```
import pandas as pd
from bertopic import BERTopic
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import CountVectorizer
```

```
c:\Users\ramirolobo\AppData\Local\Programs\Python\Python312\Lib\site-packages\tqdm\auto.py:21: TqdmWarning: IProgress not found. Ple
  from .autonotebook import tqdm as notebook_tqdm
```

```
nltk.download('stopwords', download_dir='C:/Users/ramirolobo/nltk_data')
nltk.download('punkt', download_dir='C:/Users/ramirolobo/nltk_data')
nltk.data.path.append('C:/Users/ramirolobo/nltk_data')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:/Users/ramirolobo/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:/Users/ramirolobo/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
#load texts
text_chunks = pd.read_csv("data/text_chunks.csv")
texts = text_chunks["text"].tolist()
```

```
#define stopwords
stop_words = set(stopwords.words('english'))
custom_stopwords = {
    "chapter", "book", "section", "article",
    "i", "ii", "iii", "iv", "v", "vi", "vii", "viii",
    "ix", "x", "xi", "xii", "xiii", "xiv", "xv", "xvi", "xvii",
    "xviii", "xix", "xx", "xxi", "xxii", "xxiii", "xxiv", "xxv",
    "xxvi", "xxvii", "xxviii", "xxix", "xxx", "xxxii", "xxxiii",
    "xxxiv", "xxxv", "xxxvi", "xxxvii", "xxxviii", "xxxix", "xl",
}
all_stopwords = stop_words.union(custom_stopwords)

#function to preprocess text
def preprocess_text(text):
    text = text.lower()
    text = re.sub(r"[^a-z\s]", "", text)  # remove punctuation and numbers
    words = text.split()  # split by whitespace instead of word_tokenize
    filtered_words = [w for w in words if w not in all_stopwords]
    return " ".join(filtered_words)
```

```
#apply preprocessing
text_chunks["cleaned_text"] = text_chunks["text"].apply(preprocess_text)
```

```
text_chunks.head()
```

	Unnamed: 0	author	title	chunk	text	year	century	cleaned_text
0	1	Arrian	The Anabasis of Alexander\r\nor, The History o...	0	the anabasis of alexander	130	2nd c. CE	anabasis alexander
1	2	Arrian	The Anabasis of Alexander\r\nor, The History o...	1	the anabasis of alexander	130	2nd c. CE	anabasis alexander
2	3	Arrian	The Anabasis of Alexander\r\nor, The History o...	2	or the history of the wars and conquests of ...	130	2nd c. CE	history wars conquests alexander great
3	4	Arrian	The Anabasis of Alexander\r\nor, The History o...	3	literally translated with a commentary from th...	130	2nd c. CE	literally translated commentary greek arrian n...
4	5	Arrian	The Anabasis of Alexander\r\nor,	4	e j chinnock ma llb london rector	130	2nd c.	e j chinnock llb london rector

```
cleaned_texts = text_chunks["cleaned_text"].dropna().tolist()
```

```
#get vectorizer model to remove any residual stopwords
vectorizer_model = CountVectorizer(stop_words="english")

#fit BERTopic model
topic_model = BERTopic(
    #vectorizer_model=vectorizer_model,
```

```
    #calculate_probabilities=True,
    #verbose=True,
    #nr_topics="auto",  # Automatically determine the number of topics
    #language="english"  # Specify the language for stopwords
)

topics, probs = topic_model.fit_transform(cleaned_texts)


#topic_model.save("bertopic_model") #save model
topic_model = BERTopic.load("bertopic_model")  # load model


topics = topic_model.get_topic_info()
#topics.to_csv("data/topics.csv", index=False)


#avoid training model each time
topics = pd.read_csv("data/topics.csv")


# top 15 topics
topics.loc[1:20, :]
```

	Topic	Count	Name	Representation	Representative_Docs
1	0	4446	0_french_english_england_france	[french, english, england, france, ships, sea,...	[advantage wind english day fortyfour ships ac...
2	1	2679	1_athenians_lacedaemonians_athens_athenian	[athenians, lacedaemonians, athens, athenian, ...	[terms armistice ships delivered number sixty ...
3	2	928	2_operations_lines_strategic_line	[operations, lines, strategic, line, point, en...	[front operations space separates two armies u...
4	3	921	3_theory_mind_must_us	[theory, mind, must, us, may, things, therefor...	[influence theoretical principles upon real li...
5	4	554	4_sun_wu_tz_chi	[sun, wu, tz, chi, yu, tsao, chinese, chang, c...	[return elder sun tz mentioned two passages sh...
6	5	353	5_object_war_destruction_therefore	[object, war, destruction, therefore, politica...	[destruction enemys military force leading pri...
