[Hyperparameter Tuning](https://maartengr.github.io/BERTopic/getting_started/parameter%20tuning/parametertuning.html)[¶](https://maartengr.github.io/BERTopic/getting_started/parameter%20tuning/parametertuning.html#hyperparameter-tuning)

Although BERTopic works quite well out of the box, there are a number of hyperparameters to tune according to your use case. This section will focus on important parameters directly accessible in BERTopic but also hyperparameter optimization in sub-models such as HDBSCAN and UMAP.

**BERTopic**[¶](https://maartengr.github.io/BERTopic/getting_started/parameter%20tuning/parametertuning.html" \l "bertopic" \o "Permanent link)

When instantiating BERTopic, there are several hyperparameters that you can directly adjust that could significantly improve the performance of your topic model. In this section, we will go through the most impactful parameters in BERTopic and directions on how to optimize them.

**language**[¶](https://maartengr.github.io/BERTopic/getting_started/parameter%20tuning/parametertuning.html#language)

The language parameter is used to simplify the selection of models for those who are not familiar with sentence-transformers models.

In essence, there are two options to choose from:

* language = "english" or
* language = "multilingual"

The English model is "all-MiniLM-L6-v2" and can be found [here](https://www.sbert.net/docs/pretrained_models.html). It is the default model that is used in BERTopic and works great for English documents.

The multilingual model is "paraphrase-multilingual-MiniLM-L12-v2" and supports over 50+ languages which can be found [here](https://www.sbert.net/docs/pretrained_models.html). The model is very similar to the base model but is trained on many languages and has a slightly different architecture.

**top\_n\_words**[¶](https://maartengr.github.io/BERTopic/getting_started/parameter%20tuning/parametertuning.html" \l "top_n_words" \o "Permanent link)

top\_n\_words refers to the number of words per topic that you want to be extracted. In practice, I would advise you to keep this value below 30 and preferably between 10 and 20. The reasoning for this is that the more words you put in a topic the less coherent it can become. The top words are the most representative of the topic and should be focused on.

**n\_gram\_range**[¶](https://maartengr.github.io/BERTopic/getting_started/parameter%20tuning/parametertuning.html" \l "n_gram_range" \o "Permanent link)

The n\_gram\_range parameter refers to the CountVectorizer used when creating the topic representation. It relates to the number of words you want in your topic representation. For example, "New" and "York" are two separate words but are often used as "New York" which represents an n-gram of 2. Thus, the n\_gram\_range should be set to (1, 2) if you want "New York" in your topic representation.

**min\_topic\_size**[¶](https://maartengr.github.io/BERTopic/getting_started/parameter%20tuning/parametertuning.html" \l "min_topic_size" \o "Permanent link)

min\_topic\_size is an important parameter! It is used to specify what the minimum size of a topic can be. The lower this value the more topics are created. If you set this value too high, then it is possible that simply no topics will be created! Set this value too low and you will get many microclusters.

It is advised to play around with this value depending on the size of your dataset. If it nears a million documents, then it is advised to set it much higher than the default of 10, for example, 100 or even 500.

**nr\_topics**[¶](https://maartengr.github.io/BERTopic/getting_started/parameter%20tuning/parametertuning.html" \l "nr_topics" \o "Permanent link)

nr\_topics can be a tricky parameter. It specifies, after training the topic model, the number of topics that will be reduced. For example, if your topic model results in 100 topics but you have set nr\_topics to 20 then the topic model will try to reduce the number of topics from 100 to 20.

This reduction can take a while as each reduction in topics activates a c-TF-IDF calculation. If this is set to None, no reduction is applied. Use "auto" to automatically reduce topics using HDBSCAN.

**low\_memory**[¶](https://maartengr.github.io/BERTopic/getting_started/parameter%20tuning/parametertuning.html" \l "low_memory" \o "Permanent link)

low\_memory sets UMAP's low\_memory to True to make sure that less memory is used in the computation. This slows down computation but allows UMAP to be run on low-memory machines.

**calculate\_probabilities**[¶](https://maartengr.github.io/BERTopic/getting_started/parameter%20tuning/parametertuning.html" \l "calculate_probabilities" \o "Permanent link)

calculate\_probabilities lets you calculate the probabilities of each topic in each document. This is computationally quite expensive and is turned off by default.

