Discovering Traceability between Business Process and Software Component using Latent Dirichlet Allocation

Andreyan Rizky Baskara[1], Riyanarto Sarno[2], Adhatus Solichah[3]
Informatics Department, Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
andreyan09@mhs.if.its.ac.id[1], riyanarto@if.its.ac.id[2], adhatus@if.its.ac.id[3]

***Abstract*— Software system is built to support business process. Software system needs to evolve over time because there are some changes on business processes. A relationship exists between business processes and supporting software system which can help maintainers to understand the system and carried maintenance tasks. Such kind of relation is called traceability links. One approach to discover traceability links is analyzing the similarity of textual content. This paper proposed an approach to discover a traceability links between two software artefacts, which are business processes and software components, using Latent Dirichlet Allocation (LDA). In the proposed method, each label of business process model and software components model are formed into documents. Then, the topic probability distributions are calculated using LDA. The similarities between those two artefacts are calculated using Jensen-Shannon (JS) Divergence The result of LDA is compared to the real software components and business process documents and it shows that LDA and JS Divergence are applicable for discovering traceability links with average Cohens Kappa value of 0.67.**

Keywords—business process, LDA, traceability, supporting software system.

I. Introduction

Business process is a set of activities that organizations must do to reach a certain goal or to provide a certain service. However, business processes are always changing over time. The changes in business process are happened because their organizational structure or their operating standards are changing as well [1]. Some research have been proposed to discover alignment between business processes [2]–[4]. The purpose of this study is to find similarity between business processes and reduce the redundancy of business processes within repository. It will help organization to manage their business processes when a business process change occurs.

Nowadays, Organizations also have a software system that supports their business process. It is an application that supports employee activities within organization to finish their tasks. In service oriented system, service components selection is important, in order to fulfill the best service quality of an organization [5]. It indicates that there is a relationship between business process and software system component. When organizations change their business processes, it may affect the software system components.

A strong correlation existed between two software artefacts, i.e. business process and software component, because a software system is built to support organization to performs business process [6]. In such cases of business process change, the relation between business process and software system component may become helpful to maintainers. It will help software maintainers to understand which software component implements certain business activity. This kind of relation is called traceability links.

Unfortunately, the documentation of those relation may not exist. To discover traceability links, information from different sources, such as business process documentation, requirement documentation, design documentation, and source code are required. Discovering traceability links is not an easy task, because it is difficult to analyze natural language sentences [7]. One approach to discover traceability links between software artefacts is analyzing the textual content. After the textual content known, the similarity of content between them can be calculated. If their textual contents are similar in a certain degree, then it indicates that there is a relation between them. In order to achieve such thing, Information Retrieval (IR) Method is used. Traditional IR methods, like Vector Space Model, probabilistic IR, and LSA often used to find similarity of textual content between documents. These methods also have been applied to some software engineering task [6]–[8].

If extracting the business topic from software system components is possible, then it would be a great help to the software maintainer who without any knowledge of the software system to find the related function and classes then carry out the maintenance task [9]. An approach to mines topics is called topic modelling. One of topic modelling method that quite popular is Latent Dirichlet Allocation. In LDA, documents are formed from multiple topics, and topics are mixture of words. Topic can be interpreted as “meaning” behind word. If two different words are placed on a same topic, they might be similar in term of “meaning”. LDA usually used to extract topic from documents. However, it can be applied not only limited to documents. In this paper, we proposed an approach to discover traceability links existing between business processes and software system components by analyzing their textual content using LDA.

This paper is organized as follows: Section 2 presents the related works which has been done by other researchers in the same domain. Section 3 presents the proposed approach of this paper to discover the traceability links. Section 4 presents the experiment result of the proposed approach using a case study. And Section 5, gives concluding remarks.

II. Related Work

Some methods to recover traceability have been presented in the literature. IR methods named VSM and Probabilistic IR Model are used in [7] to find traceability links between two software artefacts which are C++ source code to manual pages, and Java code to functional requirements. The result shows that recovering traceability links can be done between documentation and code semi-automatically using information retrieval method.

Reference [8] used LSI and perform experiment on the same case studies as in [7] to compare the performance to VSM and probabilistic IR. The result indicates that LSI can perform well without full parsing while VSM and probabilistic IR required to do it to achieve similar result.