7	6	311	6_river_indus_called_rivers	[river, indus, called, rivers, name, hebrew, e...	[unquestionable therefore let record indus lar...
8	7	307	7_eordaea_rha_ulm_crateas	[eordaea, rha, ulm, crateas, ariaspians, mount...	[ulm magnesia victory malplaquet, peithon son ...
9	8	177	8_military_generals_general_spirit	[military, generals, general, spirit, virtue, ...	[assistance military virtue parts genius comma...
10	9	177	9_darius_persians_alexander_persian	[darius, persians, alexander, persian, king, g...	[reflections fate darius alexander sent body d...
11	10	174	10_rhine_archduke_austrians_austrian	[rhine, archduke, austrians, austrian, danube,...	[leaving austria rear russia front austria lau...
12	11	171	11_divisions_battalions_columns_deployed	[divisions, battalions, columns, deployed, col...	[whole armies deployed skirmishers still neces...
13	12	149	12_wall_mound_engines_towers	[wall, mound, engines, towers, walls, stones, ...	[round city able bring military engines walls ...

Topics seems to primarily identify authors rather than common themes between them.

```
topic_model.get_representative_docs(topics)
```

```
{-1: ['mahan december',
  'dumfries december',
  'january thirty twentytwo april thirty twenty difference time trincomalee saints nine hours half'],
 0: ['varus general horse man uncommon spirit skill encouraged men pursuing enemy disposed troops convenient places rest gave
battle enemy enemys cavalry made bold stand foot relieving making general halt assist horse battle',
  'austrians directed operations upon meuse sarre moselle concert prussians part useless army upper rhine force one hundred twenty
thousand men flanks protected troops could pushed forward even probable without changing direction war',
  'result first combat wiped account possible stop first success put second part battle coming day even king lost advantages first
would always set second'],
 1: ['advantage wind english day fortyfour ships action dutch eighty many english said larger two fleets passed opposite tacks
english windward tromp rear seeing dutch order battle badly formed ships two three lines overlapping',
  'time arrived strong english squadron six shipsoftheline admiral hughes absence similar french force gave entire control sea
english arrival suffren nearly three years later mean holland drawn war stations negapatam',
  'england sea hour fast approaching great colonial expeditions made last year war illustrious triumph sea power england france
spain united first'],
 2: ['notice', 'disastrous', 'index']}
```

```
#topics over time
timestamps = text_chunks['year'].tolist()
```

```
topics_over_time = topic_model.topics_over_time(cleaned_texts, timestamps=timestamps, nr_bins=7, global_tuning=False)
```

```
#visualize topics over time
topics_over_time_vis = topic_model.visualize_topics_over_time(topics_over_time)
topics_over_time_vis
```

```
#another topic model with fixed number of topics
topic_model_2 = BERTopic(nr_topics=50)
topics_2, probs_2 = topic_model_2.fit_transform(cleaned_texts)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[13], line 3
      1 #another topic model with fixed number of topics
      2 topic_model_2 = BERTopic(nr_topics=50)
----> 3 topics_2, probs_2 = topic_model_2.fit_transform(cleaned_texts)

File c:\Users\ramirolobo\AppData\Local\Programs\Python\Python312\Lib\site-packages\bertopic\_bertopic.py:454, in
BERTopic.fit_transform(self, documents, embeddings, images, y)
    452     logger.info("Embedding - Transforming documents to embeddings.")
    453     self.embedding_model = select_backend(self.embedding_model, language=self.language, verbose=self.verbose)
--> 454     embeddings = self._extract_embeddings(
    455         documents.Document.values.tolist(),
    456         images=images,
    457         method="document",
    458         verbose=self.verbose,
    459     )
    460     logger.info("Embedding - Completed \u2713")
    461 else:

File c:\Users\ramirolobo\AppData\Local\Programs\Python\Python312\Lib\site-packages\bertopic\_bertopic.py:3711, in
BERTopic._extract_embeddings(self, documents, images, method, verbose)
   3709     embeddings = self.embedding_model.embed_words(words=documents, verbose=verbose)
   3710 elif method == "document":
-> 3711     embeddings = self.embedding_model.embed_documents(documents, verbose=verbose)
   3712 elif documents[0] is None and images is None:
   3713     raise ValueError(
   3714         "Make sure to use an embedding model that can either embed documents"
   3715         "or images depending on which you want to embed."
   3716     )

File c:\Users\ramirolobo\AppData\Local\Programs\Python\Python312\Lib\site-packages\bertopic\backend\_base.py:62, in
BaseEmbedder.embed_documents(self, document, verbose)
     50 def embed_documents(self, document: List[str], verbose: bool = False) -> np.ndarray:
     51     """Embed a list of n words into an n-dimensional
     52     matrix of embeddings.
     53
   (...)
