Patterns for topic modeling

Michael Weiss

Carleton University, Ottawa, Canada

weiss@sce.carleton.ca

Abstract

Topic modeling is a probabilistic technique for identifying the latent topics in a corpus of documents. A core challenge in topic modeling is the interpretation of the topics generated by the model. This paper identifies patterns for the creation and interpretation of topic models. The majority of the patterns focus on the latter. The audience for these patterns includes anyone who needs to quickly get an overview of a the content of a large collection of documents, but particularly, students and researchers.

Keywords

Patterns, text mining, topic modeling

1 Introduction

Topic modeling is a probabilistic technique for identifying the latent topics in a corpus of documents. The topics are latent because they are not explicitly stated, but need to be discovered.

A core challenge in topic modeling is the interpretation of the topics generated by the model. To goal of interpretation is to find out what topics the documents in the corpus are about.

This paper identifies patterns for the creation and interpretation of topic models. Table 1 contains a summary of the patterns and Figure 1 shows a map of the links between the patterns.

The patterns were mined from the author's experience using topic modeling, as well as from examples in the literature. This process involved brainstorming contexts, challenges, and solutions to those challenges, and mapping them visually to detect commonalities and links between them. The patterns were also used as the foundation of a tool for exploring topic models.

The audience for these patterns includes anyone who needs to quickly get an overview of a the content of a large collection of documents, but particularly, students and researchers.