VSM is used in [6] to recover traceability between higher level of requirement which is business processes and software components (source code). The result indicates that IR method is applicable on recovering traceability links between business processes and software components.

The advance development of research to solve problems on IR task, a topic modelling is introduced. A topic modelling named Latent Dirichlet Allocation (LDA) was introduced [10].

Some literatures have applied LDA to recover traceability links or on other software engineering tasks. LDA is used in [11] to mining concepts in source code. Their approach extract the functional concept founds within source code including comments. Then the source code grouped by comparing its document topics distributions. The result also shows that LDA automatically assign topic without the need to put manual annotation to the source code.

Reference [9] used LDA to mining business concepts in source code. The approached proposed a weighting scheme based on place of the domain specific keyword (e.g. class name, attribute name, method name). The weighted matrix used as input on LDA. From experiment result, The approach is able to extract business topic and implementation topic from source code.

In [12], LDA is used to recover traceability on software documentation artefacts prospectively. It means traceability links created and managed while software development occurs. While it is good to maintain traceability while software development occurs, the fact that some legacy software suffers from poor traceability links documentation can't be neglected.

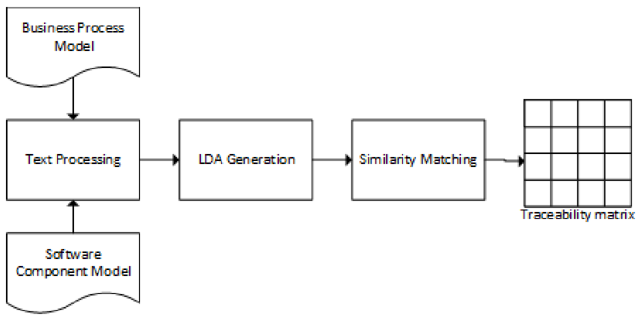

Fig. 1. Traceability Recovery Process of Proposed Approach

This paper contribution are as follows: (1) Discover traceability between business processes and software system components by using topic modelling, LDA (2) The discovery process occurs retrospectively, which means the discovering process is performed when the software is already exist. The approach was done retrospectively, in respect to help maintainers to do maintenance task to a legacy software system which doesn't have traceability links documentation and to discuss the effectiveness of LDA on discovering traceability between business process and software component retrospectively.

III. Proposed Approach

The proposed approach in this paper consist of three main step, i.e. text processing, LDA generation, and similarity matching process. The output of these processes is traceability matrix between business activities and software components. Fig. 1 illustrate the flow of proposed approach.

First, business process models keywords and software system component models keywords extracted using text processing and then joined into a set of documents. Then using LDA, the document-topic and topic-word distribution is generated. After that, each software system components are mapped to the relevant business process based on their similarity.

A. Text Processing

Business processes model is in form of Activity Diagram and used as an input. Each label of activities in diagram is extracted and formed as documents. As for software component model, class diagram is used as an input. Every class inside the class diagram represent a software. Before LDA generates the topic distribution of documents, some text processing methods are required, i.e. tokenization, stop word elimination, lemmatization and stemming, to prepare the documents as input.

Then LDA generation is applied in the next step to get topics over documents probability distribution and words over topic probability distributions.

B. LDA Generation

The next step of the proposed approach is to generates the topic distribution of set of documents using LDA. The idea of LDA is that documents are formed from multiple topics, while

TABLE. I Notation in LDA

Notation	Description
α , β	Hyper parameter, estimated by user.
θ	Probability distribution of topic on documents.
Φ	Probability distribution of word on topic.
z	Topic distribution of word in documents.
T or K	Number of topic

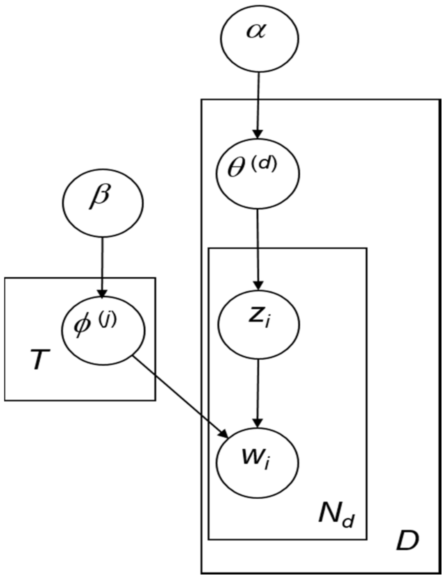

Fig. 2. LDA Generative Model

topics are probability distribution over words. The extraction of probability distribution between document and topic follows dirichlet prior distribution and the extraction of probability distribution between topic and words follows polynomial distribution. Fig. 2. shows the generative model of LDA. There are some notations used in LDA as shown in Table 1.