     60         that each have an embeddings size of `m`
     61     """
---> 62     return self.embed(document, verbose)

File c:\Users\ramirolobo\AppData\Local\Programs\Python\Python312\Lib\site-
packages\bertopic\backend\_sentencetransformers.py:84, in SentenceTransformerBackend.embed(self, documents, verbose)
     72 def embed(self, documents: List[str], verbose: bool = False) -> np.ndarray:
     73     """Embed a list of n documents/words into an n-dimensional
     74     matrix of embeddings.
     75
   (...)
     82         that each have an embeddings size of `m`
     83     """
---> 84     embeddings = self.embedding_model.encode(documents, show_progress_bar=verbose)
     85     return embeddings

File c:\Users\ramirolobo\AppData\Local\Programs\Python\Python312\Lib\site-
packages\sentence_transformers\SentenceTransformer.py:685, in SentenceTransformer.encode(self, sentences, prompt_name, prompt,
batch_size, show_progress_bar, output_value, precision, convert_to_numpy, convert_to_tensor, device, normalize_embeddings,
**kwargs)
    682 features.update(extra_features)
    684 with torch.no_grad():
--> 685     out_features = self.forward(features, **kwargs)
    686     if self.device.type == "hpu":
    687         out_features = copy.deepcopy(out_features)

File c:\Users\ramirolobo\AppData\Local\Programs\Python\Python312\Lib\site-
packages\sentence_transformers\SentenceTransformer.py:758, in SentenceTransformer.forward(self, input, **kwargs)
    756     module_kwarg_keys = self.module_kwargs.get(module_name, [])
    757     module_kwargs = {key: value for key, value in kwargs.items() if key in module_kwarg_keys}
--> 758     input = module(input, **module_kwargs)
    759 return input

File c:\Users\ramirolobo\AppData\Local\Programs\Python\Python312\Lib\site-packages\torch\nn\modules\module.py:1751, in
Module._wrapped_call_impl(self, *args, **kwargs)
   1749     return self._compiled_call_impl(*args, **kwargs)  # type: ignore[misc]
   1750 else:
-> 1751     return self._call_impl(*args, **kwargs)

File c:\Users\ramirolobo\AppData\Local\Programs\Python\Python312\Lib\site-packages\torch\nn\modules\module.py:1762, in
Module._call_impl(self, *args, **kwargs)
   1757 # If we don't have any hooks, we want to skip the rest of the logic in
   1758 # this function, and just call forward.
   1759 if not (self._backward_hooks or self._backward_pre_hooks or self._forward_hooks or self._forward_pre_hooks
   1760         or _global_backward_pre_hooks or _global_backward_hooks
   1761         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1762     return forward_call(*args, **kwargs)
   1764 result = None
   1765 called_always_called_hooks = set()

File c:\Users\ramirolobo\AppData\Local\Programs\Python\Python312\Lib\site-
packages\sentence_transformers\models\Transformer.py:442, in Transformer.forward(self, features, **kwargs)
```

```
    435 """Returns token_embeddings, cls_token"""
    436 trans_features = {
    437     key: value
    438     for key, value in features.items()
    439     if key in ["input_ids", "attention_mask", "token_type_ids", "inputs_embeds"]
    440 }
--> 442 outputs = self.auto_model(**trans_features, **kwargs, return_dict=True)
    443 token_embeddings = outputs[0]
    444 features["token_embeddings"] = token_embeddings

File c:\Users\ramirolobo\AppData\Local\Programs\Python\Python312\Lib\site-packages\torch\nn\modules\module.py:1751, in
Module._wrapped_call_impl(self, *args, **kwargs)
   1749     return self._compiled_call_impl(*args, **kwargs)  # type: ignore[misc]
   1750 else:
-> 1751     return self._call_impl(*args, **kwargs)

File c:\Users\ramirolobo\AppData\Local\Programs\Python\Python312\Lib\site-packages\torch\nn\modules\module.py:1762, in
Module._call_impl(self, *args, **kwargs)
   1757 # If we don't have any hooks, we want to skip the rest of the logic in
   1758 # this function, and just call forward.
