Investigation of IR based Topic Models on Issue Tracking Systems to Infer Software-Specific Semantic Related Term Pairs

Denzil Correa\textsuperscript{1}  Sangeeta Lal\textsuperscript{2}  Ashish Sureka\textsuperscript{3}

\textsuperscript{1}IIIT Delhi, India (denzilc@iiitd.ac.in)
\textsuperscript{2}JIIT Noida, India (sangeeta@jiit.ac.in)
\textsuperscript{3}ABB India, India (ashish.sureka@in.abb.com)
Table of Contents

1 Research Motivation and Aim
   • Objectives and Context Setting

2 Related Work and Research Contributions
   • Related Work
   • Research Contributions

3 Background on LSI and LDA
   • Latent Semantic Indexing (LSI)
   • Latent Dirichlet Allocation (LDA)

4 Experimental Dataset

5 Experimental Setup and Design
   • Solution Approach and Framework
   • Pre-processing Steps
   • LSI and LDA Tools

6 Experimental Results
   • Software Maintenance Application
   • Performance Evaluation

7 Conclusion

8 References

Denzil Correa, Sangeeta Lal, Ashish Sureka

Topic Modeling on Issue Tracking Systems
# Research Motivation and Aim

## Objectives and Context Setting

# Related Work and Research Contributions

## Related Work

## Research Contributions

# Background on LSI and LDA

## Latent Semantic Indexing (LSI)

## Latent Dirichlet Allocation (LDA)

# Experimental Dataset

# Experimental Setup and Design

## Solution Approach and Framework

## Pre-processing Steps

## LSI and LDA Tools

# Experimental Results

## Software Maintenance Application

## Performance Evaluation

# Conclusion

# References

Denzil Correa, Sangeeta Lal, Ashish Sureka

Topic Modeling on Issue Tracking Systems
Search and Exploration of Software Repositories

Searching software depositories by developers

Software maintenance activities involve non-trivial tasks like bug fixing, feature enhancements and software reengineering.

Requires searching code bases and repositories like version control systems, issue tracking systems and software documentation.

Search and exploration of large information repositories is non-trivial as the terms used to query the knowledge base could be highly domain-specific [13].
Domain-specific semantic vocabulary gap

Developer may have little or no knowledge about the query terms to search the knowledge base to fulfill their information need [11].

Existing lexical resources in the English language like WordNet are insufficient due to domain-specific issues [28].

Need to bridge the domain-specific semantic vocabulary gap and build domain-specific lexical resources.
We propose a novel approach to automatically infer semantically related terms from in a software context to facilitate the construction of a domain-specific lexical resource.

We infer semantically related terms from free form natural language textual data contained in defect tracking systems by use of IR based topic models.

We investigate the application of two popular and widely used IR based topic models Latent Semantic Indexing (LSI) and Latent Dirichlet Allocation (LDA).
Research Motivation and Aim
Objectives and Context Setting

Related Work and Research Contributions
Related Work
Research Contributions

Background on LSI and LDA
Latent Semantic Indexing (LSI)
Latent Dirichlet Allocation (LDA)

Experimental Dataset

Experimental Setup and Design
Solution Approach and Framework
Pre-processing Steps
LSI and LDA Tools

Experimental Results
Software Maintenance Application
Performance Evaluation

Conclusion

References
Wang et al. [34]
Wang et al. propose an algorithm based on an holistic method of context-based and co-occurrence based term scoring to extract technical paraphrases from noisy bug report corpora [34]

Yang et al. [36]
Yang et al. propose a simple similarity based technique based on clustering comments and code to infer semantically related words in software context [36]
Shepherd et al. [26]
Shepherd et al. propose a NLP based method to locate and understand high-level concepts in source code repositories [26].

