Interactive Topic Modeling for Source Code Analysis

Patrik Ehrencrona Kjellin
Abstract

Interactive Topic Modeling for Source Code Analysis

Patrik Ehrencrona Kjellin

Trying to make sense of large sets of data is becoming a task very central to computer science in general. Topic models, capable of uncovering the semantic themes pervading through large collections of documents, have seen a surge in popularity in recent years, with applications in a variety of domains. In this thesis, topic models are applied to source code repositories, specifically for the purpose of concept location - offering an overview of which features are contained within a system, the relationships between such features, and their locality within the system.

Topic models are high level statistical tools; their raw output is given in terms of probability distributions, suited neither for simple interpretation nor deep analysis. Interpreting an inferred model in an intuitive manner requires significant post-processing and tools suited for such purposes.

Additionally, topic models rarely produce perfectly sensible and coherent topics without some level of supervision - some measure of human interaction is thus typically required for refining the output.

Our objective is to simplify the process of topic modeling as it pertains to source code analysis, by addressing the aforementioned issues. First, by implementing existing methods of semi-supervised topic modeling, offering users tools for iteratively refining an inferred model. Second, by tightly integrating topic modeling with high level visual representations of inferred models, capable of capturing the relationship between terms, documents and features related to a source code repository.

We have implemented a fully working prototype of such a system. Through a survey, we have put the tool in the hands of users, thereby demonstrating the system to offer several perceived benefits from a user perspective - in terms of easily comprehending large-scale repositories and in terms of facilitating the process of topic modeling.
Acknowledgment

Thanks to the Innovation Project at Tongji University Graduate Education (2014JYJG016) for supporting this work.

I would like to express my sincere gratitude

To my supervisor Prof. Liu of the Software Engineering school of Tongji University, for her continued support and helpful guidance throughout my work.

To mom, dad, Sofie, Maja, and Hanna, without whom this would never have been possible.

To my understanding girlfriend Danting, for her love and encouragement.
Contents

1 Introduction ................................................. 1
  1.1 Problem Description ................................. 2
  1.2 Motivation ........................................... 2
  1.3 Delimitations ......................................... 3
  1.4 Contributions ........................................ 4
  1.5 Structure ............................................. 4

2 Background ................................................. 6
  2.1 Topic Models ........................................... 6
    2.1.1 General Methods ................................ 6
    2.1.2 Latent Dirichlet Allocation ...................... 7
  2.2 Interactive Visual Analysis ......................... 8

3 Related Work ................................................ 10
  3.1 Survey Methodology ................................... 10
  3.2 Human Knowledge Injection ......................... 11
  3.3 Interactive Visualisation ............................. 13
    3.3.1 Matrix & Text Based ............................. 14
    3.3.2 Graph Based .................................... 15
    3.3.3 Changes Over Time ............................... 17
  3.4 Topic Models and Source Code Analysis ............. 18
  3.5 Summary & Discussion ................................. 19

4 Methodology & Technical Route ......................... 21
  4.1 Summary of Work ...................................... 21
  4.2 Toolchain ............................................. 21

5 Topic Model & Knowledge Injection .................... 23
  5.1 Tree-Based LDA using Dirichlet Tree Priors ........ 23
  5.2 Encoding Knowledge ................................... 24
  5.3 Inference ............................................. 26
    5.3.1 Standard LDA ..................................... 27
    5.3.2 Tree-Based LDA .................................. 27
    5.3.3 Hyper-parameters \{\alpha, \beta, \eta\} .............. 28
1 Introduction

*Topic models* have in recent years emerged as a powerful set of techniques for discovering the underlying semantic structure of large, unstructured collections of documents. The general term topic model refers to a suite of Bayesian or linear-algebraic approaches to discerning the latent topics (themes) pervading in collections of documents (Blei, 2012). Through the results of such analysis, individual documents can then in turn be organised and explored in accordance with their themes. While most commonly applied in a natural language context, such methods have also seen successful application in other domains, including source code analysis for a variety of tasks (Chen et al., 2015).

Common topic model techniques are not perfect; they tend to suffer from some problems that may deter ostensible end users, or at the very least prevent them from reaping the full benefits of the methods. Sometimes, inferred topics do not make sense conceptually. Other times, the resulting topics may not align with the specific goals of the user. Fundamentally, such problems stem from the fact that the objective functions subject to optimisation in common topic model techniques do not necessarily reflect the expectations felt on behalf of a user (Boyd-Graber et al., 2009).

The problem is only further exacerbated when dealing with source code, wherein arbitrary language conventions are by no means guaranteed to make sense to a third party.

While topic model papers may sometimes paint a picture of perfectly coherent and sensible topics, such is rarely the case without some measures to inject additional information into the model – commonly done in an *a priori* fashion. More recently, methods for refining inferred models interactively and iteratively have been proposed (Hu et al., 2011).

Another shortcoming is the fact that interpretation of inferred models is not necessarily a simple task. Topic models such as Latent Dirichlet Allo-
cition (LDA) are high level statistical tools. The raw, numerical output of an inferred model is normally given in terms of multinomial probability distributions, over words in topics, and over topics in specific documents. The difficulty of interpreting such output has motivated the use of a wide range of visual representations (Chaney and Blei, 2012).

1.1 Problem Description

The issues described have motivated the use of both knowledge injection and interactive visualisation as an additional step toward more coherent and sensible results (Yang et al., 2014). The focus of this thesis is therefore two-fold. First, a user interface will be created for the purpose of exploring a topic model fitted to source code repositories interactively. Second, an additional utility will be provided, implementing a partially supervised extension of LDA wherein topic models may be refined by the user at request. Here, given some user-defined constraints, topic model results can be improved in an iterative fashion.

Note that we are not opting to offer improvements or new techniques relative to current state of the art methods for semi-supervised topic model inference. Our goal is rather to implement a simplistic variation one such technique, and tightly combine it with a visual and interactive interface geared for the specific purpose of source code analysis.

1.2 Motivation

Visual interactive analysis in conjunction with topic models has come under more focus in recent years with some promising results found in user studies, but the area still remains somewhat unexplored. Topic models have been successfully applied for the purpose of source code analysis in several previous studies, but most commonly, the output is simply displayed in terms of tabular views or word-clouds. The main motivation for this work is to make topic models, specifically for the purpose of source code analysis, more read-
ily available to end users, by using interactive visual analysis to provide more interesting visual representations, and by simplifying the process of injecting domain knowledge into an inferred model.

The hope is to tie domain knowledge injection and visuals together in a way that makes it easier to obtain meaningful results from topic models. Since most frameworks for interactively refining topic models are based on simplistic text or tabular views, using more interesting graphical representations to guide the user through such a process will thus be the novel contribution of this work, with special focus on source code analysis. Since this work focuses on source code explicitly, design decisions will be made accordingly.

1.3 Delimitations

The aim of this work is not a fully functional system, but rather a prototype incorporating some specified features.

The visual representation of models has been limited to a single form of visualisation. We have opted for a graph-based approach.

Source code analysis has been restricted to a select few repositories. Due to time constraints, some aspects have been designed with the structure of these specific repositories in mind, rather than more generic solutions. For example, omitting licensing and author information present in source code documents has been done specifically according to the conventions found in the selected repositories.

A single topic model method has been used (namely, LDA). No other algorithms have been employed, and no comparison between different topic model approaches are given in this thesis. Additionally, topic modeling results have not be evaluated in any computational manner, as no gold standard for such evaluation exists. However, the prototype implementing semi-supervised modeling and visual representations has been evaluated through case studies and a brief user study.

While focus has been placed on Python repositories primarily, the work
is generic in the sense that it could easily be applied to other programming languages without any major modification.

While topic models can be used for a variety of tasks in source code analysis (Chen et al., 2015), we focus exclusively on concept location. Briefly put, we are interested in providing a good overview of repositories such that a user can easily identify the specific features and purposes that relate to different parts of the system.

1.4 Contributions

The main contribution of this thesis is in the form of combining a semi-supervised topic model with visual representations suited for the purpose of concept location, with the aim of simplifying the use of topic models for analysing large scale repositories. Since most extended topic models that incorporate some form of user supervision do not provide much in terms of visualisation, and vice versa; current tools providing visualisation do not offer interactive measures of model refinements, combining these two features is thus in itself a novel contribution.

A secondary contribution is the implementation of a (simplistic) semi-supervised extension of LDA using user-designated constraints on words, adhering to the standards of the MALLET machine learning library, which already incorporates a variety of topic model tools.

1.5 Structure

The thesis is structured as follows. A comprehensive background of topic models in general and LDA in particular is given in Chapter 2. In Chapter 3, we present the results of a literature review, attempting to get a grasp on the current state of topic model research as it pertains to interactive domain knowledge incorporation and interactive visualisation. The findings thereof have served as a basis for the choice of appropriate methods and techniques used to accomplish the goals of this thesis. In Chapter 4 we present the
general methods and technical choices relevant to this thesis. Chapter 5 outlines our approach to topic modeling, using existing methods to incorporate domain knowledge iteratively. In Chapter 6, we detail the source code repository subject to analysis and any pre-processing steps applied on the data. A prototype, the vitm tool is presented in Chapter 7, which is subsequently evaluated through a user survey and a brief example detailed in Chapter 8, with discussion on the results deferred to Chapter 9. We conclude the thesis with some final thoughts and propose future work in Chapter 10.
2 Background

Here some background is presented, summarising concepts found in this thesis. The section is structured into a brief introduction to topic models in general, with more in-depth descriptions of the model to be employed for this project (LDA).