   1759 if not (self._backward_hooks or self._backward_pre_hooks or self._forward_hooks or self._forward_pre_hooks
   1760         or _global_backward_pre_hooks or _global_backward_hooks
   1761         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1762     return forward_call(*args, **kwargs)
   1764 result = None
   1765 called_always_called_hooks = set()

File c:\Users\ramirolobo\AppData\Local\Programs\Python\Python312\Lib\site-
packages\transformers\models\bert\modeling_bert.py:1144, in BertModel.forward(self, input_ids, attention_mask, token_type_ids,
position_ids, head_mask, inputs_embeds, encoder_hidden_states, encoder_attention_mask, past_key_values, use_cache,
output_attentions, output_hidden_states, return_dict)
   1137 # Prepare head mask if needed
   1138 # 1.0 in head_mask indicate we keep the head
   1139 # attention_probs has shape bsz x n_heads x N x N
   1140 # input head_mask has shape [num_heads] or [num_hidden_layers x num_heads]
   1141 # and head_mask is converted to shape [num_hidden_layers x batch x num_heads x seq_length x seq_length]
   1142 head_mask = self.get_head_mask(head_mask, self.config.num_hidden_layers)
-> 1144 encoder_outputs = self.encoder(
   1145     embedding_output,
   1146     attention_mask=extended_attention_mask,
   1147     head_mask=head_mask,
   1148     encoder_hidden_states=encoder_hidden_states,
   1149     encoder_attention_mask=encoder_extended_attention_mask,
   1150     past_key_values=past_key_values,
   1151     use_cache=use_cache,
   1152     output_attentions=output_attentions,
   1153     output_hidden_states=output_hidden_states,
   1154     return_dict=return_dict,
   1155 )
   1156 sequence_output = encoder_outputs[0]
   1157 pooled_output = self.pooler(sequence_output) if self.pooler is not None else None

File c:\Users\ramirolobo\AppData\Local\Programs\Python\Python312\Lib\site-packages\torch\nn\modules\module.py:1751, in
Module._wrapped_call_impl(self, *args, **kwargs)
   1749     return self._compiled_call_impl(*args, **kwargs)  # type: ignore[misc]
   1750 else:
-> 1751     return self._call_impl(*args, **kwargs)

File c:\Users\ramirolobo\AppData\Local\Programs\Python\Python312\Lib\site-packages\torch\nn\modules\module.py:1762, in
Module._call_impl(self, *args, **kwargs)
   1757 # If we don't have any hooks, we want to skip the rest of the logic in
   1758 # this function, and just call forward.
   1759 if not (self._backward_hooks or self._backward_pre_hooks or self._forward_hooks or self._forward_pre_hooks
   1760         or _global_backward_pre_hooks or _global_backward_hooks
   1761         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1762     return forward_call(*args, **kwargs)
   1764 result = None
   1765 called_always_called_hooks = set()

File c:\Users\ramirolobo\AppData\Local\Programs\Python\Python312\Lib\site-
packages\transformers\models\bert\modeling_bert.py:695, in BertEncoder.forward(self, hidden_states, attention_mask, head_mask,
encoder_hidden_states, encoder_attention_mask, past_key_values, use_cache, output_attentions, output_hidden_states,
return_dict)
    684     layer_outputs = self._gradient_checkpointing_func(
    685         layer_module.__call__,
    686         hidden_states,
   (...)
    692         output_attentions,
    693     )
    694 else:
--> 695     layer_outputs = layer_module(
    696         hidden_states,
    697         attention_mask,
    698         layer_head_mask,
    699         encoder_hidden_states,
    700         encoder_attention_mask,
    701         past_key_value,
    702         output_attentions,
    703     )
    705 hidden_states = layer_outputs[0]
    706 if use_cache:
```

```
    760 if use_cache:

File c:\Users\ramirolobo\AppData\Local\Programs\Python\Python312\Lib\site-packages\torch\nn\modules\module.py:1751, in
Module._wrapped_call_impl(self, *args, **kwargs)
   1749     return self._compiled_call_impl(*args, **kwargs)  # type: ignore[misc]
   1750 else:
-> 1751     return self._call_impl(*args, **kwargs)

File c:\Users\ramirolobo\AppData\Local\Programs\Python\Python312\Lib\site-packages\torch\nn\modules\module.py:1762, in
Module._call_impl(self, *args, **kwargs)
   1757 # If we don't have any hooks, we want to skip the rest of the logic in
   1758 # this function, and just call forward.
