant_fullpdfs+=1
123 
124         if options.download_missing_only==True:
125             abstract_filename = url_abstract.split("/")[-1]
126             base_filename = abstract_filename.split(".")[0]
127             local_base_path = row[field['year']]+"/"+base_filename
128             local_path = row[field['year']]+"/"+abstract_filename
129             if os.path.isfile(local_path) and os.path.getsize(
                local_path)!=0:
130                 if download_abstract=="" and url_abstract_is_duplicate
                    =="":
131                     irrelevant_abstracts+=1
132                     #print local_path, url_abstract
133                 download_abstract=""
134                 found_abstracts+=1
135             local_path = row[field['year']]+"/"+base_filename+".full.
                pdf"
136             if os.path.isfile(local_path) and os.path.getsize(
                local_path)!=0:
137                 if download_fullpdf=="" and url_abstract_is_duplicate==
                    "": irrelevant_fullpdfs+=1
138                 download_fullpdf=""
139                 found_fullpdfs+=1
140 
141         if options.clean_titles==True:
142             row[field['title']] = row[field['title']].replace("From the
                 Cover: ","")
143             row[field['title']] = row[field['title']].replace("From the
                 Academy: ","")
144             row[field['title']] = row[field['title']].replace("
                Inaugural Article: ","")
145             row[field['title']] = row[field['title']].replace("
                Colloquium Paper: ","")
146             row[field['title']] = row[field['title']].replace("Special
                Feature: ","")
147             row[field['title']] = row[field['title']].replace("From the
                 Cover: ","")
148 
```

```
149         if download_abstract=="yes": marked_abstracts+=1
150         if download_fullpdf=="yes": marked_fullpdfs+=1
151         rows_additional =[url_abstract_is_duplicate, download_abstract,
                download_fullpdf]
152         if options.reorder_columns==True:
153             row = [row[index] for index in [field[name] for name in
                    firstline]]
154         writer.writerow(row + rows_additional)
155
156         if options.check_duplicate_abstracts==True and url_abstract!=u"
                ": processed_abstract_fields.append(url_abstract)
157
158     if options.check_duplicate_abstracts==True: print "Duplicate
            url_abstract entries:", url_abstract_duplicates
159     print ".abstract: relevant: %5d, found: %5d, found but irrelevant:
            %5d, marked for download: %5d" % (relevant_abstracts,
            found_abstracts, irrelevant_abstracts, marked_abstracts)
160     print ".fullpdf:  relevant: %5d, found: %5d, found but irrelevant:
            %5d, marked for download: %5d" % (relevant_fullpdfs,
            found_fullpdfs, irrelevant_fullpdfs, marked_fullpdfs)
161     print "Irrelevant files (non-duplicate): leftovers from other
            scrapers or manual download selection"
162     print "Relevant files: these include Research articles with unknown
             category. Some files will later be recognized as irrelevant (5-
            zip.py)."
163
164 if __name__== "__main__":
165     main()
```

D.3.7. 3-get.py

```
1  #! /usr/bin/python
2  #import os.path
3  #print os.path.supports_unicode_filenames
4  from time import sleep
5  from csv_unicode import *
6  import os
7  import sys
8  import pycurl
9  import codecs
10
11 def download_file_urllib(urlbase, urlpath, tofolder):
12     from urllib2 import Request, urlopen
13     filename = urlpath.split("/")[-1]
14     if not os.path.exists(tofolder):
15         os.makedirs(tofolder)
16     # http://www.voidspace.org.uk/python/articles/urllib2.shtml#
           handling-exceptions
17     req = Request(urlbase + urlpath)
18     try:
19         response = urlopen(req)
20     except IOError, e:
21         if hasattr(e, 'reason'):
```

```
22              print urlbase+urlpath + ': We failed to reach a server.'
23              print 'Reason: ', e.reason
24          elif hasattr(e, 'code'):
25              print urlbase+urlpath + ': The server couldn\'t fulfill the
                    request.'
26              print 'Error code: ', e.code
27          return 1
28      try:
29          fileout = open(tofolder + "/" + filename, "wb")
30          fileout.write(response.read())
31          fileout.close
32      except:
33          print tofolder + filename + ": Error saving"
34          return 1
35      return 0
36
37  def download_file(url, tofolder, socks=False):
38      filename = url.split("/")[-1]
39      if not os.path.exists(tofolder):
40          os.makedirs(tofolder)
41      try:
42          fileout = open(tofolder + "/" + filename, "wb")
43      except:
44          print tofolder +"/"+ filename + ": Error opening file for
                saving"
45          return 1
46      curl = pycurl.Curl()
47      curl.setopt(pycurl.FOLLOWLOCATION, 1)
48      curl.setopt(pycurl.MAXREDIRS, 5)
49      curl.setopt(pycurl.CONNECTTIMEOUT, 30)
50      curl.setopt(pycurl.TIMEOUT, 300)
51      curl.setopt(pycurl.NOSIGNAL, 1)
52      curl.setopt(pycurl.URL, url)
53      curl.setopt(pycurl.WRITEDATA, fileout)
54      if socks==True:
55          curl.setopt(pycurl.PROXY, 'localhost')
56          curl.setopt(pycurl.PROXYPORT, 8080)
57          curl.setopt(pycurl.PROXYTYPE, pycurl.PROXYTYPE_SOCKS5)
58      try:
59    curl.perform()
60      except:
61          print url+': Error downloading.'
62    import traceback
63    traceback.print_exc(file=sys.stderr)
64    sys.stderr.flush()
65          return(1)
66      curl.close()
67      fileout.close()
68      return 0
69
70
71  def missing_loop(missinglist, filename, sockssupport):
```

```
72      # the following is a loop that runs only if there were downloading
            errors.
73      downloaded = 0
74      missingloops = 1
75      try:
76          while missinglist != []:
77              print "-"*40
78              print "Retry loop", str(missingloops), " -", "Items missing
                    :", len(missinglist)
79              print "-"*40
80              missinglist2 = []
81              for missing in missinglist:
82                  sleep(1)
83                  if download_file(*missing, sockssupport)!= 0:
84                      missinglist2.append(missing)
85                  else:
86                      downloaded += 1
87              # overwrite old list
88              missinglist = missinglist2
89              missingloops += 1
90      except KeyboardInterrupt:
91          for missing in missinglist:
92              filemissing = open(filename ,"wb")
93              filemissing.write(missing[0]+"\n")
94              filemissing.close()
95              print "Successfully downloaded:", downloaded
96              print "Sill missing:", len(missinglist)
97              print filename, "written!"
98      return downloaded
99
100 def main():
101     # more options?: verbose (Downloading...), different input file
102     url_base = "http://www.pnas.org"
103     interval = 10
104     filename = "selected.csv"
105     reader = UnicodeReader(open(filename, "rb"))
106     firstline = next(reader)
107     field = dict(zip(firstline, range(len(firstline))))
108     sockssupport = False
109     if "socks" in sys.argv:
110         sockssupport=True
111
112     downloaded = 0
113     missinglist = []
114
115     if "abstracts" in sys.argv:
116         print "abstracts:"
117         for row in reader:
118             if row[field['download_abstract']]=="yes":
119                 print "Downloading:", row[field['url_abstract']], row[
                        field['year']]
```

```
120                 if download_file(row[field['url_abstract']].encode("
                        ascii"), row[field['year']], sockssupport)!= 0:
121                     missinglist.append((row[field['url_abstract']], row
                            [field['year']]))
122                 else:
123                     downloaded += 1
124                 sleep(interval) # seconds
125             #if downloaded == 3:
126                 #break
127         print "="*40
128         print "Successful downloads:", downloaded
129         if missinglist != []:
130             missingdownloaded = missing_loop(missinglist, "missing-
                    abstracts.txt", sockssupport)
131             print "Successful downloads in missing loop:",
                    missingdownloaded
132
133     elif "fullpdfs" in sys.argv:
134         print "fullpdfs:"
135         for row in reader:
136             if row[field['download_fullpdf']]=="yes":
137                 url = row[field['url_fullpdf']].replace("+html","")
138                 print "Downloading:", url, row[field['year']]
139                 if download_file(url.encode("ascii"), row[field['year'
                        ]], sockssupport)!= 0:
140                     missinglist.append((url, row[field['year']]))
141                 else:
142                     downloaded += 1
143                 sleep(interval) # seconds
144             #if downloaded == 3:
145             #    break
146         print "="*40
147         print "Successful downloads:", downloaded
148         if missinglist != []:
149             missingdownloaded = missing_loop(missinglist, "missing-
                    fullpdf-res.txt", sockssupport)
150             print "Successful downloads in missing loop:",
                    missingdownloaded
151
152     else:
153         print "Fields in", filename, ":", field
154         print
155         print "Available arguments: 'fullpdfs' and/or 'abstracts' and/
                or 'socks'"
156         print "Socks support: ssh -D 8080 -v -N h0053049@login.wu.ac.at
                "
157
158         #print "Test download:"
159         #if download_file("http://www.pnas.org/content/98/26/14745.full
                .pdf", "test-curl") != 0:
160         #    print "Error..."
161
```

```
162 if __name__== "__main__":
163     main()
```

D.3.8. 4-scrub.py

```
1  #!/usr/bin/python
2
3  from BeautifulSoup import BeautifulSoup
4  import codecs
5  import os
6  import glob
7  import sys
8  import subprocess
9  import re
10
11 def soup_flatten(soup):
12     # http://stackoverflow.com/questions/753052/strip-html-from-strings
       -in-python
13     soupstring = "".join(soup.findAll(text=True))
14     soupstring = soupstring.replace(u"\n",u" ")
15     # remove whitespace : http://bytes.com/topic/python/answers/590441-
       how-strip-mulitiple-white-spaces
16     soupstring = " ".join(soupstring.split())
17     return soupstring
18
19 # see also: Python Cookbook, Recipe 1.18. Replacing Multiple Patterns
    in a Single Pass
20 # however: order is important for the first few items, therefore I will
     leave it as it is.
21 def category_scrub(category):
22     category = category.replace(u"(rheumatic disease/nuclear
       autoantigen/epitope)", u"Immunology")
23     category = category.replace(u"This paper is based on a presentation
        given during the 130th Annual Meeting of the National Academy
       ofSciences",u"Review")
24     category = category.replace(u"Vol. 89, pp. 6639-6643, July 1992", u
       "Neurobiology")
25     category = category.replace(u"bull. At 40-48 hr postfertilization,
       embryos were manually",u"Developmental Biology")
26     category = category.replace(u",",u"")
27     category = category.replace(u"'",u"")
28     category = category.replace(u".",u"")
29     category = category.replace(u"Anthropo ogy",u"Anthropology")
30     category = category.replace(u"Applied 1iological Sciences",u"
       Applied Biological Sciences")
31     category = category.replace(u"Applied 1iological Sciences",u"
       Applied Biological Sciences")
32     category = category.replace(u"Applied Biologcal Sciences",u"Applied
        Biological Sciences")
33     category = category.replace(u"Biocem stry",u"Biochemistry")
34     category = category.replace(u"Biochemlstry",u"Biochemistry")
35     category = category.replace(u"Biochemisty",u"Biochemistry")
36     category = category.replace(u"Biochemnistry",u"Biochemistry")
```