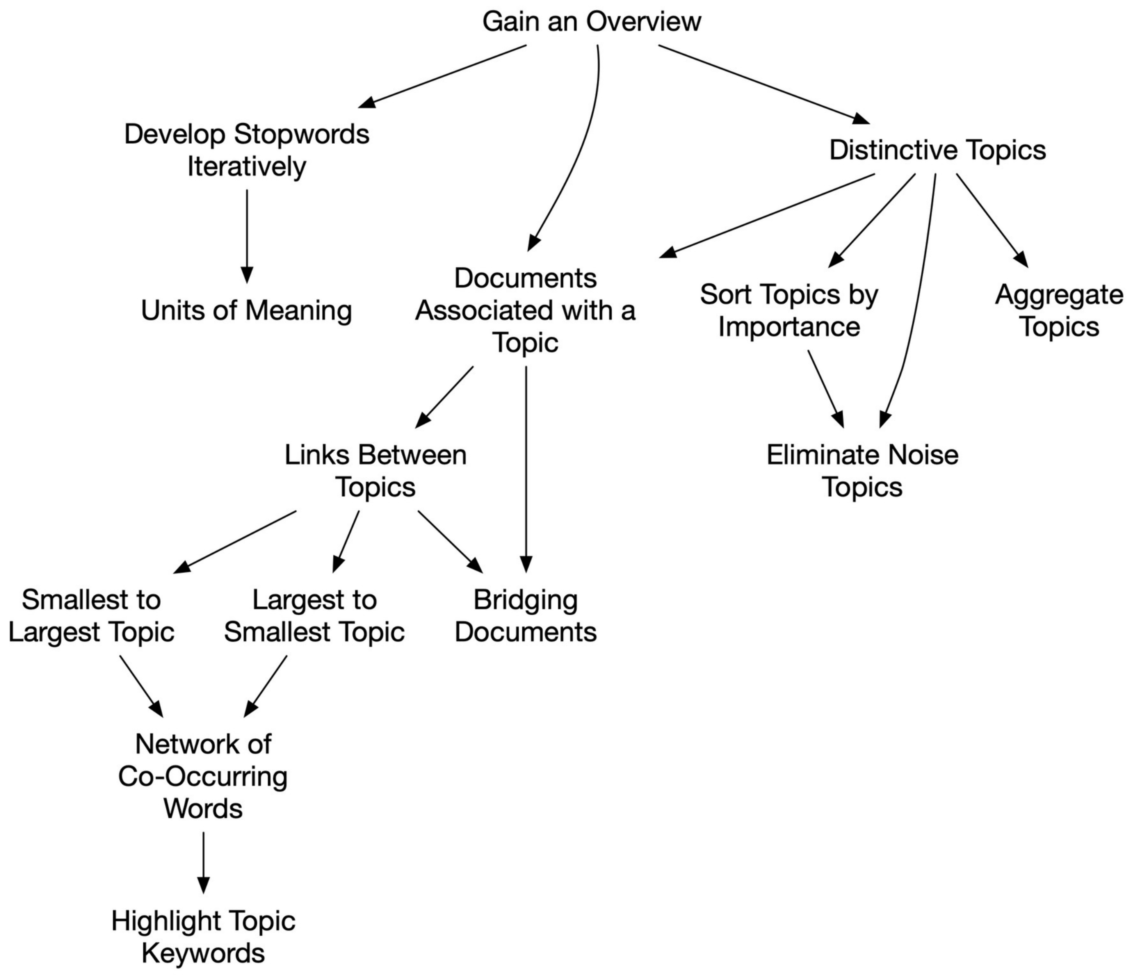

Gain an Overview
Develop Stopwords Iteratively
Distinctive Topics
Units of Meaning
Documents Associated with a Topic
Sort Topics by Importance
Aggregate Topics
Links Between Topics
Eliminate Noise Topics
Smallest to Largest Topic
Largest to Smallest Topic
Bridging Documents
Network of Co-Occurring Words
Highlight Topic Keywords

Figure 1: Pattern map that shows the patterns and the suggested order of applying them

Table 1: Summary of the patterns

Pattern	Description
Gain an Overview	Keeping the content of many documents in your head at the same time is difficult. Therefore, create a topic model to gain an overview of the content of the documents.
Distinctive Topics	Selecting the number of topics can be a bit of an art. If you choose too few topics, the topics will be overly broad, and if you choose too many topics, they will be too specific. Therefore, generate a range of topic models of different sizes and choose the number of topics that maximizes the distinctiveness between topics.
Develop Stopwords Iteratively	Coming up with good stopwords is hard and depends on the domain. Therefore, develop stopwords iteratively as you explore the topic model.
Units of Meaning	Not all units of meaning can be captured by single words. Therefore, include important compound words as single “units” of meaning.
Sort Topics by Importance	You want to get a sense of which topics are most central. Therefore, order the topics by weight.
Documents Associated with a	Keywords only give you a general idea of what a topic is about. Therefore, identify the key documents associated with the topic and read them closely.

Topic	
Links Between Topics	Keywords alone are often not sufficient to distinguish between topics. Therefore, visualize the connections between topics in order to show each topic in the context of other, related topics.
Network of Co-Occurring Words	Trying to grasp the content of multiple documents associated with a topic at once can be taxing. Therefore, visualize a network of the co-occurring keywords in those documents.
Highlight Topic Keywords	Support close reading of the top documents associated with a topic by highlighting topic keywords in the document.
Bridging Documents	Examine documents “bridging” between topics.
Largest to Smallest Topic	Examine topics in order of weight, from largest to smallest.
Smallest to Largest Topic	Examine topics in reverse order of weight, from smallest to largest.
Eliminate Noise Topics	Eliminate irrelevant “noise” topics.
Aggregate Topic Models	Aggregate multiple topic models to improve topic stability.

2 Running example

The use of the patterns will be illustrated with a running example. As part of a recent research project, we needed to get an overview of the current research on the triage of security bugs in open source projects. Triage determines which bugs requires most attention and which developers will be assigned the responsibility of resolving a bug (Anvik et al., 2006). We collected a corpus of abstracts from Google Scholar by searching for the keywords “security”, “bug”, and “triage”. From this we created a shortlist of papers by retaining only papers from a select list of journals and conferences. We then automatically extracted abstracts from the publishers’ websites and manually scanned them to remove duplicates and obviously unrelated papers (for example, papers that discuss triage in a hospital).

To *Gain an Overview* of the corpus create a topic model. This gives you a list of topics and their associated keywords. When you create a topic model, you need to choose the number of topics. Finding *Distinctive Topics* helps you with that task. You are looking for the number of topics that best separates the topics from one another. To make the topics meaningful, you also want to remove uninformative words (called stopwords) from the topics (*Develop Stopwords Iteratively*). To gain a deeper understanding of each topic you next want to read the top *Documents Associated with a Topic*.

3 Patterns

3.1 Gain an Overview

You need to digest the contents of a large number of documents in a corpus.

❖❖❖

Keeping the content of many documents in your head at the same time is difficult.

First of all, it takes a long time to read many documents and you may not have that not much time available. Recalling the contents of a large number of documents as you try to synthesize what those documents are about is also cognitively demanding. Finally, the order in which they read the documents and their prior experience and interests may cause readers to selectively pay attention to some documents and not others, which creates a risk of cognitive bias.