   1759 if not (self._backward_hooks or self._backward_pre_hooks or self._forward_hooks or self._forward_pre_hooks
   1760         or _global_backward_pre_hooks or _global_backward_hooks
   1761         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1762     return forward_call(*args, **kwargs)
   1764 result = None
   1765 called_always_called_hooks = set()

File c:\Users\ramirolobo\AppData\Local\Programs\Python\Python312\Lib\site-
packages\transformers\models\bert\modeling_bert.py:627, in BertLayer.forward(self, hidden_states, attention_mask, head_mask,
encoder_hidden_states, encoder_attention_mask, past_key_value, output_attentions)
    624     cross_attn_present_key_value = cross_attention_outputs[-1]
    625     present_key_value = present_key_value + cross_attn_present_key_value
--> 627 layer_output = apply_chunking_to_forward(
    628     self.feed_forward_chunk, self.chunk_size_feed_forward, self.seq_len_dim, attention_output
    629 )
    630 outputs = (layer_output,) + outputs
    632 # if decoder, return the attn key/values as the last output

File c:\Users\ramirolobo\AppData\Local\Programs\Python\Python312\Lib\site-packages\transformers\pytorch_utils.py:253, in
apply_chunking_to_forward(forward_fn, chunk_size, chunk_dim, *input_tensors)
    250     # concatenate output at same dimension
    251     return torch.cat(output_chunks, dim=chunk_dim)
--> 253 return forward_fn(*input_tensors)

File c:\Users\ramirolobo\AppData\Local\Programs\Python\Python312\Lib\site-
packages\transformers\models\bert\modeling_bert.py:640, in BertLayer.feed_forward_chunk(self, attention_output)
    638 def feed_forward_chunk(self, attention_output):
    639     intermediate_output = self.intermediate(attention_output)
--> 640     layer_output = self.output(intermediate_output, attention_output)
    641     return layer_output

File c:\Users\ramirolobo\AppData\Local\Programs\Python\Python312\Lib\site-packages\torch\nn\modules\module.py:1751, in
Module._wrapped_call_impl(self, *args, **kwargs)
   1749     return self._compiled_call_impl(*args, **kwargs)  # type: ignore[misc]
   1750 else:
-> 1751     return self._call_impl(*args, **kwargs)

File c:\Users\ramirolobo\AppData\Local\Programs\Python\Python312\Lib\site-packages\torch\nn\modules\module.py:1762, in
Module._call_impl(self, *args, **kwargs)
   1757 # If we don't have any hooks, we want to skip the rest of the logic in
   1758 # this function, and just call forward.
   1759 if not (self._backward_hooks or self._backward_pre_hooks or self._forward_hooks or self._forward_pre_hooks
   1760         or _global_backward_pre_hooks or _global_backward_hooks
   1761         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1762     return forward_call(*args, **kwargs)
   1764 result = None
   1765 called_always_called_hooks = set()

File c:\Users\ramirolobo\AppData\Local\Programs\Python\Python312\Lib\site-
packages\transformers\models\bert\modeling_bert.py:552, in BertOutput.forward(self, hidden_states, input_tensor)
    551 def forward(self, hidden_states: torch.Tensor, input_tensor: torch.Tensor) -> torch.Tensor:
--> 552     hidden_states = self.dense(hidden_states)
    553     hidden_states = self.dropout(hidden_states)
    554     hidden_states = self.LayerNorm(hidden_states + input_tensor)

File c:\Users\ramirolobo\AppData\Local\Programs\Python\Python312\Lib\site-packages\torch\nn\modules\module.py:1751, in
Module._wrapped_call_impl(self, *args, **kwargs)
   1749     return self._compiled_call_impl(*args, **kwargs)  # type: ignore[misc]
   1750 else:
-> 1751     return self._call_impl(*args, **kwargs)

File c:\Users\ramirolobo\AppData\Local\Programs\Python\Python312\Lib\site-packages\torch\nn\modules\module.py:1762, in
Module._call_impl(self, *args, **kwargs)
   1757 # If we don't have any hooks, we want to skip the rest of the logic in
   1758 # this function, and just call forward.
   1759 if not (self._backward_hooks or self._backward_pre_hooks or self._forward_hooks or self._forward_pre_hooks
   1760         or _global_backward_pre_hooks or _global_backward_hooks
   1761         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1762     return forward_call(*args, **kwargs)
```

```
#not meaningful topics
topics_2 = topic_model_2.get_topic_info()
topics_2.head(15)
```

	Topic	Count	Name	Representation	Representative_Docs
0	-1	10	-1_ttttttttttttttttttttttttttttttttttttttt_co...	[ttttttttttttttttttttttttttttttttttttttt, con...	[conquest difficulties national wars wars inst...
1	0	155	0_disastrous_notice__	[disastrous, notice, , , , , , , ,]	[, notice, disastrous]
2	1	4448	1_french_english_ships_sea	[french, english, ships, sea, england, france,...	[ship first last might brought action beginnin...
3	2	17122	2_army_one_upon_would	[army, one, upon, would, war, men, enemy, may,...	[danger could save baggage whole army cohorts ...