```
37      category = category.replace(u"Biochemnstry",u"Biochemistry")
38      category = category.replace(u"Biochemsstry",u"Biochemistry")
39      category = category.replace(u"Biochemstr",u"Biochemistry")
40      category = category.replace(u"Biochemustry",u"Biochemistry")
41      category = category.replace(u"Biochenistry",u"Biochemistry")
42      category = category.replace(u"Biochenmstry",u"Biochemistry")
43      category = category.replace(u"Biochenustry",u"Biochemistry")
44      category = category.replace(u"Ccll Biology",u"Cell Biology")
45      category = category.replace(u"Ceil Biology",u"Cell Biology")
46      category = category.replace(u"Celi Biology",u"Cell Biology")
47      category = category.replace(u"Cell Biofogy",u"Cell Biology")
48      category = category.replace(u"Cell BioIlogy",u"Cell Biology")
49      category = category.replace(u"Cell Bioiogy",u"Cell Biology")
50      category = category.replace(u"Cell Biol[ogy",u"Cell Biology")
51      category = category.replace(u"Cell BioLogy",u"Cell Biology")
52      category = category.replace(u"CeLl Biology",u"Cell Biology")
53      category = category.replace(u"Cell Bio ogy",u"Cell Biology")
54      category = category.replace(u"Cell Bio[ogy",u"Cell Biology")
55      category = category.replace(u"Cell Biotogy",u"Cell Biology")
56      category = category.replace(u"Colioquium Paper",u"Colloquium Paper"
            )
57      category = category.replace(u"Development Biology",u"Developmental
            Biology")
58      category = category.replace(u"exonuclease BAL-31 to generate
            ordered sets ofdeletions in", u"Microbiology")
59      category = category.replace(u"ficrobiology",u"Microbiology")
60      category = category.replace(u"ficrobiology",u"Microbiology")
61      category = category.replace(u"fMicrobiology",u"Microbiology")
62      category = category.replace(u"Immunoiogy",u"Immunology")
63      category = category.replace(u"Inmunology",u"Immunology")
64      category = category.replace(u"Immuno ogy",u"Immunology")
65      category = category.replace(u"JBiophysics",u"Biophysics")
66      category = category.replace(u"Medical ciences",u"Medical Sciences")
67      category = category.replace(u"Medical Science", "Medical Sciences")
68      category = category.replace(u"Medical sciences",u"Medical Sciences"
            )
69      category = category.replace(u"Medical Sclences",u"Medical Sciences"
            )
70      category = category.replace(u"Medical Sciencess", u"Medical
            Sciences")
71      category = category.replace(u"Microbiology 0",u"Microbiology")
72      category = category.replace(u"Microbiology a",u"Microbiology")
73      category = category.replace(u"Ncurobiology",u"Microbiology")
74      category = category.replace(u"Neurobiofogy",u"Microbiology")
75      category = category.replace(u"NeurobioLogy",u"Microbiology")
76      category = category.replace(u"Neurobio ogy",u"Microbiology")
77      category = category.replace(u"Pharmaology",u"Pharmacology")
78      category = category.replace(u"Phraoogy",u"Pharmacology")
79      return category
80
81  def abstract_scrub(filename):
82      inputfile = codecs.open(filename, "r", encoding='utf-8').read()
```

```
83      soup = BeautifulSoup(inputfile)
84      #print soup.prettify()
85
86      # title, already in meta
87      #h1 = soup.find("h1", {"id":"article-title-1"})
88      #print soup_flatten(h1)
89
90      # abstract content
91      abstractdiv = soup.find("div", { "class" : "section abstract" })
92      if abstractdiv == None: return 1
93
94      if abstractdiv.p != None:
95          outputabstract = codecs.open(filename+".txt","w", encoding="utf
                -8")
96          #print soup_flatten(abstractdiv.p)
97          outputabstract.write(soup_flatten(abstractdiv.p)+"\n")
98          outputabstract.close()
99
100     # keywords, optional
101     if abstractdiv.ul != None:
102         outputkeywords = codecs.open(filename+".keywords","w", encoding
                ="utf-8")
103         #print soup_flatten(abstractdiv.ul)
104         outputkeywords.write(soup_flatten(abstractdiv.ul)+"\n")
105         outputkeywords.close()
106
107     ## categories, again
108     majorlist = soup.find("ul", { "class": re.compile("subject-headings
            ") })
109     if majorlist==None:
110         print filename, ": error parsing categories"
111         return 1
112
113     outputcategories = codecs.open(filename+".categories","w", encoding
            ="utf-8")
114     for major in majorlist.findAll("li", recursive=False):
115         majorstring= major.contents[0].string.strip()
116         #print majorstring
117         found=0
118         if major.findAll("li")!=[]:
119             for minor in major.findAll("li"):
120                 cat_line = majorstring+","+minor.a.string.strip()
121                 outputcategories.write(cat_line+"\n")
122                 found += 1
123         else:
124             outputcategories.write(majorstring+",\n")
125     outputcategories.close()
126
127     return 0
128
129 def main():
130
```

```
131     if len(sys.argv) < 2:
132         print "No argument supplied. Available are: 'abstracts' and/or
                'fullpdfs'."
133         return 1
134 
135     errorfile = open("errors-scrub-pdf.txt","w")
136     for folder in ["1991","1992","1993","1994","1995","1996","1997","
            1998","1999","2000","2001"]:
137     #for folder in
            ["1993","1994","1995","1996","1997","1998","1999","2000","2001"]:

138     #for folder in ["2000","2001"]:
139     #for folder in ["2001"]:
140         if "abstracts" in sys.argv:
141             errors = 0
142             successes = 0
143             print "\n", "="*60
144             print folder, "Converting .abstract to .abstract.txt"
145             print "-"*40
146             for filename in glob.glob(folder+"/"+"*.abstract"):
147                 # print filename
148                 if abstract_scrub(filename)!=0:
149                     print filename,": Could not be parsed. Empty or
                            partial abstract?"
150                     errors += 1
151                 else: successes +=1
152             print "="*60
153             print "Successes:", successes, "Errors:", errors
154 
155         if "fullpdfs" in sys.argv:
156             errors = 0
157             successes = 0
158             print "\n", "="*60
159             print folder, "Extracting .full.category from .full.pdf"
160             print "-"*40
161             for filename in glob.glob(folder+"/"+"*.pdf"):
162                 if subprocess.call(['pdftotext', '-raw',filename])!=0:
163                     print filename,": pdftotext error."
164                     errorfile.write(filename)
165                     errors += 1
166                 else:
167                     filebase = filename.replace(".pdf","")
168                     try:
169                         fulltext = codecs.open(filebase+".txt","r",
                                encoding="utf-8")
170                         # Skip first 2 lines
171                         next(fulltext); next(fulltext)
172                         category = fulltext.readline().strip()
173                         category = category_scrub(category)
174                         fulltext.close()
175                         os.remove(filebase+".txt")
176                     except:
```

```
177                         print filename, ": Error reading .full.txt. Is
                                the PDF empty or not OCRed? "
178                         errors += 1
179                         continue
180                     try:
181                         categoryfile = codecs.open(filebase+".category"
                                ,"w", encoding="utf-8")
182                         categoryfile.write(category)
183                         categoryfile.close()
184                         successes +=1
185                     except:
186                         print filename, ": Error writing .full.category
                                ."
187                         errors += 1
188             print "="*60
189             print "Successes:", successes, "Errors:", errors
190     errorfile.close()
191
192 if __name__== "__main__":
193     main()
```

D.3.9. 5-zip.py

```
1  #! /usr/bin/python
2
3  from csv_unicode import *
4  import os.path
5  import sys # for args
6
7  category_merged_dict = {u"Agricultural Sciences":"Agricultural Sciences
       ",
8                          u"Anthropology":"Anthropology",
9                          u"Applied Biological Sciences":"Applied
                               Biological Sciences",
10                         u"Applied Mathematics":"Applied Mathematics",
11                         u"Applied Physical Sciences":"Applied Physical
                               Sciences",
12                         u"Astronomy":"Astronomy",
13                         u"Biochemistry":"Biochemistry",
14                         u"Biophysics":"Biophysics",
15                         u"Botany":"Plant Biology", #!: only 1991, a few
                                dozen
16                         u"Cell Biology":"Cell Biology",
17                         u"Chemistry":"Chemistry",
18                         u"Computer Sciences":"Computer Sciences",
19                         u"Developmental Biology":"Developmental Biology
                               ",
20                         u"Ecology":"Ecology",
21                         u"Economic Sciences":"Economic Sciences",
22                         u"Engineering":"Engineering",
23                         u"Evolution":"Evolution",
24                         u"Genetics":"Genetics",
25                         u"Geology":"Geology",
```

```
26                          u"Geophysics":"Geophysics",
27                          u"Immunology":"Immunology",
28                          u"Mathematics":"Mathematics",
29                          u"Medical Sciences":"Medical Sciences",
30                          u"Molecular Biology":"Biochemistry", #! only
                                one instance
31                          u"Microbiology":"Microbiology",
32                          u"Neurobiology":"Neurobiology",
33                          u"Neurosciences":"Neurobiology",#! only one
                                instance, 1994
34                          u"Pharmacology":"Physiology/Pharmacology", #!
                                see below
35                          u"Physics":"Physics",
36                          u"Physiology":"Physiology/Pharmacology",#! to
                                account for category Phys/Pharm from 1991
37                          u"Plant Biology":"Plant Biology",
38                          u"Plant Sciences":"Plant Biology", #! one
                                instance in 1992
39                          u"Political Sciences":"Social Sciences", #! two
                                 instances in 1999
40                          u"Population Biology":"Population Biology",
41                          u"Psychology":"Psychology",
42                          u"Social Sciences":"Social Sciences",
43                          u"Statistics":"Statistics"}
44
45  def main():
46      url_base = "http://www.pnas.org"
47      filename = "selected.csv"
48      reader = UnicodeReader(open(filename, "rb"))
49      firstline = next(reader)
50      field = dict(zip(firstline, range(len(firstline))))
51      categories_major = [u"Biological Sciences", u"Physical Sciences", u
            "Social Sciences"]
52
53      writer = UnicodeWriter(open("meta.csv", "wb")) #, delimiter="\t")
54
55      ## Will: in CSV: .abstract.txt Location
56      # category, keywords
57      # option to ignore all lines that have no abstract.txt available
58      # checks to show if all research articles now have a category
59
60      ### All relevant abstracts
61      notselected = 0
62      processed_abstract_fields = []
63      output_list = []
64      duplicate_abstracts = 0
65      downloads_selected=0
66      lines_written=0
67
68      if len(sys.argv) == 1:
69          print "Available arguments:"
70          print "'all' ......... process all input lines."
```

```
71          print "'skiperrors'... only return complete metadata."; exit(1)
72          print "Warnings are also stored in the resulting file."
73      elif len(sys.argv) == 2 and "all" in sys.argv: errorlineskip = 0
74      elif len(sys.argv) == 2 and "skiperrors" in sys.argv: errorlineskip
            = 1
75      else: print "Wrong argument"; exit(1)
76      print "Remember to previously run ./2-select.py without -m flag to
           ensure that all relevant abstracts are tagged for download!"
77
78      firstline = []
79      for value in range(len(field)):
80          for key in field:
81              if field[key] == value:
82                  firstline.append(key)
83      writer.writerow(firstline + ["abstract_local_path", "
           local_fullpdf_path", "category_fullpdf", "
           category_abstract_1_major", "category_abstract_1_minor", "
           category_abstract_2_major", "category_abstract_2_minor","
           category_merged", "keywords", "warning"])
84
85      for row in reader:
86          add_local_abstract_path = ""
87          add_local_fullpdf_path = ""
88          add_category_fullpdf = ""
89          add_keywords = ""
90          add_warning = ""
91          add_category_abstract_1_major=""
92          add_category_abstract_1_minor=""
93          add_category_abstract_2_major=""
94          add_category_abstract_2_minor=""
95          add_category_merged =""
96          url_abstract = row[field['url_abstract']]
97
98          if url_abstract!="" and url_abstract in
               processed_abstract_fields:
99              add_warning += "CSV: duplicate abstract path, "
100             duplicate_abstracts += 1
101             if errorlineskip: continue
102         processed_abstract_fields.append(url_abstract)
103         if url_abstract=="": add_warning+= "CSV: Empty url_abstract, "
104
105         ## Only lines that were previously selected
106         if row[field["download_abstract"]] == u"yes":
107             downloads_selected+=1
108             abstract_filename = url_abstract.split("/")[-1]
109             base_filename = abstract_filename.split(".")[0]
110             local_base_path = row[field['year']]+"/"+base_filename
111             local_abstracttxt_path = row[field['year']]+"/"+
                   abstract_filename+".txt"
112
113             if os.path.isfile(local_abstracttxt_path):
114                 if os.path.getsize(local_abstracttxt_path)==0:
```