Therefore,

Create a topic model to gain an overview of the content of the documents.

A topic model helps you identify the topics that the documents belong to. These are also known as latent topics, because they may not be not explicitly stated in the documents. Topic modeling works by looking at frequencies and co-occurrences of words within documents and words that are common across documents and that distinguish different topics at the same time. A popular technique for topic modeling is Latent Dirichlet Allocation (Blei et al., 2003). It is a probabilistic method that reconstructs the distribution of topics across documents and that of words within each topic.

The overall flow of creating a topic model is shown in Figure 2. The first step is to create a corpus of documents. These could be abstracts of articles, social media posts, websites, etc. The second step is to create a topic model. This produces a list of topics with associated keywords and topic weights. The weight of a topic indicates how many of the documents in the corpus are explained by this topic. Most topic modeling algorithms (such as LDA) require you to choose a number of topics. Therefore, you need to experiment with different numbers of topics. As you interpret the topics, you also want to remove words that add "noise" to the topics. These words are called stopwords.

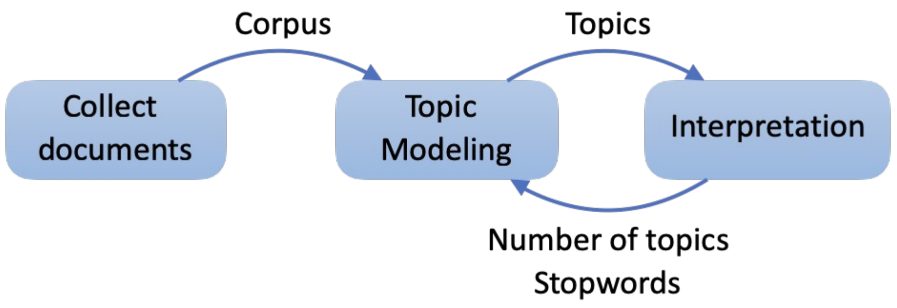
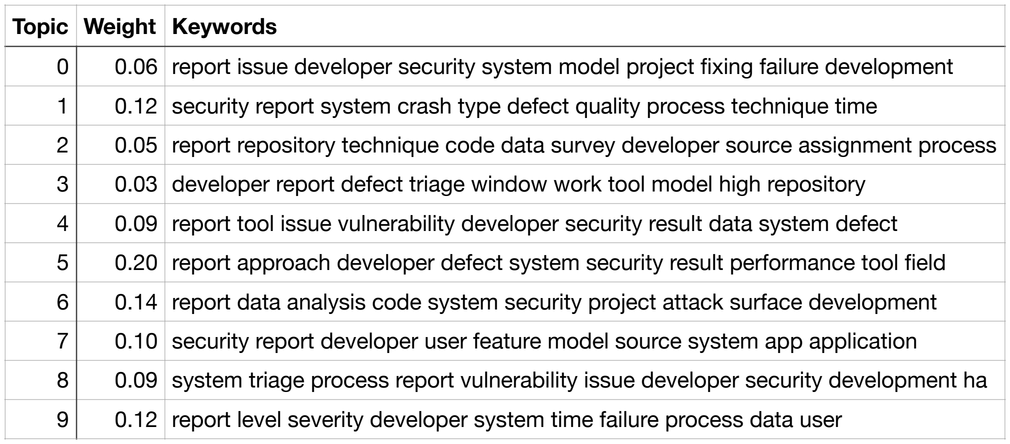


Figure 2: Overall flow of creating a topic model

The outcome of topic modeling is a list of topics with associated weights and keywords. Figure 3 provides an example. This topic model was generated from a corpus of abstracts collected from Google Scholar by searching for the keywords "security", "bug", and "triage". This model has 10 topics. A list of English stopwords and a list of domain-specific words (including words like "bug" and "paper") were applied to the corpus before creating the model. Weights indicate the percentage of documents covered or explained by those topics. Keywords are the top 10 words associated with each topic.



