Topic Model on Stack Overflow Data

Benjamin Dutton, Liuyi Hu

1 Introduction

Document clustering has been extensively studied in information retrieval. There are distance-based clustering algorithms which use a similarity function to measure the closeness between the text objects. Agglomerative clustering algorithms like hierarchical clustering and partitioning algorithms such as K-means are typical distance-based clustering algorithms.

Another popular approach is probabilistic document clustering and topic modeling is one of them. The idea behind probabilistic document clustering is to assume that the data were generated by some probabilistic process and then to infer the parameters of the process. Most topic modeling methods attempt to learn the above parameters using maximum likelihood methods, or some refinement of this principle such as maximum a posteriori (MAP) probability. There are two basic methods which are used for topic modeling, which are Probabilistic Latent Semantic Indexing (PLSI) (Hofmann, 1999) and Latent Dirichlet Allocation (LDA)(Blei et al., 2003) respectively. PLSI has the disadvantage that the number of model parameters grows linearly with the size of the collection. Therefore, LDA has been used more extensively than PLSI. In LDA framework, each document is modeled as a mixture over an underlying set of topics and each topic is, in turn, modeled as an mixture over an underlying set of topic probabilities (a latent meta-category of words). The topic probabilities thus provide an explicit representation of a document. Besides, the multinomial indicator for each words in a document, by the idea of data augmentation, directly shows the probability of this word that belongs to a specific topic.

In this project, we use LDA model for document clustering of the Stack Overflow (S/O) dataset and apply the collapsed Gibbs sampling method (Griffiths and Steyvers, 2004) for the estimation and inference. In section 2, we describe the LDA model and then we introduce the collapsed Gibbs sampling method in section 3. In section 4, we talk about the interpretation of the topic model and go over the details of the real data and how we pre-process the data in section 5 and 6. In section 7, we describe how we implement the collapsed Gibbs sampling with streaming data in Scala. And in the last section we demonstrate the results of our topic model.

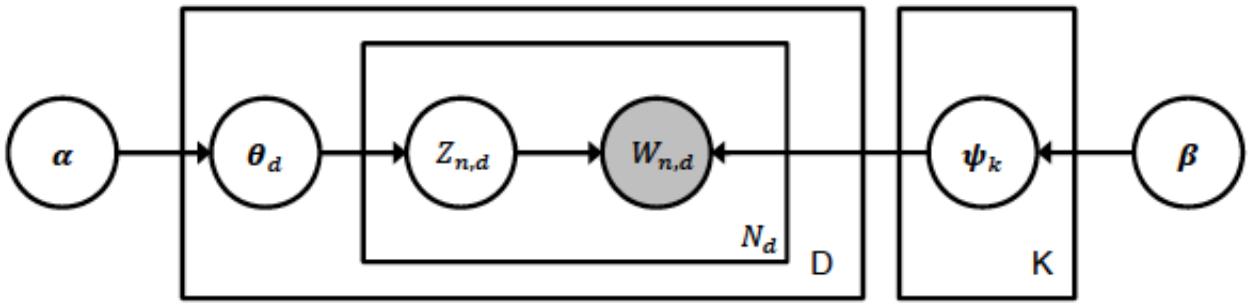


Figure 1: Graphical model representation of LDA.

2 LDA Model

Latent Dirichlet Allocation(LDA) was originally introduced by Blei et al. (2003), which is based on the intuition that each document contains words from multiple topics; the proportion of each topic in each document is different, but the topics themselves are the same for all documents.

Let D denotes the number of documents and N_d, $d = 1, \ldots, D$ denotes the length of each document. And let V be the dimension of the vocabulary. The LDA assumes the following generative process:

1. Choose $\boldsymbol{\theta}_d \sim \text{Dir}(\boldsymbol{\alpha})$, where $d \in \{1, \ldots, D\}$ and $\text{Dir}(\boldsymbol{\alpha})$ is the Dirichlet distribution for parameter $\boldsymbol{\alpha}$
2. Choose $\boldsymbol{\phi}_k \sim \text{Dir}(\boldsymbol{\beta})$, where $k \in \{1, \ldots, K\}$
3. For each of the word positions d, i, where $i \in \{1, \ldots, N_d\}$, and $d \in \{1, \ldots, D\}$
   (a) Choose a topic $z_{d,i} \sim \text{Multinomial}(\boldsymbol{\theta}_d)$
   (b) Choose a word $w_{d,i} \sim \text{Multinomial}(\boldsymbol{\phi}_{z_{d,i}})$