```
115                     add_warning += "FILE: .abstract.txt is empty, "
116                     if errorlineskip: continue
117                 else:
118                     add_local_abstract_path = local_abstracttxt_path
119                     add_warning += "FILE: OK: .abstract.txt, "
120             else:
121                 add_warning += "FILE: .abstract.txt not found, "
122                 if errorlineskip: continue
123 
124             ## get pdf path
125             local_meta_path = local_base_path + ".full.pdf"
126             if os.path.isfile(local_meta_path):
127                 add_local_fullpdf_path = local_meta_path
128 
129             ## get .category
130             local_meta_path = local_base_path + ".full.category"
131             if os.path.isfile(local_meta_path):
132                 metafile = open(local_meta_path, "rb")
133                 add_category_fullpdf = unicode(metafile.readline().
                        strip())
134                 metafile.close()
135 
136             ## get .keywords
137             local_meta_path = local_base_path + ".abstract.keywords"
138             if os.path.isfile(local_meta_path):
139                 metafile = codecs.open(local_meta_path, "rb", encoding=
                        'utf-8')
140                 add_keywords = metafile.readline()
141                 add_keywords = add_keywords.replace(u"\u2016","|")
142                 metafile.close()
143 
144             ## get .categories from abstract
145             local_meta_path = local_base_path + ".abstract.categories"
146             if os.path.isfile(local_meta_path):
147                 cat_reader = UnicodeReader(open(local_meta_path, "rb"))
148                 linesread=0
149                 for line in cat_reader:
150                     if line[0] in categories_major:
151                         if linesread==0:
152                             add_category_abstract_1_major=line[0]
153                             add_category_abstract_1_minor=line[1]
154                         if linesread==1:
155                             add_category_abstract_2_major=line[0]
156                             add_category_abstract_2_minor=line[1]
157                     linesread+=1
158 
159         #else:
160 
161         ### Some more general checks
162         #"""
163         if row[field['category']] in categories_major:
164             ## generate merged categories
```

```
165             if row[field['category_minor']]!="":
166                 add_category_merged = category_merged_dict[row[field['
                        category_minor']]]
167             else:
168                 add_warning += "META: no minor category, "
169                 if errorlineskip: continue
170         elif row[field["category"]]=="Research Article":
171             if add_category_fullpdf in [u"Colloquium Paper",u"
                    Commentary",u"Review",u"Symposium Paper"]:
172                 add_warning += "META: 'Research article': irrelevant
                        add_category_fullpdf, "
173                 if errorlineskip: continue
174             else:
175                 if add_category_fullpdf!="":
176                     add_category_merged = category_merged_dict[
                            add_category_fullpdf]
177                 else:
178                     add_warning += "META: 'Research article': empty
                            add_category_fullpdf, "
179                     if errorlineskip: continue
180         ## very few cases for 2000, Special Feature, Research Article
                that have second categories in abstract
181         elif row[field["category"]]!="" and row[field["category_minor"
                ]]!="" and \
182             add_category_abstract_1_major=="" and
                    add_category_abstract_1_minor=="" and \
183             add_category_abstract_2_major in categories_major:
184                 add_category_merged = category_merged_dict[
                        add_category_abstract_2_minor]
185
186         else:
187             add_warning += "META: no category details, "
188             if errorlineskip: continue
189
190         if row[field["download_abstract"]] == u"":
191             add_warning += "CSV: not selected for download, "
192             notselected += 1
193             if errorlineskip: continue
194         #"""
195         writer.writerow(row + [add_local_abstract_path,
                add_local_fullpdf_path, add_category_fullpdf,
196                                add_category_abstract_1_major,
                                       add_category_abstract_1_minor,
197                                add_category_abstract_2_major,
                                       add_category_abstract_2_minor,
198                                add_category_merged, add_keywords,
                                       add_warning])
199         lines_written+=1
200
201     print "-"*80
202     print "Errors resulted in skipping a line:", errorlineskip and "yes
            , only good data is saved to .csv" or "no, all warnings can be
```

```
        read in .csv output"
203     print "Duplicate url_abstract:", duplicate_abstracts, errorlineskip
            and "(skipped)" or "(not skipped)"
204     print "Lines with unique url_abstract:", len(
        processed_abstract_fields), errorlineskip and "." or "(including
            empty)"
205     print "Abstracts originally selected for download:",
        downloads_selected, "(including 'Research Articles')"
206     print "Output: lines written:", lines_written
207     # print "Lines that were not selected for download:", notselected,
        errorlineskip and "(skipped, but full meta available (from
        previous scraping), including empty url_abstract)" or "(
        complement to selected)"
208
209     print "-"*80
210
211 if __name__== "__main__":
212     main()
```

D.3.10. csv_unicode.py

```
1  #! /usr/bin/python
2  import csv
3  import codecs
4  import cStringIO
5
6  class UTF8Recoder:
7      """
8      Iterator that reads an encoded stream and reencodes the input to
       UTF-8
9      """
10     def __init__(self, f, encoding):
11         self.reader = codecs.getreader(encoding)(f)
12
13     def __iter__(self):
14         return self
15
16     def next(self):
17         return self.reader.next().encode("utf-8")
18
19 class UnicodeReader:
20     """
21     A CSV reader which will iterate over lines in the CSV file "f",
22     which is encoded in the given encoding.
23     """
24
25     def __init__(self, f, dialect=csv.excel, encoding="utf-8", **kwds):
26         f = UTF8Recoder(f, encoding)
27         self.reader = csv.reader(f, dialect=dialect, **kwds)
28
29     def next(self):
30         row = self.reader.next()
31         return [unicode(s, "utf-8") for s in row]
```

```
32
33      def __iter__(self):
34          return self
35
36  class UnicodeWriter:
37      """
38      A CSV writer which will write rows to CSV file "f",
39      which is encoded in the given encoding.
40      """
41
42      def __init__(self, f, dialect=csv.excel, encoding="utf-8", **kwds):
43          # Redirect output to a queue
44          self.queue = cStringIO.StringIO()
45          self.writer = csv.writer(self.queue, dialect=dialect, **kwds)
46          self.stream = f
47          self.encoder = codecs.getincrementalencoder(encoding)()
48
49      def writerow(self, row):
50          self.writer.writerow([s.encode("utf-8") for s in row])
51          # Fetch UTF-8 output from the queue ...
52          data = self.queue.getvalue()
53          data = data.decode("utf-8")
54          # ... and reencode it into the target encoding
55          data = self.encoder.encode(data)
56          # write to the target stream
57          self.stream.write(data)
58          # empty queue
59          self.queue.truncate(0)
60
61      def writerows(self, rows):
62          for row in rows:
63              self.writerow(row)
64
65  def main():
66      pass
67
68  if __name__== "__main__":
69      main()
```

D.3.11. opt-categories.py

```
1  #! /usr/bin/python
2  from csv_unicode import *
3
4  """
5  Create a separate categories.csv, for deciding which categories to
       merge.
6  Not needed for a regular run of the scraper.
7  """
8
9  def main():
10     categories_major = [u"Biological Sciences", u"Physical Sciences", u
           "Social Sciences"]
```

```
11
12      filename = "meta.csv"
13      reader = UnicodeReader(open(filename, "rb"))
14      firstline = next(reader)
15      field = dict(zip(firstline, range(len(firstline))))
16      firstline = []
17      for value in range(len(field)):
18          for key in field:
19              if field[key] == value:
20                  firstline.append(key)
21
22      categories={}
23      for year in range(1991,2002):
24          categories[str(year)]=[]
25
26      for row in reader:
27          if row[field["category_fullpdf"]]!=u"":
28              categories[row[field['year']]].append(row[field["
                    category_fullpdf"]])
29          if row[field["category_minor"]]!=u"":
30              categories[row[field['year']]].append(row[field["
                    category_minor"]]+" ("+row[field["category"]]+")")
31      writer = UnicodeWriter(open("categories.csv", "wb")) #, delimiter
            ="\t")
32      writer.writerow(["year", "category"])
33      for year in categories:
34          for cat in set(categories[year]):
35              print year,cat
36              writer.writerow([year, cat])
37
38  if __name__== "__main__":
39      main()
```

D.3.12. tm-corpus-make.R

```
1   # setwd("/media/NTFS500GB_A/pnas-scrapy/pnas/")
2
3   meta_df <- read.table("meta.csv", header = T, sep=",",
4           quote="\"", encoding = "UTF-8", stringsAsFactors = F)
5
6   require("tm")
7
8   abstracts_reader <- FunctionGenerator(function(...)
9       function(elem, language, id) {
10          PlainTextDocument(x = elem$content,
11                  datetimestamp = as.POSIXlt(Sys.time(), tz = "GMT"),
12                  #heading = elem$content[[1]],
13                  id = id,
14                  origin = elem$uri,
15                  language = language)
16      })
17
18  abstracts_source <- DirSource(".",
```