Topic	Weight	Keywords
0	0.06	report issue developer security system model project fixing failure development
1	0.12	security report system crash type defect quality process technique time
2	0.05	report repository technique code data survey developer source assignment process
3	0.03	developer report defect triage window work tool model high repository
4	0.09	report tool issue vulnerability developer security result data system defect
5	0.20	report approach developer defect system security result performance tool field
6	0.14	report data analysis code system security project attack surface development
7	0.10	security report developer user feature model source system app application
8	0.09	system triage process report vulnerability issue developer security development ha
9	0.12	report level severity developer system time failure process data user

Figure 3: List of topics with associated weights and keywords

❖❖❖

Creating a topic model takes far less time than reading a large number of documents in a corpus. The approach also scales easily: a topic model for a corpus 10,000 documents does not take much longer to create than if the corpus only had 100 documents. As a semi-automatic method, topic modeling does not eliminate the need for interpretation, but expects the user to be involved. However, by classifying documents into topics, it provides input into the interpretation of the corpus (Asmussen et al., 2019; Yla-Antilla et al., 2021). It facilitates the process of detecting structure in a corpus (Lindgren, 2020). It can also help identify topics that we may not have detected otherwise (Lindgren, 2020).

There are several issues that need to be considered to create a good topic model and to come up with meaningful interpretations. First, you need to choose the number of topics. Your goal in choosing the number of topics should be to create *Distinctive Topics*. Next, not all of the keywords are informative. Removing such "noise" words will lead to topics that are easier to interpret. Since you usually don't know up-front which keywords are noise, you should *Develop Stopwords Iteratively.* Finally, while the keywords associated with each topic give you an overall sense of what the topics are about, the keywords on their own do not provide you with a full understanding of the topics. To gain a deeper understanding of each topic, you should also read the top *Documents Associated with a Topic*.

There are many known uses of topic modeling. Among other applications, topic modeling has been used to prepare literature reviews (Asmussen et al., 2019), identify topics in news articles (Di Maggio et al., 2013), study framing in public debates (Yla-Antilla et al., 2021), and extract the key concepts of a domain and their emergence over time (Muegge & Weiss, 2019). In each of these cases, researchers were faced with a large corpus of documents and an incomplete existing understanding of a domain.

3.2 Distinctive Topics

You are trying to *Gain an Overview* over a corpus of documents using topic modeling. Most topic modeling algorithms (such as LDA) require you to choose a number of topics.

❖❖❖

Selecting the number of topics can be a bit of an art. If you choose too few topics, the topics will be overly broad, and if you choose too many topics, they will be too specific.

There is no single criterion for choosing the number of topics. The main driver for the number of topics will be the number of documents in your corpus. Generally, the higher the number of topics, the more fine-grained your topic model will be. However, if you choose too many topics, the topics will start to look very similar, especially if the corpus is small. The number of topics will also depend on what types of distinctions you are interested in (eg given a corpus of social media posts, one goal might be to identify what types of content the posts discuss, and a very different goal would be to identify which posts express positive and which negative sentiments).

Therefore,

Generate topic models of different sizes and choose the number of topics that maximizes the distinctiveness between topics.

If you have a large number of documents (>> 100), you can create a fine-grained topic model, whereas you should aim for a small number of topics if only have a few documents (< 100). A large topic model can have 50 or even 500 topics, whereas a small model might only have 5 topics. A good strategy is to generate topic models with 5, 10, 15, etc. topics. As you increase the number of topics, some topics will stay the same (that is, they are stable), whereas other, more specific topics will emerge. You can stop when the topics become too fragmented. A good default is to use 10 topics (Mathew et al., 2017).

In general, it is better to aim for a smaller number of topics, at least initially. This gives you an overall sense of the topics in the corpus. As you explore more refined topic models, you can assess the distinctiveness between topics either through manual inspection or by using performance metrics. As an example of the former, you can evaluate how cohesive the topics are by checking the associated top keywords (do they make sense together?). As an example of the latter, you could compute the coherence of the topic model using one of several metrics and look for an optimal value of this metric.

As alluded to above, what you consider “distinctive” depends on the kinds of insights you want to extract from the corpus. Topic models of different size can create different distinctions. Di Maggio et al. (2013) describe a case where two topic models of different size created from a corpus of news on funding of the arts were considered. The first topic model gave insights into the different types of arts. If that is what the authors were interested in, this topic model would be the right one. The second topic model with a different number of topics produced insights into funding policy. Since the authors’ goal was to understanding policy, they proceeded with the second topic model.