Relevant literature has been identified and studied, both in terms of some books comprehensively dealing with the subjects (such as natural language processing (NLP), LDA, and interactive visual analysis (IVA)), and in terms of current research papers on interactive visualisation and human knowledge injection within the context of topic models.

2.1 Topic Models

In information retrieval (IR), the general term *topic model* refers to a suite of algorithmic approaches to discovering the latent topics present in a collection of documents.

2.1.1 General Methods

Some basic vocabulary is necessary for describing the general concept of topic models. Formally,

- A word (or term, used interchangeably) $w$ is a basic unit of data (commonly a string of alphanumerical characters, but topic modeling can be applied to other domains than natural language processing).

- A document $d$ is an ordered sequence of $N$ words, $w_1, w_2, \ldots, w_N$.

- A corpus is an unordered set of $M$ documents, denoted by $D = \{d_1, d_2, \ldots, d_M\}$. 
CHAPTER 2. BACKGROUND

Topic modeling then consists of taking a corpus $D$ as input and computing $K$ topics (typically in terms of multinomial distributions over the words in the vocabulary), and associating each document with relevant topics (again, in terms of a multinomial distribution over the different emerging topics).

Early attempts at tackling this problem were however largely concerned with creating a term-document matrix, describing the relative frequency of words in each document $d \in D$. This method is useful for many applications, but insufficient in terms of topic modeling, as such a matrix provides little size reduction w.r.t. the original corpus, and does not take into account the relationships between words within a document or documents within a corpus (Blei et al., 2003).

Further research resulted in the strictly linear algebraic approach Latent Semantic Indexing (LSI), which uses singular-value decomposition in order to significantly minimise the term-document matrix (Deerwester et al., 1990). This was later on extended by Probabilistic LSI (PLSI) (Hofmann, 1999), an early generative model attempting to correct some of the statistically unsound aspects of LSI.

2.1.2 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is a generative statistical model loosely based on earlier work on LSI and probabilistic variations thereof. LDA attempts to address some perceived shortcomings found in the previous generative model pLSI.

Namely, in pLSI, parameters to be estimated grow linearly with the size of the corpus. It also has a strong tendency for overfitting, and of even greater consequence, the model is unable to generalise topic mixtures onto previously unseen documents (not part of the training data) (Blei et al., 2003; Chen et al., 2015). Through correcting these problems with a truly generative model, LDA has seen a surge in popularity and has acted like a springboard for numerous other advancements in IR.

In LDA, documents are regarded as mixtures of a finite set of $K$ underlying topics, where the parameter $K$ must be specified either by the user or
determined through computational inference on the corpus to be analysed. Topics, in turn, are seen as multinomial distributions over the words of the vocabulary.

Inherent to LDA is the assumption that a corpus $D$ of $M$ documents, consisting of $N_M$ words and $K$ topics is generated accordingly (Blei et al., 2003);

1. Choose $\phi_k \sim Dir(\beta)$ for each topic $k \in \{1, \ldots, K\}$.

2. Choose $\theta_i \sim Dir(\alpha)$ where $i \in \{1, \ldots, M\}$, $Dir(\alpha)$ denotes the Dirichlet distribution for $\alpha$.

3. For each word position $i, j, j \in \{1, \ldots, N_i\}, i \in \{1, \ldots, M\}$:
   
   (a) Choose a topic $z_{i,j} \sim Multinomial(\theta_i)$

   (b) Choose a word $w_{i,j} \sim Multinomial(\phi_{z_{i,j}})$

Here, $\alpha$ and $\beta$ are smoothing factors for document-topic and topic-term distributions, respectively. LDA is then concerned with inferring the relevant posterior distribution (i.e., given the terms present in the corpus, what are the topics) through a latent variable model (the latent variables being the distribution over words for each topic). The normal approach for sampling in LDA is by collapsed Gibbs sampling, but we will defer inference descriptions to Section 5.3.1. For a more in depth description of LDA in general, we refer to the article of (Blei et al., 2003), for instance.

### 2.2 Interactive Visual Analysis

Interactive visual analysis (IVA) is a set of techniques incorporating visual analytics (VA) and user interaction in computational or statistical analysis.

Typically, IVA is employed in the task of analysing and attempting to obtain deeper insights from large and possibly complex sets of data, where certain information may not easily be extracted from looking at the data set alone.
IVA is particularly useful for hypotheses generation and validation, since it equips the users with tools enabling them to look at data sets in a variety of different ways and perspectives (Oeltze et al., 2012).

A good paper summarising about a decade worth of human-computer interaction research is the paper by (Thomas and Cook, 2005). Important concepts applicable to this thesis include the lessons on computational efficiency. For an operation of some complexity in an interactive system (such as inferring the LDA posterior of a corpus, or refining it) the authors state the users should not be kept waiting for periods longer than approximately 10 seconds, or on the order of 1 second for simpler operations.
CHAPTER 3. RELATED WORK

3 Related Work

Here, the literature is surveyed as it pertains to interactivity and visualisation within the context of topic models, with the goal of finding current research trends in this area serving as a basis for decisions in our own work. We also provide a brief discussion on topic models in a source code analysis context. Finally, based on the findings of this literature review, we discuss our choices for techniques applied in our work.

3.1 Survey Methodology

Visualisation of fitted topic models is a relatively young field; while many topic model papers include some degree of output visualisation, it is rarely the main focus of the paper. Papers purely dealing with the subject are somewhat sparse, and to the knowledge of the author, no summarised overview of such papers exist. Additionally, we are interested in techniques and methods that not only visualise topic models, but also provide the user with some degree of interactivity.

Papers candidate for review have been found primarily through common search engines (i.e., Google) and the digital libraries of ACM and IEEE. Search terms used are various mixtures of \{topic, model, IR, interactive, visualisation, visual, IVA\}. Papers have then been selected based on their relevance, as deemed by the author upon glancing over the contents. As a basic criteria for relevancy, the papers must be focused on topic modeling, while also including some aspect related to interactive visualisation, possibly incorporating human algorithm supervision.

In this literature review section, no comparison is made between methods surveyed against other methods of visualisation that do not include much of an interactive component, as no such papers are reviewed. It simply attempts to create an overview of current research trends within this specific subset of
visualisation techniques, as they relate to topic models.

In the surveyed articles, interactivity was most commonly incorporated for addressing one of two concerns:

1. **Human knowledge injection.** The first use case concerns integrating human knowledge in topic models in some manner. Parameter tuning and model constraints through user interactions can enhance models in various ways. Topic models like LDA rely on parameters that can not easily be estimated through computation alone, although there are methods for doing so with varying levels of success. Often, some emerging topics will be nonsensical to a human user (Hu et al., 2011). Through interactivity, a topic model can be guided towards achieving more meaningful results.

2. **Topic visualisation.** Visualisation of the emerging topics generated by a topic model appeared in many of the surveyed articles. Graphical tools of many varieties have proved helpful in the task of exploring and attempting to make sense of the results of topic modeling, in order to get an overarching grasp of the various topics spanning some literary corpus. IVA, in the form of allowing users to navigate the corpus and discover the relationships between topics and documents, has been shown to allow users to gain deeper insight in studies (Chaney and Blei, 2012).

Apart from this, many different task specific measures are to be found in topic model related papers. For instance, topic modeling for the purpose of source code analysis may benefit from visual interactive analysis for displaying the relationships between actual code, requirements documents, and change logs. Here, we focus on general-purpose methods.

### 3.2 Human Knowledge Injection

Some researchers have attempted to improve on the results offered by topic models by correcting some of their common shortcomings through incorpo-
CHAPTER 3. RELATED WORK

Figure 3.1: Diagram from (Yang et al., 2014), outlining the general process of user-guided constrained topic models.

rating human knowledge in the process. Shortcomings of topic models identified in previous research include non-sensible and incoherent topics (Newman et al., 2010), certain terms wrongly belonging to a topic, terms not belonging to a specific topic when they sensibly should (Andrzejewski et al., 2011), etc. At its heart, the problem is due to the fact that the objective function subject to optimisation in LDA does not necessarily reflect the expectations on topic quality felt on behalf of a human (Boyd-Graber et al., 2009).

Many different extensions to the normal methods (LDA, in particular) have been proposed for improving the results offered by different models. One such approach is by directly incorporating domain knowledge into the model, typically in an \textit{a priori} fashion, thereby introducing a degree of supervision to an otherwise unsupervised model.

In the paper by (Yang et al., 2014), the authors describe constrained LDA (cLDA), a framework for allowing users to add constraints to a model in order to improve it iteratively. Constraints are defined on the documents in terms of \textbf{must link}, indicating that two documents semantically belong to the same topic, or \textbf{cannot link}, representing the opposite.