```
#add topics to text_chunks df
document_info = topic_model.get_document_info(cleaned_texts)
text_chunks['topic'] = document_info['Topic']
```

```
topic_by_author = (
    text_chunks.groupby("author")["topic"]
    .value_counts()
    .unstack(fill_value=0)
)
```

```
topic_by_author
```

topic	-1	0	1	2	3	4	5	6	7	8	...	72	73	74	75	76	77	78	79	80	81
author																					
Arrian	1858	66	157	2	43	10	1	287	292	1	...	12	0	0	0	1	11	0	0	11	3
Caesar	1916	135	37	12	58	16	0	17	4	4	...	0	0	0	2	9	0	10	0	0	0
Clausewitz	934	30	2	258	444	1	324	1	0	76	...	0	0	2	0	0	0	0	0	0	0
Jomini	1616	247	28	521	43	6	11	1	7	61	...	0	10	1	1	0	1	0	11	0	6
Mahan	299	3782	10	39	41	8	8	1	2	12	...	0	2	0	0	0	0	0	0	0	0
Sun Tzu	560	21	3	83	42	512	5	0	2	18	...	0	0	0	1	0	0	1	0	0	2
Thucydides	702	165	2442	13	250	1	4	4	0	5	...	0	0	9	8	2	0	0	0	0	0

7 rows × 83 columns

```
#topics with more than n occurrences across all chunks
n = 100
top_topics = topics[topics["Count"] > n]["Topic"].tolist()
top_topics.remove(15) # exclude noisy topic 15
top_topics = top_topics[1:]  # exclude noisy topic -1
```

```
#get top topics by author
top_topics_by_author = topic_by_author[top_topics]
```

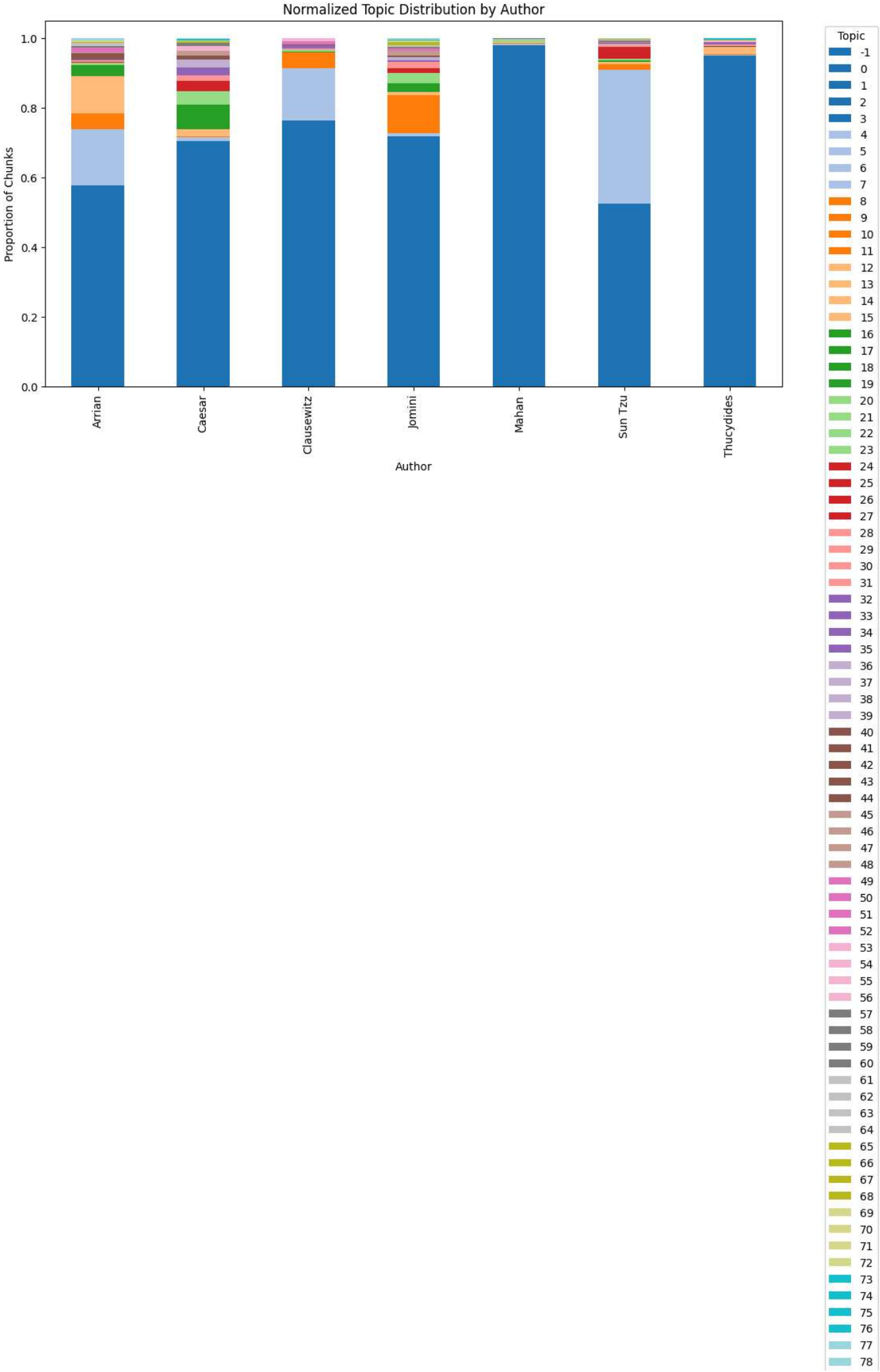
```
# convert to proportion of total document counts per topic belonging to each author
topics_by_author_norm = topic_by_author.div(topic_by_author.sum(axis=1), axis=0)
top_topics_by_author_norm = top_topics_by_author.div(top_topics_by_author.sum(axis=1), axis=0) # get top topics by author normalized
```

```
top_topics_by_author_norm.head(10)
```

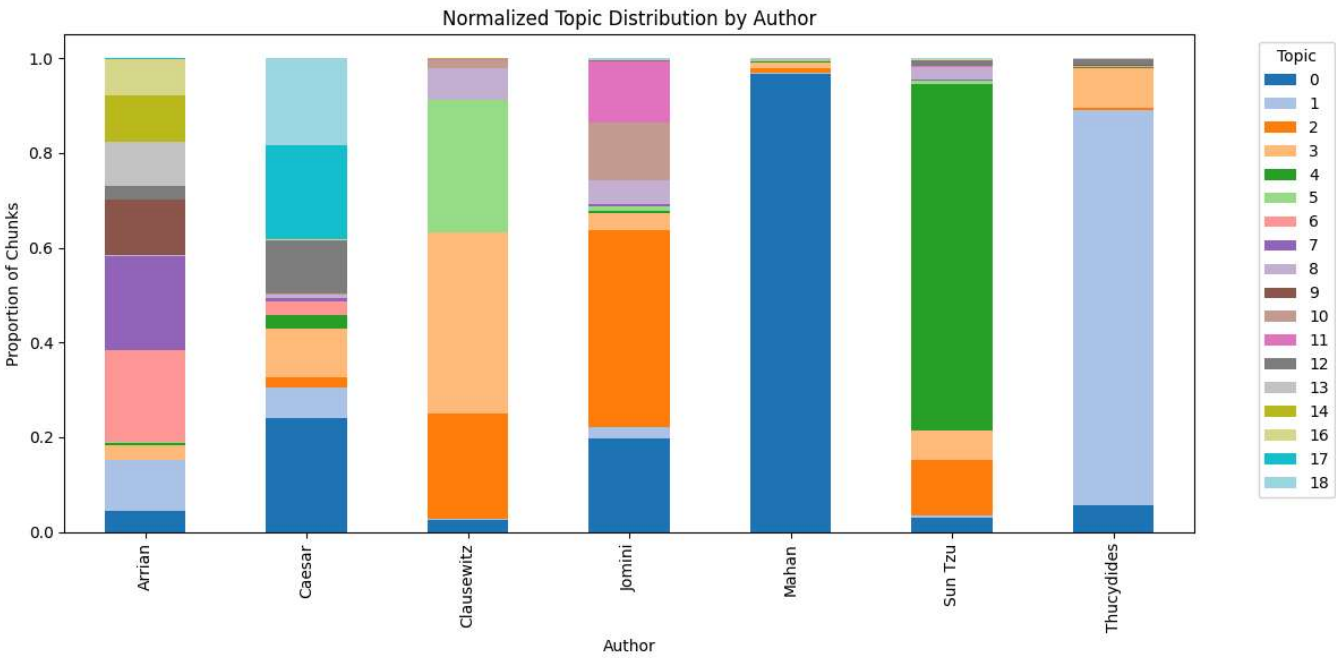
topic	0	1	2	3	4	5	6	7	8	9	10	11
author												
Arrian	0.044867	0.106730	0.001360	0.029232	0.006798	0.000680	0.195105	0.198504	0.000680	0.116927	0.000000	0.000680	0.0278
Caesar	0.239362	0.065603	0.021277	0.102837	0.028369	0.000000	0.030142	0.007092	0.007092	0.000000	0.001773	0.000000	0.1117
Clausewitz	0.025840	0.001723	0.222222	0.382429	0.000861	0.279070	0.000861	0.000000	0.065461	0.000861	0.016365	0.003445	0.0000