Figure 1 illustrates the process of generating words in LDA model. The joint distribution is thus

$$P(\text{Data} \mid \boldsymbol{\alpha}, \boldsymbol{\beta}) = \prod_{k=1}^{K} P(\boldsymbol{\phi}_k \mid \boldsymbol{\beta}) \prod_{d=1}^{D} P(\boldsymbol{\theta}_d \mid \boldsymbol{\alpha}) \prod_{i=1}^{N_d} P(w_{d,i} \mid z_{d,i} = k, \boldsymbol{\phi}_k) P(z_{d,i} = k \mid \boldsymbol{\theta}_d).$$

Making inference on the LDA model involves both the document-topic distribution parameters $\boldsymbol{\theta}_d$ and the topic-word distribution parameters $\boldsymbol{\phi}_k$. Two common estimation methods are variational Bayes (Blei et al., 2003) and collapsed Gibbs sampling (Griffiths and Steyvers, 2004). In our analysis, we adopt collapsed Gibbs sampling method and we give a detailed description of this method in the following section.

3 Collapsed Gibbs Sampling

A collapsed Gibbs sampler marginalizes over one or more variables when sampling for some other variable to make computation convenient. For LDA model, the collapsed Gibbs sampling method first integrates out document-topic distribution parameters $\boldsymbol{\theta}_d$ and the topic-word distribution parameters $\boldsymbol{\phi}_k$ and consider the posterior distribution over the assignments of words to topic $P(\boldsymbol{z} \mid \boldsymbol{w})$ instead. Then estimation of $\boldsymbol{\theta}_d$ and $\boldsymbol{\phi}_k$ can be obtained by examing the posterior distribuion $P(\boldsymbol{z} \mid \boldsymbol{w})$.

What collapsed Gibbs sampling essentially does is to iteratively sample a topic for every token in the document from a full conditional probability:

$$P(z_{d,i} = k \mid \boldsymbol{W}_{\neg w_{d,i}}, \boldsymbol{Z}_{\neg z_{d,i}}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \propto (C_k^{d,-i} + \alpha_k) \frac{C_{k,w}^{\cdot,-i} + \beta_w}{\sum_{w'} (C_{k,w'}^{\cdot,-i} + \beta_{w'})},$$

where $w_{d,i}$ is the ith word in document d, $C_k^{d,-i}$ is number of words in document d assgined to topic k in not including the current ith word and $C_{k,w}^{\cdot,-i}$ is the number of type w that are assigned to topic k not including the current ith word.

With a set of samples from $p(\boldsymbol{z} \mid \boldsymbol{w})$, we can estimate $\boldsymbol{\phi}_k$ and $\boldsymbol{\theta}_d$ by:

$$\hat{\phi}_{k,w} = \frac{C_{k,w} + \beta_w}{C_k + \sum_{w'} \beta_{w'}}, \ w = 1, \ldots, V,$$

and

$$\hat{\theta}_{d,k} = \frac{C_k^d + \alpha_k}{C^d + \sum_{k'} \alpha_{k'}}, \ k = 1, \ldots, K,$$

where $C_{k,w}$ is the number of tokens of type w assigned to topic k, C_k is the number of words assigned to topic k, C_k^d is the number of words from topic k in document d and C^d is the number of words in document d, i.e. the length of document d.

4 Model Interpretation

When fitting a topic model, the most important question is: What is the meaning of each topic? Since each topic is characterized by its distribution over words, we need to determine the most useful terms for interpreting a given topic. The typical way is to examine the most probable terms in the topic. The problem with interpreting topics this way is that common terms in the whole corpus might turn out to be the most probable words in some topics, making it hard to differentiate the meanings of these topics. Therefore, people propose ranking the terms in the topic in terms of both *frequency* and *exclusivity*. Taddy (2011) used *lift* to rank the terms for each topic. The *lift* is defined as $P(w \mid k)/p(w)$, i.e. ratio of a terms probability within a topic to its marginal probability across the corpus. The

ranking of the globally frequent terms are decreased when using *list*, however, this measure can also be noisy because it gives very high rank to rare terms. In order to overcome this problem, Bischof and Airoldi (2012) came up with a measure called FREX score, which is a weighted harmonic mean of a terms rank within a given topic with respect to *frequency* and *exclusivity*.