The general process of this semi-supervised learning, outlined in Figure 3.1, consists of first performing LDA analysis, then presenting selected documents to the user who adds constraints based on the output, upon which a specialised constrained LDA is computed. The constraints are encoded as \textit{soft constraints}, which is to say they will be satisfied to some specified degree, but not necessarily fully satisfied.
Table 3.1: Summary of Surveyed Papers

<table>
<thead>
<tr>
<th>Type</th>
<th>Papers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix based</td>
<td>Termite (Chuang et al., 2012), A. Chaney, et al. (Chaney and Blei, 2012), The Topic Browser (Gardner et al., 2010), H. Yuening, et al. (Hu et al., 2011), Y. Yang, et al. (Yang et al., 2014), D. Andrzejewski, et al. (Andrzejewski et al., 2011)</td>
</tr>
<tr>
<td>Graph based</td>
<td>Topicnets (Gretarsson et al., 2012), ParallelTopics (Dou et al., 2011), LDAVis (Sievert and Shirley, 2014), LDAexplore (Ganesan et al., 2015), TopicPanorama (Liu et al., 2014), S. Rönqvist, et al. (Rönqvist et al., 2014), Hierarchie (Smith et al., 2014)</td>
</tr>
<tr>
<td>Time based</td>
<td>TextFlow(Cui et al., 2011), TIARA(Liu et al., 2009), ThemeRiver(Havre et al., 2002), RoseRiver(Cui et al., 2014)</td>
</tr>
</tbody>
</table>

Similarly, in the paper by (Andrzejewski et al., 2011) the authors implement user interaction through allowing users to add constraints to the model formulated in first-order logic (FOL). The FOL constraints are similar to the **must link** and **cannot link** constraints found in (Yang et al., 2014), but defined on word-pairs rather than documents. In some cases, real-time interactive knowledge injection has been applied, such as in the paper by (Hu et al., 2011), where the authors have used similar concepts as in the paper by (Andrzejewski et al., 2011) to create a framework allowing users to iteratively and interactively improve topic modeling results.

While the aforementioned papers are general-purpose solutions, many of the specialised variations of LDA which incorporate domain knowledge are custom-built, single-purpose methods. All the methods described here for semi-supervised, user guided LDA use only limited visualisation, in the form of matrices or word clouds.

### 3.3 Interactive Visualisation

Topic models are high level statistical tools; the raw numerical distributions produced alone are not particularly well suited for intuitive analysis (Chaney...
and Blei, 2012). The visualisation of topic models is an area previously relatively unexplored, which has come under more scrutiny in recent times. Here, some of the concepts and techniques found in the literature (summarised in Table 3.1) are described in terms of their respective unique contributions. There are many ways of visualising topic models, however in this survey of interactive visualisations, the most common representations were found to be either graph based, or matrix and text based, along with a few other novel visualisation techniques. The following subsections are organised accordingly.

### 3.3.1 Matrix & Text Based

Matrix or tabular representations are generally easily understood from a user perspective (Ganesan et al., 2015). In *Termite*, the authors present a visual analysis system for quickly assessing fitted topic models computed with LDA, for the specific purpose of user-guided, iterative topic modeling. A corpus is represented in matrix form, wherein rows correspond to words, and columns to topics. While, in its current state, *Termite* is merely a visual tool, the authors outline future work consisting of expanding it into a complete framework for user-guided, iterative topic modeling with the addition of user interaction (in terms of topic deletion and merging, model parameter adjustments, et cetera) (Chuang et al., 2012).

As mentioned, LDA requires several input parameters, the smoothing variables $\beta$, $\alpha$ and the number of topics $K$. There are no strict guidelines for setting these parameters; tuning is usually done through experimentation. This emphasizes the need for good visualisation, allowing users to quickly evaluate the results of their model and tuning parameters accordingly, as is the ambition presented in (Chuang et al., 2012).

In (Gardner et al., 2010), an interactive tool *The Topic Browser* is introduced, with the addition of incorporating document attributes (such as date and authors). Additionally, in their method, a variety of different topic and document metrics are computed and displayed – ranging from simple word counts to pairwise topic and document correlations. Visualisation is done through a mixture of *word clouds* (i.e., terms listed with font sizes de-
CHAPTER 3. RELATED WORK

termined by their respective probability within a topic) and other text-based views, which may be filtered.

Though many of the existing approaches serve to give a good overview of topics, they seldom capture the relationships between individual documents present in the data. In (Chaney and Blei, 2012), a topic navigation method for fitted topic models is presented where, in comparison with other methods, greater emphasis is put on individual documents, rather than just topics. Moreover, contrary to previous similar methods, the authors attempt to use visuals rather than numerical data to convey meaning.

The authors provide not only a summarised overview of the corpus as a whole, but also an interactive method for uncovering the discovered structure of the corpus in more detail; in terms of document-document and document-topic relationships. Visualisation is here done entirely through several tabular, text-based views. The technique is validated through a (small) user survey, indicating that the interactive visualisation gives rise to additional insight and discovery.

3.3.2 Graph Based

In the topic browser TopicNets (Gretarsson et al., 2012), among other things, a novel visualisation approach is presented allowing users to navigate the corpus through a high-level graph-based representation of topics, wherein semantically similar topics are positionally clustered. Another interesting addition seen in TopicNets is the ability of users to perform additional topic modeling on subsets of the corpus for more fine-grained analysis.

In the paper by (Rönnqvist et al., 2014), a corpus is similarly organised and explored through a graph-based visual approach; topics are displayed along with relevant terms, and are linked together with similar topics through shared keywords. The authors note that topic models are imperfect; review by domain experts is often necessary for perfecting the fitted models – a fact that should be accounted for in further research on topic model visualisation.

ParallelTopics (Dou et al., 2011) presents several novel representations of fitted topic models generated through LDA. The main distinguishing feature
of ParallelTopics is that it displays documents in terms of the number of topics pervading through them; documents are plotted in accordance with the number of associated topics, and a document’s distribution over different topics can be viewed in more detail. An additional interactive view exists in ParallelTopics, which presents topics in terms of their evolution over time. Users gain an overview of the pervasiveness of each topic at some particular moment in time, and are able to “zoom in” on specific periods and topics, thereby accessing documents of that time period that have a high probability of containing said topic.

Not uncommonly, the resulting topics are displayed simply by listing the $n$ most probable terms from each topic, and analogously, listing the $m$ most common topics present in each document. This method often leaves a lot to be desired, as it does not comprehensively capture the document and topic relationships discovered in a way that is easily interpreted. A user study (Sievert and Shirley, 2014) suggests that measuring word relevance purely on the basis of word probability is suboptimal for topic interpretation, as common terms may then appear at the top of several topics. The authors present LDAvis, wherein new ideas are introduced on defining term relevancy in a more useful way. The topic browser LDAVis allows users to visually explore a corpus using such relevancy scores.

TopicPanorama (Liu et al., 2014) differs from previous graph-based approaches in that it visualises not just one, but several corpora. A topic graph is generated for each corpus through a topic model algorithm Correlated Topic Models (CTM). These are then combined through graph matching. The authors wish to address the concern that many topic model visualisation tools are unfit to scale for growing data sets.

Hierarchical LDA (hLDA) is a variation of LDA which, contrary to LDA, captures the relationships between different topics (Blei et al., 2004). Effectively, the method results in topic trees allowing for simpler analysis and greater scalability for large data sets. Recently, some studies have proposed new visual and interactive tools specifically based on such models. For instance, Hierarchie (Smith et al., 2014) uses a sunburst chart (see Figure 3.2) for displaying the hierarchical topic trees in a compact and simple fashion.
3.3.3 Changes Over Time

Beyond simply visualising fitted topic models statically, recently, plenty of research has been conducted on the visualisation of topics changing over time (Cui et al., 2011). Examples of early such methods include ThemeRiver (Havre et al., 2002), where topic evolution over time is displayed in terms of a metaphorical river (see Figure 3.3 for example) made up of smaller streams (topics). Set against a time line, the river provides users with an intuitive overview of how a corpus has changed and at which point specific topics are more or less pervasive in the associated documents. In TIARA (Liu et al., 2009), a tool that resembled the work of (Havre et al., 2002) in terms of visuals, a river is similarly used as metaphor for the changing of topics over time. However, TIARA also includes a rich set of interactive tools for further analysis; users may zoom in on selected topics or topic segments for further analysis. Additionally, by selecting some keyword in the river view, a user can retrieve relevant documents for further examination.

In the paper by (Cui et al., 2011), TextFlow was introduced using a novel approach for LDA output analysis. In contrast to previous research, topics are not only displayed as they progress over time (again, in terms of a river), but the splitting and merging of topics is also captured in the visualisation. It is also highly interactive, allowing users to discover what causes the birth,
death, splitting and merging of topics throughout the time period of the associated corpus. Roseriver (Cui et al., 2014) further builds upon the work in (Cui et al., 2011), using a similar river-flow visual representation, but employs a hierarchical topic model in order to better describe large corpus, and for providing users with different overview levels as desired.

3.4 Topic Models and Source Code Analysis

While natural language processing is the canonical application of topic models, they have seen application in other domains as well, including for purposes of source code analysis.

In fact, recent research has shown that source code exhibits many features similar to those of natural language, and statistical models aimed at natural language can often achieve meaningful results when applied to source code, without modification (Hindle et al., 2012). When modeling source code, a corpus can be regarded as a collection of source code documents, classes, functions, or some other programming construct.

In the case of topic models, linear-algebraic as well as probabilistic models have been successfully applied on source code for different tasks. A very comprehensive survey on the use of topic models for purposes of source code analysis is given in (Chen et al., 2015), details the tasks most commonly tackled with topic models and which models are most commonly employed for such purposes.

Some of the source code analysis tasks where topic models have been
applied are traceability recovery (which source code documents implement which requirements), concept location (what are the features or themes dealt with in which parts of the system, and how do they relate), or for deriving various metrics, commonly for purposes of bug prediction.

The survey by (Chen et al., 2015) stresses the importance of good pre-processing. While topic models can be applied to source code as is, better results can often be achieved by making sure to extract only the most relevant data, discarding programming-related constructs that are unlikely to contribute to the semantic themes of the resulting topics.

3.5 Summary & Discussion

Topic models have seen a surge in popularity in recent years and have provided a new way of discerning useful information from big, complex sets of data, with applications in several different fields.

Recently, much of the effort put into researching topic models, as has been summarised in this chapter, is focused on providing users with tools for interacting with and visualising topic models, both in order to improve results in terms of topic coherence and sensibility, and also for allowing users to fully comprehend and benefit from the model outputs.