Aggarwal et al. [2]
Aggarwal et al. present an application of mining bug report description and threaded discussion comments using Latent Dirichlet Allocation (LDA) which is a topic modeling technique [2]
Novel and Unique Contributions

Infer semantically related term pairs
We propose IR based topic model approach to infer semantically related term pairs in software as an aid to construction of a domain-specific lexical resource construction

Experiments on Google Chromium
We demonstrate the efficacy of our inferred lexical resource of a large bug report repository of a popular open-source web browser, Google Chromium, on duplicate bug report detection which is one of the significant software maintenance task.
# Topic Modeling on Issue Tracking Systems

## Table of Contents

1. Research Motivation and Aim
   - Objectives and Context Setting
2. Related Work and Research Contributions
   - Related Work
   - Research Contributions
3. **Background on LSI and LDA**
   - Latent Semantic Indexing (LSI)
   - Latent Dirichlet Allocation (LDA)
4. Experimental Dataset
5. Experimental Setup and Design
   - Solution Approach and Framework
   - Pre-processing Steps
   - LSI and LDA Tools
6. Experimental Results
   - Software Maintenance Application
   - Performance Evaluation
7. Conclusion
8. References

Denzil Correa, Sangeeta Lal, Ashish Sureka
Latent Semantic Indexing (LSI)

Figure: Synonym or related keyword of any primary keyword (Source: Internet)
Latent Semantic Indexing (LSI)

Information Retrieval based Method

IR based method to extract meaningful word representations from a collection of documents [8]

Objective of LSI is to infer latent relationships for words occurring within and across documents

LSI helps uncover words with similar meanings (synonymy) and words which may have more than one meaning (polysemy).
### Latent Dirichlet Allocation (LDA)

<table>
<thead>
<tr>
<th>Term</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>game</td>
<td>0.014</td>
</tr>
<tr>
<td>team</td>
<td>0.011</td>
</tr>
<tr>
<td>hockey</td>
<td>0.009</td>
</tr>
<tr>
<td>play</td>
<td>0.008</td>
</tr>
<tr>
<td>games</td>
<td>0.007</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Term</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>space</td>
<td>0.021</td>
</tr>
<tr>
<td>nasa</td>
<td>0.006</td>
</tr>
<tr>
<td>earth</td>
<td>0.006</td>
</tr>
<tr>
<td>henny</td>
<td>0.005</td>
</tr>
<tr>
<td>launch</td>
<td>0.004</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Term</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>drive</td>
<td>0.021</td>
</tr>
<tr>
<td>card</td>
<td>0.015</td>
</tr>
<tr>
<td>system</td>
<td>0.013</td>
</tr>
<tr>
<td>scsi</td>
<td>0.012</td>
</tr>
<tr>
<td>hard</td>
<td>0.011</td>
</tr>
</tbody>
</table>

**Figure:** Topic modeling with LDA (Source: Internet Databricks)
Latent Dirichlet Allocation (LDA)

Information Retrieval based Method

Probabilistic generative model that helps *discovers document-topic and topic-word distributions* from a collection of documents [7]

The words occurring in the same topic of the output of LDA are connected to each other via a *semantic relationship*. 

Denzil Correa, Sangeeta Lal, Ashish Sureka

Topic Modeling on Issue Tracking Systems
Download Date | August 12, 2012
---|---
Number of Reports Available in Issue Tracker | 142,175
Number of Reports Available for Download | 134,410
Number of Reports excluding WontFix, Unconfirmed, Invalid, Untriaged, Unknown, FixUnreleased | 90,431
Number of Open Reports | 29,596
Number of Closed Reports | 104,814
Number of Duplicate Reports | 28,460
Number of Duplicate Reports with Master report available | 20,936
Timestamp of first bug report | 2008-08-30 16:00:21
Timestamp of last bug report | 2012-08-11 21:16:59

**Table:** Experimental Dataset Details for Google Chromium DTS
Dataset Details

Publicly Available Google Chromium Dataset

Google Chromium software, a popular open-source web browser, which contains 134,000+ bug reports.\(^a\)

We had 142,175 bug reports available but only 134,410 bug reports were available for download.

We exclude reports which are marked as - WontFix, Unconfirmed, Invalid, Untriaged, Unknown and FixUnreleased.