Recently, Sievert and Shirley (2014) introduced a similar measurement called *relevance*, which is defined as

$$\text{Relevance}(w, k \mid \lambda) = \lambda \log P(w \mid k) + (1 - \lambda) \log \frac{P(w \mid k)}{p(w)}, \tag{1}$$

where λ is a weight parameter between 0 and 1. This measurement of a weighted average of *frequency* and *lift*. When $\lambda = 1$, then ranking based on relevance is equivalent to ranking based on *frequency*; when $\lambda = 0$, then relevance gives the same ranking results as *lift*.

5 Data

Stack Overflow is a popular programming question-answer site. Users of S/O ask programming and tech-related questions that other members of the community can answer. Both questions and answers can be voted on to help identify important questions and insightful answers and questions are tagged by moderators by the topics they pertain to. Users with good questions and answers get rewarded by accumulating badges that demonstrate their proficiency in various domains. Since at least 2009, S/O has made their data publicly available (Atwood, 2009).

S/O uses a SQL database behind the scenes so their data is organized into tables, 8 of which are publicly available: Badges, Comments, PostHistory, PostLinks, Posts, Tags, Users, and Votes. We focused exclusively on the Posts (which includes both the questions and answers) since this is where most of the semantic content lies.

6 Pre-Processing

One of the challenges of applying NLP techniques to S/O data is that English words are often mixed together with code. Since S/O stores their posts with HTML tags, as a pre-processing step, we remove all code between `<code>` HTML tags. Unfortunately, these tags aren't always used so there is still some code that remains. We also attempted to filter out numbers, which wasn't entirely successful but led to some interesting results demonstrating the power of LDA - topics were found that correspond almost entirely to the notion of a number.

After removing the HTML and code, we took the 10,492 most commonly occurring words across all Posts and ran them through the SMART information retrieval system to remove some common stop words. After pre-processing, the data consists of $D = 4000$ documents, a vocabulary of $V = 10004$ terms, and $W = 128639$ total tokens in the corpus.

Figure 2 shows the word cloud of the top 1000 frequent words in the corpus and Table 1 summarizes the counts of the top 10 frequent words.

Figure 2: Word cloud of top 1000 frequent words in the corpus

Word	Count in the whole corpus
code	1156
file	805
br	745
pi	728
dont	697
work	677
data	655
good	639
server	616
net	590

Table 1: Frequency table of the top 10 words in the corpus

7 Implementation

We organized our data into a MySQL database running on a separate server. In order to take advantage of this project as a learning opportunity, we decided to try and implement the algorithm discussed in Yao and McCallum (2009) (henceforth referred to as "Efficient Gibbs"), which relies on Gibbs sampling, in Scala and run it on Spark - two technologies we were relatively unfamiliar with which have become popular for big data. We soon realized that Spark wasn't very suitable for Efficient Gibbs because the algorithm is inherently sequential - topics get assigned sequentially during each Gibbs sweep. So, we ended up not using Spark, but still implementing the algorithm in Scala.

Unfortunately, the code was pretty slow and for just 1,000 posts, took roughly 33 hours. I don't think this is a failure of the algorithm - simply the effect of poor understanding of how to write efficient Scala code. Nevertheless, the code worked, but unfortunately, the prior parameters we chose - α and β corresponding to the document-topic and topic-word distributions, respectively, were too small. These parameters act as "smoothing parameters"

and because their values were too small, the topic-word distributions were very sparse, leading to poor results. If we had used a larger number of documents, this would have helped, but would have taken much more time to execute.

In the meantime, we took our pre-processed text and tried using R for LDA, which took a fraction of the time. As a result, we decided to only use the Scala code for pre-processing and then use the `lda` library in R to do the inference.

8 Results

8.1 Topic Number Selection

For fitting the LDA model to a given document collection, the number of topics needs to be fixed a-priori. Collapsed Gibbs sampling method also requires specification of the parameters of the prior distribution, and Griffiths and Steyvers (2004) suggest using $\alpha = 50/K$ (K is the number of topics) and $\beta = 0.1$ for the symmetric Dirichlet priors, Therefore, in our analysis, we follow the suggestion and use a symmetric Dirichlet with $\beta_w = 0.1, w = 1, \ldots, V$ for all word types and $\alpha_k = 50/K, k = 1, \ldots, K$. To select the optimal number of topics K, we have tried $K = 10, 20, \ldots, 100$.