Many of the attempts at visualising the results of topic models are not limited to simple visualisation; they also provide varying degrees of user interaction with demonstrably improved results (Gardner et al., 2010), suggesting that IVA may play a central role in making these models more available and intuitive to end users.

Research suggests that different representations may aid in different tasks and lead to discovery at different levels (Chuang et al., 2012). Whereas an overarching graph or matrix based topic overview may provide a deeper understanding of the corpus as a whole, other visuals displaying word relatedness and topic-topic or topic-document relationships may aid in providing other forms of insight or discovery. Currently, most studies focus on a specific level of representation. In future work, several of the proposed representa-
tions could be integrated for a more comprehensive view.

We find that future work in this area may also include more comprehensive frameworks in terms of combining the interactive elements of semi-supervised LDA described in Section 3.2 with interactive visual aids for output analysis – both have demonstrable value in terms of increased usability.

The literature review has served as a basis for choosing techniques and methods in our own topic modeling tool. Having looked at a variety of visualisation techniques, we select the force-directed graph based approach for this project. While it has problems with scaling for large number of topics, it has served to give a good overview of the corpus as a quick glance, which is desirable in our application for concept location.

For implementing interactive refinements, we will use methods originally described in (Hu et al., 2011; Hu et al., 2014) among others, using Dirichlet tree priors to implement must-link constraints iteratively. While there are many methods of incorporating domain knowledge in models, few of them can be applied after a LDA posterior has been derived in a user-guided way. The model described in the papers (Hu et al., 2011; Hu et al., 2014) also has the added benefit of treating constraints as soft constraints, which will be satisfied only as long as they are sufficiently consistent with the underlying statistical model.

The main purpose of our tool in source code analysis will be concept location, mostly due to the fact that LDA is so naturally applicable for this purpose. LDA captures not only the topics found within a corpus, but also relates them to individual documents, seen as mixtures over several topics. We will opt to analyse repositories at a document level, rather than at a class or function level.
4 Methodology & Technical Route

Here we present the general workflow of the thesis work, along with any tools used for achieving the purposes of this project.

4.1 Summary of Work

Thesis work has been conducted in an incremental fashion, with the goal of quickly having a working prototype of the system and refining it iteratively. As such, a simple prototype implementing only the visualisation and basic user inputs was constructed using a standard MALLET LDA implementation.

Subsequently, we have extended the tool with a new implementation of LDA incorporating domain knowledge, obtained through the user interface.

The model has been verified to have the desired effects through testing on synthetic data. Further testing on the tool itself is conducted through a case study, demonstrating its usage in some common scenarios when applied to the specified python repositories.

Finally, we have evaluated the prototype by presenting it to two potential users, to gauge their reactions and for recording their modeling sessions for further analysis.

4.2 Toolchain

Programming has entirely been done in Java using Eclipse. Libraries employed include MALLET, a Java library including a wide variety of machine learning tools, here used for topic modeling. Specifically, the naïve implementation of LDA shipped with MALLET has been used as a starting point for implementing the topic model used in this project. MALLET has also been utilized in part for the parsing and pre-processing of source code. Other Java
natural language processing (NLP) libraries employed is the Stanford NLP library, useful for things like word stemming or lemmatisation (i.e., reducing a word to its dictionary form, removing any inflectional endings).

Apache commons has been used for certain matrix arithmetic operations, and for calculating Pearson correlation between topics.

The user interface component is implemented as a Java Swing application. MATLAB has been used to some extent for analysing and plotting the outputs of topic modeling.
5 Topic Model & Knowledge Injection

In this chapter, we describe the approach to topic modeling taken in this work, which is a simplified variation of previous methods (Hu et al., 2011; Hu et al., 2014; Andrzejewski et al., 2009), where Dirichlet tree priors are used in order to encode user-defined must-link constraints on terms within the corpus, termed Dirichlet Forest LDA (DFLDA).

The main goal of our implementation is to allow users to refine inferred models by improving certain topics through the linking of related words. At the same time, we wish to keep satisfactory topics intact. The model described in this chapter implements soft constraints, with strength controlled by the introduction of a new parameter. The benefit of these types of constraints is that they will only affect the model in the case when the constraints are sufficiently aligned with the underlying statistical model of the data.

5.1 Tree-Based LDA using Dirichlet Tree Priors

Encoding domain knowledge in order to introduce partial supervision in LDA has been done in several ways, with the typical approach being a priori strategies of encoding things like related terms (one example being LDA with topic-in-set (Andrzejewski and Zhu, 2009)). In more recent work, supervision has been implemented through Dirichlet tree distributions for an in situ approach, wherein topic models can be refined iteratively.

In this work, we will use a method similar to tree-based LDA using Dirichlet tree priors as seen in (Andrzejewski et al., 2009; Hu et al., 2011; Hu et al., 2014), among others, except we will limit ourselves to must-link constraints, and make simplifications wherever possible. The purpose of this thesis is not
CHAPTER 5. TOPIC MODEL & KNOWLEDGE INJECTION

to offer improvements on existing semi-supervised extensions of LDA, but to tightly combine such a model with an interactive, visual tool for the purpose of source code analysis. Additionally, we will refrain from performing hyper-parameter optimisation, and from using the various optimisation strategies existing for DFLDA (Hu et al., 2014)

Tree-based structures are well suited for LDA-esque models since they preserve conjugacy (just as the Dirichlet distribution, the Dirichlet tree is conjugate to the multinomial), which allows us to implement a Gibbs sampler not unsimilar to that of standard LDA (Hu et al., 2014). Second, contrary to various a priori knowledge injection techniques, they allow for incremental refinements, which is more compatible with our goal of an interactive topic model approach.

5.2 Encoding Knowledge

In DFLDA, knowledge is of the form must-link or cannot link relationships between words, and is encoded using tree structures. For the purposes of this thesis, we limit ourselves to must-link constraints, omitting the more complex cannot-link constraints (Hu et al., 2014; Andrzejewski et al., 2009). While of similar nature, the implementation of cannot-link constraints is much less straight-forward, stemming from the fact that cannot-link constraints do not possess the transitive property of must-link constraints. Indeed, given two constraints \{t_i, t_j\} and \{t_j, t_k\} where \( t \) denotes terms from the vocabulary, we can conclude that \{t_i, t_k\} shall also be treated as linked. We therefore combine must-link constraints according to their transitive closure, which in this case would result in a single constraint \{t_i, t_j, t_k\}.

In our simplified version of DFLDA, the general process of topic modeling is as follows. We first infer a posterior distribution analogous to that of regular LDA, since no constraints are present. Once constraints have been solicited from the user based on the initially inferred model, in terms of \( n \)-tuples of the form \{\( t_0, \ldots, t_n \)\} where \( t_i \) indicates the \( i \)th term of the vocabulary, we find the transitive closures of the initial constraints using depth-first
Figure 5.1: Prior tree transformations after the addition of two constraints. The tree represents correlated distributions. With $\eta$ set to a value much larger than that of $\beta$, terms commonly belonging to a constraint are unlikely to have highly varying probabilities within a topic.

Let $\Omega_l$ denote the $l^{th}$ constraint of $\Omega$. Conceptually, using the notions found in (Hu et al., 2011), due to the occlusion of must-link constraints, we have a single tree containing a root $r$, under which we have children of the form:

- All terms $t_i$ not belonging to any constraint, i.e., $\forall l : t_i \notin \Omega_l$, will occupy a single node directly under the root.

- All terms $t_i$ belonging to constraint $\Omega_l$ are children of an internal node, which in turn is connected to the root.

An illustration of the tree structure given a set of constraints is provided in Figure 5.1.

In DFLDA, each topic $i \in \{1, \ldots, K\}$ has a distribution $\phi_i$ over words and constraints, rather than just over words. Each constraint $\Omega_l$, in turn, has a distribution $\pi_l$ over the terms belonging to the constraint.

Recall the generative process of LDA, described in equations 1–3 of Section 2.1.2. The generative process for DFLDA is similar, but involves the addition of generating terms given a constraint, when necessary.

1. Choose $\phi_k \sim \text{Dir}(\beta)$ for each topic $k \in \{1, \ldots, K\}$.

2. Choose $\pi_{l,k} \sim \text{Dir}(\eta)$ for each constraint $\Omega_l$ and $k \in \{1, \ldots, K\}$. 
3. Choose $\theta_i \sim \text{Dir}(\alpha)$ where $i \in \{1, \ldots, M\}$. $\text{Dir}(\alpha)$ denotes the Dirichlet distribution for $\alpha$.

4. For each word $w_{i,j}$, $j \in \{1, \ldots, N_i\}$, $i \in \{1, \ldots, M\}$:
   
   (a) Choose a topic $z_{i,j} \sim \text{Multinomial}(\theta_i)$
   
   (b) Choose either a constraint $\Omega_l \sim \text{Multinomial}(\phi_{z_{i,j}})$ or a word $w_{i,j} \sim \text{Multinomial}(\phi_{z_{i,j}})$
       
       i. If a word $w_{i,j}$ was chosen, we are done,
       
       ii. otherwise, choose a word from the constraint-term distribution $w_{i,j} \sim \text{Multinomial}(\pi_{\Omega_l, z_{i,j}})$

The full description including derivations of the generative process of Dirichlet forests can be found in (Andrzejewski et al., 2009; Hu et al., 2014; Hu et al., 2011), for instance.