❖❖❖

Often, it is difficult to distinguish between topics based on their associated keywords alone. In this case, it helps to visualize the *Links Between Topics* in a topic co-occurrence network. When the topics in such a network formed by the relationships between topics form clear clusters, these clusters give you a sense of the broad thematic groups within the corpus, and you can now focus your attention on the more nuanced distinctions between topics within each thematic cluster. For example, in the running example, the topics include topics related to themes like security, process, and data.

Since what you consider “distinctive” depends on your interests, ie on the question you are trying to answer by creating the topic model, the absolute number of topics is not as relevant, but rather how many meaningful topics are being created that help you answer your question. Topics that are not relevant to your question can be considered “noise” topics (*Eliminate Noise Topics*). Consider again the example of social media posts discussed above: if you are interested in the types of content discussed, then topics related to the sentiments of the posts are not relevant to your question.

When exploring the topics in a topic model, it also helps to *Sort the Topics by Importance*. This gives you a sense of how representative each topic is of the corpus. As you examine individual topics, the keywords give you an initial idea of what each topic is about. However, the keywords alone are usually not sufficient to interpret the topic, that is, to assign a label or description to the topic. To form a better impression of what a topic is about, you should explore the *Documents Associated with each Topic*.

3.3 Develop Stopwords Iteratively

You are trying to *Gain an Overview* over a corpus of documents. You want to make the topics distinct by removing stopwords (words that add “noise” to the topics).

❖❖❖

Coming up with good stopwords to remove is hard and depends on the context.

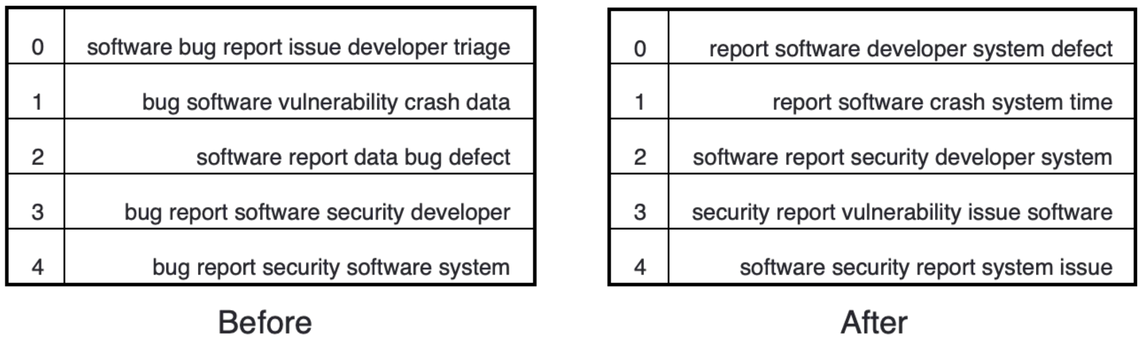
Topics should be distinct. One way of making topics more distinct is to remove words that just add “noise” to the topics. Good stopwords are often words that are central to a domain but also frequent. You do not want such words to dominate a topic model. However, you also do not want to remove too many central words, as this can introduce a bias towards what you view as important. Some words also have multiple meanings, and may be used in different senses within the same corpus. While a common practice, reusing the same stopword list across different corpora may also cause you to exclude words that may be highly relevant in a specific context, and thus lose nuances of meaning in the model.

Therefore,

Develop stopwords iteratively as you explore the topic model.

Stopwords help create topics that are easier to interpret and distinguish from other topics. They can include both common English words like “the” (which can be applied to any corpus) and words that are unique to the domain of the corpus. Candidates for stopwords should include both frequent and infrequent or rare terms. You want to introduce domain-specific stopwords iteratively and test their impact on the generated topic model. You should also use stopwords sparingly. Often a short list of stopwords achieves the desired effect without unduly biasing the model.

In the example on the left of Figure 3, all topics contain the word “bug”. This topic model was generated from a corpus of abstracts that match the keywords “security”, “bug”, and “triage”. Knowing that, you can remove the word “bug”, as it does not add much information for creating the topics. Even though the word “bug” has multiple meanings, in this corpus, it is only used in one sense – an “issue”. The right side of Figure 2 shows the topics that result from removing the word “bug”.