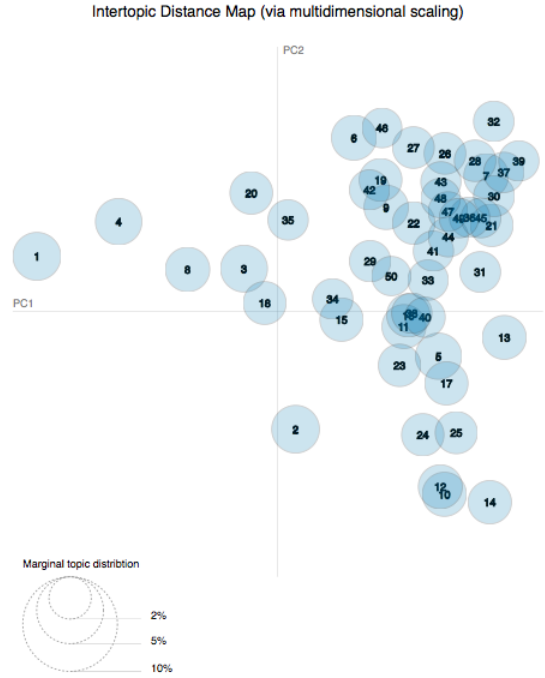
Figure 3 shows the similarity between topics when $K = 20, 50, 70, 100$. The topics are represented by circles in the two-dimensional plane whose centers are determined by computing the distance between topics, and then by using multidimensional scaling to project the inter- topic distances onto two dimensions (Chuang et al., 2012). The size of the circles represent each topics overall prevalence, and the topics are sorted in decreasing order of prevalence. We can see from the plots that many topics are very close to each other when the topic number $K = 70$ or $K = 100$. After a close examination of the meaning of each topic using different number of topics, $K = 50$ tends to give us a better interpretation of the model. Therefore, we end up using 50 topic numbers and we elaborate on the details of topic interpretation when $K = 50$ in the following section.

8.2 Topic Interpretation

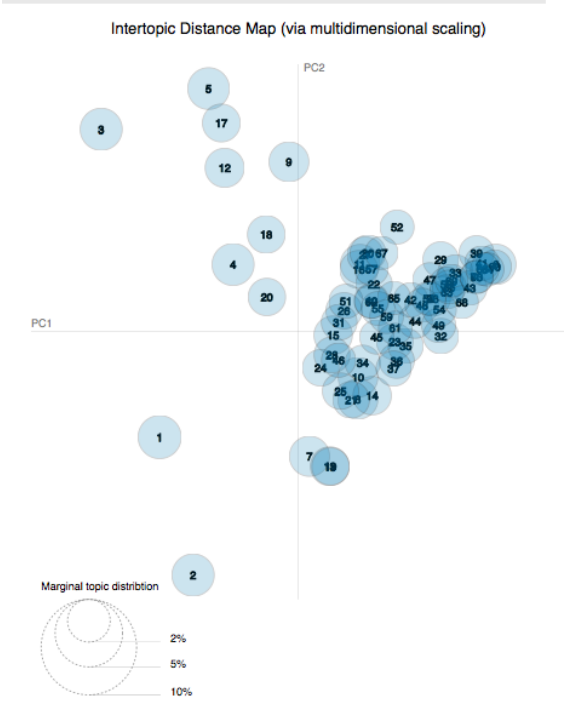
In this section, we talk about the results when fitting a 50-topic model to the Stack Overflow data.

For the topic interpretation, we need to figure out the most useful terms that can be representative of each topic. Here we use *relevance* introduced in Sievert and Shirley (2014) to quantify the usefulness of terms in each topic. First we can see how different values of λ in (1) result in different ranked term lists in Figure 4. In this figure, the terms Topic 1 are ranked based on the *relevance* and corpus-wide frequency and topic-specific frequency

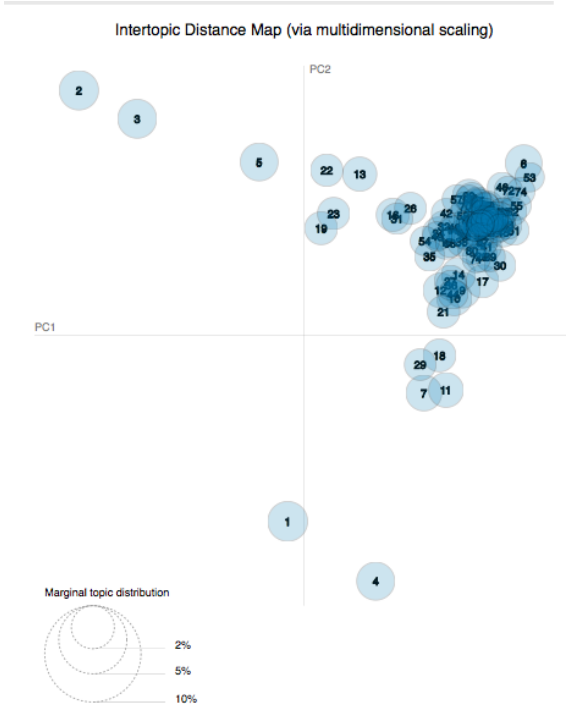
(a) K = 20



(b) K = 50



(c) K = 70



(d) K = 100

Figure 3: Inter-topic distance map with different topic number specification.