Parameter estimation is required in LDA. [13] proposed to use Gibbs Sampling to extract the approximation of the posterior distribution. The generative processes for LDA model using Gibbs Sampling according to [14] are as follow:

```
Initialization phase:
1. For every documents m in set D
   a. For every words w in document m
      i. Draw sample topic z randomly ~ Mult(1/K).
      ii. Increment $n^{(w)}_{-i,t}$, $n^{()}_{-i,t}$, $n^{(m_i)}_{-i,t}$, and $n^{m_i}_{-i}$.
Gibbs Sampling phase:
1. For each iteration do:
   a. For all documents m in set D do:
      i. For all words w in document m do:
         1. Cancel the current value of z in w.
         2. Decrement $n^{(w)}_{-i,t}$, $n^{()}_{-i,t}$, $n^{(m_i)}_{-i,t}$, and $n^{m_i}_{-i}$
         3. For topic j = 0 to K-1 do:
            a. Calculate $p(z_i = t|z_{-i}, w_i)$
         4. Draw new topic z ~ $p(z_i = t|z_{-i}, w_i)$
         5. Assign new topic z to word w
         6. Increment $n^{(w)}_{-i,t}$, $n^{()}_{-i,t}$, $n^{(m_i)}_{-i,t}$, and $n^{m_i}_{-i}$.
```

The computing formula to extract the approximation of the posterior distribution $p(z_i = t|z_{-i}, w_i)$ is shown in (1).

$$p(z_i = t|z_{-i}, w_i) \propto \frac{n^{(w)}_{-i,j}+\beta}{n^{()}_{-i,t}+W\beta}\frac{n^{(m_i)}_{-i,t}+\alpha}{n^{m_i}_{-i}+T\alpha} \quad (1)$$

$n^{(w)}_{-i,t}$is the number of word w_i put to topic t. $n^{()}_{-i,t}$ is the total number of word put to topic t. $n^{(m_i)}_{-i,t}$ is the number of word put to topic t in document m_i. $n^{m_i}_{-i}$ is the total number of word within document m_i.

After the estimation process using Gibbs Sampling is finished, probability distribution of topics over documents, $\vartheta^{(d_i)}$, is calculated using (2) and probability distribution of word over topic for each word in vocab, φ^{w}_{j}, is calculated using (3).

$$\vartheta^{(d_i)} = \frac{n^{(d_i)}_{-i,j}+\alpha}{n^{d_i}_{-i}+T\alpha} \quad (2)$$

$$\varphi^{w}_{j} = \frac{n^{(w)}_{-i,j}+\beta}{n^{()}_{-i,j}+W\beta} \quad (3)$$

C. Similarity Matching

After the topic distribution of each document is generated, a similarity between documents are calculated by measuring the difference between each documents topic distributions [15]. Some researchers used Kullback-Lieber (KL) divergence to measure distance between topic distributions. Let $q[x]$ and $d[x]$ are set of topics distributions, the KL distance is calculated using (4).

$$D_{KL}(q,d) = \sum_{i=1}^{T} q_i \ln\frac{q_i}{d_i} \quad (4)$$

However, KL distance measurement is asymmetrical. It means that the measurement result can't be reversed, $D_{KL}(q,d) \neq D_{KL}(d,q)$. Therefore, the proposed approach used Jensen-Shannon (JS) divergence so that the distance measurement becomes symmetrical which means the measurement result reversible on both document, $D_{JS}(q,d) = D_{JS}(d,q)$. The formula of JS divergence is shown in (5).

$$D_{JS}(q,d) = \frac{1}{2}D_{KL}\left(q,\frac{q+d}{2}\right) + \frac{1}{2}D_{KL}\left(d,\frac{q+d}{2}\right) \quad (5)$$

The similarity of business activity documents is checked to each software component documents. If the similarity is greater than a given threshold, then it indicates that the software component is related to the business activity.