\(^a\)http://crbug.com/
Table of Contents

1. Research Motivation and Aim
   - Objectives and Context Setting

2. Related Work and Research Contributions
   - Related Work
   - Research Contributions

3. Background on LSI and LDA
   - Latent Semantic Indexing (LSI)
   - Latent Dirichlet Allocation (LDA)

4. Experimental Dataset

5. Experimental Setup and Design
   - Solution Approach and Framework
     - Pre-processing Steps
     - LSI and LDA Tools

6. Experimental Results
   - Software Maintenance Application
   - Performance Evaluation

7. Conclusion

8. References
Bug report extraction, text pre-processing, application of LSI and LDA, extraction of semantically related terms
Multi-Step Pre-processing - I

**Case Folding**
We convert all the terms in the DTS corpora into lower case.

**Stopword Removal**
We use two types of stop word removal techniques. We use a standard *stopword list* in the English language. Due to the noisy nature of DTS corpora, we consider the top 30 frequently occurring words in the corpus also as stopwords.

http://www.link-assistant.com/seo-stop-words.html
Word Level Filtering

We eliminate terms which are lesser than 3 characters in length.

Tokenization

We use a white space tokenizer to form tokens to our IR based topic models.
Table of Contents

1. Research Motivation and Aim
   - Objectives and Context Setting

2. Related Work and Research Contributions
   - Related Work
   - Research Contributions

3. Background on LSI and LDA
   - Latent Semantic Indexing (LSI)
   - Latent Dirichlet Allocation (LDA)

4. Experimental Dataset

5. Experimental Setup and Design
   - Solution Approach and Framework
   - Pre-processing Steps
   - LSI and LDA Tools

6. Experimental Results
   - Software Maintenance Application
   - Performance Evaluation

7. Conclusion

8. References
Latent Semantic Indexing (LSI)
We used the implementation from Gensim, python framework to perform our experiments.\textsuperscript{a}
\textsuperscript{a}http://radimrehurek.com/gensim/

Latent Dirichlet Allocation (LDA)
We used the implementation from the Stanford Topic Modeling Toolbox for our experiments.\textsuperscript{a}
\textsuperscript{a}http://nlp.stanford.edu/software/tmt/tmt-0.4/
### Automatically Inferred Semantically Related Terms - 1

<table>
<thead>
<tr>
<th>Term Pairs</th>
<th>Context</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssl – certificate</td>
<td>Software</td>
</tr>
<tr>
<td>omnibox – search</td>
<td>Google Chromium</td>
</tr>
<tr>
<td>html – background</td>
<td>Google Chromium</td>
</tr>
<tr>
<td>css – element</td>
<td>Google Chromium</td>
</tr>
<tr>
<td>int–bool</td>
<td>Code</td>
</tr>
<tr>
<td>com.google.chrome.framework</td>
<td>Code</td>
</tr>
<tr>
<td>std:allocator – memcheck</td>
<td>Code</td>
</tr>
<tr>
<td>renderblock.cpp – resourcedispatcher</td>
<td>Code</td>
</tr>
</tbody>
</table>
### Automatically Inferred Semantically Related Terms - II

<table>
<thead>
<tr>
<th>Term Pairs</th>
<th>Context</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>time – event</strong></td>
<td>English</td>
</tr>
<tr>
<td><strong>flaky – failure</strong></td>
<td>English</td>
</tr>
<tr>
<td><strong>error – failure</strong></td>
<td>English</td>
</tr>
<tr>
<td><strong>window – tab</strong></td>
<td>Software</td>
</tr>
<tr>
<td><strong>mouse – drag</strong></td>
<td>Software</td>
</tr>
<tr>
<td><strong>hash – base/message</strong></td>
<td>Software</td>
</tr>
<tr>
<td><strong>ssl – certificate</strong></td>
<td>Software</td>
</tr>
</tbody>
</table>
**Successful Extraction of Semantically Related Terms - I**

### English Context

Term pairs *time* – *event* and *error* – *failure* are semantically similar in the English language. In other words, these terms are synonyms in the English dictionary.

### General Software Context

The term pairs *window* – *tab* and *ssl* – *certificate* are semantically similar terms in a general software context on web browsers. SSL commonly used in the form security certificates in web browsing to encrypt communication by web browsers.
Successful Extraction of Semantically Related Terms - II

Google Chromium Context

The term pairs *omnibox* – *search* and *css* – *element* are semantically related in the Google Chromium software context in particular.