```
19      pattern = "\\.abstract\\.txt", recursive = TRUE)
20
21  abstracts <- Corpus(abstracts_source, readerControl = list(
22      reader = abstracts_reader, language = "en_US"))
23
24  ## reformat paths
25  meta_df$abstract_local_path <- paste("./",
26          meta_df$abstract_local_path, sep="")
27
28  ## we have complete and unique metadata, now find abstracts in corpus
29  abstracts_origin_chr <- sapply(abstracts, Origin)
30  corpus_reordered_index <- match(meta_df$abstract_local_path,
31          abstracts_origin_chr)
32  ## select abstracts by available metadata
33  abstracts <- abstracts[corpus_reordered_index]
34
35  ## select metadata to match the corpus
36  abstracts_origin_chr <- sapply(abstracts, Origin)
37  meta_reordered_index <- match(abstracts_origin_chr,
38          meta_df$abstract_local_path)
39  meta_df <- meta_df[meta_reordered_index,]
40
41  # now metadata can be merged into corpus
42  #names(meta_df)
43  for (name in names(meta_df)[-c(10,13,14,15,25)])
44      meta(abstracts, tag = name) <- meta_df[name]
45
46  # does not work:
47  # sapply(names(meta_df), function(x) meta(abstracts, tag=x) <- meta_df[
        x])
48
49  ## also:
50  # DMetaData(abstracts)$title <- meta_df$title
51
52  #meta(abstracts)$title[1]
53
54  save(abstracts, file = "tm-corpus-pnas-abstracts.rda", compress = TRUE)
```

E. Appendix: PNAS Journal 1991-2001 Corpus: 300 Topics and Their Twelve Most Probable Terms from the LDA Model Sample

The twelve most probable terms for all 300 topics of the model sample that was used in the application part of this thesis are collected in the following tables. The source code for generating the tables is reproduced in Appendix C.21.

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
1	fraction	sperm	possibility	phosphatase	hiv
2	total	suggest	found	protein	virus
3	contained	fertilization	question	activity	immunodeficiency
4	content	chemotaxis	address	pp	human
5	amount	results	data	dephosphorylation	infection
6	amounts	mechanism	raise	phosphatases	viral
7	detected	chey	observations	pten	type
8	obtained	egg	strongly	alkaline	tat
9	found	hpr	finding	regulation	replication
10	fractions	report	revealed	ptp	infected
11	wheat	previously	findings	ptpase	ccr
12	barley	involved	raises	ppa	aids

	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10
1	receptors	survival	binding	variants	oxidative
2	receptor	donor	affinity	variant	oxygen
3	antagonist	transplantation	bind	hybrid	superoxide
4	agonist	transplanted	binds	hybrids	ho
5	antagonists	recipients	bound	parental	oxidation
6	agonists	rejection	site	resulting	free
7	acetylcholine	mixed	sites	compared	radical
8	gaba	grafts	ligand	single	reactive
9	muscarinic	host	highaffinity	data	sod
10	subtypes	allogeneic	interaction	identical	dismutase
11	desensitization	normal	ligands	based	species
12	acid	longterm	specifically	due	damage

	Topic 11	Topic 12	Topic 13	Topic 14	Topic 15
1	development	activation	species	coli	dnabinding
2	neural	activated	diversity	escherichia	proteins
3	expression	stat	global	bacterial	factors
4	embryos	nfb	marine	lambda	binding
5	embryonic	factor	ecological	phage	transcription
6	formation	transcription	abundance	repressor	basic
7	embryo	ifn	community	operon	protein
8	dorsal	activate	carbon	bacteriophage	factor
9	mesoderm	interferon	climate	strain	domain
10	bmp	traf	forest	bacteria	dna
11	ectopic	response	ecosystem	purified	family
12	wnt	constitutive	islands	operator	motif

Table E.1.: Twelve most probable terms of Topics 1 to 15.

	Topic 16	Topic 17	Topic 18	Topic 19	Topic 20
1	ii	apoptosis	gp	delta	thymocytes
2	iii	death	glycoprotein	density	cell
3	iv	cell	kappa	low	selection
4	following	bcl	nfkappa	transformation	mature
5	results	apoptotic	envelope	degree	thymic
6	ang	caspase	opioid	cs	development
7	vi	induced	mu	foci	immature
8	angiotensin	cells	morphine	population	thymus
9	independent	survival	rel	growth	dendritic
10	determined	fas	scd	assay	maturation
11	type	programmed	ankyrin	produced	rag
12	observations	bax	opiate	clones	positive

	Topic 21	Topic 22	Topic 23	Topic 24	Topic 25
1	immune	synthesis	oxide	primary	virulence
2	mice	labeling	nitric	results	salmonella
3	responses	de	synthase	similar	pseudomonas
4	response	synthesized	nos	suggest	bacterial
5	antigen	labeled	inos	peroxisome	aeruginosa
6	immunization	novo	inducible	peroxisomal	typhimurium
7	immunity	incorporation	inhibitor	demonstrate	secretion
8	vaccine	newly	production	secondary	genes
9	antigens	synthesize	activity	data	required
10	challenge	indicating	cgmp	report	bacteria
11	protective	pool	effect	proliferatoractivated	strains
12	ctl	label	larginine	ppar	host

	Topic 26	Topic 27	Topic 28	Topic 29	Topic 30
1	dna	activity	promoter	male	active
2	repair	activities	gene	female	reductase
3	adducts	enzymatic	transcription	males	inactive
4	mismatch	assay	element	females	activity
5	adduct	active	region	sexual	thioredoxin
6	sos	exhibited	elements	sex	reduction
7	lesion	dependent	upstream	mating	biologically
8	lesions	assays	enhancer	pheromone	ribonucleotide
9	glycosylase	exhibits	site	germ	form
10	mutagenesis	assayed	expression	offspring	enzyme
11	base	detectable	regulatory	reproductive	cofactor
12	mutator	biological	sequence	behavior	dihydrofolate

Table E.2.: Twelve most probable terms of Topics 16 to 30.

	Topic 31	Topic 32	Topic 33	Topic 34	Topic 35
1	density	selection	brain	factor	method
2	cholesterol	library	rat	pc	approach
3	low	libraries	situ	ngf	using
4	lipoprotein	approach	cortex	growth	assay
5	plasma	selected	regions	nerve	detection
6	ldl	strategy	immunoreactivity	nt	developed
7	apolipoprotein	phage	distribution	neurotrophic	technique
8	apob	novel	localized	bdnf	analysis
9	hdl	design	brains	survival	rapid
10	atherosclerosis	screening	neuronal	neurons	methods
11	apoe	method	hippocampus	neuronal	detect
12	lipoproteins	display	localization	neurotrophin	quantitative

	Topic 36	Topic 37	Topic 38	Topic 39	Topic 40
1	compounds	cd	major	reverse	mice
2	analogs	cells	parasite	wt	transgenic
3	potent	cell	leishmania	rnase	mouse
4	derivatives	tcell	parasites	transcriptase	expression
5	analog	lymphocytes	host	rt	transgene
6	compound	surface	trypanosoma	nucleoside	expressing
7	chemical	antigen	brucei	tc	control
8	selectivity	activation	cruzi	purine	animals
9	selective	anticd	eb	wilms	levels
10	analogues	activated	shown	type	generated
11	derivative	ligand	hosts	ribonuclease	overexpression
12	synthetic	lymphocyte	protozoan	azt	development

	Topic 41	Topic 42	Topic 43	Topic 44	Topic 45
1	cc	transfer	sequence	sp	cell
2	results	energy	sequences	olfactory	cells
3	melanocytes	fluorescence	conserved	glucocorticoid	motility
4	ci	time	motif	substance	dictyostelium
5	found	nm	regions	gr	bodies
6	tyrosinase	cm	motifs	glucocorticoids	waves
7	skin	electron	identified	mouse	wave
8	lead	spectroscopy	consensus	found	body
9	pas	spectra	similarity	gc	cellular
10	due	dynamics	highly	dexamethasone	movement
11	coat	absorption	comparison	specific	form
12	indicate	charge	analysis	response	discoideum

Table E.3.: Twelve most probable terms of Topics 31 to 45.

	Topic 46	Topic 47	Topic 48	Topic 49	Topic 50
1	vivo	populations	blood	nuclear	transcription
2	vitro	population	peripheral	localization	transcriptional
3	skin	genetic	cells	nucleus	factor
4	keratinocytes	variation	lymphocytes	cytoplasm	activation
5	epidermal	alleles	spleen	protein	factors
6	epidermis	diversity	lymphoid	export	promoter
7	demonstrate	polymorphism	lymph	cytoplasmic	repression
8	results	loci	mononuclear	nuclei	protein
9	ex	selection	nodes	localized	binding
10	keratin	polymorphisms	organs	subcellular	activator
11	keratinocyte	individuals	lymphocyte	import	transactivation
12	basal	natural	detected	signal	gal

	Topic 51	Topic 52	Topic 53	Topic 54	Topic 55
1	neurons	receptor	elements	fragment	repeat
2	spinal	hormone	element	fragments	repeats
3	cord	acid	insertion	restriction	sequence
4	nerve	thyroid	maize	length	sequences
5	sensory	retinoic	genome	endonuclease	bp
6	motor	receptors	insertions	bt	terminal
7	axons	ra	transposition	digestion	tandem
8	ganglia	nuclear	transposon	field	repeated
9	dorsal	tr	transposable	ecori	short
10	schwann	rar	genomes	studies	length
11	ganglion	retinoid	sequences	produced	units
12	peripheral	response	sequence	thuringiensis	contain

	Topic 56	Topic 57	Topic 58	Topic 59	Topic 60
1	chain	trna	single	fc	visual
2	chains	rrna	multiple	cells	motion
3	heavy	synthetase	individual	epsilon	stimulus
4	immunoglobulin	ribosomal	generation	mast	cortex
5	variable	ribosome	generated	gamma	orientation
6	light	ribosomes	study	ige	spatial
7	ig	trnas	generate	igg	field
8	region	anticodon	independent	ri	information
9	vh	synthetases	combination	mc	stimuli
10	heavychain	acceptor	derived	associated	responses
11	bcell	codon	analyzed	rii	color
12	ch	coli	independently	highaffinity	neural

Table E.4.: Twelve most probable terms of Topics 46 to 60.

	Topic 61	Topic 62	Topic 63	Topic 64	Topic 65
1	liver	camp	mice	genes	mass
2	rat	cyclic	normal	gene	spectrometry
3	hepatocytes	creb	gene	expression	identified
4	hepatic	adrenal	wildtype	encoding	analysis
5	epo	levels	development	identified	vhl
6	erythropoietin	pde	lacking	encode	molecular
7	hepatocyte	phosphodiesterase	homozygous	involved	sea
8	gm	production	knockout	identify	gas
9	epor	cgmp	disruption	analysis	determined
10	bile	regulation	targeted	expressed	urchin
11	livers	cre	role	products	chromatography
12	hepatocellular	response	mouse	genome	identification

	Topic 66	Topic 67	Topic 68	Topic 69	Topic 70
1	cellular	endogenous	deletion	previously	dopamine
2	biological	exogenous	truncated	identified	neurons
3	processes	results	deletions	reported	da
4	functions	suggest	analysis	described	cholinergic
5	including	demonstrate	deleted	shown	dopaminergic
6	function	role	region	recently	basal
7	report	data	fulllength	report	striatal
8	variety	ir	functional	additional	striatum
9	various	whereas	lacking	characterized	nucleus
10	implicated	cb	pkr	identification	pd
11	activities	addition	revealed	novel	forebrain
12	diverse	furthermore	mutants	furthermore	substantia

	Topic 71	Topic 72	Topic 73	Topic 74	Topic 75
1	analysis	isolated	cells	site	expression
2	revealed	isolation	transformation	sites	mrna
3	blot	intact	cmyc	binding	levels
4	northern	characterization	transformed	located	level
5	demonstrated	using	expression	near	gene
6	detected	report	fibroblasts	adjacent	increased
7	indicated	crf	ea	close	low
8	western	characterized	growth	location	expressed
9	using	describe	nih	distance	mrnas
10	hybridization	including	oncogenic	identified	regulation