5.3 Inference

Here we describe the method of inference used for standard LDA and DFLDA in this project. A variety of methods exist for the inference of LDA posterior distributions. We have opted for the most common one, being Markov Chain Monte Carlo (MCMC) processes by the means of Gibbs sampling. Non-stochastic methods include variational inference and iterated conditional modes (ITM). While generally faster, and having the benefit of determinism, such methods also introduce bias and have a strong tendency to get stuck within local optima. Nevertheless, due to fast convergence, it is worth noting that they could be of interest for a real-time, interactive application such as ours. Since users are unlikely to wait for the Markov Chain to converge, bias will be present regardless. Faster results would also contribute to less frustration when iteratively refining an inferred model.
5.3.1 Standard LDA

In LDA, collapsed Gibbs sampling is the standard method for inferring the posterior distribution of the latent variables in terms of a Markov chain. Using the vocabulary of Section 2.1 and definitions of sampling equations given in the paper by (Hu et al., 2011), given \( M \) documents of size \( M_N \) and \( K \) topics with a vocabulary of \( V \) words, the state of the Gibbs sampler for LDA consists of topic assignments for each token of the corpus \( z_{d,n} = \{1, \ldots, K\} \) for all \( d \in \{1, \ldots, M\} \) and \( n \in \{1, \ldots, M_N\} \), with \( z_{d,n} \) initialised to random topics.

Inference is then done through, in each iteration and for each token of the corpus, resampling the topic assignment \( z_{d,n} \) given the current topic assignments of all tokens excluding that of \( z_{d,n} \) (denoted by \( Z_{-(d,n)} \)) with conditional probability:

\[
p(z_{d,n} = k|Z_{-(d,n)}, \alpha, \beta) \propto \frac{T_{d,k} + \alpha}{T_{d,(\cdot)} + K\alpha} \cdot \frac{P_{k,w_{d,n}} + \beta}{P_{k, (\cdot)} + V\beta}
\] (5.1)

Here, \( T_{d,k} \) refers to the number of times topic \( k \) occurs in document \( d \). \( P_{k,w_{d,n}} \) denotes the number of times type \( w_{d,n} \) has been assigned to topic \( k \). In our notation, a variable indexed with a dot of the form \( T_{d,(\cdot)} \) implies marginalisation over the missing index (i.e. \( T_{d,\cdot} = \sum_{k=1}^{K} T_{d,k} \)).

5.3.2 Tree-Based LDA

Given that we only consider must-link relationships, our single Dirichlet tree has two sampling equations; one for the case when a word is unconstrained, which is identical to that of standard LDA, and one for the case when a word is part of a must-link constraint. These equations are taken from (Hu et al., 2011), with derivations given in (Andrzejewski et al., 2009). We jointly sample the latent variables \( z \) and \( \Omega \) in one go, making our implementation a blocked Gibbs sampler. Notice that the sampling equation for words not
belonging to any constraint is the same as that of standard LDA.

\[
p(z_{d,n} = k|Z_{-(d,n)}, \alpha, \beta, \eta) \propto \begin{cases} 
    \frac{T_{d,k} + \alpha}{T_{d,()} + K\alpha} \cdot \frac{P_{k,w_{d,n}} + \beta}{P_{k,()} + V\beta}, & \text{if } \forall l : w_{d,n} \notin \Omega_l \\
    \frac{T_{d,k} + \alpha}{T_{d,()} + K\alpha} \cdot \frac{P_{k,l} + |\Omega_l|\beta}{P_{k,()} + V|\Omega_l|\beta} \cdot \frac{W_{k,l,w_{d,n}} + \eta}{W_{k,l,()} + \eta}, & \exists l : w_{d,n} \in \Omega_l
\end{cases}
\]  

(5.2)

Here, \( P_{k,l} \) denotes the number of times any term \( w \in \Omega_l \) has been assigned to topic \( k \). \( W_{k,l,w_{d,n}} \) denotes the number of times \( w_{d,n} \) occurs in constraint \( \Omega_l \) in topic \( k \). For intuition, the first term of the equation represents the proportion of topic \( k \) in document \( d \). The second term essentially captures the combined contribution of all words in constraint \( \Omega_l \) to topic \( k \), whereas the last term captures the contribution of term \( w_{d,n} \) to the probability of \( \Omega_l \) belonging to topic \( k \).

### 5.3.3 Hyper-parameters \( \{\alpha, \beta, \eta\} \)

We keep hyper-parameters fixed, as hyper-parameter optimisation has a tendency to reduce the effectiveness of the constraints. The constraint word-distribution parameter \( \eta \) should be kept high relative to \( \beta \), and optimisation of \( \beta \) may bring it to a level comparable to \( \eta \). There are however successful uses of hyper-parameter optimisation in DFLDA (Hu et al., 2011; Hu et al., 2014; Andrzejewski et al., 2009). An \( \eta \) value of 1 corresponds to regular LDA. Note that these constraints are not hard, but merely preferences, with strength controlled by the constraint-term distribution parameter \( \eta \).

### 5.4 Ablation

The goal of our interactive topic modeling, and DFLDA in general, is to retain the good topics while improving those that suffer from shortcomings. When resuming inference, with constraints added to the model, we need to decide from which state we should proceed – the identical state of the previous Markov chain or one with some \( z \) variables strategically unassigned. We
will refer to such strategies as ablation (Hu et al., 2011). Revoking all topic assignments is clearly incompatible with our goal of retaining good previous topics; we are left with a priori knowledge inference and any interactivity is lost. On the other hand, revoking no topic assignments will make it difficult for the constraints to significantly impact the results. A compromise suggested in the literature (Hu et al., 2011) is to either revoke topic assignment for any newly constrained term, or revoking topic assignments for each token in any document containing one or more newly constrained terms.

We have experimented with all different strategies and found that document level ablation is preferable for our purposes, as the constraints are not sufficiently satisfied in other strategies.

5.5 Evaluating the Model

In this section we evaluate the model using a synthetic corpus, and describe some of our findings upon applying it to real source code data.

5.5.1 Synthetic Data

We conduct some basic experiments on a synthetic corpus in a manner similar to that of (Andrzejewski et al., 2009), in order to check that the must-link constraints have the desired effect. We model the simple corpus shown in Table 5.1 using different configurations of $\eta$, with smoothing factors $\beta = 0.01$ and $\alpha = 0.5$. The results are projected through PCA.

<table>
<thead>
<tr>
<th>doc</th>
<th>contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>red blue red blue</td>
</tr>
<tr>
<td>2</td>
<td>cat dog cat dog</td>
</tr>
<tr>
<td>3</td>
<td>apple apple apple apple</td>
</tr>
<tr>
<td>4</td>
<td>red blue red blue</td>
</tr>
<tr>
<td>5</td>
<td>cat dog cat dog</td>
</tr>
<tr>
<td>6</td>
<td>apple apple apple apple</td>
</tr>
</tbody>
</table>

Table 5.1: A simple corpus.
For each experiment, we begin with running our sampler for 100,000 iterations (burn-in). Then, we sample for another 150,000 iterations, collecting the current topic term distribution $\phi$ at every $10^6$th iteration for a total of 150 samples. Our reasoning is that due to the stochastic nature of Gibbs sampling, looking at single samples is insufficient. We achieve dimensionality reduction through PCA, and add some randomness to each term-probability in each sample in order to make clusters with big overlap more clearly visible in the plots. Figure 5.2 is the plot for $\eta = 1$, with no constraints added to the model, identical to standard LDA.

Figure 5.2: $\eta = 1$ unconstrained LDA.

For $K = 2$, the six clusters seen in Figure 5.2, emerging at about equal probability, correspond to permutations of the following three topic distributions for the vocabulary [red, blue, cat, dog, apple]

$\phi_1 = (\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, 0), \phi_2 = (0, 0, 0, 0, 1)$

$\phi_1 = (\frac{1}{4}, \frac{1}{4}, 0, 0, \frac{1}{2}), \phi_2 = (0, 0, \frac{1}{2}, \frac{1}{2}, 0)$
\[ \phi_1 = (0, 0, \frac{1}{4}, \frac{1}{4}, \frac{1}{2}), \phi_2 = (\frac{1}{2}, \frac{1}{2}, 0, 0, 0) \]

We then add a constraint \{red, cat\} to the model, and perform the same experiment with \( \eta = 10, 50, 100 \). The results can be seen in Figure 5.3; with \( \eta = 10 \), the constraints are not strongly enforced, and the six clusters found previously still exist in the PCA projection, although they are clearly diminished. As we tune \( \eta \) upward, only two clusters remain. These are permutations of the distributions \( \phi_1 = (\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, 0), \phi_2 = (0, 0, 0, 0, 1) \) seen previously. This is the desired result; whereas apple has become isolated in a topic, \{red, cat\} end up with similar probability mass in a topic along with their co-occurring terms blue and dog.

![Figure 5.3: \( \eta = 10, 50, \text{ and } 100 \) with constraint \{red, cat\}.](image)

### 5.5.2 Shortcomings of the Model

We have also tested the model using real data (Python repositories which will be described in Chapter 6), and found that the constraints have some undesired effects under certain circumstances.

We will not provide any specific results, as they should be evident from the formulation of DFLDA given previously.

The main shortcoming arises with the addition of must-links defined on terms that were previously of high probability within some topic, combined with terms of low probability within the same topic. This may result in the high-probability term getting dragged down somewhat, possibly making the topic worse as a whole.
This is by design. The entire goal of the tree priors is to assign similar probabilities to words within a common constraint. Still, it is problematic when highly relevant terms that are defining for a topic get assigned low probabilities in subsequent iterations due to a **must-link**, and is likely not the effects desired on behalf of a user.

On the other hand, constraints entered are sometimes entirely ineffectual, since constraints will only be satisfied when there is sufficient support in the data. While this is also a desirable feature, it may not align with the expectations of a user.