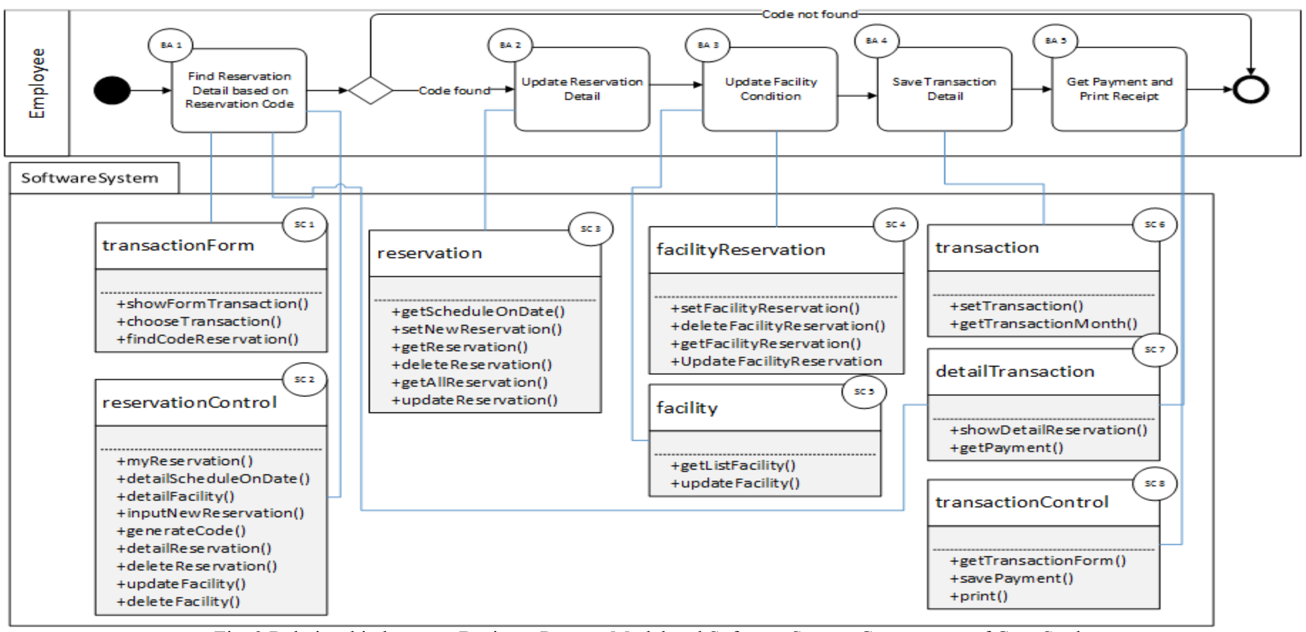

Fig. 3 Relationship between Business Process Model and Software System Components of Case Study

IV. EXPERIMENT AND RESULT

A. *Case Study*

The proposed approach is applied in a case study. A software system which is used by campus organization to manage the reservation of sport facilities and its business processes, is taken as input. The business process model named payment transaction and its software system components model are shown in Fig. 3.

There are 5 business activities and 8 software system components that are used in experiment. The business activities are Find Reservation Detail based on Reservation Code, Update Reservation Detail, Update Facility Condition, Save Transaction Detail, Get Payment and Print Receipt. The software system components are transactionForm, reservationControl, reservation, facilityReservation, facility, transaction, detailTransaction, and transactionControl. Each business activities and software system components are formed into documents as shown in Table 2.

TABLE. II List of Business Process and Software Component Elements

Identifier	Elements
BA 1	Find Reservation Detail based on Reservation Code
BA 2	Update Reservation Detail
BA 3	Update Facility Condition
BA 4	Save Transaction Detail
BA 5	Get Payment and Print Receipt
SC 1	transactionForm
SC 2	reservationControl
SC 3	reservation
SC 4	facilityReservation
SC 5	facility
SC 6	transaction
SC 7	detailTransaction
SC 8	transactionControl

Topic 0

transaction	0.45222
save	0.226486
form	0.151242
detail	0.075997
month	0.075997

Topic 1

get	0.389699
payment	0.292517
print	0.195335
receipt	0.098154
all	0.000971

Topic 2

reservation	0.232819
facility	0.211673
update	0.127088
get	0.105942
delete	0.084796

Topic 3

reservation	0.288596
detail	0.247427
code	0.123919
find	0.08275
show	0.08275

Fig. 4 Top 5 of Words Probability Distributions over Topics from Case Study.