11	analyses	previously	cellular	identify	transcripts
12	confirmed	identification	oncogene	localized	detected

Table E.5.: Twelve most probable terms of Topics 61 to 75.

	Topic 76	Topic 77	Topic 78	Topic 79	Topic 80
1	matrix	cell	protein	epithelial	evolution
2	extracellular	cdc	proteins	intestinal	species
3	collagen	division	ras	mammary	phylogenetic
4	tissue	mitotic	rho	epithelium	evolutionary
5	mmp	cycle	gtpase	gap	sequences
6	type	mitosis	rgs	gland	divergence
7	cartilage	spindle	signaling	junctions	related
8	basement	cells	activation	cells	origin
9	short	required	gtpbinding	intestine	molecular
10	laminin	segregation	gap	glands	evolved
11	chondrocytes	pcdc	effector	junction	lineages
12	sox	checkpoint	gtp	cell	analyses

	Topic 81	Topic 82	Topic 83	Topic 84	Topic 85
1	base	loss	mutation	estrogen	kinase
2	pairs	resulted	mutations	splicing	activation
3	dna	partial	gene	nuclear	protein
4	duplex	complete	phenotype	progesterone	pathway
5	pair	lost	normal	pr	mitogenactivated
6	sequence	observed	mutant	er	pi
7	strand	result	allele	estradiol	signaling
8	bases	imprinting	syndrome	sr	activated
9	strands	contrast	patients	premrna	kinases
10	complementary	maternal	dominant	steroid	mapk
11	minor	imprinted	autosomal	sc	map
12	groove	bc	defect	ribonucleoprotein	erk

	Topic 86	Topic 87	Topic 88	Topic 89	Topic 90
1	combined	protein	mammalian	domain	rats
2	ring	fkbp	elegans	domains	animals
3	ku	calcineurin	caenorhabditis	cterminal	treatment
4	scid	fk	mammals	nterminal	administration
5	severe	pa	similar	region	days
6	rings	cypa	function	cytoplasmic	injection
7	ftsz	rapamycin	conserved	terminus	treated
8	indicate	cyclosporin	nematode	contains	dose
9	data	cyp	found	transmembrane	weeks
10	results	csa	identified	residues	control
11	protein	target	homolog	tail	levels
12	composed	cyclophilin	genetic	aa	day

Table E.6.: Twelve most probable terms of Topics 76 to 90.

	Topic 91	Topic 92	Topic 93	Topic 94	Topic 95
1	prostate	chromosome	serum	rate	muscle
2	ar	chromosomal	bovine	rates	skeletal
3	human	gene	albumin	kinetics	myosin
4	erbb	chromosomes	lif	constant	smooth
5	androgen	region	dg	slow	fibers
6	cancer	kb	study	kinetic	muscles
7	dp	hybridization	cattle	dissociation	contraction
8	results	located	experiments	fast	dystrophin
9	demonstrate	situ	sheep	time	force
10	prostatic	genomic	addition	rapid	actin
11	neu	human	bsa	faster	achr
12	data	locus	related	times	fiber

	Topic 96	Topic 97	Topic 98	Topic 99	Topic 100
1	synaptic	mitochondrial	il	size	volume
2	glutamate	mitochondria	interleukin	selection	pressure
3	neurons	ga	cytokine	evolution	hydroxylase
4	hippocampal	mtdna	factor	fitness	blood
5	postsynaptic	potential	production	social	anp
6	synapses	respiration	necrosis	species	salt
7	longterm	isolated	cytokines	behavior	results
8	potentiation	import	tumor	competition	phenylalanine
9	nmda	respiratory	tnf	evolutionary	natriuretic
10	excitatory	suggesting	ilr	traits	osmotic
11	ltp	cytosolic	cells	reproductive	nacl
12	presynaptic	dwarf	inflammatory	population	contrast

	Topic 101	Topic 102	Topic 103	Topic 104	Topic 105
1	dna	drosophila	glutathione	cells	interaction
2	topoisomerase	melanogaster	gsh	migration	interactions
3	bending	pax	lysozyme	cerebellar	protein
4	circular	development	stransferase	granule	proteins
5	bend	flies	ascorbate	cell	interact
6	doublestranded	eye	exposed	purkinje	interacts
7	topo	insect	levels	zone	binding
8	sequencespecific	larvae	hel	cerebellum	proteinprotein
9	supercoiled	insects	highly	layer	twohybrid
10	dnas	larval	nrf	adult	using
11	top	body	found	parallel	vitro
12	containing	fly	gst	pacap	yeast

Table E.7.: Twelve most probable terms of Topics 91 to 105.

	Topic 106	Topic 107	Topic 108	Topic 109	Topic 110
1	suggests	region	presence	treatment	nmr
2	result	regions	absence	agents	spectroscopy
3	observed	segment	results	effective	solution
4	study	segments	observed	therapeutic	resonance
5	switch	located	addition	therapy	magnetic
6	indicates	sequence	due	drug	using
7	observation	coding	experiments	drugs	chemical
8	finding	corresponding	indicate	agent	spectra
9	process	conserved	dependent	toxicity	experiments
10	analysis	contains	indicating	effects	observed
11	demonstrates	identified	contrast	clinical	obtained
12	occur	homologous	examined	potential	spin

	Topic 111	Topic 112	Topic 113	Topic 114	Topic 115
1	water	expression	differences	lung	gene
2	energy	gene	pattern	fluid	transfer
3	free	reporter	patterns	pulmonary	vector
4	hydrogen	promoter	distribution	mdm	vectors
5	transition	construct	observed	csf	expression
6	interactions	cat	similar	lungs	delivery
7	kcalmol	activity	distinct	airway	therapy
8	temperature	constructs	data	alveolar	recombinant
9	hydrophobic	luciferase	individual	associated	vivo
10	solvent	genes	differential	respiratory	transduced
11	molecular	acetyltransferase	relative	cerebrospinal	adenovirus
12	phase	containing	heterogeneity	lc	efficient

	Topic 116	Topic 117	Topic 118	Topic 119	Topic 120
1	enzyme	cells	cox	protease	insulin
2	enzymes	cell	prostaglandin	inhibitor	secretion
3	activity	natural	mediator	inhibitors	pancreatic
4	ec	nk	cyclooxygenase	serine	cells
5	purified	target	synthesis	proteases	irs
6	activities	cytotoxic	pge	proteinase	islets
7	enzymatic	killer	pg	activity	islet
8	catalytic	lymphocytes	induced	trypsin	rat
9	active	cytotoxicity	antiinflammatory	cysteine	pancreas
10	substrate	lysis	production	calpain	suggest
11	hydrolase	effector	inflammation	suggesting	cck
12	properties	inhibitory	ep	active	glucose

Table E.8.: Twelve most probable terms of Topics 106 to 120.

	Topic 121	Topic 122	Topic 123	Topic 124	Topic 125
1	substrate	results	virus	ca	methylation
2	catalytic	pig	viral	release	dna
3	active	brca	infection	cai	cpg
4	site	guinea	viruses	channels	silencing
5	enzyme	cocaine	infected	influx	methylated
6	substrates	suggest	host	intracellular	methyltransferase
7	catalysis	found	replication	calmodulin	gene
8	enzymes	pigs	herpes	calcium	cg
9	mechanism	major	hepatitis	concentration	xist
10	phosphate	similar	genome	entry	sites
11	hydrolysis	previously	simplex	cadependent	demethylation
12	bond	properties	vp	cytosolic	inactivation

	Topic 126	Topic 127	Topic 128	Topic 129	Topic 130
1	sigma	mechanism	gel	studies	assembly
2	subtilis	action	electrophoresis	previous	core
3	bacillus	mechanisms	mobility	results	electron
4	tryptophan	via	hb	demonstrated	particles
5	trp	process	shift	shown	microscopy
6	biotin	mediated	hemoglobin	indicate	particle
7	trap	involving	gradient	study	nm
8	presence	occurs	sucrose	experiments	microscopic
9	sporulation	novel	analysis	recent	diameter
10	form	involves	electrophoretic	demonstrate	contain
11	dependent	occur	twodimensional	indicated	revealed
12	found	remains	bands	suggested	assemble

	Topic 131	Topic 132	Topic 133	Topic 134	Topic 135
1	response	negative	signal	human	influenza
2	induction	positive	signaling	murine	results
3	responses	dominant	pathway	humans	using
4	induced	act	transduction	report	ha
5	stress	regulator	pathways	similar	ab
6	induce	results	signals	mouse	cam
7	exposure	feedback	via	rodent	found
8	stimuli	directly	response	speciesspecific	shown
9	mediated	acts	involved	demonstrated	studies
10	mediate	loop	events	potential	suggest
11	increased	appears	intracellular	functional	data
12	increase	suggest	downstream	various	ns

Table E.9.: Twelve most probable terms of Topics 121 to 135.

	Topic 136	Topic 137	Topic 138	Topic 139	Topic 140
1	ph	resistance	structure	family	lipid
2	proton	resistant	crystal	related	membranes
3	exchange	sensitivity	resolution	conserved	liposomes
4	nah	drug	structural	identified	membrane
5	bacteriorhodopsin	sensitive	xray	proteins	lipids
6	br	mdr	structures	novel	phospholipid
7	exchanger	drugs	determined	highly	bilayer
8	phi	confers	conformation	closely	phospholipids
9	base	susceptible	residues	putative	ceramide
10	pump	pgp	threedimensional	sequence	bilayers
11	nhe	confer	molecule	expressed	cationic
12	protons	multidrug	interactions	superfamily	phosphatidylcholine

	Topic 141	Topic 142	Topic 143	Topic 144	Topic 145
1	mrna	dehydrogenase	amp	macrophages	common
2	translation	alcohol	actin	factor	structural
3	mrnas	ethanol	sci	macrophage	features
4	initiation	nad	cytoskeleton	igfi	similar
5	protein	results	filaments	growth	unique
6	translational	coenzyme	al	insulinlike	properties
7	polya	adh	acad	role	organization
8	untranslated	pyruvate	usa	igf	share
9	region	found	proc	igfii	including
10	factor	nadh	natl	igfbp	unusual
11	cap	ec	polymerization	murine	feature
12	eukaryotic	oxidoreductase	nature	peritoneal	characteristic

	Topic 146	Topic 147	Topic 148	Topic 149	Topic 150
1	suppression	vitamin	tcell	role	movement
2	suppressed	results	tcr	play	microtubules
3	multiple	accessory	cells	plays	motor
4	experimental	suggest	antigen	critical	microtubule
5	ms	carboxylase	receptor	roles	kinesin
6	myelin	avp	clones	key	tubulin
7	basic	ohd	tolerance	suggest	taxol
8	suppress	rat	repertoire	results	dynein
9	protein	oc	zeta	regulating	cytoplasmic
10	mbp	dihydroxyvitamin	specific	crucial	mts
11	sclerosis	va	betachain	demonstrate	rotation
12	autoimmune	demonstrate	variable	central	squid

Table E.10.: Twelve most probable terms of Topics 136 to 150.

	Topic 151	Topic 152	Topic 153	Topic 154	Topic 155
1	epsteinbarr	clones	concentrations	recognition	yeast
2	lymphoma	map	concentration	specificity	cerevisiae
3	ebv	genome	nm	specific	saccharomyces
4	latent	chromosome	low	recognize	protein
5	ebna	artificial	mm	determinants	essential
6	virus	mapping	microm	lens	gene
7	expression	physical	m	determinant	required
8	lmp	kb	values	ag	homolog
9	latency	mb	ic	specificities	growth
10	herpesvirus	yeast	respectively	whereas	function
11	hhv	genomic	value	recognized	pombe
12	lymphomas	cloning	pm	determined	conserved

	Topic 156	Topic 157	Topic 158	Topic 159	Topic 160
1	mutant	component	transport	surface	er
2	wildtype	components	uptake	molecules	endoplasmic
3	mutants	major	transporter	surfaces	golgi
4	defective	complement	hamster	beads	reticulum
5	mutation	found	transporters	polymer	proteins
6	mutations	minor	ovary	directly	glycosylation
7	phenotype	data	chinese	polymers	apparatus
8	essential	principal	cho	contact	transport
9	required	shown	rc	molecule	oligosaccharides
10	mutated	independent	membrane	adsorption	lectin
11	unable	findings	highaffinity	onto	srp
12	lacking	indicating	transported	properties	carbohydrate

	Topic 161	Topic 162	Topic 163	Topic 164	Topic 165
1	recombinant	strains	growth	factor	form
2	purified	strain	factor	platelets	forms
3	vitro	tuberculosis	tgfbeta	platelet	distinct
