

Natural-language processing (NLP) and indexing, Part 2. Topic modeling

Donald Howes

This article explores how topic modeling can support book index creation, comparing a classic machine learning method (Latent Dirichlet Allocation or LDA) with a large language model (LLM) approach (BERTopic, which builds on SBERT-derived sentence embeddings). Using the public-domain text ‘The cliff ruins of Canyon de Chelly, Arizona’ as a test corpus, the article covers document preprocessing, pipeline construction, and a suite of visualizations that help interpret the latent topics each model discovers. The results show that both techniques generate coherent, interpretable topics, but BERTopic yields richer, more nuanced topic clusters as it leverages larger text spans and preserves grammatical structure. Consequently, the author recommends LLM-based topic modeling over traditional LDA for topic discovery in book indexing.

Introduction

This article follows on from the first in this series, which provided an overview of natural language processing (NLP) and natural language understanding (NLU) tasks, and how they may be applied to the creation of book indexes (Howes, 2025). Here, the technique of *topic modeling* is explored, using two different methods. The first is a traditional machine learning (ML) approach, using Latent Dirichlet Allocation (Blei et al., 2003), while the second uses BERTopic (Grootendorst, 2022), a topic modeling derivation of the BERT (Devlin et al., 2018) large language model (LLM).¹

The sample manuscript used for both the LDA and BERTopic examples is ‘The cliff ruins of Canyon de Chelly, Arizona’ by Cosmos Mindeleff (1897), the first major report on the Ancestral Pueblo ruins in what is now Canyon de Chelly

Donald Howes has an academic background in archaeology. He has worked as a software engineer for companies large and small and has owned and operated an online antiquarian bookstore. He currently works as a freelance indexer, specializing in back-of-book and embedded indexes for scholarly and trade publications. He is a member of the Indexing Society of Canada/Société canadienne d’indexation (ISC/SCI), and of the American Society for Indexing (ASI).

Email: dwhowes@shaw.ca

National Monument in Arizona. This document is in the public domain and can be downloaded from Project Gutenberg.²

Topic maps

There have been several articles published in *The Indexer* dealing with the application of topic maps in indexing. Northedge (2008) discusses topic maps, an ISO-standard model born from back-of-book indexing,³ which provides a formal way to store index information independently of its visual form.⁴ By separating content from presentation, a single data store can be transformed into printed, web, mobile, or audio versions, eliminating duplication, errors, and re-editing.

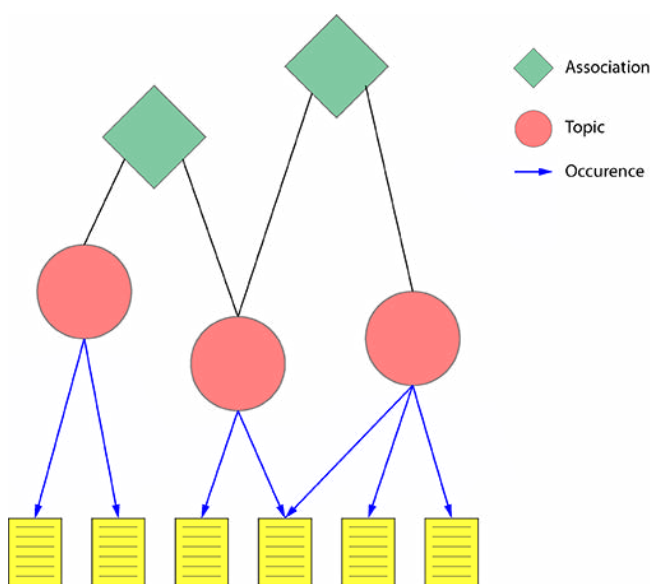


Figure 1. Basics of topic maps (see endnote 4 for an explanation of terms)

Biezunski (2018) points out that the topic maps model that emerged from the SGML/XML communities in the early 1990s makes indexes, thesauri, taxonomies, and other navigational aids interoperable across systems. A topic map captures the semantic structure of an index (terms, occurrences, associations, and scopes), allowing traditional print indexes to be expressed as graph-based topic maps (Figure 1). By representing index entries as topics, occurrences, and associations, topic maps reveal the underlying semantics of indexing.

Finally, Provo (2019) reports on the Enhanced Networked Monographs (ENM) project, which aimed to provide free web access to 110 scholarly monographs and to transform their back-of-book indexes into a unified, machine-readable meta-index. Using the Topic Curation Toolkit, built on the topic maps model, individual EPUB indexes were parsed and combined into a single ENM topic map.

What is topic modeling?

As applied to book indexing, topic modeling can be viewed as an automated technique that attempts to discover the topic map latent in the documents under investigation. Topic modeling is a statistical modeling technique that uses unsupervised learning⁵ to identify latent semantic clusters in a document corpus (Pykes, 2023),⁶ using the assumption that semantically similar documents are indicative of that underlying structure (Figure 2). These clusters must be assessed by the individual undertaking the modeling to see whether they can be interpreted as topics. This is done by analyzing the terms (individual words, bigrams, or trigrams) within each cluster and assigning a topic designation.

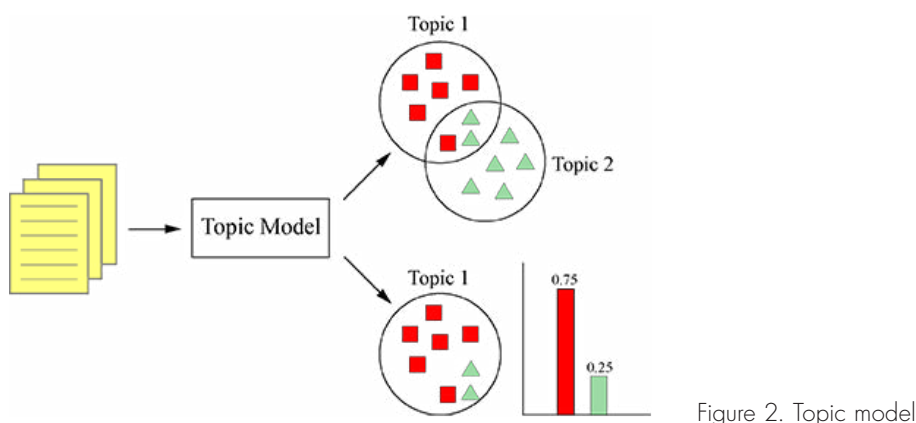


Figure 2. Topic model

The statistical nature of this technique is self-evident – similar words can be present in more than one cluster at different frequencies. Conversely, documents may contain more than one topic, again at different frequencies. A document is classified using its dominant topic (the topic with the highest frequency) to associate it with a topic.

Topic modeling implementation

The topic modeling pipeline is roughly the same for both traditional ML and LLM implementations (Figure 3). Where they vary is in the amount of data preprocessing

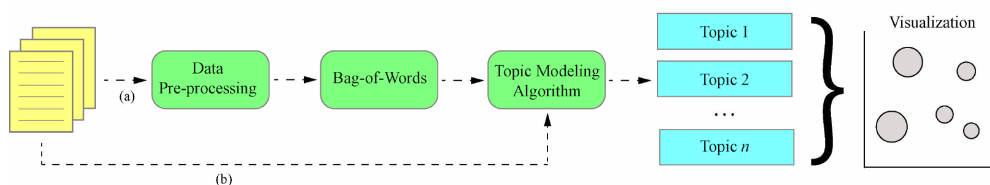


Figure 3. Topic modeling pipeline.

Note: data preprocessing may not be required: (a) path followed by traditional techniques such as LDA, (b) path followed by LLM techniques, such as BERTopic, which are designed to work with unprocessed data.

required to create input texts in the appropriate format for the topic-modeling algorithm. The implementation of this pipeline for each of the two techniques explored in this article is covered in detail in the appendices (see Appendix A for LDA and Appendix B for BERTopic).

Preprocessing

Text preprocessing is the essential process of cleaning and transforming unstructured text data into a structured, normalized format that machines can understand and analyze. Key techniques include tokenization, lowercasing, stop word removal, and lemmatization to reduce words to their root forms. This standardization and noise reduction improve the accuracy and efficiency of traditional NLP ML models.

Bag-of-words (BoW)

The BoW model is a technique for representing text data numerically by counting the frequency of each word in a document, ignoring word order and grammatical context. It creates a fixed-length, sparse vector where each dimension corresponds to a unique word in the vocabulary (summed over all documents). The value in that dimension indicates how many times the word appears in the text of a document. This vector allows ML models to process and analyze text.

Visualization

Topic and document visualizations can be accomplished in a number of different ways. The Python package pyLDAVis⁷ uses LDA topic data to create a bubble chart of topics and the most common terms associated with each topic. Custom plots can be created using packages such as plotly⁸ or matplotlib.⁹ BERTopic has a number of built-in visualization methods¹⁰ that produce output as either plotly or matplotlib figure classes. Additional custom visualizations can be created from BERTopic topic data using plotly or matplotlib.

Latent Dirichlet Allocation (LDA)

LDA (Figure 4) is a generative probabilistic model that uses Bayesian inference (Wikipedia, 2025) to find the underlying topics in a corpus of documents. It assumes that these documents were created through a random sampling of pre-existing topics. It then attempts to reverse engineer this process in order to discover those pre-existing topics (Murel and Kavlakoglu, 2024).

Because of its probabilistic nature, LDA assumes that each document is a combination of a small number of latent topics, and that each word is generated by a particular topic. Documents are mapped to a list of topics by assigning each word in the document to different topics. The model creates a BoW vocabulary of the unique words in the corpus. This model deals strictly with word frequency and co-occurrence within documents. See Appendix A for a detailed discussion of LDA.

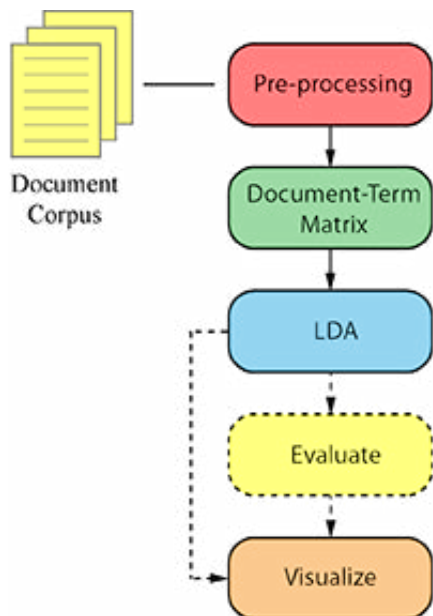


Figure 4. LDA pipeline

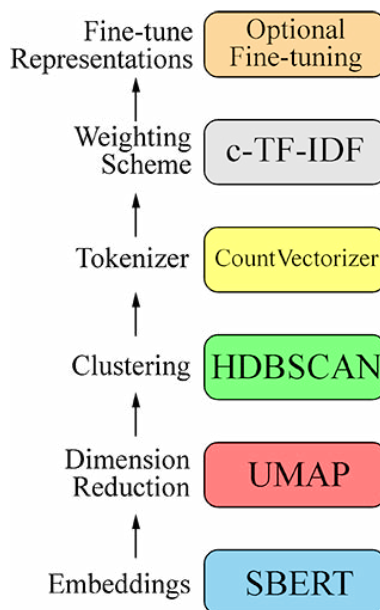


Figure 5. BERTopic pipeline with default modules

BERTopic

BERTopic (Grootendorst, 2022) is a topic-modeling technique that generates sentence embeddings using pre-trained transformer-based language models, clusters these embeddings, and generates dense topics with a class-based TF-IDF¹¹ procedure (c-TF-IDF). Important key phrases are associated with each topic (Figure 5).

BERTopic is designed to be highly modular, with the capacity to replace the module at each level with an alternative method. For example, SBERT could be replaced by SpaCy, and HDBSCAN by K-Means, and so on. This allows the BERTopic user to create a wide range of different topic model implementations – although for the example here the defaults are used. See Appendix B for a detailed discussion of this implementation.