Another problem occurs when dealing with polysemy (i.e., a word which has multiple semantical meanings determined by the context), stemming from the fact that we treat **must-link** relationships as transitive. For instance, consider the word *man*, which in some context (man contra child) has diametrically different meaning than in another (man contra animal). As such, if multiple constraints were to be defined on *man* within different contexts, our model would be unequipped to deal with polysemy, and would treat all words within those constraints as related.
6 Data & Preprocessing

Apart from the synthetic corpus seen previously, we have evaluated our tool using two python source code repositories. Preprocessing is applied on any corpus evaluated through our tool, with the steps described in Section 6.2.

6.1 Repository Selection

Two Python machine learning libraries (scikit-learn¹ and pybrain²) have been selected for the purpose of topic model based source code analysis (for no particular reason, other than that the author has some previous familiarity with these repositories).

scikit-learn is a machine learning (ML) library for python, providing simple tools for data mining and analysis. scikit-learn contains a total of 395 python source code documents, encompassing about 385000 tokens of 6500 unique types after unwanted terms have been discarded in preprocessing. As such, it is a repository of relatively small size, motivating a smaller number of topics, which is convenient for testing purposes.

pybrain contains 390 documents consisting of about 82000 tokens of 4000 unique types. The latter thus contains significantly fewer tokens per document, yielding some different results.

6.2 Preprocessing

Source code can be subjected to topic modeling as-is, but more generally, some preprocessing steps are first applied. For our purposes, we are not interested in syntactical elements specific to python, as such words do not

¹See official website at http://scikit-learn.org/
²See official website at http://pybrain.org/
carry semantical meaning. We therefore extract identifier names, comments, and docstrings.

All files ending with `.py` contained in a specified directory or any of its sub-directories are imported. The following processing steps are then taken.

### 6.2.1 Removal of licence and author information

In the specified Python libraries (and many others), an initial comment or docstring is located at the top of source code documents containing licensing and authorship information. Such sections are identified through regular expressions and, if found, are omitted in subsequent steps.

### 6.2.2 Tokenisation

Words are tokenised according to the regular expression `[aA-zZ][aA-zZ]+`, i.e., only strings with a minimum length of two characters, consisting solely of normal letters, are kept. Because, by convention, Python identifiers are of the form `something_something`, this method should sufficiently separate identifiers consisting of multiple words.

Upon inspecting the results of processing some Python repositories, variable names using CamelCase convention are found to be frequently occurring as well, so we have opted to handle such cases in an additional step.

### 6.2.3 Stop word and python syntax removal

Words contained in a standard English stopword list (words such as *there* or *about*, for instance) are pruned from the input, as they do not contribute anything in terms of semantical meaning. Semantically, Python keywords (language-syntactical constructs such as *while* and *if*) are akin to stopwords. Such information is therefore also removed in preprocessing.
6.2.4 Lemmatisation

Lemmatisation is the process of removing any inflectional endings of words, keeping only the base or dictionary form of a word (known as a lemma). Lemmatisation is rarely used when analysing source code (out of the 167 papers surveyed in the paper by (Chen et al., 2015), none of them included lemmatisation on the vocabulary (though often they did include a computationally cheaper word-stemming step)), however the authors of (Chen et al., 2015) suggest that lemmatisation may be of interest in future work. While this operation is computationally expensive, we have found that cost acceptable as preprocessing must only be performed once on any given repository. We apply the Stanford NLP library for this purpose. As an example of lemmatisation, a sequence [cat, cats, different, mice] transforms into [cat, cat, differ, mouse]. This is desirable, since otherwise, no thematic relationship would be inferred between two documents containing, for instance, the word model and models, despite the fact that the words carry similar or identical meaning.

6.2.5 Common word removal

Words occurring in a majority (say, 95 percent) of documents are sometimes removed prior to topic modeling. Similarly, words that are very infrequent, only occurring in a small subset of the corpus, may also be removed. While this strategy has been shown to produce better topics in some cases, we have refrained from using this step. After experimentation, we have determined that the results of pruning frequent or infrequent terms is not always positive, and may conceal information that could be of interest in the inferred model.
CHAPTER 7. THE VITM TOOL

7 The vitm Tool

A desktop application vitm (Visual Interactive Topic Modeling) has been implemented in Java, integrating interactive visual analysis and interactive topic modeling. The tool allows users to perform LDA analysis on a selected python repository, upon which the results are visualised. Visualisation is done primarily through the means of a force-directed graph, wherein topics

Figure 7.1: The vitm tool.
(represented as nodes of the graph) are configured in accordance with the correlation between them.

*vitm*, after analysis of the *pybrain* repository, is shown in Figure 7.1. It contains a control panel for selecting repositories and running LDA and refining LDA models given some parameters entered by the user. Topics are visualised through a graph along with terms ranked according to their probability within each topic at the bottom of the window. A file browser is supplied to analyse topic-document relationships. Finally, a utility for entering words into **must-link** constraints is provided.

### 7.1 Software Architecture

A simplified UML class diagram (with insignificant classes removed and attributes and methods hidden) has been generated using the *ObjectAid* extension for Eclipse IDE, as shown in Figure 7.2.

The architecture follows the basic outlines of a model-view-control design scheme. The *Vitm* and *MainPanel* classes serve to initialise the system and put the relevant Swing components into an overarching panel, contained in a Swing JFrame.

*TopicModel* is a class containing an actual MALLET topic model class (in our case *DtLDA*, implementing the extension of LDA described in Chapter 5), implementing methods to initialise, run, and extract data from such a topic model class. *VitmModel* serves as the model part of the MVC scheme, containing the relevant data, performing preprocessing through the class *PreProcessor*, and instantiating the *TopicModel* class when requested through the control classes.

Control classes are:

- **ControlPanel**, the upper panel with buttons and editable fields seen in Figure 7.1.
- **WordTable**, the bottom view containing word distributions for each topic.
• **FileTree**, the folder structure displayed on the left.

The **GraphView** class implements a force directed graph through classes **Node** and **Edge**, and draws the graph.

### 7.2 LDA Implementation

An implementation of the version of DFLDA described in Chapter 5 has been programmed in Java, following the examples set in the MALLET library of a standard, naïve (w.r.t. computational complexity) model. The implementation adheres to MALLET conventions and uses data structures inherent to MALLET topic modeling.
The control panel in the upper part of the application allows users to select directories for analysis. Repositories are analysed at a per document level, where we regard the corpus as a collection of source code documents, rather than a collection of classes or methods.

With the inclusion of lemmatisation and several regular expression matchings, the preprocessing step can be timeconsuming and should be avoided if possible. In case a directory has not previously been subjected to analysis through the tool, preprocessing is performed. The results thereof is stored in a file in the directory. Otherwise, if preprocessing has already been performed such that the file is present in the directory, we simply import the processed data and proceed with topic modeling. If the repository has changed, the user may explicitly specify that preprocessing should be performed again.

Users may enter the number of iterations and the $\eta$ parameter for DFLDA, as well as the desired number of topics.

7.3 Visualisation

The topics are visualised primarily through a force-directed graph, capturing the correlation between different topics. The graph in turn is linked to two additional views; a file browser allowing users to visualize the topic distribution of each document, and a tabular view of the word distributions of each topic. An additional utility is provided for selecting individual words and linking them through must-link constraints.

7.3.1 Force Directed Graph

A complete graph $G = \{V, E\}$ is constructed and displayed for the purpose of providing an overview of the inferred topics. A force directed graph approach is used, wherein graph layouts are computed as a function of the information contained within the graph, such as data assigned to edges. A good summarisation of such techniques is given by (Kobourov, 2012). Here, each vertex $v \in V$ represents a topic (named according to the most highly ranked terms within the topic). Edges in between nodes are assigned values based
CHAPTER 7. THE VITM TOOL

Figure 7.3: A force directed graph, illustrating topic correlations for LDA fitted to the pybrain repository with $K = 40$.

on the topic correlation, computed as the logarithm of a Pearson correlation (of the topic-word distributions of each topic). Pearson correlation produces a value $r \in [-1, 1]$ where $r = 1$ corresponds to a total positive correlation, $r = -1$ to a total negative correlation, and $r = 0$ indicating no correlation.

With such a value $r_{i,j}$ assigned to each edge $\{v_i, v_j\} \in E$, in each iteration, we use the forces inflicted on each node to calculate velocity and acceleration, and update their position accordingly. The result is a visual graph that will initially move about for some time before settling in a configuration of minimal energy (as illustrated in Figure 7.3). A user may move nodes around by clicking and dragging them. Nodes are coloured according to an elliptical gradient, such that nodes near the centre appear blue, whereas the colour of nodes further away from the centre shifts to red. The colour of the edges corresponds to the correlation between two nodes, with stronger topic-topic correlation represented by brighter edges. Users may freely zoom in and move around for a closer inspection of the results.

Upon visual inspection of the results, the force-directed graph method produces visually appealing representations of models fitted to a low to moderate amount of topics approximately in the range of $10 \leq K \leq 50$. For very large corpora, requiring hundreds of topics, a graph representation appears
to be less suitable.

7.4 Document Browser

In the left-hand view of the application (see Figure 7.1), the folder tree specified for analysis is displayed in a browsable format. Selecting individual documents (with a .py extension) triggers the node size for each topic to change. Node sizes are derived from the probability of each topic occurring within the specified document, such that more probable topic nodes grow and less probable topic nodes shrink.

The document browser is necessary since we do not want to limit users to exploring the topic-term distributions, but also the topic distribution of each document.