4	extracts	isolates	transforming	thrombin	found
5	soluble	mycobacterium	tgf	tf	whereas
6	system	infection	smad	plasminogen	respectively
7	using	pathogenic	egr	activator	suggesting
8	insect	virulence	beta	ix	short
9	extract	mycobacterial	superfamily	fibrinogen	studied
10	reconstituted	isolate	previously	coagulation	appears
11	sf	virulent	activin	par	contrast
12	prepared	bcg	suggest	human	additional

Table E.11.: Twelve most probable terms of Topics 151 to 165.

	Topic 166	Topic 167	Topic 168	Topic 169	Topic 170
1	toxin	body	fold	monkeys	structure
2	cyclase	leptin	approximately	humans	structures
3	coupling	fat	gt	primates	secondary
4	adenylyl	tissue	respectively	primate	structural
5	pertussis	weight	similar	world	predicted
6	gs	adipose	compared	evidence	tertiary
7	guanine	obesity	lower	african	threedimensional
8	gi	cebp	determined	late	interactions
9	ac	adipocytes	single	modern	local
10	coupled	obese	times	species	hairpin
11	adenylate	tl	relative	middle	elements
12	nucleotidebinding	food	nearly	rhesus	primary

	Topic 171	Topic 172	Topic 173	Topic 174	Topic 175
1	host	heat	release	endothelial	molecular
2	bacterial	shock	hormone	vascular	basis
3	bacteria	hsp	pituitary	cells	provide
4	infection	temperature	gh	hypoxia	understanding
5	aureus	degrees	lhrh	vegf	mechanisms
6	defense	chaperone	lh	blood	weight
7	antimicrobial	temperatures	gnrh	angiogenesis	study
8	killing	protein	secretion	vessels	provides
9	antibiotic	dnak	prolactin	smooth	studies
10	bacterium	proteins	effect	endothelium	biochemical
11	microbial	stress	released	flow	insights
12	gramnegative	cold	luteinizing	arterial	low

	Topic 176	Topic 177	Topic 178	Topic 179	Topic 180
1	mouse	stable	potential	injury	dna
2	embryonic	stability	inactivation	protection	replication
3	embryos	results	results	ischemia	polymerase
4	development	stabilization	suggest	damage	pol
5	stem	indicate	found	ischemic	origin
6	embryo	unstable	produced	cerebral	synthesis
7	day	addition	addition	recovery	strand
8	es	instability	indicate	protects	initiation
9	stage	due	similar	effects	helicase
10	germ	halflife	inactivated	protected	singlestranded
11	detected	extremely	escape	neuronal	template
12	yolk	properties	mv	reperfusion	required

Table E.12.: Twelve most probable terms of Topics 166 to 180.

	Topic 181	Topic 182	Topic 183	Topic 184	Topic 185
1	regulation	cortex	murine	target	exon
2	regulatory	brain	chicken	antisense	gene
3	regulated	subjects	integration	targeted	intron
4	control	functional	avian	targeting	alternative
5	expression	regions	retroviral	oligonucleotides	splicing
6	regulate	task	retrovirus	targets	exons
7	regulates	imaging	integrase	oligonucleotide	introns
8	controlled	flow	retroviruses	sense	transcripts
9	involved	prefrontal	leukemia	specific	splice
10	level	temporal	nc	oligodeoxynucleotides	reading
11	suggest	left	virus	vitro	spliced
12	regulator	tomography	previously	manner	coding

	Topic 186	Topic 187	Topic 188	Topic 189	Topic 190
1	cells	ability	cleavage	chloroplast	data
2	bone	results	processing	light	method
3	hematopoietic	able	precursor	photosynthetic	using
4	marrow	induce	cleaved	reaction	methods
5	stem	capacity	proteolytic	center	set
6	cell	demonstrate	mature	chlorophyll	based
7	progenitors	suggest	processed	photosystem	analysis
8	progenitor	promote	site	electron	information
9	gata	direct	cleaves	centers	statistical
10	myeloid	directly	precursors	chloroplasts	average
11	factor	capable	form	photosynthesis	algorithm
12	murine	sufficient	protease	ii	estimate

	Topic 191	Topic 192	Topic 193	Topic 194	Topic 195
1	kidney	effects	reduced	increase	antibodies
2	renal	effect	reduction	increased	antibody
3	apical	enhanced	decrease	fold	monoclonal
4	water	enhancement	significantly	increases	antigen
5	proximal	results	decreased	levels	mab
6	epithelial	enhance	effect	decreased	epitope
7	basolateral	modulation	results	decrease	human
8	cells	whereas	significant	effect	mabs
9	tubules	examined	reduces	treatment	specific
10	aqp	ct	reduce	increasing	igg
11	membrane	influence	caused	observed	epitopes
12	tubular	observed	indicate	level	directed

Table E.13.: Twelve most probable terms of Topics 181 to 195.

	Topic 196	Topic 197	Topic 198	Topic 199	Topic 200
1	ap	range	complex	time	stimulation
2	cfos	variety	complexes	hr	activation
3	sulfate	wide	formation	min	pkc
4	cjun	including	cp	rapid	phorbol
5	jun	found	assembly	exposure	stimulated
6	heparin	types	formed	initial	protein
7	fos	broad	form	period	acetate
8	furthermore	diverse	stable	followed	kinase
9	results	various	ternary	rapidly	ester
10	heparan	rap	components	course	induced
11	suggest	shown	interactions	occurred	myristate
12	involved	examined	bound	delayed	treatment

	Topic 201	Topic 202	Topic 203	Topic 204	Topic 205
1	kda	level	cells	growth	xenopus
2	protein	correlation	differentiation	factor	oocytes
3	purified	adenosine	cell	factors	laevis
4	molecular	editing	proliferation	epidermal	injected
5	polypeptide	observed	expression	egf	oocyte
6	mass	extent	differentiated	fibroblast	maturation
7	purification	correlated	induced	fgf	frog
8	chromatography	found	terminal	proliferation	injection
9	approximately	degree	differentiate	egfr	eggs
10	identified	relationship	induces	pdgf	expressed
11	antibodies	significant	hl	cells	npc
12	apparent	strong	proliferative	bfgf	egg

	Topic 206	Topic 207	Topic 208	Topic 209	Topic 210
1	evidence	leukemia	kinase	membrane	protein
2	provide	acute	phosphorylation	plasma	proteins
3	direct	translocation	protein	membranes	function
4	studies	fusion	phosphorylated	translocation	report
5	data	myeloid	kinases	protein	binds
6	provides	gene	activity	outer	rnabinding
7	directly	human	ser	integral	encoded
8	strong	leukemic	pka	inner	consistent
9	results	mll	serine	cytoplasmic	properties
10	provided	associated	vitro	transmembrane	preferentially
11	recent	patients	catalytic	soluble	alternative
12	providing	chromosomal	substrate	proteins	associates

Table E.14.: Twelve most probable terms of Topics 196 to 210.

	Topic 211	Topic 212	Topic 213	Topic 214	Topic 215
1	type	diabetes	cluster	mechanisms	beta
2	types	diabetic	iron	understood	gamma
3	wild	nod	clusters	poorly	alpha
4	hpv	gad	fes	involved	tgf
5	papillomavirus	mellitus	clustering	suggest	whereas
6	similar	autoimmune	ferritin	process	chain
7	contrast	islet	protein	underlying	respectively
8	major	associated	transferrin	remain	role
9	lack	islets	hfe	mechanism	seen
10	ee	involved	clustered	aba	ark
11	suggesting	rhizobium	aconitase	events	furthermore
12	iia	nodules	irp	results	studied

	Topic 216	Topic 217	Topic 218	Topic 219	Topic 220
1	receptor	ht	degradation	amino	cell
2	receptors	neutrophils	telomerase	acid	lines
3	ligand	serotonin	telomere	acids	line
4	ligands	leukocyte	proteasome	sequence	cells
5	extracellular	leukocytes	pathway	residues	types
6	transmembrane	human	ubiquitin	nucleic	derived
7	proteincoupled	monocytes	ligase	substitutions	various
8	signaling	neutrophil	telomeres	five	express
9	binding	inflammatory	vivo	carboxylterminal	established
10	tm	recruitment	telomeric	aromatic	neuroblastoma
11	expressed	inflammation	ubiquitination	substitution	expresses
12	functional	mcp	degraded	glutamic	contribute

	Topic 221	Topic 222	Topic 223	Topic 224	Topic 225
1	tumor	channel	peptide	atp	theory
2	tumors	channels	peptides	hydrolysis	time
3	cells	current	synthetic	atpase	space
4	growth	na	residues	presence	random
5	melanoma	currents	corresponding	arf	shape
6	human	ion	derived	nucleotide	local
7	mice	sodium	sequence	adp	size
8	cancer	conductance	synthesized	gtp	linear
9	metastasis	potassium	containing	binding	systems
10	nude	voltage	using	activity	distribution
11	vivo	membrane	gif	atpdependent	probability
12	metastatic	pore	neuropeptide	nucleotides	terms

Table E.15.: Twelve most probable terms of Topics 211 to 225.

	Topic 226	Topic 227	Topic 228	Topic 229	Topic 230
1	residues	circadian	system	hair	cortical
2	residue	oscillations	central	mechanical	cortex
3	position	clock	nervous	frequency	neurons
4	positions	hz	neuronal	auditory	lateral
5	amino	time	neurons	adaptation	nucleus
6	conserved	frequency	brain	ear	layer
7	substitution	period	astrocytes	inner	layers
8	mutagenesis	firing	glial	process	sensory
9	acid	sleep	cns	bundle	thalamic
10	arginine	network	neural	bundles	nuclei
11	lysine	temporal	developing	sound	organization
12	alanine	light	suggest	cochlear	somatosensory

	Topic 231	Topic 232	Topic 233	Topic 234	Topic 235
1	fusion	plants	memory	mutations	th
2	protein	plant	learning	mutation	song
3	exocytosis	arabidopsis	behavior	frequency	pylori
4	secretory	tobacco	behavioral	somatic	gastric
5	vesicle	thaliana	spatial	substitutions	pthrp
6	release	leaves	hippocampal	mutational	parathyroid
7	granules	tomato	longterm	spontaneous	birds
8	proteins	transgenic	hippocampus	mutants	juvenile
9	vesicles	leaf	rats	wildtype	production
10	fusions	defense	training	hot	hormone
11	synaptic	seed	task	affect	zebra
12	snap	pollen	performance	genetic	helicobacter

	Topic 236	Topic 237	Topic 238	Topic 239	Topic 240
1	age	locus	function	adhesion	fluorescent
2	aging	genetic	functions	cell	fluorescence
3	life	loci	unknown	molecule	microscopy
4	months	linkage	role	integrin	green
5	decline	markers	functional	molecules	force
6	senescence	susceptibility	suggest	icam	imaging
7	significantly	analysis	results	integrins	single
8	diet	linked	remains	ncam	using
9	aged	chromosome	andor	intercellular	molecules
10	risk	alleles	report	cellcell	living
11	study	genes	recently	fibronectin	gfp
12	dietary	mapping	suggests	surface	dye

Table E.16.: Twelve most probable terms of Topics 226 to 240.

	Topic 241	Topic 242	Topic 243	Topic 244	Topic 245
1	gene	disease	model	rna	hcv
2	product	patients	models	polymerase	eg
3	expression	clinical	based	transcription	ie
4	genes	syndrome	experimental	rnas	red
5	encodes	diseases	data	elongation	hepatitis
6	encoding	associated	proposed	ribozyme	study
7	products	chronic	simple	vitro	gpi
8	encoded	severe	dynamics	transcripts	obtained
9	expressed	pathogenesis	results	transcript	evidence
10	coding	disorder	explain	termination	found
11	cloned	autoimmune	predicted	nucleotides	major
12	carrying	disorders	modeling	template	revealed

	Topic 246	Topic 247	Topic 248	Topic 249	Topic 250
1	histone	cancer	cdna	phospholipase	proteins
2	chromatin	tumor	sequence	inositol	protein
3	nucleosome	tumors	amino	phosphate	family
4	histones	human	encoding	phosphatidylinositol	involved
5	hdac	breast	cloning	ip	encoded
6	linker	carcinoma	protein	trisphosphate	biochemical
7	acetylation	suppressor	isolated	plc	components
8	deacetylase	normal	acid	insp	shown