Text processing is performed to extract the keyword identifier on each documents. Then, topics over documents probabilities are generated using LDA. The hyper parameter used in this paper are $\alpha = 0.01$, $\beta = 0.01$ and $T = 4$. We used this setting because it provides the best result from our observation. Fig. 4 shows the top five words probability distribution over topic that have been generated by LDA. From the words probability distribution, we can interpret that Topic 0 is about transaction, Topic 1 is about payment, Topic 2 is about reservation and facility, and Topic 3 is about reservation detail.

After topics over documents probability distributions generated by using LDA, the similarities of topics are calculated. As mentioned in section 3, Jensen-Shannon (JS) Divergence is used to measure the similarity of topic probability distributions. The threshold is set to 0.6. because it gives the best result on finding the related software components in this case.

TABLE. III Traceability Matrix

SC/BA	BA 1	BA 2	BA 3	BA 4	BA 5
SC 1	#				
SC 2	#	x	x		
SC 3		#	x		
SC 4		x	#		
SC 5		x	#		
SC 6				#	
SC 7	#				#
SC 8				x	#

TABLE. IV Topics over Documents Probability Distributions

	Topic 0	Topic 1	Topic 2	Topic 3
BA 1	0.001984	0.001984	0.001984	0.994048
BA 2	0.003289	0.003289	0.661184	0.332237
BA 3	0.003289	0.003289	0.990132	0.003289
BA 4	0.990132	0.003289	0.003289	0.003289
BA 5	0.002475	0.992574	0.002475	0.002475
SC 1	0.374378	0.001244	0.001244	0.623134
SC 2	0.0004990	0.0004990	0.499501	0.499501
SC 3	0.00066489	0.00066489	0.998005	0.00066489
SC 4	0.00083056	0.00083056	0.997508	0.00083056
SC 5	0.001984	0.001984	0.994048	0.001984
SC 6	0.795635	0.200397	0.001984	0.001984
SC 7	0.001984	0.39881	0.001984	0.597222
SC 8	0.498344	0.498344	0.001656	0.001656

TABLE. V Result of Relevant and Retrieved Relation

Activities Name	Relevant & Retrieved	Relevant & Not Retrieved	Not Relevant & Retrieved	Not Relevant & Not Retrieved	Kappa
Find Reservation Detail based on Reservation Code	3	0	0	5	1
Update Reservation Detail	1	0	3	4	0.25
Update Facility Condition	2	0	2	4	0.5
Save Transaction Detail	1	0	1	6	0.6
Get Payment and Print Receipt	2	0	0	6	1
Total average					0.67

If the similarity measurement is greater than the threshold, then business process is related to software components in terms of textual content. From this similarity matching process, a set of software system components that has a relation to business activity are obtained.

Next, the relevance of these results to business activities are traced and they presented as a matrix as shown in Table 3. '#' denotes that software components are relevant retrieved (true positive) and 'x' denotes that they are not relevant retrieved (true negative). Then, the result of LDA is compared to the real software component and business processes documents and evaluated using Cohen's Kappa Coefficient. Cohen's Kappa is intended to measure the degree of agreement on a same observation between two or more people. In this case, it is used to measure the agreement degree between the result of LDA and the real document. Cohen's Kappa Coefficient is calculated using (6).

$$K = \frac{Pr(a) - Pr(e)}{1 - Pr(e)} \quad (6)$$

$Pr(a)$ denotes the total agreement probability and $Pr(e)$ denotes the hypothetical probability of change agreement based on the observed data. The result of retrieved and relevant software components on business activities is shown in Table 5. The average Kappa Coefficient value is 0.67. Some software components are not relevant retrieved. It is happened because those components are having nearly the same topic to the business activity. As shown in Table 4, Business Activity (BA) 1 is dominant to Topic 3, which is about reservation detail and code. Software components which also dominant to Topic 3 are Software Component (SC) 1, 2, and 7. When the similarity is calculated, these 3 software components are produced a high similarity to BA 1 and they are retrieved. And as shown in Fig. 3, BA 1 is related to SC 1, 2, and 7. The Cohens Kappa value is 1 which means the result of LDA and the real document are the same.