7.5 User Feedback

In the bottom of the application is a tabular view displaying the words (ranked according to probability) of each topic (see Figure 7.4). The topic names, at the top of the matrix, are top-rated terms by default. A user may edit those names which will be reflected in the graph view. Terms in the matrix may be added to the term list to the right, which in turn can then be added to a must-link constraint in the bottom-right corner.

![Figure 7.4: The topic term view and interface for entering constraints.](image)
8 Evaluation

Evaluating topics computationally is a nontrivial task. While several methods have been proposed and put into practice (metrics of model quality include log-likelihood and perplexity), such methods do not necessarily reflect good topic quality as assessed by a human – in fact, studies have shown that models performing well in terms of such metrics, may in some cases result in less semantically meaningful topics (Boyd-Graber et al., 2009).

In this chapter, we first present a case study, illustrating the workflow and usage of our tool. To evaluate the tool itself, we present it to two potential users, collect their results, and gather their feedback.

Generally, when we say "top 10 words of topic k", we are referring to a set of words \( w_i \) being the most frequent in terms of \( p(w_i|k) \).

8.1 Case Study

To demonstrate some ways in which \textit{vitm} can be used, we provide a walkthrough of a typical modeling session. We apply our tool to the smaller \texttt{PyBrain} repository with \( K = 12 \) for a demonstration that is easy to follow. Parameter settings here are \( \alpha = 0.1 \), \( \beta = 0.1 \), and \( \eta = 100 \) (when applicable).

The initial standard LDA process is run for a total of 500 iterations. In subsequent steps of refining the results, we settle for 100 iterations. While 100 iterations is unlikely to be sufficient for convergence, we limit ourselves to a smaller number of iterations for two reasons. First, for realism, since most users would likely refrain from waiting for convergence. Second, while the model as a whole will improve with further inference, the constraints are sometimes less effective given a large number of iterations, as a result of the model getting stuck in local extrema (Hu et al., 2011).

The initial state after 500 iterations of regular LDA can be seen in Table 8.1. While \( K = 12 \) is probably insufficient for the corpus subject to
CHAPTER 8. EVALUATION

Table 8.1: Standard LDA fit to the PyBrain repository with $K = 12$ and 500 iterations.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Top 15 words</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>agent action reward task ob maze observation pybrain episode environment reset state init rl env</td>
</tr>
<tr>
<td>2</td>
<td>test list module file set key find node error val tag str index filename item</td>
</tr>
<tr>
<td>3</td>
<td>individual fitness population evaluable evaluation key init list size mutation evolino evaluator set step optimize</td>
</tr>
<tr>
<td>4</td>
<td>dataset datum set sequence target trainer index field module error label append output epoch seq</td>
</tr>
<tr>
<td>5</td>
<td>point sigma range size gray len inv mm norm update plot dot batch log center</td>
</tr>
<tr>
<td>6</td>
<td>network connection layer module net add pybrain output test input build recurrent full dim bias</td>
</tr>
<tr>
<td>7</td>
<td>gl joint body insert set geom sensor stop world mass axis cole finger po link</td>
</tr>
<tr>
<td>8</td>
<td>action state num learner explorer reward module policy learn gradient sigma theta init dot pybrain</td>
</tr>
<tr>
<td>9</td>
<td>sensor env task environment reward action len angle step epi init pole renderer count po</td>
</tr>
<tr>
<td>10</td>
<td>game player size po capture opponent move color task env pybrain result random gomoku win</td>
</tr>
<tr>
<td>11</td>
<td>function dim xdim blob environment blobtransformation permutation multi sqrt penalize ball init sum val base</td>
</tr>
<tr>
<td>12</td>
<td>offset module dim inbuf outbuf slice index size error neuron implementation layer outerr arg input</td>
</tr>
</tbody>
</table>

We first turn our attention to topic 3, largely comprised of terms related to genetic algorithms (GA) (with words such as individual, fitness, and population). While the theme is obvious, we find that some unrelated terms like key, list, size and set have unreasonably high probability of occurring within topic 3. We can rectify this by linking such unwanted terms with words of some other topic.

Since the words deal with data structures such as lists and sets, topic 2 may be a more suitable fit for these terms. We thus formulate a constraint between unwanted terms of topic 3 and highly probably terms of topic 2: \{set, list, size, key, test, module\}, marked in red in Table 8.1. Here, we hope to rid topic 2 of some terms while simultaneously drawing them into a better fitting topic.

The results of this procedure and another 100 iterations of DFLDA is shown in Table 8.2. The terms related to lists and sets are now to be found

Table 8.2: DFLDA fit to PyBrain repository with $K = 12$, 100 iterations, and a constraint $\{set, list, size, key, test, module\}$.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Top 15 words</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>action agent reward task ob pybrain observation state maze environment reset episode tom perform episodic</td>
</tr>
<tr>
<td>2</td>
<td>dim item list test set module key permutation arr scipy front row len find dict</td>
</tr>
<tr>
<td>3</td>
<td>individual fitness population kwarg evaluation evaluable search init mutation evolino step genome set evaluator</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>function dim xdim blob dot environment init len core blobtransformation sum log sqrt exp nu</td>
</tr>
<tr>
<td>12</td>
<td>layer connection module network dim offset input output slice inbuf weight outbuf outdim bias pybrain</td>
</tr>
</tbody>
</table>
in topic 2, rather than topic 3, as is our desire. More terms highly relevant to genetic algorithms have replaced them in topic 3, creating a more coherent topic. Note that not all terms of the constraint \( \{ \text{set, list, size, key, test, module} \} \) have been pruned from topic 3 – the word set remains, although it is not as highly ranked. This is a case of the data somewhat overriding the constraint, which will only be fully enforced if it is sufficiently consistent with the data of the model.

Another undesired term \textit{kwarg} has obtained a high rank, so we repeat the procedure with a new constraint \( \{ \text{kwarg, function} \} \) (marked in blue), in an attempt to draw the term into the more relevant topic 10, the results of which is shown in Table 8.3.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Top 15 words</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>action agent state reward task pybrain module learner ri ob num episode observation maze init</td>
</tr>
<tr>
<td>2</td>
<td>list key test item dim len set grid point find permutation arr log front search</td>
</tr>
<tr>
<td>3</td>
<td>fitness individual population evaluation evaluable init mutation evalino evaluator genome pybrain optimizer key copy</td>
</tr>
<tr>
<td>4</td>
<td>data set datum set sequence target module trainer index field error label seq output append epoch</td>
</tr>
<tr>
<td>5</td>
<td>g point gray center range norm gni vertex summe gaze matrix texture color normal image</td>
</tr>
<tr>
<td>6</td>
<td>network module connection pybrain net add test layer build recurrent structure linear tom full dim</td>
</tr>
<tr>
<td>7</td>
<td>joint body insert set geom sensor stop mass osle axis finger node po object link</td>
</tr>
<tr>
<td>8</td>
<td>sigma num update size gradient dot batch inv alpha rate range log factor learn len</td>
</tr>
<tr>
<td>9</td>
<td>sensor env task environment reward len action step position theta angle list epi pole count</td>
</tr>
<tr>
<td>10</td>
<td>game size player po capture opponent move task color env result random gomoku winner pop</td>
</tr>
<tr>
<td>11</td>
<td>function dim xdim plot blob len environment dot mapblob transformation kwarg temp penalize range cos</td>
</tr>
<tr>
<td>12</td>
<td>layer module offset connection dim output input inbuf network slice outbuf outdim weight size structure</td>
</tr>
</tbody>
</table>

Table 8.3: DFLDA fit to \textbf{PyBrain} repository with \( K = 12 \), 100 iterations, and a constraint \( \{ \text{function, kwarg} \} \).

The final result, as displayed through \textit{vitm}, can be seen in Figure 8.1. Having selected a document highly related to genetic algorithms in the document browser, we obtain relative node sizes in the graph, which serves to give a sense of which topics pervade through this particular document.

### 8.2 User Survey

A brief survey was performed to gauge the reaction of two potential users upon modeling a repository using our prototype.
8.2.1 Putting vitm in the Hands of Users

The tool was distributed to two users, both familiar with Python coding and with some basic notion of machine learning. After a brief introduction to the tool, they were asked to model the pybrain repository, first to familiarise themselves with the tool, and finally to refine a model to the best of their ability, with results recorded and collected for analysis.

8.2.2 Parameter Settings

For LDA and DFLDA, we use $\alpha = 0.1$ and $\beta = 0.1$. These parameters, which we do not optimise in our current prototype, must usually be tuned experimentally. Upon visual inspection of modeling the two corpora we found that these values produce relatively sensible topics, and that small changes to $\alpha$ and $\beta$ had little impact on the results. We do not currently allow users to change these variables, as parameter estimation is difficult, and we would prefer their focus elsewhere. The constraint strength parameter $\eta$ was set by
the users, by our recommendation, to different values in the range of [10, 200]. We left it up to the users to decide $K$, the number of topics.

### 8.2.3 Results

The perplexity of the model, defined as \( \text{perplexity}(w) = \exp\left(\frac{-L(w)}{\text{count of tokens}}\right) \) where $L$ is the log-likelihood of the model and $w$ a set of data (in our case, the training set), was estimated and recorded at each iteration of sampling, shown in Figures 8.2 and 8.3, as compared to a baseline consisting of standard LDA over a comparable number of iterations. The word constraints were also recorded, to see which types of constraints were typically added. The top 15 words of each topic for the final models of the two users are given in Tables 8.4 and 8.5.

Additionally, the users were asked for their general experience of using the tool, on whether it helped them gain insight into the repository, and on whether they could offer any suggestions (see questionnaire in Appendix A).