9	core	cancers	cloned	activity	function
10	nucleosomes	colon	clone	specific	bind
11	structure	loss	library	levels	interact
12	sir	carcinomas	encodes	plcgamma	roles

	Topic 251	Topic 252	Topic 253	Topic 254	Topic 255
1	intracellular	hypothesis	cells	cells	pcr
2	calcium	support	culture	transfected	dna
3	extracellular	test	cultures	expressing	amplification
4	increase	results	medium	expression	sequences
5	free	data	cell	expressed	using
6	concentration	consistent	fibroblasts	human	amplified
7	accumulation	et	cultured	cdna	genomic
8	mobilization	tested	secreted	cos	sequencing
9	activation	proposed	vitro	cell	primers
10	rapid	supports	grown	stably	reaction
11	induced	evidence	production	transfection	polymerase
12	cytosolic	idea	conditions	hela	chain

Table E.17.: Twelve most probable terms of Topics 241 to 255.

	Topic 256	Topic 257	Topic 258	Topic 259	Topic 260
1	recombination	isoform	domain	inhibition	eukaryotic
2	homologous	isoforms	domains	inhibited	eukaryotes
3	dna	protein	sh	effect	organisms
4	meiotic	prion	homology	inhibit	genomes
5	reca	prp	proteins	inhibits	genome
6	sitespecific	conversion	src	inhibitory	archaea
7	events	prpsc	protein	inhibitor	archaeal
8	breaks	prpc	containing	effects	prokaryotic
9	exchange	scrapie	motif	activity	complete
10	repair	cellular	binding	inhibitors	bacteria
11	doublestrand	studies	found	block	prokaryotes
12	strand	diseases	prolinerich	blocked	bacterial

	Topic 261	Topic 262	Topic 263	Topic 264	Topic 265
1	dna	ps	amyloid	functional	subunit
2	damage	arthritis	disease	distinct	alpha
3	repair	notch	ad	properties	subunits
4	radiation	found	alzheimer	functionally	nakatpase
5	uv	presenilin	tau	classes	composed
6	rad	rheumatoid	app	defined	heterodimer
7	irradiation	suggest	a	structural	ouabain
8	cells	results	protein	functions	encoding
9	exposure	function	precursor	structurally	previously
10	ionizing	synovial	plaques	results	addition
11	atm	associated	alzheimers	related	heterodimeric
12	light	disease	fibrils	equivalent	consisting

	Topic 266	Topic 267	Topic 268	Topic 269	Topic 270
1	cardiac	class	dimer	modification	folding
2	heart	major	crosslinking	terminus	protein
3	myocytes	mhc	form	modified	native
4	ventricular	molecules	dimerization	carboxyl	unfolding
5	hypertrophy	ii	dimers	protein	transition
6	failure	complex	monomer	proteins	structure
7	rat	histocompatibility	monomeric	modifications	proteins
8	hearts	peptide	dimeric	termini	intermediate
9	ne	antigen	monomers	posttranslational	folded
10	myocardial	peptides	crosslinked	cysteine	energy
11	function	molecule	forms	covalently	groel
12	hypertension	presentation	tetramer	covalent	unfolded

Table E.18.: Twelve most probable terms of Topics 256 to 270.

	Topic 271	Topic 272	Topic 273	Topic 274	Topic 275
1	residues	retinal	zinc	specific	system
2	loop	light	chimeric	using	systems
3	helix	rhodopsin	finger	highly	model
4	arg	retina	containing	selective	using
5	asp	rod	domains	demonstrated	study
6	glu	photoreceptor	chimeras	demonstrate	useful
7	residue	photoreceptors	fingers	specifically	developed
8	lys	degeneration	proteins	selectively	studying
9	ala	cone	chimera	various	module
10	position	pigment	domain	results	provide
11	tyr	blue	protein	established	provides
12	ser	visual	constructed	sensitive	modules

	Topic 276	Topic 277	Topic 278	Topic 279	Topic 280
1	metal	species	vesicles	acid	cycle
2	ions	vertebrate	membrane	fatty	cell
3	mg	fish	cells	acids	phase
4	ion	vertebrates	intracellular	aa	cyclin
5	copper	found	endocytosis	arachidonic	arrest
6	mt	zebrafish	compartment	lipoxygenase	rb
7	zn	related	trafficking	fa	ef
8	divalent	mammals	rab	acyl	cdk
9	cations	suggests	compartments	lo	progression
10	ligands	invertebrate	localized	desaturase	cells
11	coordination	conservation	surface	polyunsaturated	cyclindependent
12	metallothionein	evolution	cell	products	retinoblastoma

	Topic 281	Topic 282	Topic 283	Topic 284	Topic 285
1	associated	conditions	cytochrome	wall	changes
2	association	nitrogen	oxidase	fungus	change
3	found	low	heme	fungal	conformational
4	involved	physiological	co	fungi	altered
5	related	carbon	electron	albicans	alterations
6	andor	source	redox	cellulose	significant
7	significant	fixation	enzyme	involved	associated
8	suggest	environment	nadph	chitin	causes
9	data	nitrate	reduction	analysis	conformation
10	associate	nh	transfer	found	structural
11	evidence	environmental	oxygen	candida	cr
12	associations	starvation	reduced	filamentous	morphological

Table E.19.: Twelve most probable terms of Topics 271 to 285.

	Topic 286	Topic 287	Topic 288	Topic 289	Topic 290
1	reaction	tyrosine	lt	biosynthesis	development
2	step	kinase	control	synthase	adult
3	intermediate	phosphorylation	compared	pathway	fetal
4	reactions	receptor	significantly	enzyme	developmental
5	mechanism	kinases	levels	enzymes	stages
6	formation	signaling	controls	biosynthetic	day
7	chemical	activation	increased	synthesis	stage
8	steps	phosphorylated	vs	acc	developing
9	products	proteintyrosine	significant	involved	postnatal
10	product	jak	ratio	step	neonatal
11	initial	btk	lower	catalyzes	days
12	molecule	stimulation	elevated	product	life

	Topic 291	Topic 292	Topic 293	Topic 294	Topic 295
1	plasmid	expressed	five	cl	cells
2	transfer	tissues	found	cftr	cell
3	gene	expression	six	hox	normal
4	plasmids	tissue	occurring	chloride	cultured
5	transformation	tissuespecific	seven	fibrosis	express
6	dna	distinct	naturally	regulator	types
7	containing	restricted	eight	cystic	treatment
8	integration	identified	nine	homeodomain	contrast
9	tdna	novel	tested	cf	intact
10	agrobacterium	mouse	contained	conductance	secrete
11	marker	unique	determined	transmembrane	examined
12	transformants	tag	specific	homeobox	exposure

	Topic 296	Topic 297	Topic 298	Topic 299	Topic 300
1	glucose	formation	required	falciparum	aggregation
2	metabolic	disulfide	essential	plasmodium	expansion
3	metabolism	bond	sufficient	malaria	hd
4	glut	cys	requires	parasite	aggregates
5	flux	cysteine	requirement	erythrocytes	formation
6	glycogen	form	function	erythrocyte	expanded
7	control	bonds	minimal	invasion	containing
8	rate	formed	efficient	parasites	gd
9	transporter	cysteines	require	pe	results
10	glucokinase	sulfur	demonstrate	mosquito	study
11	energy	function	results	midgut	caused
12	insulin	shown	requirements	surface	polyglutamine

Table E.20.: Twelve most probable terms of Topics 286 to 300.

Bibliography

E. Airoldi, D. Blei, S. Fienberg, and E. Xing. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9:1981–2014, 2008.

M. Alexander and C. Wolff. Einleitung: Perspektiven und Positionen des Text Mining. *Zeitschrift für Computerlinguistik und Sprachtechnologie*, 20(1):1–18, 2005.

A. A. Alsheikh-Ali, W. Qureshi, M. H. Al-Mallah, and J. P. A. Ioannidis. Public availability of published research data in high-impact journals. *PLoS ONE*, 6(9):e24357, Jan. 2011.

D. Ascher, A. Martelli, and A. Ravenscroft. *Python Cookbook*. O'Reilly, second edition, 2005.

A. Asuncion, M. Welling, P. Smyth, and Y. Teh. On smoothing and inference for topic models. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, pages 27–34, 2009.

L. Azzopardi, M. Girolami, and C. van Rijsbergen. Topic based language models for ad hoc information retrieval. In *2004 IEEE International Joint Conference on Neural Networks*, pages 3281–3286. IEEE, 2004.

R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, first edition, 1999.

N. Balakrishnan and V. Nevzorov. *A Primer on Statistical Distributions*. Wiley-Interscience, 2003.

D. J. Barrett and R. E. Silverman. *SSH, the Secure Shell - The Definitive Guide*. O'Reilly, first edition, 2001.

M. W. Berry and J. Kogan. *Text Mining*. Wiley, first edition, 2010.

D. Blei and J. Lafferty. Dynamic topic models. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 113–120. ACM, 2006.

D. M. Blei. Topic models. Video lecture, 2009. URL `http://videolectures.net/mlss09uk_blei_tm/`. (accessed on: 2011-07-03).

D. M. Blei. Introduction to probabilistic topic models, 2011. URL `http://www.cs.princeton.edu/~blei/papers/Blei2011.pdf`. (accessed on: 2012-05-04).

D. M. Blei and J. D. Lafferty. A correlated topic model of Science. *Annals of Applied Statistics*, 1(1): 17–35, 2007.

D. M. Blei and J. D. Lafferty. Topic models. In A. Sahami and M. Srivastava, editors, *Text Mining: Theory and Applications*, pages 71–93. Taylor and Francis, 2009.

D. M. Blei and J. D. McAuliffe. Supervised topic models. *Advances in Neural Information Processing Systems*, 21:121–128, 2007.

D. M. Blei, A. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

W. Buntine. Estimating likelihoods for topic models. In Z.-H. Zhou and T. Washio, editors, *Advances in Machine Learning*, volume 5828 of *Lecture Notes in Computer Science*, pages 51–64. Springer Berlin / Heidelberg, 2009.

W. Buntine and A. Jakulin. Applying discrete PCA in data analysis. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 59–66. AUAI Press Arlington, Virginia, United States, 2004.

K. Burnham and D. Anderson. *Model Selection and Multimodel Inference: a Practical Information-theoretic Approach*. Springer-Verlag, second edition, 2002.

B. Carpenter. Integrating out multinomial parameters in latent Dirichlet allocation and naive Bayes for collapsed Gibbs sampling, 2010. URL `http://lingpipe.files.wordpress.com/2010/07/lda1.pdf`. (accessed on: 2011-11-11).

H.-A. Chang and C.-H. Yu. Latent Dirichlet allocation. Seminar paper, 2007. URL `http://cassava.fudan.edu.cn/seminars/ppt/lecture-lda.pdf`. (accessed on: 2011-11-11).

J. Chang. lda: Collapsed Gibbs sampling methods for topic models. R package version 1.3.1, 2011. URL `http://cran.r-project.org/package=lda`.

J. Chang and D. Blei. Relational topic models for document networks. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, pages 81–88, 2009.

S. Chib. Marginal likelihood from the Gibbs output. *Journal of the American Statistical Association*, 90:1313–1321, 1995.

J.-T. Chien and M.-S. Wu. Adaptive Bayesian latent semantic analysis. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(1):198–207, 2008.

P. S. Cowpertwait and A. V. Metcalfe. *Introductory Time Series with R*. Springer, 2009.

D. B. Dahl. xtable: Export tables to LaTeX or HTML. R package version 1.7-0, 2009. URL `http://cran.r-project.org/package=xtable`.

S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

P. Del Moral, A. Doucet, and A. Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B*, 68(3):411–436, 2006.