NLP pipelines

An NLP pipeline is a sequence of processes that transform raw text into structured data. The pipeline typically involves such steps as text preprocessing, feature extraction, and modeling. The pipeline is designed to handle the complexity of human language, ensuring that data is transformed for specific NLP tasks.¹² Two pipelines have already been shown, those for LDA (Figure 4, Figure A1) and BERTopic (Figure 5, Figure B1). A third, SpaCy, remains to be examined, which is used in text preprocessing and exploratory data analysis.

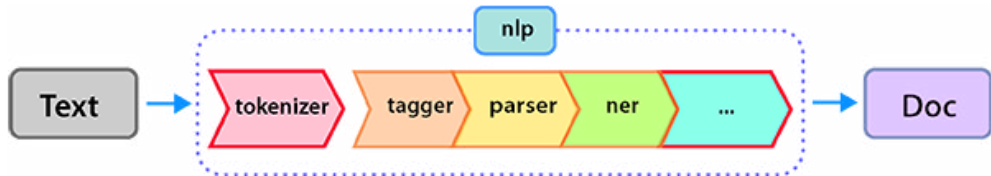


Figure 6. SpaCy NLP pipeline

SpaCy pipeline

SpaCy is a modular and highly flexible library for performing NLP tasks.¹³ The SpaCy pipeline includes a number of steps, only the first of which is mandatory (Figure 6).

It is possible to create an instance of a SpaCy `nlp` object with nothing but the tokenizer,¹⁴ which creates a `doc` object with only token information. The other default pipeline components can be individually removed or replaced. Custom components can also be added to the end of the default pipeline (following `ner`).

The components of the default pipeline are:

- **tokenizer:** segments the text into tokens¹⁵
- **tagger:** predicts part-of-speech tags for any part-of-speech tag set. Taggers are language-specific and trainable
- **parser:** analyzes the grammatical structure of a sentence by identifying the syntactic relationships between words. It assigns a ‘head’ and ‘dependent’ relationship to each pair of related words and labels the dependency type¹⁶
- **ner:** performs named entity recognition (ner) on the text.¹⁷

Exploratory data analysis (EDA)

The *Text Exploration* page of both example applications¹⁸ provides basic EDA information about the corpus of documents being used for topic modeling. This gives a different view of the data set as it examines the structure of the corpus as a whole, rather than the structure of latent topics derived from the corpus and the relationship of those topics to documents. All the EDA techniques on this page allow the user to display more or less information.

Document structure

Basic document structure is shown by bar chart distributions of the number of characters¹⁹ and words per document (Figure 7).

Most common words

The most common words in the corpus are displayed using a vertical bar chart (Figure 8).

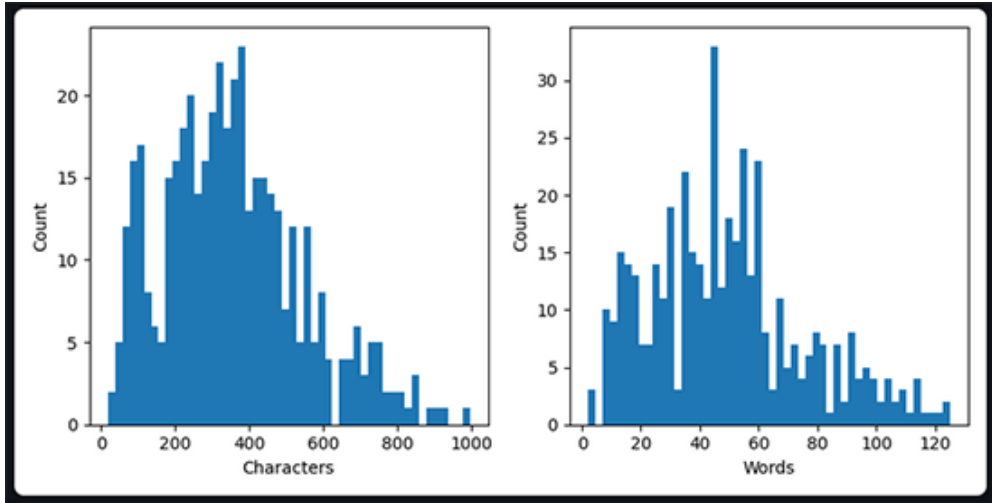


Figure 7. Structure of documents

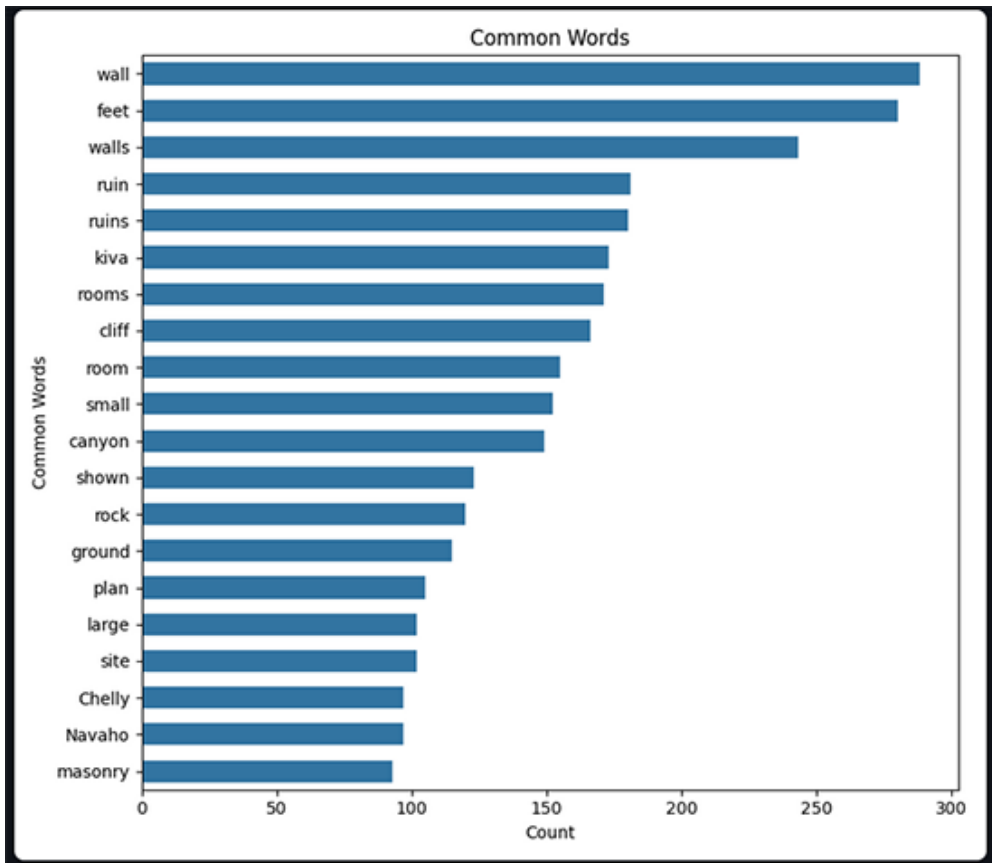


Figure 8. Most common words

Note: Readers of the print volume are advised to download the digital version of this article for better resolution and readability of the more detailed figures.

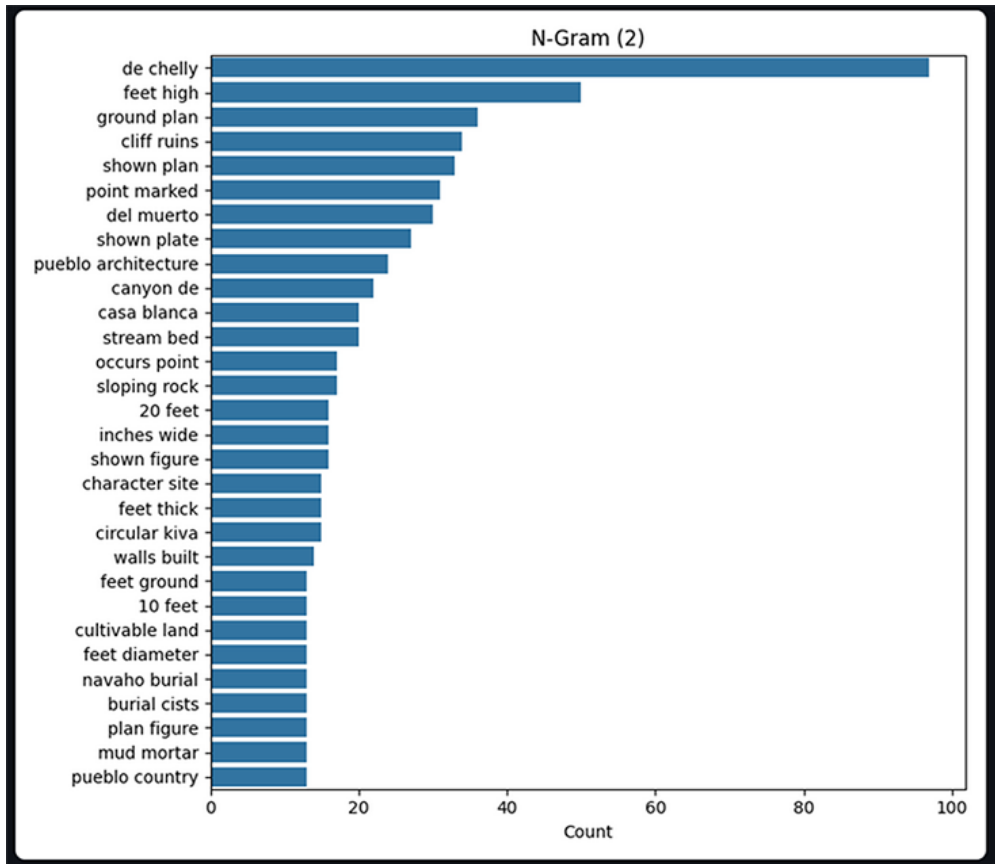


Figure 9a. Most common bigrams

The list of common words is dominated by words used to describe the architecture of the Ancestral Puebloan ruins present in Canyon de Chelly and possibly the geography of the canyon itself. Note, however, the word ‘Navaho’ is present in the manuscript approximately 100 times, suggesting that a later Navaho occupation of the valley was detected.

Ngrams

The user can examine ngrams up to five words long, but bigrams (two words) and trigrams (three words) usually provide the most information as longer repetitive word sequences are relatively uncommon (Figure 9).

As with common words, the charts are dominated by phrases dealing with Ancestral Puebloan architecture, the geography of the valley, and the wider Puebloan culture area. Because these are phrases, there is a degree of descriptive

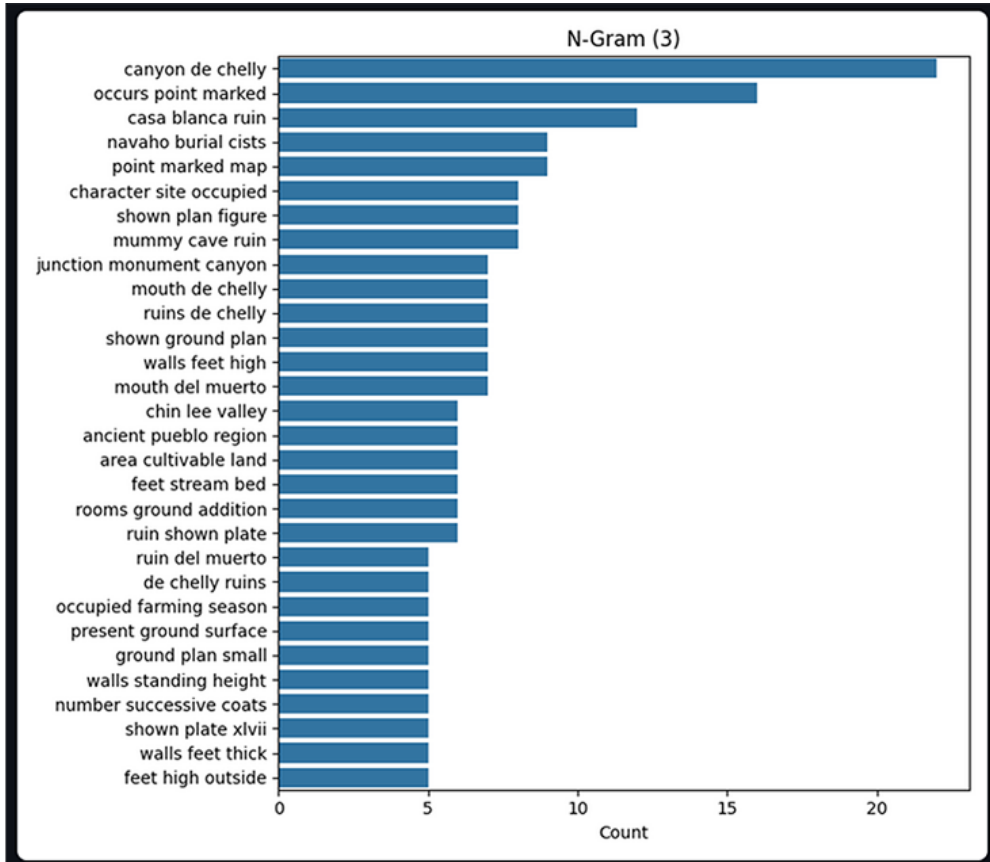


Figure 9b. Most common trigrams

specificity about Puebloan architecture (‘casa blanca’, ‘circular kiva’, ‘mud mortar’, ‘mummy cave ruin’, ‘number successive coats’), agricultural practices (‘cultivable land’, ‘area cultivable land’), geography (‘stream bed’, ‘sloping rock’, ‘junction monument canyon’, ‘chin lee valley’, ‘present ground surface’), and the wider region (‘pueblo country’, ‘ancient pueblo region’). The later Navaho occupation of the valley is also shown in more detail (‘navaho burial’, ‘navaho burial cists’).