Code Context

*int* – *bool* are data types in code which are related to each other as they have the same data type. In addition, *std::allocator* – *memcheck* is a library and function definitions which are used for memory leakage checks.
# Table of Contents

1. Research Motivation and Aim
   - Objectives and Context Setting

2. Related Work and Research Contributions
   - Related Work
   - Research Contributions

3. Background on LSI and LDA
   - Latent Semantic Indexing (LSI)
   - Latent Dirichlet Allocation (LDA)

4. Experimental Dataset

5. Experimental Setup and Design
   - Solution Approach and Framework
   - Pre-processing Steps
   - LSI and LDA Tools

6. **Experimental Results**
   - Software Maintenance Application
   - Performance Evaluation

7. Conclusion

8. References
Duplicate Bug Report Detection - I

Snapshot of Eclipse Project BUG ID: 319000

Bug 319000 - Toolbar does not wrap on Linux

Status: RESOLVED DUPLICATE of bug 46025

Product: Platform
Component: SWT
Version: 3.6
Platform: PC Linux

Importance: P3 normal (vote)
Target Milestone: ---

Reported: 2010-07-06 08:35 EDT by Michael Barkhouse
Modified: 2010-07-06 09:11 EDT (History)
CC List: 1 user

--- BUG DESCRIPTION ---

I've attached a small plug-in based on the PDE plug-in template for a plug-in with a view. I've modified the view to include a tool bar inside the editor component. On Windows the tool bar will wrap when the view is maximized. On Linux, the tool bar does not wrap. The buttons simply disappear off the right side of the tool bar.

Steps to Reproduce:
1. Manually select Window -> Show View -> Other...
2. In the dialog select Simple Category -> Simple View and press the OK button.
3. The view will open. Notice it has a tool bar on the left (open in code view, but I wanted a quick example so I didn't make it look nice). Make the view smaller by dragging the left edge of the view toward the right. Continue doing so until the tool bar runs out of horizontal UI real estate.

Result:
On Windows, the tool bar will wrap resulting in two rows of buttons. On Linux,
Duplicate Bug Report Detection - II

Use-case to demonstrate the efficacy of our approach

Experimental dataset contains 20,936 duplicate bug reports. We only use the description of the bug report for our task.

Procedure to Compute Similarity

The output of LSI and LDA is a term-term pair each consisting of a similarity score. We iterate through each document in our experimental duplicate bug report dataset and compare the similarity with all the other bug reports in the corpus by using these scores from LSI and LDA.
Table of Contents

1 Research Motivation and Aim
   • Objectives and Context Setting
2 Related Work and Research Contributions
   • Related Work
   • Research Contributions
3 Background on LSI and LDA
   • Latent Semantic Indexing (LSI)
   • Latent Dirichlet Allocation (LDA)
4 Experimental Dataset
5 Experimental Setup and Design
   • Solution Approach and Framework
   • Pre-processing Steps
   • LSI and LDA Tools
6 Experimental Results
   • Software Maintenance Application
   • Performance Evaluation
7 Conclusion
8 References
Recall Rate as Evaluation Metrics

<table>
<thead>
<tr>
<th>Top-K list</th>
<th>LSI Recall Rate</th>
<th>LDA Recall Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>44.14</td>
<td>34.32</td>
</tr>
<tr>
<td>20</td>
<td>49.12</td>
<td>35.12</td>
</tr>
<tr>
<td>30</td>
<td>53.47</td>
<td>37.73</td>
</tr>
<tr>
<td>40</td>
<td>55.62</td>
<td>40.51</td>
</tr>
<tr>
<td>50</td>
<td>57.72</td>
<td>40.97</td>
</tr>
</tbody>
</table>

Table: Recall Rates of IR based topic models LSI and LDA on various top-k list sizes for duplicate bug report detection task.
Experimental Result Analysis

Key Takeaways

LSI performs significantly better than LDA while the recall rates achieved are not significantly high.

There are various factors which affect the recall rates and this may not be specifically due to the nature of our lexical resource.
We investigate the effectiveness of IR based topic models like LSI and LDA to infer semantically related term pairs in software domain context.

We apply our technique on Google Chromium DTS, an open source popular browser, and are able to infer semantically related term pairs in various contexts like English language, web browser, Google Chromium specific and code snippets.
We use the automatically inferred semantic term pairs (as is) to look into duplicate bug report detection.

Initial experiments suggest that IR based topic models are an interesting direction to pursue for construction of lexical resources.
References I


References II


References III