C. Elkan. Text mining and topic models. Lecture notes, 2010. URL `http://cseweb.ucsd.edu/~elkan/250B/topicmodels.pdf`. (accessed on: 2011-11-11).

E. Erosheva, S. Fienberg, and J. Lafferty. Mixed-membership models of scientific publications. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, 101(Supplement 1):5220–5227, Apr. 2004.

W. Fan, L. Wallace, S. Rich, and Z. Zhang. Tapping the power of text mining. *Communications of the Association for Computing Machinery (ACM)*, 49(9):82, 2006.

I. Feinerer. *A Text Mining Framework in R and Its Applications*. PhD thesis, Vienna University of Economics and Business, 2008.

E. A. Fox and S. Sharat. A comparison of two methods for soft Boolean interpretation in information retrieval. Technical report, Virginia Tech, Dept. of Computer Science, 1986.

A. Gelman, J. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*. CRC Press, second edition, 2004.

S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

M. Girolami and A. Kabán. On an equivalence between PLSI and LDA. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 434, 2003.

T. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, 101(Supplement 1):5228–5235, Apr. 2004.

T. Griffiths, M. Steyvers, D. Blei, and J. Tenenbaum. Integrating topics and syntax. *Advances in Neural Information Processing Systems*, 17:537–544, 2005.

B. Grün and K. Hornik. topicmodels: an R package for fitting topic models. *Journal of Statistical Software*, 40(13), 2011.

M. Hearst. Untangling text data mining. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 3–10, 1999.

G. Heinrich. Parameter estimation for text analysis. Technical report, 2005. URL `http://www.arbylon.net/publications/text-est2.pdf`. (accessed on: 2011-11-11).

T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57, New York, NY, USA, 1999.

D. C. Ince, L. Hatton, and J. Graham-Cumming. The case for open computer programs. *Nature*, (482): 485–488, 2012.

T. Iwata, K. Saito, N. Ueda, S. Stromsten, T. Griffiths, and J. Tenenbaum. Parametric embedding for class visualization. *Neural Computation*, 19(9):2536–2556, 2007.

M. I. Jordan. Graphical models. *Statistical Science*, 19(1):140–155, Feb. 2004.

D. Jurafsky and J. H. Martin. *Speech and Language Processing*. Prentice Hall, 1999.

R. Kass and A. Raftery. Bayes factors. *Journal of the American Statistical Association*, 90(430):773–795, 1995.

U. Kruschwitz. *Intelligent Document Retrieval: Exploiting Markup Structure*. Springer, 2005.

D. Kuropka. Modelle zur Repräsentation natürlichsprachlicher Dokumente - Information-Filtering und -Retrieval mit relationalen Datenbanken. *Advances in Information Systems and Management Science*, (10), 2004.

W. Li and A. McCallum. Pachinko allocation: DAG-structured mixture models of topic correlations. In *Proceedings of the 23rd International Conference on Machine Learning.*, pages 577–584, 2006.

D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge UP, 4th edition, 2005.

C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, second edition, 1999.

C. D. Manning, P. Raghavan, and H. Schütze. *An Introduction to Information Retrieval*. Cambridge UP, online edition, 2009.

A. K. McCallum. MALLET: A Machine Learning for Language Toolkit, 2002. URL `http://mallet.cs.umass.edu`. (accessed on: 2011-11-11).

T. Minka and J. Lafferty. Expectation-propagation for the generative aspect model. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, pages 352–359, 2002.

I. Mukherjee and D. Blei. Relative performance guarantees for approximate inference in latent Dirichlet allocation. *Advances in Neural Information Processing Systems*, 21:1129–1136, 2009.

I. Murray and R. Salakhutdinov. Evaluating probabilities under high-dimensional latent variable models. *Advances in Neural Information Processing*, 21:1137–1144, 2009.

R. M. Neal. Annealed importance sampling. *Statistics and Computing*, 11:125–139, 2001.

R. E. Neapolitan. *Learning Bayesian Networks*. Prentice-Hall, 2003.

D. Newman, C. Chemudugunta, and P. Smyth. Statistical entity-topic models. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 680–686, 2006.

D. Newman, A. Asuncion, P. Smyth, and M. Welling. Distributed inference for latent Dirichlet allocation. *Advances in Neural Information Processing Systems*, 20:1081–1088, 2007.

D. Newman, A. Asuncion, P. Smyth, and M. Welling. Distributed algorithms for topic models. *Journal of Machine Learning Research*, 10:1801–1828, 2009.

M. A. Newton and A. E. Raftery. Approximate Bayesian inference with the weighted likelihood bootstrap. *Journal of the Royal Statistical Society Series B*, 56:3–48, 1994.

PNAS. Proceedings of the National Academy of Sciences of the United States of America, 2010. URL `http://www.pnas.org`. (accessed on: 2010-11-11).

R Development Core Team. Writing R extensions, 2010. URL `http://cran.r-project.org/doc/manuals/R-exts.pdf`. (accessed on: 2010-11-01).

R Development Core Team. R: A Language and Environment for Statistical Computing, 2011. URL `http://www.r-project.org`. (accessed on: 2011-11-11).

S. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27:129–146, 1976.

M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth. The author-topic model for authors and documents. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 487–494, 2004.

M. Rosen-Zvi, T. Griffiths, P. Smyth, and M. Steyvers. Learning author-topic models from text corpora. Technical report, 2005.

G. Salton, A. Wong, and C. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):620, 1975.

G. Salton, E. Fox, and H. Wu. Extended Boolean information retrieval. *Communications of the ACM*, 26:1022 – 1036, 1983.

D. Sarkar. *Lattice - Multivariate Data Visualization with R.* Springer, 2008.

D. Shen, J.-t. Sun, Q. Yang, and Z. Chen. Latent friend mining from blog data. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 552–561, Dec. 2006.

R. M. Shiffrin and K. Bo. Mapping knowledge domains. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, 101(Supplement 1):5183–5185, 2004.

A. Srivastava and M. Sahami, editors. *Text Mining: Classification, Clustering, and Applications*, volume 78. Chapman Hall /CRC, Apr. 2009.

M. Steyvers and T. Griffiths. Probabilistic topic models. In T. Landauer, D. Mcnamara, S. Dennis, and W. Kintsch, editors, *Latent Semantic Analysis: A Road to Meaning*, page 427. Lawrence Erlbaum, 2007.

M. Steyvers, P. Smyth, M. Rosen-Zvi, and T. Griffiths. Probabilistic author-topic models for information discovery. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '04)*, pages 306–315, 2004.

W. G. Stock. *Information Retrieval: Informationen suchen und finden.* Oldenbourg, 2007.

Y. Teh, D. Newman, and M. Welling. A collapsed variational Bayesian inference algorithm for latent Dirichlet allocation. *Advances in Neural Information Processing Systems*, 19:1353–1360, 2006a.

Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101:1566–1581, 2006b.

D. Temple Lang. RCurl: General network (HTTP/FTP/...) client interface for R. R package version 1.91-1, 2012. URL `http://cran.r-project.org/package=RCurl`.

I. Titov and R. McDonald. Modeling online reviews with multi-grain topic models. In *Proceedings of the 17th International Conference on World Wide Web (WWW '08)*, pages 111–120, New York, New York, USA, 2008. ACM Press.

C. J. van Rijsbergen. A non-classical logic for information retrieval. *The Computer Journal*, 29(6): 481–485, 1986.

W. N. Venables, D. M. Smith, and R Development Core Team. An Introduction to R, 2011. URL `http://cran.r-project.org/doc/manuals/R-intro.html`. (accessed on: 2011-11-11).

H. M. Wallach. Topic modeling: Beyond bag-of-words. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 977–984. ACM, 2006.

H. M. Wallach. *Structured topic models for language.* PhD thesis, University of Cambridge, 2008.

H. M. Wallach, I. Murray, R. Salakhutdinov, and D. Mimno. Evaluation methods for topic models. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09)*, pages 1105–1112, 2009.

X. Wang and A. McCallum. A note on topical n-grams. Technical report, 2005. URL `http://ciir.cs.umass.edu/pubfiles/ir-515.pdf`. (accessed on: 2011-11-11).

X. Wang and A. McCallum. Topics over time: a non-Markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 424–433, 2006.

Y. Wang. Distributed Gibbs sampling of latent topic models: The gritty details. Technical report, 2008. URL http://dbgroup.cs.tsinghua.edu.cn/wangyi/lda/index.html. (accessed on: 2010-11-01).

Y. Wang, P. Sabzmeydani, and G. Mori. Semi-latent Dirichlet allocation: A hierarchical model for human action recognition. *Lecture Notes in Computer Science*, 4814:240–254, 2007.

X. Wei and W. Croft. LDA-based document models for ad-hoc retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 178–185. ACM, 2006.

X. Wei and W. Croft. Investigating retrieval performance with manually-built topic models. In *Proceedings of Recherche d'Information Assistée par Ordinateur (RIAO)*, pages 333–349, 2007.

X. Wei, J. Sun, and X. Wang. Dynamic mixture models for multiple time series. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2909–2914, 2007.

S. M. Weiss, N. Indurkhya, T. Zhang, and F. J. Damerau. *Text Mining*. Springer, 2005.

M. Welling, Y. Teh, and B. Kappen. Hybrid variational/Gibbs collapsed inference in topic models. In *Proceedings of the Conference on Uncertainty in Artifical Intelligence (UAI)*, pages 587–594, 2008.

Wikipedia. Information Retrieval, 2011a. URL http://en.wikipedia.org/wiki/Information_retrieval. (accessed on: 2011-02-28).

Wikipedia. Text mining. English article, 2011b. URL http://en.wikipedia.org/wiki/Text_mining. (accessed on: 2011-11-11).

Wikipedia. Text mining. German article, 2011c. URL http://de.wikipedia.org/wiki/Text_Mining. (accessed on: 2011-11-11).

Wikipedia. N-gram, 2011d. URL http://en.wikipedia.org/wiki/N-gram. (accessed on: 2011-11-11).

S. K. M. Wong, W. Ziarko, and P. C. N. Wong. Generalized vector spaces model in information retrieval. In *Proceedings of the 8th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 18 – 25, 1985.

X. Yi and J. Allan. Evaluating topic models for information retrieval. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, pages 1431–1432, 2008.

C. Zhai. *Statistical Language Modeling for Information Retrieval*. Morgan & Claypool, 2009.

B. Zheng, D. McLean, and X. Lu. Identifying biological concepts from a protein-related corpus with a probabilistic topic model. *BMC Bioinformatics*, (7):58, 2006.