Named entities

The occurrence of selected SpaCy named entities can be displayed (Figure 10).²⁰ Which NER category(s) are able to provide useful information is strongly dependent on the manuscript subject under investigation. For the example manuscript, the LOC category is best for providing interpretable results. The results of the LOC analysis show that the manuscript has an emphasis on Canyon de Chelly itself,

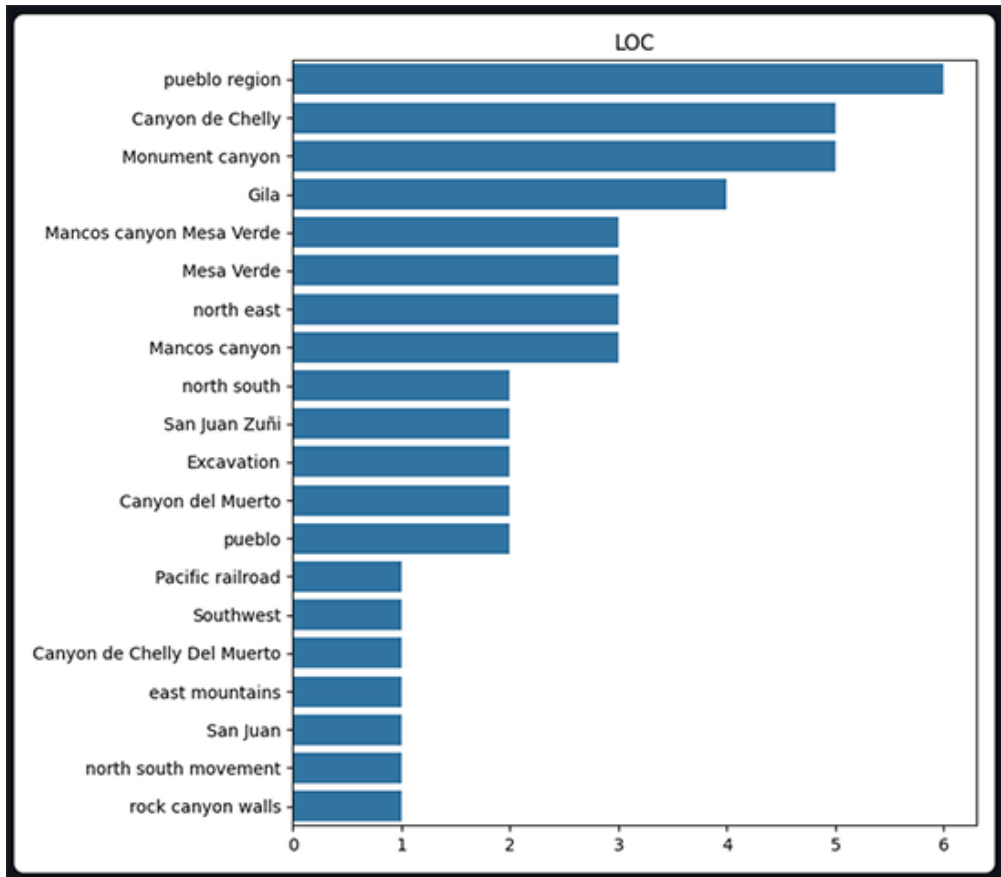


Figure 10. Occurrence of SpaCy NER LOC category

and on the comparison of Canyon de Chelly with both other Ancestral Puebloan areas (‘Mancos canyon Mesa Verde’, ‘Mesa Verde’, ‘Mancos Canyon’) and modern Puebloan settlements (‘Gila’, ‘San Juan Zuni’).

Visualizations

The example applications provide a number of different visualization methods that can be used both to fine-tune topic generation and, when fine-tuning is finished, to aid the user in understanding the topics.²¹ Methods are also provided that allow the user to examine how topics are distributed among the documents of the corpus.

Topic visualizations

Topic visualizations included in the example applications are topic maps, heat maps, bar charts, word clouds, sunburst charts, treemap charts, and hierarchical cluster dendrograms.

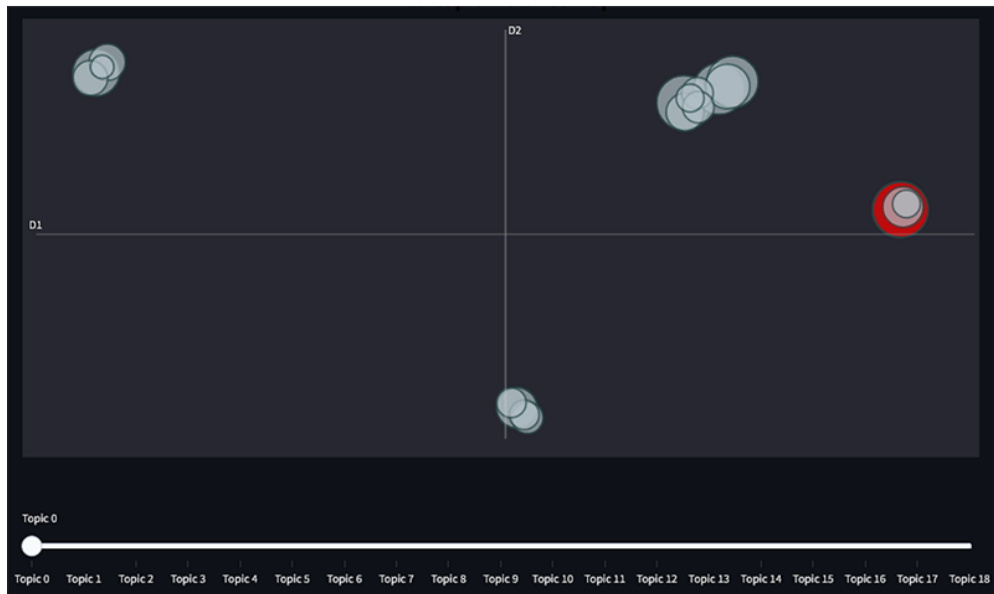
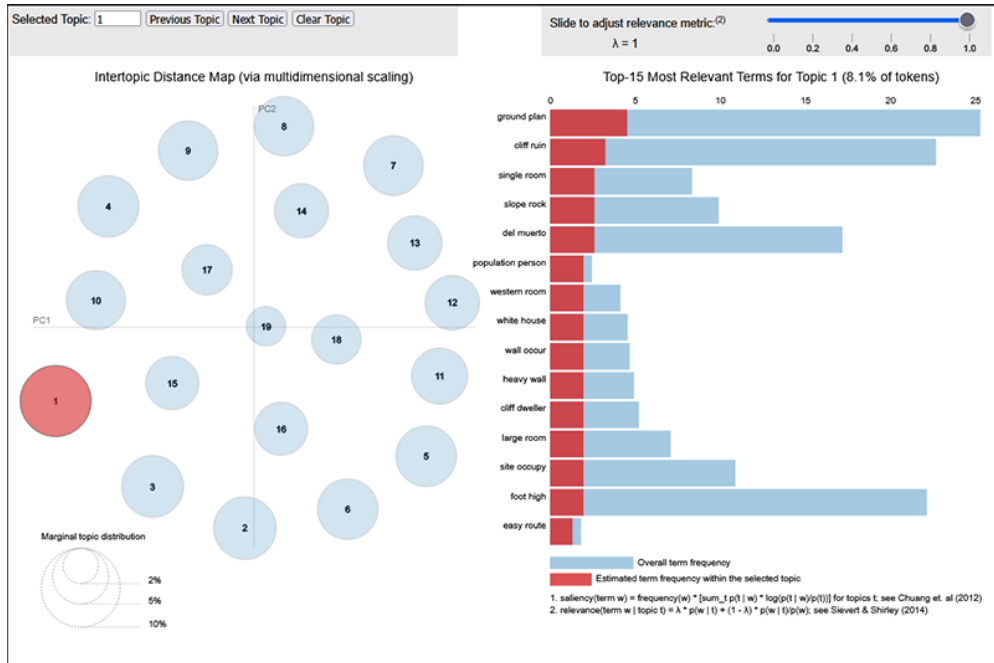


Figure 11. Topic maps generated from LDA (top) and BERTopic (bottom)

Topic maps

Both LDA²² and BERTopic provide a 2D bubble chart showing topics and their relative importance (Figure 11). Each example allows the user to examine the representative phrases returned for each topic. This is the first tool to use for topic discovery. The LDA implementation in scikit-learn makes an effort to create discrete topics, which contrasts strongly with the topic map of BERTopic, which has grouped and overlapping topics. This does not make the LDA topics any less valid, but it does make it more difficult to examine topic groupings. LDA implementations in other packages (such as gensim) produce topic maps that are more like that from BERTopic.

Topic similarity (heat map)

Topic similarity is shown using a heat map, where the higher the similarity between topics the darker the colour of the square in the map (Figure 12). This is a quick way to see higher-level topic groupings.

Topic bar charts

These consist of a set of summary bar charts, one for each topic. Each chart shows the top five topic phrases and the relative importance of each phrase (Figure 13). This visualization is very useful for determining topic generation accuracy.

Topic clouds (word clouds)

This visualization presents a set of word clouds, one for each topic. A word cloud displays important words in a category (in this case, the top ten representative phrases in each topic), where the size of the phrase is displayed based on its relative importance (Figure 14). Used in combination with the topic bar charts, it is possible to achieve an accurate interpretation of each topic.

Topic sunburst

A sunburst chart is a visualization tool that displays hierarchical data with a series of concentric rings, where the innermost ring represents the top level of the hierarchy, and each subsequent outer ring breaks down the data into subcategories. Each segment's size and colour correspond to its proportional value, allowing easy comprehension of the relationships between parent and child categories within a complex structure. This chart is an alternative to word clouds that display not just the relative importance of representative phrases in a topic, but the relative importance of topics. The example chart (Figure 15) shows topics in the inner ring and representative phrases in the outer one. Clicking on a topic opens it, exploding the view to allow closer examination of the representative phrases. This is an excellent visualization method that allows you to quickly assess the relative importance of topics and of representative words to each topic.