0	software bug report issue developer triage
1	bug software vulnerability crash data
2	software report data bug defect
3	bug report software security developer
4	bug report security software system

Before

0	report software developer system defect
1	report software crash system time
2	software report security developer system
3	security report vulnerability issue software
4	software security report system issue

After

Figure 3: Effect of using removing a stopword (“bug”)

❖❖❖

Developing your stopwords iteratively will provide you with immediate feedback about the impact of stopwords as you consider adding them to the list of stopwords. When you add stopwords one at a time, rather than all at once, you can also see which stopwords most affect the topics generated. Following this pattern will also prevent you from imposing your preconceptions on the corpus. This allows the corpus to speak for itself and you may discover topics that you may not have expected.

There is one case, however, that requires special attention. Sometimes multiple words used together have a specific meaning. Take, for example, the term “security bug”. While you may want to remove the word “bug” from the corpus, you may want to retain instances of that word where it is combined with the word “security”. Such compound words can be treated as special *Units of Meaning*.

3.4 Units of Meaning

You are selecting stopwords to remove from the corpus (*Develop Stopwords Iteratively*). You want to capture meaningful combinations of words in the documents.

❖❖❖

Not all units of meaning can be captured by single words.

Often multiple words together form units of meaning. If you only individual words when creating the topic model, the special meaning of combined words will be lost. When you use stopwords, you may also accidentally remove word combinations that you want to retain, even though the component words may be considered noise if encountered individually. A common heuristic is to include all n-grams (sequences of n words) as words, but that will not only produce meaningful combinations.

Therefore,

Include important compound words as single “units” of meaning.

Use lemmatization to identify compound words in the documents and replace them by a single token representing the combination of words in the compound word. Lemmatization is primarily a natural language processing technique for extracting the base form of words by removing inflectional endings. An example of this is changing the plural “bugs” to the singular “bug”. However, lemmatization can also be used to detect compound words in a sentence (eg the word combination “security bug”).

Figure 4 shows the effect of defining the combination of “bug” and “report” as a compound word on the topic model. The compound word is shown as “bug_report”. This is a continuation of the example from Figure 3. While individual occurrences of the word “bug” have been removed as stopwords, the combination “bug report” will be retained.

0	bug_report software developer security system approach model result study report
1	software developer system security process bug_report development paper vulnerability severity
2	system security data software issue crash time failure bug_report user
3	security developer software defect data tool paper study issue application
4	security software system issue bug_report data tool study project developer

Figure 4: Effect of defining compound words

❖❖❖

Applying this pattern ensures that word combinations of interest are retained in the corpus, even though the individual words may be removed as stopwords, if they are found on their own in the text.

One challenge applying this pattern is to discover meaningful word combinations. Text summarization techniques that extract key phrases from a text can be used to suggest compound words.

3.5 Sort Topics by Importance

You have identified *Distinctive Topics* in the corpus. However, to interpret a topic model, you also need to have a sense of how representative these topics are of the corpus.

❖❖❖

You want to get a sense of which topics are most central.

Exploring a topic model by examining the topics in the order in which they were generated (starting with topic 0, then topic 1, etc.) may lead you to give topics that are not representative of the corpus too much weight in your interpretation. It is easy to read too much into such “small” topics. Also, not all topics are meaningful, because they contain noise (for example, a topic made up of HTML tags).

Therefore,

Order the topics by weight.

To find the weight of each topic, identify the topics each document belongs to and the degree to which it belongs to those topics (*Documents Associated with each Topic*). The distribution of topics across documents can be found in the document-topic matrix. The weight of a topic is the average degree to which documents belong to the topic. It provides an indicator of how central the topic is. The higher the topic weight, the more documents in the corpus are represented by this topic.

Figure 5 shows the topic model from Figure 3 with the topics sorted by weight. Topic weights can be used to suggest the order in which the topics should be examined. Typically, you examine the topics either in the order from *Largest to Smallest Topic* or from *Smallest to Largest Topic*. When you explore topics in a top-down manner, you start with the most representative topics. When you explore them bottom-up, you start with topics that are easier to interpret, as there are comparatively few documents associated with them. Sorting topics by weight also helps *Eliminate Noise Topics* that are too small.