Jomini	0.198075	0.022454	0.417803	0.034483	0.004812	0.008821	0.000802	0.005613	0.048917	0.000802	0.121893	0.129912	0.0016
Mahan	0.967511	0.002558	0.009977	0.010489	0.002047	0.002047	0.000256	0.000512	0.003070	0.000000	0.000512	0.000256	0.0000
Sun Tzu	0.030043	0.004292	0.118741	0.060086	0.732475	0.007153	0.000000	0.002861	0.025751	0.000000	0.000000	0.002861	0.0114
Thucydides	0.056352	0.834016	0.004440	0.085383	0.000342	0.001366	0.001366	0.000000	0.001708	0.001025	0.000000	0.000342	0.0119

```
import matplotlib.pyplot as plt
#topics by author
```

```
topics_by_author_norm.plot(kind='bar', stacked=True, colormap='tab20', figsize=(12, 6))
plt.title("Normalized Topic Distribution by Author")
plt.xlabel("Author")
plt.ylabel("Proportion of Chunks")
plt.legend(title="Topic", bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```


Normalized Topic Distribution by Author
Proportion of Chunks
1.0
0.8
0.6
0.4
0.2
0.0
Arrian
Caesar
Clausewitz
Jomini
Mahan
Sun Tzu
Thucydides
Author
Topic
-1
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78


```
#top topics by author
top_topics_by_author_norm.plot(kind='bar', stacked=True, colormap='tab20', figsize=(12, 6))
plt.title("Normalized Topic Distribution by Author")
plt.xlabel("Author")
plt.ylabel("Proportion of Chunks")
plt.legend(title="Topic", bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



Almost all of Mahan's documents texts under topic 0. Similarly, for Thucydides almost all topics fall under Topic 1 and for Sun Tzu almsot all of his texts fall under Topic 4. For the remaining four authors, there is a larger degree of variation in the topics their texts were classified. Clausewitz's texts largely fall under topics 2, 3, and 5. Jomini's primarily fall under topics 0, 2, 10, and 11.

```
top_topics_by_author
```

topic author	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	16	17	18
Arrian	66	157	2	43	10	1	287	292	1	172	0	1	41	139	143	115	1	0
Caesar	135	37	12	58	16	0	17	4	4	0	1	0	63	0	0	2	112	103
Clausewitz	30	2	258	444	1	324	1	0	76	1	19	4	0	0	1	0	0	0
Jomini	247	28	521	43	6	11	1	7	61	1	152	162	2	2	0	1	1	1