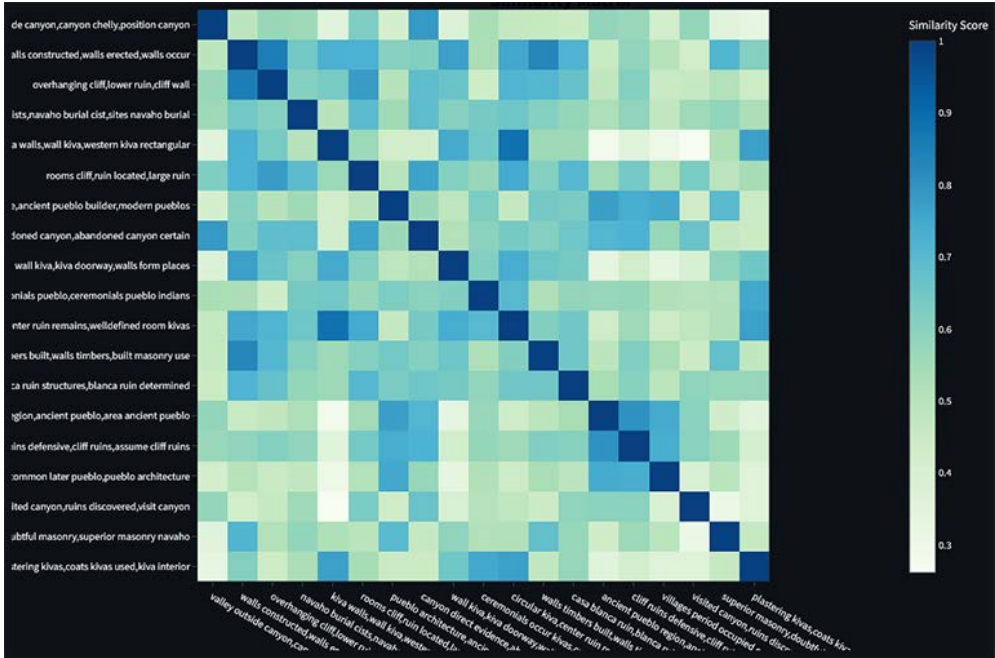


Figure 12. Heat map of topic similarity

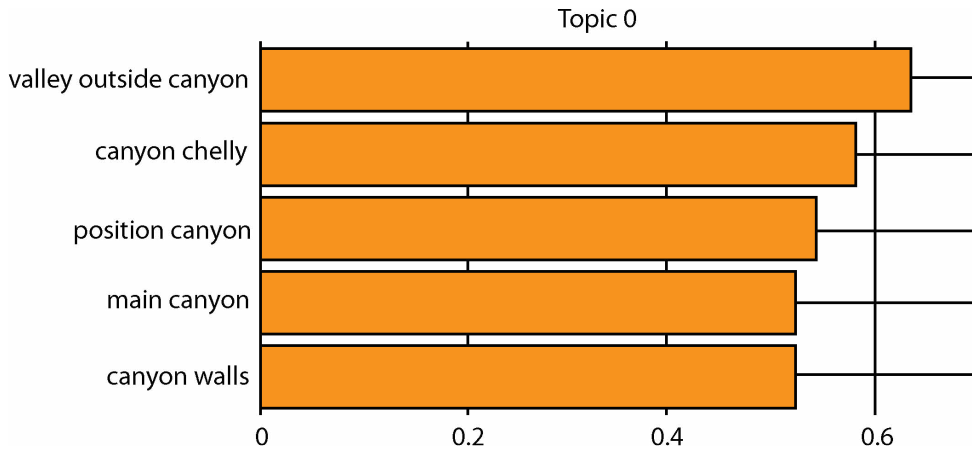


Figure 13. Bar chart with representative phrases for a single topic. There is a chart for each modeled topic



Figure 14. Word cloud for a single topic. There is a word cloud for each modeled topic

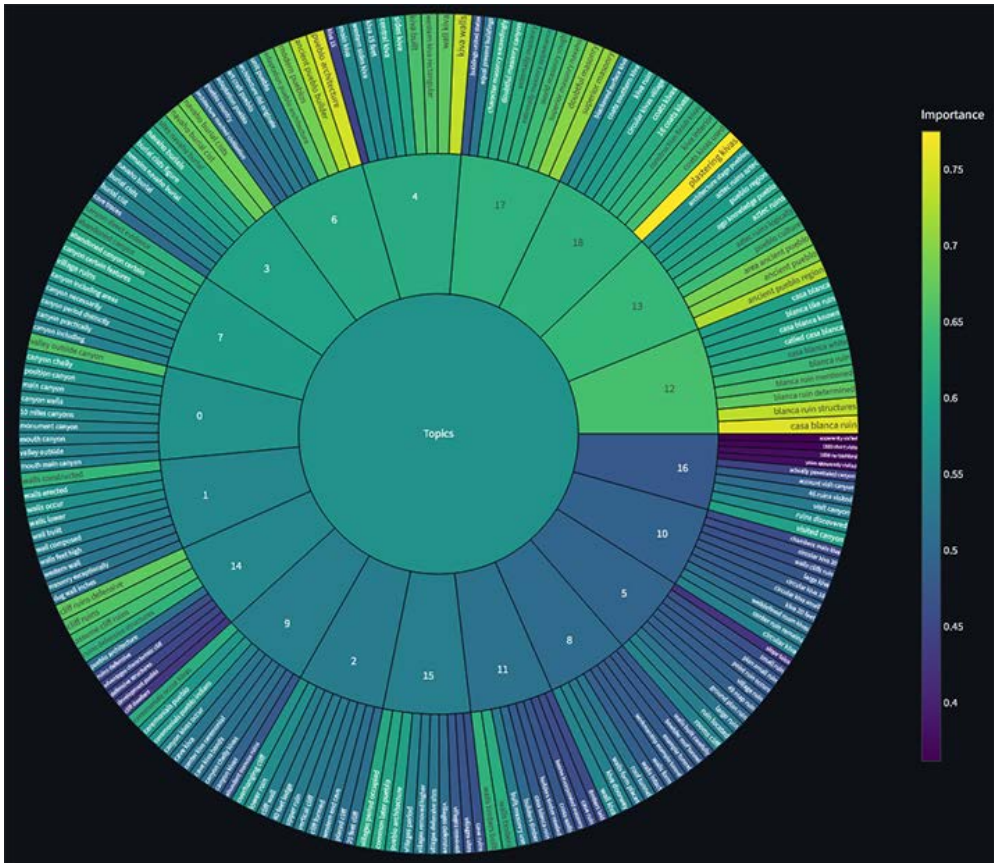


Figure 15. Sunburst chart of topics and representative phrases



Figure 16. Topic treemap of a single topic. There is a treemap for each modeled topic

Topic treemap

A treemap chart (Figure 16) visualizes hierarchical data as a set of nested rectangles, where the size and colour of each rectangle represent the relative importance of each topic and the representative phrases within a topic. This method allows for the comparison of proportions within a corpus of documents and the identification of patterns in a compact format, condensing large amounts of data into a small space. This chart is an alternative to word clouds that displays not just the relative importance of representative phrases in a topic, but the relative importance of topics.

Treemaps are also an alternative to the sunburst chart for those who are more comfortable with a tabular layout and, like the sunburst chart, clicking on a topic opens an exploded view that allows a closer examination of the relative importance of each of the top ten representative phrases for that topic.

Cluster map (dendrogram)

A cluster dendrogram (Figure 17) is a tree-like diagram that visualizes the results of a hierarchical clustering algorithm, showing how topics are progressively merged into higher-level clusters based on their similarity or distance. Each leaf node represents a topic, and the height of the U-shaped links connecting the nodes indicates the distance or dissimilarity between merged topics, with longer lines signifying greater differences.

Document visualizations

These visualizations explore the distribution of topics in the document corpus.

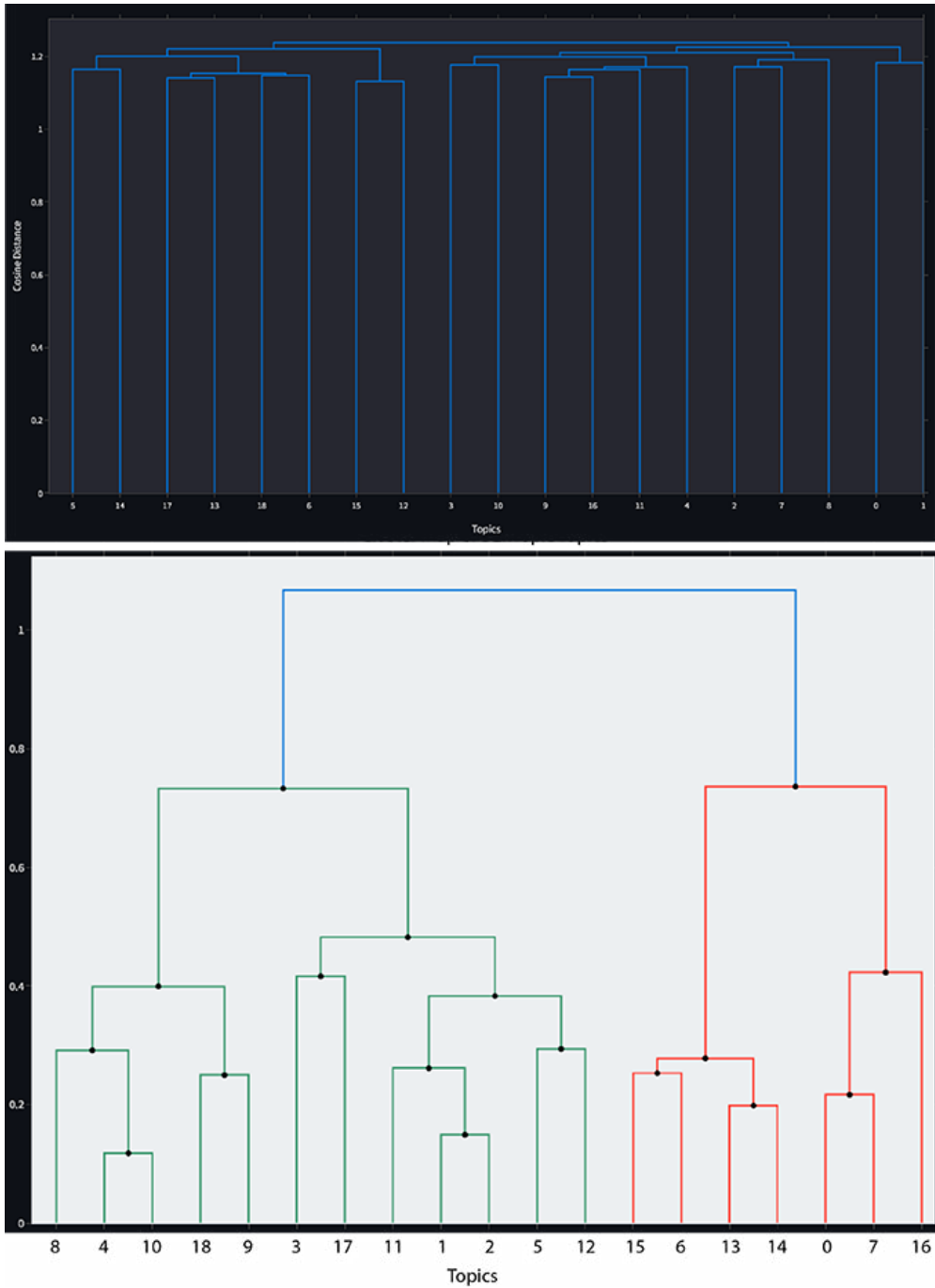


Figure 17. Topic dendrograms for LDA (top) and BERTopic (bottom)

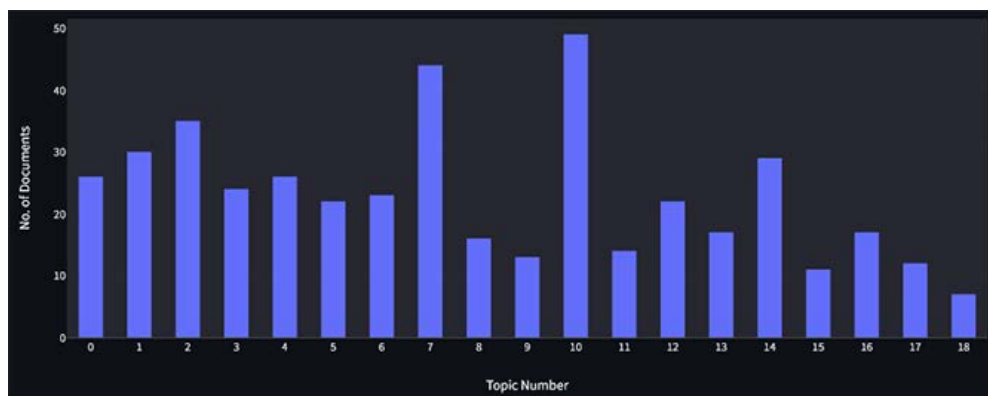


Figure 18. Distribution of documents by their dominant topic

Document topics

A bar chart groups documents by their dominant topic (Figure 18). This shows the relative importance of topics within the corpus.