are shown as well. The blue bar chars indicate the frequency over the whole corpus and the red ones represent the frequency within the selected topic. i.e. Topic 1. For example, when $\lambda = 1$, the term "code" is selected as the top 30 relevant terms; however, when $\lambda = 0$, the term "code" is no longer in the top list. This is because the term "code" is a globally frequent term, in fact it is the most frequent term over the whole corpus, and its rank is decreased when $\lambda = 0$ (ranking solely by *lift*).

Table 2 lists out the top 15 relevant words to 6 selected topics when topic number $K = 50$. Here the relevance are defined by (1) when λ are set to 0.33. For example, the most relevant words in Topic 13 are "svn", "subversion", "commit", "git", "repository" and so on. All of these are key words for version control. And in Topic 29, the top terms are "image", "element", "color", "background", "height", "left" and so on, these can be interpreted as topics related with CSS.

Rank	Topic 2	Topic 5	Topic 13	Topic 17	Topic 28	Topic 29
1	table	visual	svn	server	vista	image
2	field	studio	subversion	request	virtual	element
3	column	microsoft	commit	http	laptop	color
4	row	dll	git	port	windows	background
5	rows	windows	repository	ip	monitor	height
6	columns	net	merge	client	64	left
7	names	framework	revision	local	xp	colors
8	select	bin	version	host	screen	font
9	vendor	debug	control	response	bit	width
10	primary	msbuild	branch	vpn	ram	text
11	records	installer	cvs	ssl	pc	0000
12	company	win	working	clients	prefer	top
13	record	cs	diff	header	environment	red
14	guid	debugger	copy	url	running	pixel
15	location	assembly	source	ssh	office	space
	SQL database	Microsoft	Version Control	Network	Microsoft OS's/PCs	CSS

Table 2: Top 15 words relevant to 6 selected topics under $\lambda = 0.33$ and $K = 50$.

9 Limitations and future works

We apply LDA method in the sense of unsupervised learning and seeks to discover topics. In many situations, label information are given. For example, for the Stack Overflow data,

we have tags for each posts. In other words, we have access to the content of topics in advance. It thus becomes a problem in supervised learning (classification). Mcauliffe and Blei (2008) propose a supervised LDA method to deal with this problem. It has a very similar structure to LDA in general. Therefore, for future works, we can think of how to combine tag information to the topic models.

More generally, LDA can be applied to model other large scale discrete data. This generalization is meaningful since it introduced a useful method to deal with discrete data by building a probability model instead of defining discrepancy matrix. For example, for image data processing and genetic studies, LDA and sLDA can be useful tool to detect intrinsic clusters, as well as classification problems.

References

Atwood, J. (2009). Stack overflow creative commons data dump. `http://blog.stackoverflow.com/2009/06/stack-overflow-creative-commons-data-dump/`.

Bischof, J. and Airoldi, E. M. (2012). Summarizing topical content with word frequency and exclusivity. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 201–208.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Chuang, J., Ramage, D., Manning, C., and Heer, J. (2012). Interpretation and trust: Designing model-driven visualizations for text analysis. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 443–452. ACM.

Griffiths, T. L. and Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5228–5235.