For BA 2, the dominant topic is Topic 2. Software components which also dominant to Topic 2 are SC 2, 3, 4, 5. These software components are retrieved after matching similarity process. As shown in Fig 4, BA 2 is related only to SC 3. So Cohens Kappa value on BA 2 is 0.25 because 3 of 4 software components are not relevant retrieved.

V. Conclusion

In this paper, we have proposed an approach to discover traceability links between two software artefacts, which are business processes and software system components using Latent Dirichlet Allocation. LDA is used to discover topic probability distributions of business activities and software components. After the topic probability distributions are known, then the similarity of topic probability is calculated using JS Divergence. We decided a threshold for the similarity result to obtain the most relevant software components for a given business activity.

The result of experiment shows that LDA not only effectively mines topic on those two artefacts, but also can be applied on discovering traceability links between them. The average Cohens Kappa value is 0.67 which can be concluded that the agreement degree between the LDA result and the real document of software component and business process are high. For some business activities, the kappa value is reduced because some software components are not relevant retrieved.

This is happened because those components are having nearly the same topic.

REFERENCES

[1] R. Sarno, H. Ginardi, E. W. Pamungkas, D. Sunaryono "Clustering of ERP business process fragments," *International Conference on Computer, Control, Informatics and Its Applications* (*IC3INA*), pp. 319-324, 2013.

[2] R. Sarno, E. W. Pamungkas, D. Sunaryono, and Sarwosri, "Business process composition based on meta models," *2015 International Seminar on Intelligent Technology and Its Application (ISITIA) - Proceeding*, pp. 315–318, 2015.

[3] Z. Yan, R. Dijkman, and P. Grefen, "Fast business process similarity search with feature-based similarity estimation," *Lecture Notes in Computer Science*, vol. 6426 LNCS, no. PART 1, pp. 60–77, 2010.

[4] B. van Dongen, R. Dijkman, and J. Mendling, "Measuring Similarity between Business Process Models," *Seminal Contributions to Information Systems Engineering*, pp. 405–419, 2013.

[5] N. Y. Setiawan and R. Sarno, "Multi-Criteria Decision Making For Selecting Semantic Web Service Considering Variability And Complexity Trade-Off," *Journal of Theoretical and Applied Information Technology (JATIT)* , vol. 86, no. 2, pp. 316–326, 2016.

[6] L. Aversano, F. Marulli, and M. Tortorella, "Recovering traceability links between business activities and software components," *Communications in Computer and Information Science*, vol. 109 CCIS, no. PART 1, pp. 385–394, 2010.

[7] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo, "Recovering traceability links between code and documentation," *IEEE Transaction on Software Engineering*, vol. 28, no. 10, pp. 970–983, 2002.

[8] A. Marcus and J. I. Maletic, "Recovering documentation-to-source-code traceability links using latent semantic indexing," *25th International Conference on Software Engineering 2003. Proceedings.*, pp. 125–135, 2003.

[9] G. Maskeri, S. Sarkar, and K. Heafield, "Mining Business Topics in Source Code using Latent Dirichlet Allocation," *Proc. 1st India Software Engineering Conference (ISEC '08)*, pp. 113–120, 2008.

[10] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," *Journal of Machine Learning Research*, vol. 3, no. 4–5, pp. 993–1022, 2012.

[11] E. Linstead, P. Rigor, S. Bajracharya, C. Lopes, and P. Baldi, "Mining concepts from code with probabilistic topic models," *Proc. ASE*, no. April 2016, p. 461, 2007.

[12] H. U. Asuncion, A. U. Asuncion, and R. N. Taylor, "Software Traceability with Topic Modeling," *International Conference on Software Engineering (ICSE)*, vol 1, pp. 95-104, 2010.

[13] T. Griffiths, "Gibbs sampling in the generative model of latent dirichlet allocation," *Unpublished*, 2002.

[14] G. Heinrich, "Parameter Estimation for Text Analysis", 2008.

[15] A. Niekler and P. Jähnichen, "Matching results of latent dirichlet allocation for text," *Proc. ICCM*, pp. 317–322, 2012.