Document clusters (2D)

A 2D scatter plot of documents is colour-coded with their dominant topic (Figure 19). This shows both the coherence of topic clusters and the relationship between them. Because clusters can overlap in 2D, a 3D visualization is also provided.

Document clusters (3D)

A 3D scatter plot of documents is colour-coded by their dominant topic (Figure 20). The plot can be rotated as needed.

Comparison of LDA and BERTopic topic modeling

After exploring topic generation for both models, the best results were found to come from having LDA generate 19 topics, and a BERTopic setting of HDBSCAN hyperparameters

```
min_cluster_size = 5
```

and

```
min_samples = 4
```

(which also generated 19 clusters). The representative phrases for each topic are coherent and can be easily interpreted.²³

Table 1 shows the topics generated by LDA and Table 2 by BERTopic. Note that the topic numbers have no bearing on the relationship between topics; they are just identifiers. Topic relationships are examined using some of the visualization techniques discussed below.

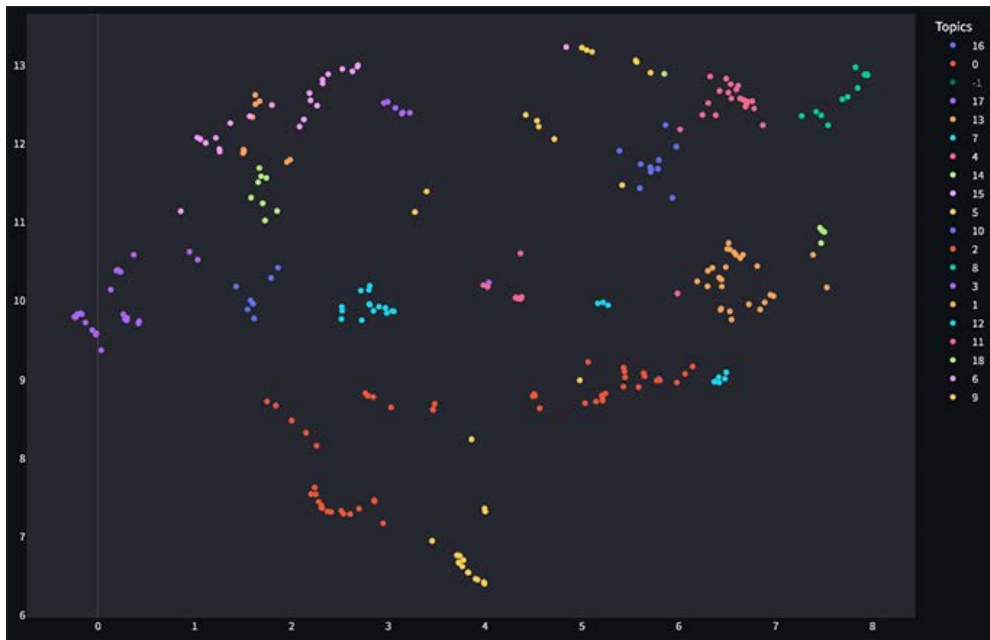


Figure 19. 2D scatter plot of documents grouped by dominant topic



Figure 20. 3D scatter plot of documents grouped by dominant topic

Table 1. Topics generated by LDA with top five representative phrases

| Topic | Representative phrases | Category | Topic | Representative phrases | Category |
|-------|--|--|-------|--|---|
| 0 | village zuni cross wall cliff ruin inclose wall successive coat | Architecture, general | 10 | cliff outlook room east inch diameter site occupy ground plan | Architecture, structure types |
| 1 | plateau country low ruin timber use cross wall tenth meter | construction techniques, general | 11 | like structure room west cliff ruin canyon chelly chimney like | Architecture, chimney-like structures |
| 2 | nearly foot foot high feet high abobe wall small room | Construction techniques, walls | 12 | ground plan inch wide area cultivable area cultivable land cultivable land | Agriculture |
| 3 | supply water compose large canyon chelly large stone stream bed | Geography, ruins and water | 13 | interior bench foot diameter foot high Navaho burial ground plan | Kivas, interior layout |
| 4 | smoke blacken cliff wall wall rest del Muerto foot high | Geography, location of ruins | 14 | defensive motive pueblo architecture wall room wall stand cliff ruin | Architecture, defensive |
| 5 | wall build foot high peach tree cliff ruin pueblo architecture | Navaho, evidence of occupation | 15 | burial cist del muerto character site stream bed ground plan | Agriculture |
| 6 | mark map end wall cliff ruin foot high western end | Architecture, general | 16 | mummy cave west tower room west tower room west rectangular room | Architecture, general |
| 7 | occur point point marked marked map point marked map second story | Architecture, mapping of | 17 | foot high figure ground figure ground plan del muerto ground plan | Canyon del Muerto, ruins |
| 8 | smoothly plaster plaster inside difficult access surface remain wall build | Construction techniques, walls | 18 | single room del muerto slope rock cliff ruin ground plan | Canyon del Muerto, ruins |
| 9 | foot ground room use feet high circular kiva rectangular room | Casa Blanca, architecture | | | |

Note: The presence of 'peach tree' as a representative term for topic 5 indicates that this topic is about the later Navaho occupation of the valley. Peaches were introduced into the US Southwest by the Spanish, long after the abandonment of the valley by Ancestral Puebloan peoples.

Table 2. Topics generated by BERTopic with top five representative phrases

| Topics | Representative phrases | Category | Topics | Representative phrases | Category |
|--------|--|--------------------------------------|--------|---|--|
| 0 | valley outside canyon canyon chelly position canyon main canyon canyon walls | Geography | 10 | circular kiva center ruin remains well defined room kivas kiva 20 feet circular kiva small | Kivas, construction |
| 1 | walls constructed walls erected walls occur walls tower walls built | Construction techniques, walls | 11 | walls timbers built walls timbers built masonry use builders timber casa blanca ruin | Construction techniques, use of timber |
| 2 | overhanging cliff lower ruin cliff wall 40 foot ledge upper ruin | Casa Blanca, layout of ruins | 12 | casa blanca ruin blanca ruin structures blanca ruin determined blanca ruin mentioned blanca ruin | Casa Blanca |
| 3 | navaho burial cists navaho burial cist sites navaho burial navaho burials burial cists figure | Navaho, burials | 13 | ancient pueblo region ancient pueblo area ancient pueblo pueblo culture aztec ruins logically | Ancestral Puebloan culture |
| 4 | kiva walls wall kiva western kiva rectangular kiva built sides kiva | Kivas, construction | 14 | cliff ruins defensive cliff ruins assume cliff ruins ruins defensive structures pueblo architecture | Ruins, defensive locations |
| 5 | rooms cliff ruin located large ruin ground plan ruin 49 map ruin | Ruins, location of | 15 | villages period occupied common later pueblo pueblo architecture villages period villages removed higher | Ruins, occupation phases |
| 6 | pueblo architecture ancient pueblo builder modern pueblo adaptation pueblo architecture unit pueblo | Architecture, modern pueblos | 16 | visited canyon ruins discovered visit canyon 46 ruins visited account visit canyon | Work summary |
| 7 | canyon direct evidence abandoned canyon abandoned canyon certain canyon certain features village ruins | Geography | 17 | superior masonry doubtful masonry superior masonry navaho world masonry rough externally masonry appearance | Construction, masonry |
| 8 | wall kiva kiva doorway walls form places roof tunnel walls intact | Kivas, construction | 18 | plastering kivas coats kivas used kiva interior construction finish kivas 16 coats kivas | Kivas, construction |
| 9 | ceremonials occur kiva ceremonials pueblo ceremonials pueblo Indians canyon kivas occur cave kiva | Kivas, ceremonial aspects | | | |

Analysis of topic generation

Both LDA and BERTopic produce coherent, interpretable topics. However, the scikit-learn implementation of LDA makes it more difficult to see how topics are related.²⁴ While the LDA topic dendrogram (Figure 17) clearly shows the effect of the discrete topics generated by scikit-learn (Figure 11), in Table 3 we are able to see the low-level topic clusters.

Table 3. Low-level LDA topic clusters derived from dendrogram

| Cluster | Topics | Description |
|---------|--------------|------------------------------------|
| 1 | 5, 14 | geographic location of ruins |
| 2 | 13, 17 | Puebloan masonry |
| 3 | 6, 18 | kiva construction and modern kivas |
| 4 | 12, 15 | Casa Blanca occupation phases |
| 5 | 3, 10 | kivas and Navaho burials |
| 6 | 4, 9, 11, 16 | kiva construction techniques |
| 7 | 2, 7, 8 | Casa Blanca |
| 8 | 0, 1 | geography and pueblo construction |

At a higher level, it is possible to see the formation of two large-scale clusters, although interpretation of those clusters is difficult.

Table 4. Low-level clusters derived from BERTopic dendrogram

| Cluster | Topics | Description |
|---------|----------|--|
| 1 | 4, 8, 10 | Kivas, construction |
| 2 | 9, 18 | Kivas, construction and ceremonial aspects |
| 3 | 3, 17 | Navaho burials and masonry construction |
| 4 | 1, 2, 11 | Casa Blanca, construction techniques |
| 5 | 5, 12 | Casa Blanca and location of ruins |
| 6 | 6, 15 | Ruins, occupation phases, and architectural comparisons |
| 7 | 13, 14 | Ruins, defensive locations, and comparisons with wider Ancestral Puebloan area |
| 8 | 0, 7 | Geography of the valley and its surroundings |
| 9 | 16 | Summary of expedition work in the valley |

The BERTopic dendrogram (Figure 17) has internally coherent topic clusters at all levels. There are nine low-level clusters (Table 4) that can then be grouped into four mid-level clusters that mirror the clusters seen on the topic map (Figure 11). Like LDA, there are two high-level clusters.

The mid-level clusters can be interpreted as (left to right on the dendrogram):

- Kiva construction and ceremonial use
- Casa Blanca architecture, construction techniques and later Navaho use
- occupation phases and areal comparisons to both other archaeological regions and modern Puebloan peoples
- geographic setting.

The high-level clusters are:

- architecture and construction techniques
- areal comparisons (both ancient and modern) and geography.

Conclusion

Both LDA and BERTopic produce coherent topics, but BERTopic has the edge. This is likely due to two factors:

- 1 LLMs use larger text spans and preserve the grammatical structure of the document. This allows for more nuanced topic generation than is possible with the BoW model used by LDA.
- 2 The ability of BERTopic to fine-tune its model output means that it creates topics that are easily interpretable.

Because of this, it is suggested that an LLM implementation of topic modeling is chosen rather than a traditional ML one.

Appendix A: Latent Dirichlet Allocation (LDA)

LDA architecture

The LDA pipeline as implemented in the sample code is shown in Figure A1, where

α = document – topics Dirichlet distribution.²⁵ Describes how documents are distributed among topics

β = topics – words Dirichlet distribution. Describes the number of words associated with a particular topic

θ = multinomial distribution for picking topics

ϕ = multinomial distribution for picking words

K = topic distribution

z = list of topics

w = list of words

\mathcal{N} = generated document

D = corpus of generated documents.

α and β are hyperparameters for the Dirichlet distributions. The user adjusts these parameters to fine-tune LDA topic generation. A third hyperparameter (not shown here) is the number of topics to be generated.