Mahan	3782	10	39	41	8	8	1	2	12	0	2	1	0	1	1	0	0	1
Sun Tzu	21	3	83	42	512	5	0	2	18	0	0	2	8	0	0	2	0	1
Thucydides	165	2442	13	250	1	4	4	0	5	3	0	1	35	4	0	1	0	0

```
topic_by_author = topic_by_author.drop(-1, axis=1)
```

```
author_by_topic_norm = topic_by_author.div(topic_by_author.sum(axis=0), axis=1).T
author_by_top_topics = author_by_topic_norm.loc[top_topics, :]
```

```
author_by_top_topics
```

author	Arrian	Caesar	Clausewitz	Jomini	Mahan	Sun Tzu	Thucydides
topic							
0	0.014845	0.030364	0.006748	0.055556	0.850652	0.004723	0.037112
1	0.058604	0.013811	0.000747	0.010452	0.003733	0.001120	0.911534
2	0.002155	0.012931	0.278017	0.561422	0.042026	0.089440	0.014009
3	0.046688	0.062975	0.482085	0.046688	0.044517	0.045603	0.271444
4	0.018051	0.028881	0.001805	0.010830	0.014440	0.924188	0.001805
5	0.002833	0.000000	0.917847	0.031161	0.022663	0.014164	0.011331
6	0.922830	0.054662	0.003215	0.003215	0.003215	0.000000	0.012862
7	0.951140	0.013029	0.000000	0.022801	0.006515	0.006515	0.000000
8	0.005650	0.022599	0.429379	0.344633	0.067797	0.101695	0.028249
9	0.971751	0.000000	0.005650	0.005650	0.000000	0.000000	0.016949
10	0.000000	0.005747	0.109195	0.873563	0.011494	0.000000	0.000000
11	0.005848	0.000000	0.023392	0.947368	0.005848	0.011696	0.005848
12	0.275168	0.422819	0.000000	0.013423	0.000000	0.053691	0.234899
13	0.952055	0.000000	0.000000	0.013699	0.006849	0.000000	0.027397
14	0.986207	0.000000	0.006897	0.000000	0.006897	0.000000	0.000000
16	0.950413	0.016529	0.000000	0.008264	0.000000	0.016529	0.008264
17	0.008772	0.982456	0.000000	0.008772	0.000000	0.000000	0.000000
18	0.000000	0.971698	0.000000	0.009434	0.009434	0.009434	0.000000

```
author_by_top_topics.plot(
    kind='bar',
    stacked=True,
    colormap='tab20',
    figsize=(12, 6)
)

plt.title("Normalized Author Distribution for Top Topics")
plt.xlabel("Topic")  # <-- Corrected
plt.ylabel("Proportion of Text Assigned to the Topic from Each Author")  # <-- Corrected
plt.legend(title="Author", bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```