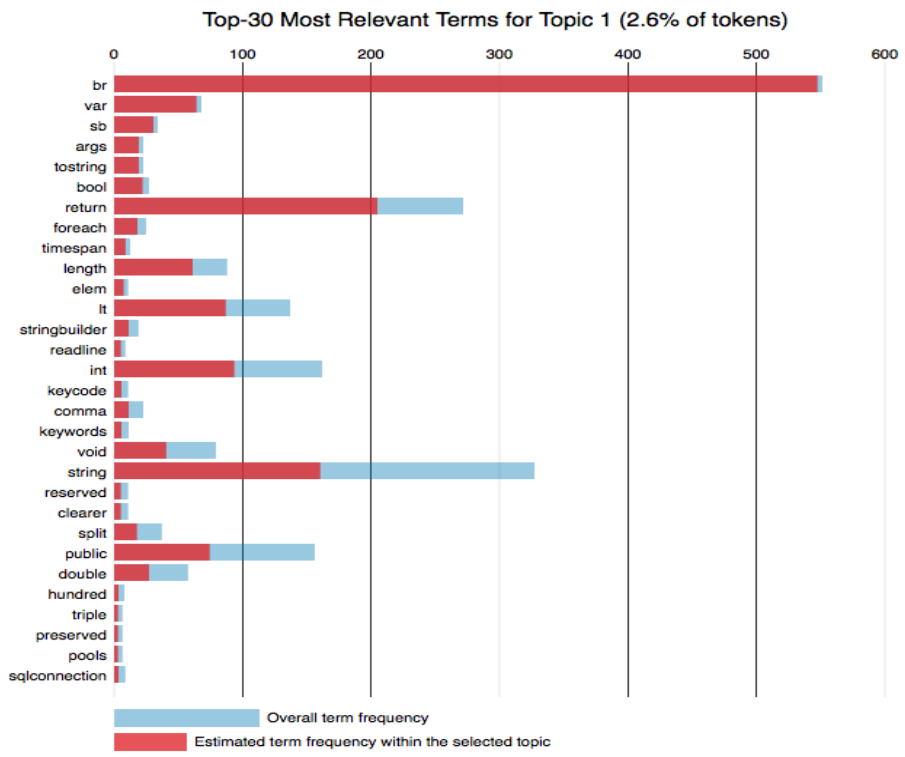
Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM.

Mcauliffe, J. D. and Blei, D. M. (2008). Supervised topic models. In *Advances in neural information processing systems*, pages 121–128.

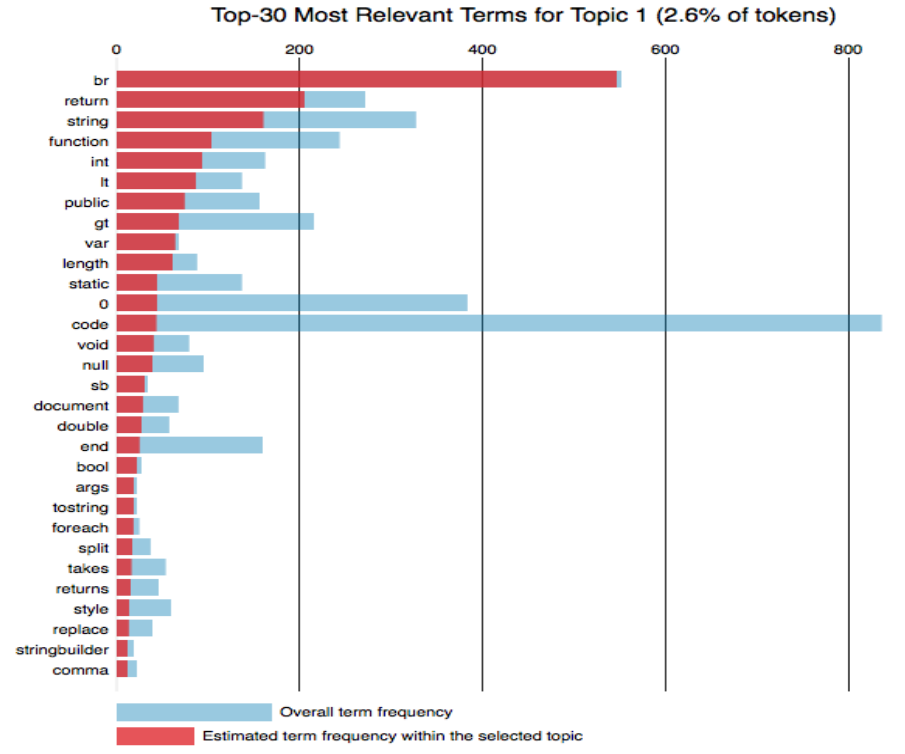
Sievert, C. and Shirley, K. E. (2014). Ldavis: A method for visualizing and interpreting topics. In *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces*, pages 63–70.

Taddy, M. A. (2011). On estimation and selection for topic models. *arXiv preprint arXiv:1109.4518*.

Yao, L., Mimno, D., and McCallum, A. (2009). Efficient methods for topic model inference on streaming document collections. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 937–946. ACM.



(a) $\lambda = 0$



(b) $\lambda = 1$

Figure 4: Top-30 Most Relevant Terms for Topic 1 using (a) $\lambda = 0$ and (b) $\lambda = 1$.