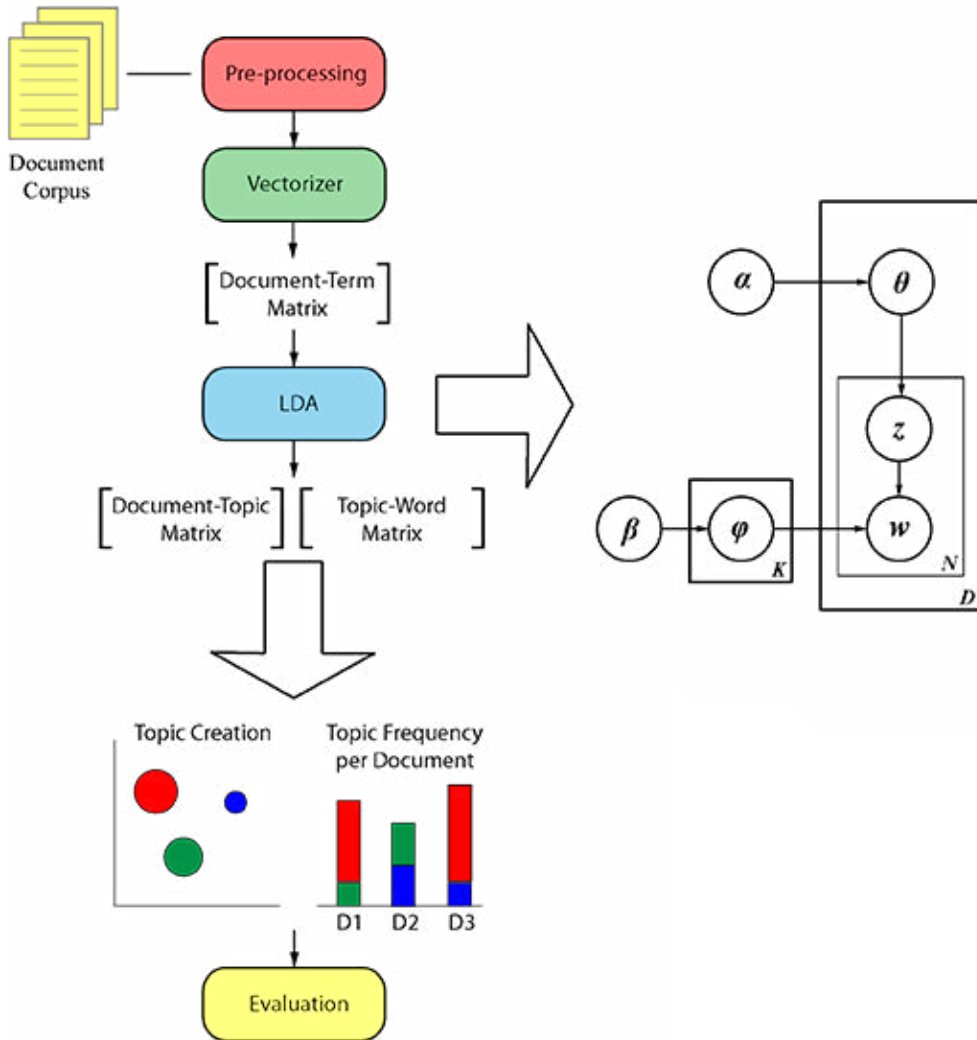


Figure A1. Latent Dirichlet Allocation architecture

Preprocessing

For LDA topic modeling, there are a number of different steps needed to format the data. These are

- 1 removal of specialized characters, URLs, HTML, in-text bibliographic citations,²⁶ and leading or trailing white space
- 2 tokenization
- 3 conversion to lower case and deletion of punctuation
- 4 lemmatization
- 5 deletion of stop words
- 6 deletion of unneeded parts of speech.

See Howes (2025: 251–2) for a detailed example of this process.

Vectorizer

LDA uses as its input a document-term matrix (DTM) where each row of the matrix is a vector of word occurrence in a document, with each dimension (cell) representing the frequency of occurrence. The DTM is used by the LDA model to create topics.

The vectorizer used to create the DTM is the scikit-learn class `CountVectorizer`.²⁷ This class creates a sparse matrix of feature vectors (a BoW), where each document in the corpus is represented by a row and each cell in a row is the number of times a feature occurs in a document. Each column is a unique feature found in the corpus.

For example, with a corpus of the following four sentences,

The plateau country is not a smooth and level region, as its name might imply; it is extremely rugged, and the topographic obstacles to travel are greater than in many wild mountain regions.

It is a country of cliffs and canyons, often of considerable magnitude and forming a bar to extended progress in any direction.

The surface is generally smooth or slightly undulating and apparently level, but it is composed of a series of platforms or mesas, which are seldom of great extent and generally terminate at the brink of a wall, often of huge dimensions.

There are mesas everywhere; it is the mesa country.

After stop word removal, `CountVectorizer` extracts 84 unique features (Table A1) comprising bigrams and trigrams.²⁸

Table A1. BoW feature matrix returned by CountVectorizer

| | Apparently level | Apparently level composed | Bar extended | Bar extended progress | ... | Wall huge | Wall huge dimensions | Wild mountain | Wild mountain regions |
|---|---------------------|------------------------------|--------------|--------------------------|-----|-----------|-------------------------|---------------|--------------------------|
| 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | ... | 0 | 0 | 0 | 0 |
| 2 | 1 | 1 | 0 | 0 | ... | 1 | 1 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 |

Note: 4 rows × 84 columns

LDA

LDA is a generative probabilistic model that represents documents as a mixture of latent topics. It makes the following assumptions about the corpus:

- A corpus is a collection of documents.
- Each document is composed of multiple latent topics.
- There are a fixed number of topics in the corpus (K), and each topic is characterized by a distribution of words.
- The words within a document are generated from a combination of topics.

How LDA Works

LDA works by

- 1 randomly assigning each word in a document to one of the K predefined topics
- 2 reassigning word topic assignment based on the Topic–Word and Document–Topic matrices
- 3 iterating over (2) until the model converges or a stop criterion is reached.

Probability of a document

The probability of a correct document being returned is calculated by

$$P(W, Z, \theta, \varphi; \alpha, \beta) = \prod_{j=1}^M P(\theta_j; \alpha) \prod_{i=1}^K P(\varphi_i; \beta) \prod_{t=1}^N P(Z_{j,t} | \theta_j) P(W_{j,t} | \varphi_{Z_{j,t}})$$

Where

$\prod_{j=1}^M P(\theta_j; \alpha)$ = Dirichlet distribution that associates documents with topics

$\prod_{i=1}^K P(\varphi_i; \beta)$ = Dirichlet distribution that relates topics to words

$P(Z_{j,t} | \theta_j)$ = topic multinomial distribution for the document/topic distribution

$P(W_{j,t} | \varphi_{z_{j,t}})$ = word multinomial distribution for the topic/word distribution.

The two multinomial distributions are used to create a set of random documents the same size as the number of documents in the corpus. The probability P is calculated by comparing the randomly generated documents to the actual documents in the corpus. This is done iteratively until the probability stabilizes.²⁹

Once stabilized, LDA returns the discovered topics and the phrases associated with each topic. What each topic actually represents must be determined by the user studying the representative phrases for each topic.

Posterior distributions

The basic problem for LDA is in determining the posterior distribution of latent variables:

$$P(\theta, z | w, \alpha, \beta) = \frac{P(\theta, z, w | \alpha, \beta)}{P(w | \alpha, \beta)}$$

This value is intractable, as the denominator cannot be precisely calculated. To solve this problem, an inference technique is used, in this case Gibbs sampling.

LDA assigns topics to words using Gibbs sampling, a Markov chain Monte Carlo technique,³⁰ which works on the relationship between objects. Each object is reordered, using the assumption that all other objects in the set are correctly ordered.

Property 1: documents are primarily about a single topic.

Property 2: words are primarily associated with a single topic.

$$P(z_i = t | z^{-i}, w) = \frac{n_{m,t}^{-1} + \alpha}{\sum_{t'=1}^T (n_{m,t'}^{-i} + \alpha)} \times \frac{n_{t,w_i}^{-i} + \beta}{\sum_{v'=1}^V (n_{t,v'}^{-i} + \beta)}$$

The first ratio gives the probability of topic t in document d and the second ratio expresses the probability of a word w belonging to topic t . The hyperparameter values α and β are added to prevent the probabilities collapsing to zero.

The sampling process is repeated multiple times until the samples converge on the true distribution.

Output matrices

The LDA model produces two matrices from the DTM:

- **Document–topic matrix.** Each row contains the proportion of topics present in each document. This matrix is used for such tasks as document clustering by topic (usually by dominant topic) and thematic analysis for topic discovery. For example, using the DTM previously created (see Vectorizer, above) and `n_topics = 3`, LDA creates the document–topic matrix in Table A2.³¹

Table A2. Document–topic matrix

| | Topic 0 | Topic 1 | Topic 2 |
|---|----------|----------|----------|
| 0 | 0.012121 | 0.011990 | 0.975890 |
| 1 | 0.018861 | 0.018651 | 0.962488 |
| 2 | 0.008921 | 0.982251 | 0.008827 |
| 3 | 0.084861 | 0.831247 | 0.083892 |

- **Topic–word matrix.** This shows the likelihood of phrases being associated with a specific topic. Each row in the matrix represents a topic and each column a phrase. The matrix assists in such tasks as thematic analysis, dimensionality reduction, and the selection of relevant key phrases for a topic. Using the same example corpus, LDA produces the topic–word matrix in Table A3.

Table A3. Topic–word matrix

| | Apparently level | Apparently level composed | Bar extended | ... | Wall huge dimensions | Wild mountain | Wild mountain regions |
|-------|------------------|---------------------------|--------------|-----|----------------------|---------------|-----------------------|
| Top 1 | 0.333487 | 0.333487 | 0.333691 | ... | 0.333487 | 0.333557 | 0.333557 |
| Top 2 | 1.333123 | 1.333123 | 0.333474 | ... | 1.333123 | 0.333421 | 0.333421 |
| Top 3 | 0.333390 | 0.333390 | 1.332835 | ... | 0.333390 | 1.333021 | 1.333021 |

Note: 3 rows × 84 columns

Evaluation

There are several ways to evaluate the output of an LDA model, both qualitatively and quantitatively.

Qualitative evaluation

- **Topic interpretability.** This examines the key phrases associated with each topic. Do the phrases form a semantically coherent set? Check whether the discovered topics align with expected themes, given the type of manuscript under investigation.
- **Visualization.** Visualizations – such as the bubble chart created by pyLDAvis, word clouds, and sunburst charts – allow for the examination of the relative importance of latent topics and the key phrases associated with those topics.

Quantitative evaluation

- **Perplexity.** This measures how well the model describes a given set of documents, with a lower perplexity score indicating a better fit to the data.
- **Topic coherence.** This measures the semantic similarity of words within a topic, indicating the interpretability and meaning of the topic. A high coherence score means that the words in a topic frequently appear together or in similar contexts across the documents in the corpus, suggesting a well-formed, meaningful concept.

Appendix B: BERTopic

The BERTopic pipeline as implemented in the example software is shown in Figure B1.

Preprocessing

In LLM topic modeling algorithms like BERTopic, the model is designed to work with text spans (typically 256 words) rather than words or short phrases, and has been trained to understand natural language. Cleaning the input data degrades the performance of the model, since the presence of all parts of speech in grammatical order aids in the syntactic and semantic understanding of the document. In the BERTopic example, I remove URLs, HTML, and citations only, text that can be considered ‘content-free’ when it comes to topic discovery.

A preprocessing step specific to LLM implementations is text chunking. In a given text span longer than 256 words, both SBERT and BERTopic discard from analysis any text that is over the 256-word limit. In order to retain all the information in the span, it is necessary to chunk a document that is longer than 256 words into two or more documents that do not exceed that limit.