**Model Perplexity**

![Figure 8.2: Perplexity of DFLDA (first user) as compared to LDA, with $K = 15$ and about 2000 iterations.](image)
While perplexity is not guaranteed to correlate with topic coherence or sensibility, the metric is still often used in topic model papers. We measured the perplexity at each iteration for both users. Since one user opted for 15 topics, and another for 10, we plot both results as compared to a baseline standard LDA with an equivalent number of iterations (as $K$ appears to impact perplexity).

The first observation is that the model perplexity will spike at the addition of new constraints, whenever DFLDA is prompted by the user. This is as we would expect, since our ablation method will strategically revoke a partition of the hidden variables $z$ at the inclusion of new constraints.

With such a small sample size, and with consideration to the stochastic nature of LDA, no certain conclusions can be drawn, but we generally find that after some number of iterations when DFLDA is starting to converge, the addition of constraints does not appear to have a big impact, positive nor negative, on perplexity.

Figure 8.3: Perplexity of DFLDA (second user) as compared to LDA, with $K = 10$ and about 1500 iterations.
Resulting Topics

Both users experienced the model as somewhat unpredictable, usually producing a mixture of good and poor topics. The final topics are shown below in Tables 8.4 and 8.5.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Top 10 words</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>function dim xdim bobb environment map bobbtransformation policy sum multi</td>
</tr>
<tr>
<td>2</td>
<td>joint body insert set geom sensor stop mass ode po</td>
</tr>
<tr>
<td>3</td>
<td>gl point gray range center norm glut vertex summe gray</td>
</tr>
<tr>
<td>4</td>
<td>render set port client format load datum step socket line</td>
</tr>
<tr>
<td>5</td>
<td>plot grid log search kwarg list point range len id</td>
</tr>
<tr>
<td>6</td>
<td>network connection layer module add pybrain net output input test</td>
</tr>
<tr>
<td>7</td>
<td>test val list key item dim set find node module</td>
</tr>
<tr>
<td>8</td>
<td>offset module dim inbuf slice outbuf index forward neuron size</td>
</tr>
<tr>
<td>9</td>
<td>sigma dot size num update gradient inv alpha rate len</td>
</tr>
<tr>
<td>10</td>
<td>game player po capture opponent move size color env task</td>
</tr>
<tr>
<td>11</td>
<td>individual fitness evolino population genome rbm weight mutation add kwarg</td>
</tr>
<tr>
<td>12</td>
<td>sensor task env action reward environment len ob init reset</td>
</tr>
<tr>
<td>13</td>
<td>evaluation evaluable fitness size population evaluator step init optimizer pybrain</td>
</tr>
<tr>
<td>14</td>
<td>dataset datum set sequence target module index error field label</td>
</tr>
</tbody>
</table>

Table 8.4: Topics of the first user with $K = 15$.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Top 10 words</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>fitness individual population genome size evaluation evaluable list evolino init</td>
</tr>
<tr>
<td>2</td>
<td>module network layer connection pybrain dim offset input add net</td>
</tr>
<tr>
<td>3</td>
<td>test plot datum list set dim module rbm index find</td>
</tr>
<tr>
<td>4</td>
<td>gl function point dim gray bobb range center norm glut</td>
</tr>
<tr>
<td>5</td>
<td>sensor task action env environment reward len ob init step</td>
</tr>
<tr>
<td>6</td>
<td>po insert set stop body axis finger joint link size</td>
</tr>
<tr>
<td>7</td>
<td>dataset datum sequence set target trainer module field error index</td>
</tr>
<tr>
<td>8</td>
<td>action agent state game module pybrain reward learner player rl</td>
</tr>
<tr>
<td>9</td>
<td>sigma num dot update alpha gradient size xdim len function</td>
</tr>
<tr>
<td>10</td>
<td>joint body geom node sensor set ode world attr tag</td>
</tr>
</tbody>
</table>

Table 8.5: Topics of the second user with $K = 10$.

Constraints

The constraints observed in the survey were most typically of two kinds. First, constraints were used to reinforce the relationship between terms of already sensible topics, with constraints such as \{\textit{mlp, layer, network}\}. Words
of sensible topics were linked together, making them less likely to appear in other topics, while at the same time giving them a more similar probability in the original topic. This strategy appears to be a natural use case of *vitm*, as it appeared in both observed sessions.

The other case concerned removal of unwanted terms, by linking ill-fitting terms of some topic together with more closely related terms in other topics.

**User Experience**

Both users felt that the tool could aid them in understanding a repository previously unknown to them, primarily through looking at topic proportions of the documents in the repository. Both users noted that topics resulting from LDA were sometimes accurate, but other times made very little sense.

One user expressed concerns that inference was slow – waiting for the specified number of iterations took longer than expected (the user correctly noticed that an increased number of topics further exacerbated the issue).

One user expressed that the DFLDA utility can be confusing, since it does not always yield the expected results. Since DFLDA constraints are not hard, and will generally not be satisfied unless they are consistent with the underlying data, user constraints will sometimes be ineffectual, despite being perfectly sensible to a user.

The tuning of $K$ also turned out to be a difficulty during the sessions; since we do not estimate $K$ computationally, it is left to users to tune the variable. It is typically far from obvious what configuration of $K$ produces the best topics, and one user suggested that automatic tuning of $K$ as an optional feature would have been preferable.

Upon one user experimenting with the tool for some time, a stated concern was that navigating a large number of topics, where each topic would have to be assessed through looking at the top words, made it difficult to get an overview of the repository. On the other hand, given that *pybrain* is a relatively small corpus, a large number of topics is bound to produce some insensible topics, so this is not entirely unexpected. However, for a larger corpus, the graph-based approach is probably sub-optimal.
One user stated that they would have liked more numerics presented to the user, such as the size of the corpus, the average number of words in the documents, and possibly more intricate statistics related to the repository.
9 Discussion

In this chapter we discuss the results found in our evaluation.

9.1 Topic Model

The implementation of tree-based LDA presented in our prototype is somewhat naïve relative to other work, which comes at a rising cost as number of topics and corpora size increase. When dealing with user interaction, computational efficiency is of especial importance (Hu et al., 2011), as any unreasonably slow task will be frustrating for the user. As such, further work in a possible continuation of this project should include algorithm modifications for improved performance.

The problems are currently evident, as refining a model fitted to a corpus of relatively large size will take an exceedingly long time. Possible solutions, apart from implementing existing optimisations for LDA and DFLDA, would be some alternate form of inference. Iterated conditional modes (ITM), could be helpful as it will typically result in significantly faster convergence than Gibbs sampling.

The omission of cannot-link constraints should also be reconsidered, as our user-guided topic modeling does not allow for the same expressiveness as current state of the art models.

Hyperparameter optimisation (especially for $K$) could also be interesting for future work, as users expressed their difficulty with finding good values for $K$.

9.2 User Interface

Users generally note that the tool would be helpful to them for exploring a repository, particularly for identifying which features or concepts are con-
tained within different parts.

The graph-based representation does not always provide a good representation of the repository. Other modes of visual representation may be interesting to explore as complements to our current approach, especially for allowing better oversight when the number of topics gets larger ($K \geq 50$). PCA is one viable alternative for such cases.

Another interesting aspect that should be considered, would be plotting (some of the) words in the graph layout with forces applied to the words, such that they are attracted to topics where they have high probability of occurring. As it currently stands, users must look through the table of words present at the bottom of the tool to fully understand the topics. This leads to users having a difficult time comprehending a model, especially with larger values for $K$.

9.3 Source Code Analysis

Topic modeling can achieve a variety of different tasks related to source code analysis (Chen et al., 2015). Here, we have focused purely on providing a good overview of a repository through concept location, allowing users to quickly assess which features are likely related to which parts of the system.

In our survey, we found that there was some desire for presenting additional statistics and numbers to the users. Continuing, interesting work may include incorporating other tasks than concept location.
10 Conclusion & Future Work

A prototype of a tool has been implemented for topic modeling of Python source code repositories using a simplistic version of tree-based LDA. The model has been verified through some experiments, and has been incorporated in the tool such that inferred models can be browsed through an interactive, visual representation.

In a possible continuation of this project, focus should be on improving existing features for better usability. A main concern of the tool in its present state is the fact that inference is not adequately fast for a satisfying user experience. Rectifying this can be done through incorporating some of the existing optimisations for LDA Gibbs sampling. Another possibility would be using other forms of inference in the case when a model is to be refined based on some constraints. For example, iterated conditional modes could be employed. This would result in faster convergence, albeit at the expense of sometimes losing the inertia required to escape local optima.

The tool can also be extended with cannot-link constraints, as well as more complex constraints derived from combinations of cannot-link and must-link constraints, which would make for a more expressive interaction granting users greater freedom to improve a model.

Further evaluation should also be considered as possible future work, perhaps using a platform such as Amazon Mechanical Turk with a much greater number of users. Preferably, computational evaluation of models could be avoided, by allowing another group of users to assess the resulting topics from such a study, as compared to standard LDA topics.

More generally, future work on topic modeling and source code analysis should incorporate more visual and interactive tools for improved usability.
Bibliography


Appendices
A Questionnaire
1. *To what extent would this tool aid you in comprehending a software repository?*  
(1 = to a very low extent, 5 = to a very high extent)

Motivation:

2. *Is topic modelling in vitm sufficiently fast?*  
(1 = much too slow, 5 = satisfactory)

Motivation:

3. *To what extent are the resulting topics sensible?*  
(1 = not at all, 5 = they make perfect sense)

Motivation:
What is your general impression of vitm? In what way would it aid you in exploring a repository?

What other features would you like to see?

Other comments or suggestions?