Topic	Weight	Keywords
5	0.20	report approach developer defect system security result performance tool field
6	0.14	report data analysis code system security project attack surface development
1	0.12	security report system crash type defect quality process technique time
9	0.12	report level severity developer system time failure process data user
7	0.10	security report developer user feature model source system app application
4	0.09	report tool issue vulnerability developer security result data system defect
8	0.09	system triage process report vulnerability issue developer security development
0	0.06	report issue developer security system model project fixing failure development
2	0.05	report repository technique code data survey developer source assignment
3	0.03	developer report defect triage window work tool model high repository

Figure 5: List of topics sorted by weight

❖❖❖

Ordering topics by weight provides you with a sense of how representative the topics are of the corpus. This, then, guides you through how you explore those topics and which ones you will consider at all.

One word of caution, though. There could be good reasons to retain topics with low weights in your exploration of the topic model. A low weight does not necessarily mean that the topic is not of potential interest. Consider, for example, topics extracted from a corpus of research articles. Some of these articles will be very recent and may discuss new topics that are perhaps not yet strongly present in the corpus. However, these could indicate important future trends and may thus merit further study.

3.6 Documents Associated with a Topic

You are trying to *Gain an Overview* over a corpus of documents using topic modeling. Just looking at the keywords associated with a topic does not provide you with a full understanding of the topic.

❖❖❖

Keywords only give you a general idea of what a topic is about.

Keywords are usually not sufficient to interpret a topic, that is, to assign a label or description to it. To get a deeper understanding of a topic you also want to look at the documents associated with a topic. Technically, each document belongs to multiple topics. However, for most documents one or two topics dominate. This means that some documents are going to be more representative of a topic than others.

Therefore,

Identify the key documents associated with the topic and read them closely.

You can identify the key documents for a given topic from the document-topic matrix. Besides the keywords for each topic and the topic weights, the document-topic matrix is the other key output of a topic model. Each row in the document-topic matrix represents one document and the distribution of topics for that document. Figure 6 shows an example of a document-topic matrix.

Sorting the documents by the degree to which they are associated with the topic – in the order from highest to lowest – gives you the list of documents that are most representative of that topic. Reading those documents will give you a better idea what a topic is about. As you review the documents associated with a topic, ask yourself what these documents have in common.

	Topics									
Document	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0.0	0.99	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.29	0.0	0.0	0.0	0.0	0.0	0.70
2	0.32	0.24	0.0	0.0	0.43	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.38	0.0	0.0	0.61	0.0
4	0.0	0.0	0.0	0.0	0.0	0.99	0.0	0.0	0.0	0.0

Figure 6: Distribution of topics across documents (only showing the top five documents)

❖❖❖

Applying this pattern allows you to develop an understand of a topic that goes beyond the keywords associated with the topic. Still, grasping the content of multiple documents at once can be hard. Tapping into the human capability of easily processing visual information, you can alternatively summarize the content of the documents in a *Network of Co-Occurring Words*.

Note that a document can belong to multiple topics. It is not uncommon for a document to have a primary, secondary, and even tertiary topic. Reading the documents associated with a single topic does not help you understand connections between the topics. For that, apply *Links between Topics*.

4 Conclusion

In this paper I documented the beginning of a pattern language for topic modeling, a technique for extracting the underlying topics or themes from a corpus. The remaining patterns identified in Table 1 will be documented in future work.

Acknowledgments

I thank Mary Lynn Manns for her helpful comments during the shepherding phase. They were very much to the point.

References

Anvik, J., Hiew, L., & Murphy, G. (2006). Who should fix this bug?. International Conference on Software Engineering (ICSE), 361-370. ACM.

Asmussen, C. B., & Møller, C. (2019). Smart literature review: a practical topic modelling approach to exploratory literature review. Journal of Big Data, 6(1), 1-18.

Blei, D. M., Ng, A. Y., & Jordan, M. I. 2003. Latent Dirichlet Allocation. Journal of Machine Learning Research, 3: 993-1022.

DiMaggio, P., Nag, M., & Blei, D. (2013). Exploiting affinities between topic modeling and the sociological perspective on culture: Application to newspaper coverage of US government arts funding. Poetics, 41(6), 570-606.

Lindgren, S. (2020). Data Theory. Polity.

Mathew, G., Agrawal, A., & Menzies, T. (2017). Trends in topics at SE conferences (1993-2013). International Conference on Software Engineering Companion: 397-398. IEEE/ACM.

Yla-Anttila, T., Eranti, V., & Kukkonen, A. (2021). Topic modeling for frame analysis: A study of media debates on climate change in India and USA. Global Media and Communication, 1-22.