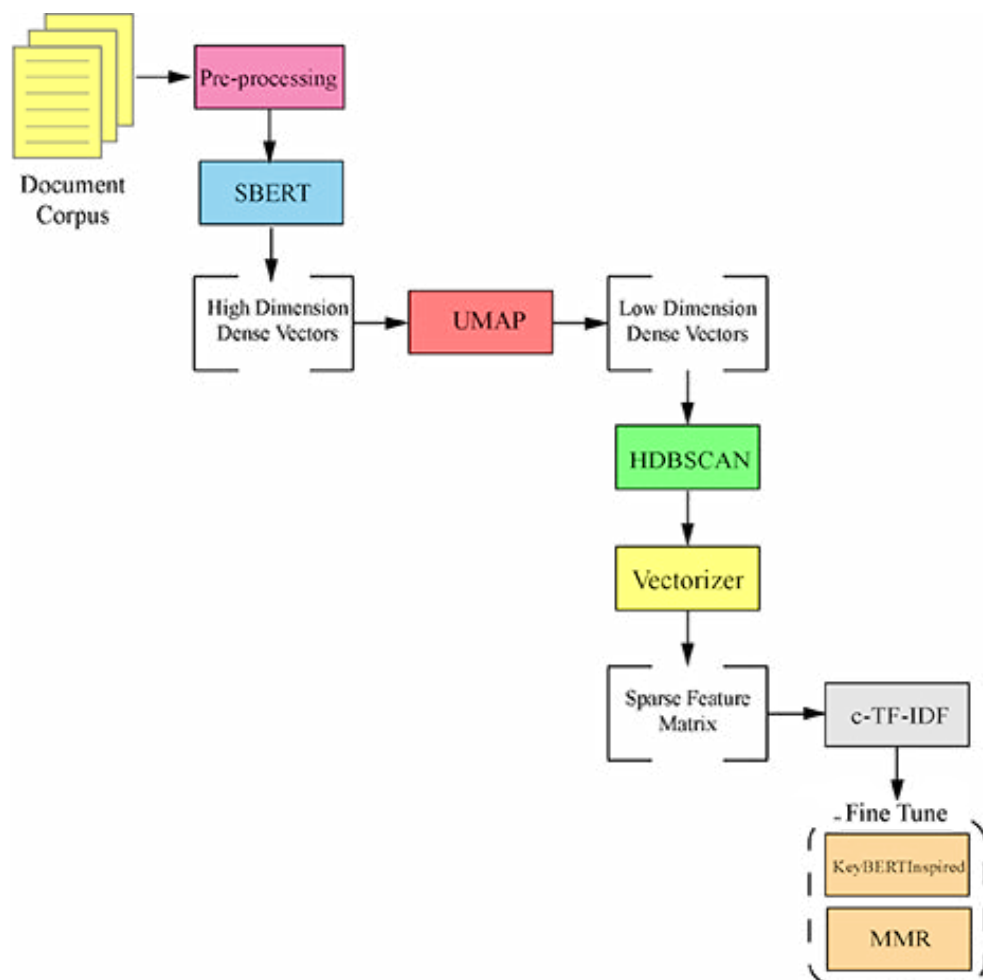


Figure B1. BERTopic pipeline

Sentence-BERT (SBERT)

The default embedding layer in BERTopic is SBERT (Reimers and Gurevych, 2019), imported through the `sentence_transformers` Python package. SBERT (Figure B2) uses Siamese³² and triplet networks to dramatically increase speed in the creation of sentence embedding.³³

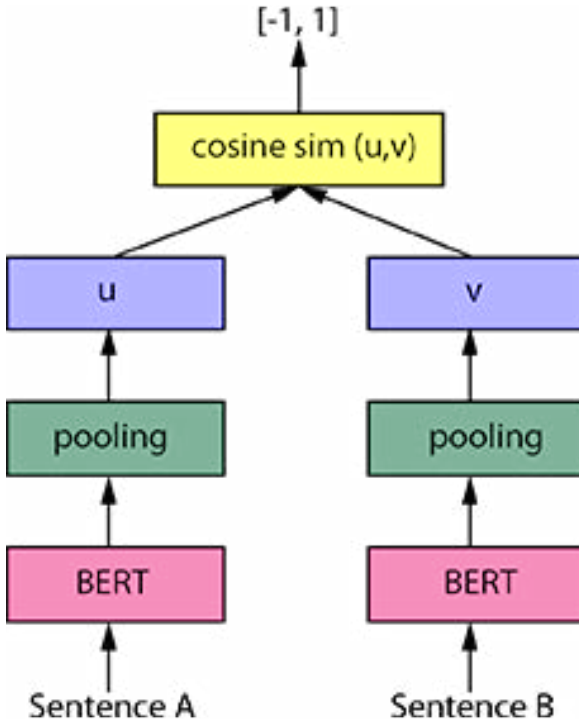


Figure B2. SBERT
Siamese architecture

SBERT compares two input sentences by first passing them to a standard BERT model, which produces a high-dimension dense feature matrix. The output of the BERT model is passed to a pooling layer, which creates a fixed-size, sentence-embedding vector. The semantic similarity of the two vectors is determined using cosine similarity. Cosine similarity is calculated by

$$\text{similarity}(u, v) = \frac{u \cdot v}{\|u\| \times \|v\|} = \frac{\sum_{i=1}^n u_i \times v_i}{\sqrt{\sum_{i=1}^n u_i^2} \times \sqrt{\sum_{i=1}^n v_i^2}}$$

In the current implementation, SBERT is loaded with the *all-MiniLM-L6-v2* model,³⁴ which maps sentences and paragraphs to a 384-dimension dense vector space. This model is suitable for the creation of topic clusters.

By default, a text span longer than 256 words is truncated. This means that text chunking is required to preserve all information in the span.

Sentence transformer models have the following characteristics.

- They calculate a fixed-size vector representation.
- The embedding calculation is efficient and embedding similarity calculation is very fast.

- They are applicable to a wide range of tasks (semantic search, clustering, classification, etc.).

Uniform manifold approximation and projection (UMAP)

In order to promote efficient clustering, BERTopic uses UMAP³⁵ (McInnes et al., 2018) as its default algorithm to accomplish dimension reduction from the 384 dimensions of the SBERT vector matrix. UMAP can capture both local and global high-dimension space in its reduction to lower dimensions. In BERTopic, the balance between local and global structure is managed by the `n_neighbors` parameter of the UMAP class. By default, BERTopic sets the number of dimensions returned by UMAP to five.

Note that UMAP is stochastic in nature and, when running it multiple times, each iteration has different weights for dimension values. This means that HDBSCAN creates different clusters on each iteration. In order to have reproducible results, a random number seed is passed to UMAP.

Hierarchical density-based spatial clustering of applications with noise (HDBSCAN)

HDBSCAN (Campello et al., 2013) is a hierarchical density-based clustering algorithm that extracts flattened clusters based on cluster stability. Unlike other clustering methods such as K-Means, HDBSCAN makes no assumptions about cluster shape and returns clusters of arbitrary shape and size. This method can find clusters of varying density.

Varying the hyperparameters for HDBSCAN has by far the greatest effect on the output of BERTopic. Note that outliers (noise) are expected in the data returned by HDBSCAN. Forcing the assignment of all points to a cluster is not recommended, as this degrades cluster coherence.

HDBSCAN works by performing the following tasks:

- 1 transforming the space according to density/sparsity;
 - 2 building the minimum spanning tree of the distance-weighted graph;
 - 3 building a cluster hierarchy of connected components;
 - 4 condensing the cluster hierarchy based on minimum cluster size;
 - 5 extracting stable clusters from the condensed tree.
- **Transforming the space.** To transform the space, the distance to the k th nearest neighbor (the core distance) is defined. The core distance for point x is denoted as $core_k(x)$. The value of k is set by the `min_samples` parameter of HDBSCAN. Clusters are found by calculating the mutual reachability distance

$$d_{reach-k}(a, b) = \max\{core_k(a), core_k(b), d(a, b)\}$$

Using this distance metric, dense points remain the same distance from each other (usually the original distance metric $d(a,b)$), but sparser points are pushed away from other points by at least their core distance. This pushing away of sparse points isolates them as noise.

- **Minimum spanning tree.** Once the mutual reachability distance has been calculated between all points, a minimum spanning tree (MST) is constructed. An MST connects all points in such a way that the sum of the mutual reachability distances is minimized.
- **Building cluster hierarchy.** The edges of the MST are sorted in ascending order based on their weight (their mutual reachability distance), constructing a hierarchical tree. Clusters are merged based on their mutual reachability distance. This hierarchy can be displayed as a dendrogram.
- **Condensing the cluster tree.** Condensing the tree relies on the notion of the stability of clusters over varying density thresholds. Clusters that are less stable over these thresholds are pruned. The result is a condensed tree that is a simplified version of the initial hierarchical tree. It retains only those clusters that have shown enough stability over the various density levels. How the algorithm prunes the tree is governed by the `min_cluster_size` parameter of HDBSCAN, which defines the minimum number of points required to form a cluster.
- **Extracting clusters.** Finally, HDBSCAN extracts stable clusters³⁶ from the condensed tree. For each cluster in the condensed tree, HDBSCAN calculates a stability score, which is the sum of the lifetimes of each point in a cluster.³⁷
- To calculate the score, the lambda value is first calculated, $\lambda = 1/\text{distance}$. Now, for each point in a cluster, its contribution to the stability of the cluster is calculated by

$$\text{Stability}(p) = \lambda_{\text{death}}(p) - \lambda_{\text{birth}}(C)$$

where

$\lambda_{\text{death}}(p)$ = the lambda value at which point p falls out of the cluster

$\lambda_{\text{birth}}(C)$ = the lambda value at which the cluster C is formed. All points in the cluster share this value.

The stability score of a cluster C is the sum of the values for all its points,

$$\text{Stability}(C) = \sum_{i=1}^n \text{Stability}(p_i) = \sum_{i=1}^n [\lambda_{\text{death}}(p_i) - \lambda_{\text{birth}}(C)]$$

Vectorizer

As in LDA, the vectorizer used is the scikit-learn class `CountVectorizer`. See the section ‘Vectorizer’ in Appendix A for a discussion of this. Unlike LDA, which uses the output of the vectorizer as the DTM input into the LDA model for the discovery of topics, `BERTopic` uses the vectorizer as part of its process to recover representative phrases for already discovered topics.

Class-based term frequency-inverse document frequency (c-TF-IDF)

c-TF-IDF³⁸ is an extension of traditional TF-IDF, which focuses on clusters of documents (topics) rather than on individual documents (Grootendorst, 2022: 3). The goal is to find representative terms for specific topics. c-TF-IDF is calculated by

$$c\text{-TF-IDF}_{x,c} = \|tf_{x,c}\| \times \log\left(1 + \frac{A}{f_x}\right)$$

Where

$tf_{x,c}$ = frequency of term x in class c

f_x = frequency of term x across all classes

A = average number of words per class.

The c-TF-IDF algorithm aggregates terms from all documents within a topic cluster and computes the importance of terms for that cluster. This allows for the identification of key terms that best represent a topic.

Fine-tuning

Fine-tuning is accomplished using a combination of two representation models: `KeyBERTInspired` and `Maximal Marginal Relevance`. Using these two models together creates highly coherent topics, each with a set of representative and non-redundant key phrases.

KeyBERTInspired

The `KeyBERTInspired` model performs fine-tuning based on the semantic relationship between key phrases and the set of documents in each topic. It leverages c-TF-IDF to create a set of representative documents per topic and uses those as updated topic embeddings. It then calculates the similarity between candidate keywords and topic embedding using the same embedding model used to embed the documents (Figure B3).

Maximal marginal relevance (MMR)

When key phrase weights are generated for topics, `BERTopic` does not, by default, take into account phrase redundancy. That is, words like ‘ruin’ and ‘ruins’ could both end up as heavily weighted key phrases for a topic, even though there is a high degree of redundancy between them.

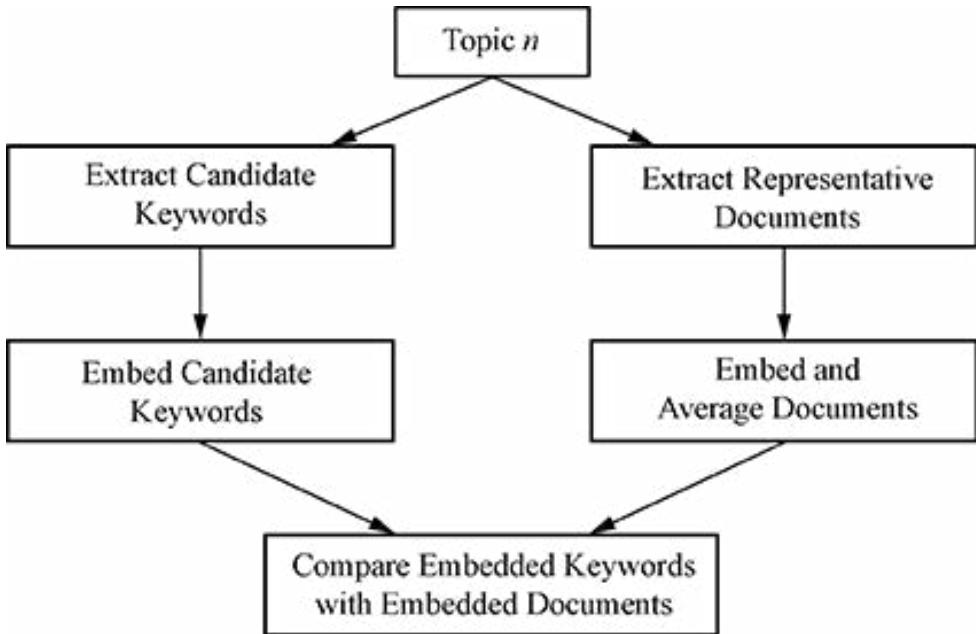


Figure B3. KeyBERTInspired workflow

To counteract this, the MMR algorithm is used (Carbonell and Goldstein, 1998). MMR is calculated by

$$R = (C, Q, \theta)$$

where

C = collection of terms

Q = query or profile

θ = relevance threshold below which a term is not returned

$$MMR = Arg \max_{D_i \in R \setminus S} [\lambda (Sim_1(D_i, Q)) - (1 - \lambda) \max_{D_j \in S} Sim_2(D_i, D_j)]$$

where

R = the ranked list of terms

S = the subset of terms in R already selected

$R \setminus S$ = the subset of terms in R not yet selected

Sim_1 = the similarity matrix used in retrieval and relevance ranking

Sim_2 = the same as Sim_1 or a different metric.

Given the above definition, MMR computes incrementally the standard relevance-ranked list when the parameter $\lambda = 1$, and computes a maximal diversity ranking among the documents in R when $\lambda = 0$. For intermediate values of λ in the interval $[0, 1]$, a linear combination of both criteria is optimized. (Carbonell and Goldstein, 1998: 335)

λ is a user-configurable hyperparameter, which varies in range $[0, 1]$. A value of 0 allows all redundant words, and a value of 1 removes all redundancy.

Acknowledgments

I would like to thank Alison Craig, who read and commented on earlier versions of this article.

Notes

- 1 The source code for the example applications is available on Github. The LDA example can be found at <https://github.com/DWHHowes/LDA-Topic-Modeling>, and the BERTopic example at <https://github.com/DWHHowes/BERTopic-Topic-Modeling>.
- 2 See <https://www.gutenberg.org/ebooks/19723>.
- 3 The ISO standard for topic maps (ISO/IEC 13250) is a multipart standard. These standards define topic maps as a technology for representing and interchanging knowledge that focuses on information findability. The original motivation was to merge book index structures, but its generalized nature led to applications in representing ontologies, integrating information sources, and improving semantic web technologies.
- 4 The model revolves around three core constructs: (1) topics, (2) occurrences (pointers to where a topic is discussed), and (3) associations (explicit relationships between topics). These correspond to index terms, locators, and cross-references, but offer a richer, context-free relationship model than that available with traditional hierarchical headings.
- 5 Unsupervised learning uses unlabeled text to identify patterns, structure, and meaning, without human-provided examples. Unlike supervised methods that require labeled data sets, unsupervised models autonomously discover underlying structures in the text.
- 6 In NLP, the meaning of the word ‘document’ differs from its common usage. For NLP purposes, a document is a discrete span of text that represents a single input to the model. A document can vary in size from a single sentence to a paragraph, a chapter, or the complete text of a book, depending on the analytical task being undertaken.
- 7 See <https://pyldavis.readthedocs.io/en/latest/readme.html>.
- 8 See <https://plotly.com/>.
- 9 See <https://matplotlib.org/>.
- 10 See https://maartengr.github.io/BERTopic/getting_started/visualization/visualization.html.
- 11 TF-IDF stands for term frequency-inverse document frequency. See Howes (2025: 253–4) for a discussion of this technique.
- 12 For a concise overview of NLP pipelines, see Ali (2023).
- 13 See <https://spacy.io/usage/spacy-101> for an overview of SpaCy.
- 14 This is done by creating a blank language object, which includes only the tokenizer for that language. To create a blank object for English, the code is `nlp = spacy.blank('en')`.

- 15 See <https://spacy.io/usage/linguistic-features#tokenization>.
- 16 See <https://spacy.io/api/dependencyparser>.
- 17 See <https://spacy.io/api/entityrecognizer>.
- 18 Both example applications use the same code and Python packages (scikit-learn and SpaCy) to generate the EDA page.
- 19 Note there is a hard 100-character floor. Documents that are less than 100 characters were excluded from analysis.
- 20 See Howes (2025: 255–6, Table 4) for a discussion of default SpaCy Named Entity categories.
- 21 For help in how to use the example applications, see the *readme.md* file present in each repository.
- 22 The Python package pyLDAvis is used to generate this chart for LDA.
- 23 Interpretation of topics is not done using just the representative phrases shown in the tables. The visualization methods available in the example applications provide a number of different views into the topic data, emphasizing different aspects, and all are used in the categorization of topics.
- 24 For those interested in pursuing LDA topic modeling, I would recommend using the gensim Python package, since it produces overlapping topics. This provides more information about the relationship between topics.
- 25 A Dirichlet distribution is a multivariate continuous probability distribution that models a vector of probabilities. It is the multivariate generalization of the Beta distribution (which models the probability of two outcomes) and is frequently used in Bayesian statistics as a conjugate prior for the multinomial distribution. See the Caltech Distribution Explorer page on this (https://distribution-explorer.github.io/multivariate_continuous/dirichlet.html) for a short mathematical description.
- 26 The removal of URLs, HTML, and citations is accomplished using regular expressions. While this works well for URLs and HTML, citations have such a wide potential variability of form that they can exceed the ability of a regular expression to recognize them. Probably the best way to handle citation removal is to use supervised learning to train an ML model so it can recognize citations and use that model in a removal tool. This, however, is something well beyond the scope of this article.
- 27 For the API, see https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html. For an explanation of feature extraction using CountVectorizer, see https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction.
- 28 Because a phrase is generally more interpretable than a single word when evaluating a cluster as a potential topic, both the LDA and BERTopic examples use bigrams and trigrams as representative key phrases.
- 29 As the probability of randomly creating a sample document that matches anything in the corpus is extremely low, the value of P is very small. It is, however, non-zero and multiple iterations increase its value.
- 30 See Speagle (2019) for a discussion of Markov chain Monte Carlo techniques.
- 31 A cursory examination of the document–topic matrix shows that there are only two topics present in this toy corpus. In an actual use case, you would rerun LDA specifying that it generate two topics.
- 32 A Siamese network is a class of neural network that contains two identical networks with the same configuration, parameters, and weights. The updating of parameters is mirrored across both networks. Siamese networks find the similarity between two inputs by comparing feature vectors.

- 33 In a standard BERT network, the analysis of 10,000 sentences to find the sentence pair with the highest similarity takes approximately 65 hours. In SBERT, this analysis is reduced to approximately five seconds (Reimers and Gurevych, 2019: 1–2).
- 34 Pre-trained LLM models provide enormous value to a user. According to the Hugging Face documentation for this model (<https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2>), the developers used the existing *MiniLM-L6-H384-uncased* model and then fine-tuned it using a data set of one billion sentence pairs.
- 35 See <https://umap-learn.readthedocs.io/en/latest/index.html>.
- 36 A stable cluster is one that persists through variation in density level.
- 37 A points lifetime is the range of density levels for which the point remains part of the cluster.
- 38 See <https://maartengr.github.io/BERTopic/api/ctfidf.html>.

References

- Ali, A. (2023) ‘Understanding the NLP pipeline: a comprehensive guide’, *Medium*. Available at https://medium.com/@asjad_ali/understanding-the-nlp-pipeline-a-comprehensive-guide-828b2b3cd4e2.
- Biezunski, M. (2018) ‘Topic maps and the essence of indexing’, *The Indexer*, **36**(4), 157–61. Available at <https://doi.org/10.3828/indexer.2018.60>.
- Blei, D. M., Ng, A. Y. and Jordan, M. I. (2003) ‘Latent Dirichlet allocation’, *Journal of Machine Learning Research*, **3**, 993–1022.
- Campello, R. J. G. B., Moulavi, D. and Sander, J. (2013) ‘Density-based clustering based on hierarchical density estimates’, in J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Zu (eds), *Advances in knowledge discovery and data mining*. Berlin and Heidelberg: Springer, pp. 160–72.
- Carbonell, J. and Goldstein, J. (1998) ‘The use of MMR, diversity-based reranking for reordering documents and producing summaries’, in *Proceedings of the 21st annual international ACM SIGIR conference on research and development in information retrieval. SIGIR98: 21st Annual ACM/SIGIR International Conference on Research and Development in Information Retrieval*. Melbourne: ACM, pp. 335–6. Available at <https://doi.org/10.1145/290941.291025>.
- Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2018) ‘BERT: pre-training of deep bidirectional transformers for language understanding’, *arXiv*. Available at <https://doi.org/10.48550/ARXIV.1810.04805>.
- Grootendorst, M. (2022) ‘BERTopic: neural topic modeling with a class-based TF-IDF procedure’, *arXiv*. Available at <https://doi.org/10.48550/ARXIV.2203.05794>.
- Howes, D. (2025) ‘Natural-language processing (NLP) and indexing, Part 1. Overview’, *The Indexer* **43**(3), 249–86.
- McInnes, L., Healy, J. and Melville, J. (2018) ‘UMAP: uniform manifold approximation and projection for dimension reduction’, *arXiv*. Available at <https://doi.org/10.48550/ARXIV.1802.03426>.
- Mindeleff, C. (1897) ‘The cliff ruins of Canyon de Chelly, Arizona’, in *Sixteenth Annual Report of the Bureau of Ethnology*. Washington, DC: Smithsonian Institution, pp. 73–198. Available at <https://www.gutenberg.org/ebooks/19723>.
- Murel, J. and Kavlakoglu, E. (2024) ‘What is Latent Dirichlet Allocation?’, *Think*. Available at <https://www.ibm.com/think/topics/latent-dirichlet-allocation>.
- Northedge, R. (2008) ‘The medium is not the message: topic maps and the separation of presentation and content in indexes’, *The Indexer* **26**(2), 60–4. Available at <https://doi.org/10.3828/indexer.2008.16>.

- Provo, A. (2019) 'From index to network: topic maps in the Enhanced Networked Monographs project', *The Indexer* **37**(1), 13–35. Available at <https://doi.org/10.3828/indexer.2019.3>.
- Pykes, K. (2023) 'What is topic modeling? An introduction with examples', *datacamp*. Available at <https://www.datacamp.com/tutorial/what-is-topic-modeling>.
- Reimers, N. and Gurevych, I. (2019) 'Sentence-BERT: sentence embeddings using Siamese BERT-networks', *arXiv*. Available at <https://doi.org/10.48550/ARXIV.1908.10084>.
- Speagle, J. S. (2019) 'A conceptual introduction to Markov Chain Monte Carlo Methods', *arXiv*. Available at <https://doi.org/10.48550/ARXIV.1909.12313>.
- Wikipedia (2025) 'Bayesian inference', *Wikipedia*. Available at https://en.wikipedia.org/wiki/Bayesian_inference.