**UMAP**[¶](https://maartengr.github.io/BERTopic/getting_started/parameter%20tuning/parametertuning.html#umap)

UMAP is an amazing technique for dimensionality reduction. In BERTopic, it is used to reduce the dimensionality of document embedding into something easier to use with HDBSCAN to create good clusters.

However, it does has a significant number of parameters you could take into account. As exposing all parameters in BERTopic would be difficult to manage, we can instantiate our UMAP model and pass it to BERTopic:

from umap import UMAP

umap\_model = UMAP(n\_neighbors=15, n\_components=10, metric='cosine', low\_memory=False)

topic\_model = BERTopic(umap\_model=umap\_model).fit(docs)

**n\_neighbors**[¶](https://maartengr.github.io/BERTopic/getting_started/parameter%20tuning/parametertuning.html" \l "n_neighbors" \o "Permanent link)

n\_neighbors is the number of neighboring sample points used when making the manifold approximation. Increasing this value typically results in a more global view of the embedding structure whilst smaller values result in a more local view. Increasing this value often results in larger clusters being created.

**n\_components**[¶](https://maartengr.github.io/BERTopic/getting_started/parameter%20tuning/parametertuning.html" \l "n_components" \o "Permanent link)

n\_components refers to the dimensionality of the embeddings after reducing them. This is set as a default to 5 to reduce dimensionality as much as possible whilst trying to maximize the information kept in the resulting embeddings. Although lowering or increasing this value influences the quality of embeddings, its effect is largest on the performance of HDBSCAN. Increasing this value too much and HDBSCAN will have a hard time clustering the high-dimensional embeddings. Lower this value too much and too little information in the resulting embeddings are available to create proper clusters. If you want to increase this value, I would advise setting using a metric for HDBSCAN that works well in high dimensional data.

**metric**[¶](https://maartengr.github.io/BERTopic/getting_started/parameter%20tuning/parametertuning.html#metric)

metric refers to the method used to compute the distances in high dimensional space. The default is cosine as we are dealing with high dimensional data. However, BERTopic is also able to use any input, even regular tabular data, to cluster the documents. Thus, you might want to change the metric to something that fits your use case.

**low\_memory**[¶](https://maartengr.github.io/BERTopic/getting_started/parameter%20tuning/parametertuning.html" \l "low_memory_1" \o "Permanent link)

low\_memory is used when datasets may consume a lot of memory. Using millions of documents can lead to memory issues and setting this value to True might alleviate some of the issues.

**HDBSCAN**[¶](https://maartengr.github.io/BERTopic/getting_started/parameter%20tuning/parametertuning.html#hdbscan)

After reducing the embeddings with UMAP, we use HDBSCAN to cluster our documents into clusters of similar documents. Similar to UMAP, HDBSCAN has many parameters that could be tweaked to improve the cluster's quality.

from hdbscan import HDBSCAN

hdbscan\_model = HDBSCAN(min\_cluster\_size=10, metric='euclidean', prediction\_data=True)

topic\_model = BERTopic(hdbscan\_model=hdbscan\_model).fit(docs)

**min\_cluster\_size**[¶](https://maartengr.github.io/BERTopic/getting_started/parameter%20tuning/parametertuning.html" \l "min_cluster_size" \o "Permanent link)

min\_cluster\_size is arguably the most important parameter in HDBSCAN. It controls the minimum size of a cluster and thereby the number of clusters that will be generated. It is set to 10 as a default. Increasing this value results in fewer clusters but of larger size whereas decreasing this value results in more micro clusters being generated. Typically, I would advise increasing this value rather than decreasing it.

**min\_samples**[¶](https://maartengr.github.io/BERTopic/getting_started/parameter%20tuning/parametertuning.html" \l "min_samples" \o "Permanent link)

min\_samples is automatically set to min\_cluster\_size and controls the number of outliers generated. Setting this value significantly lower than min\_cluster\_size might help you reduce the amount of noise you will get. Do note that outliers are to be expected and forcing the output to have no outliers may not properly represent the data.

**metric**[¶](https://maartengr.github.io/BERTopic/getting_started/parameter%20tuning/parametertuning.html#metric_1)

metric, like with HDBSCAN is used to calculate the distances. Here, we went with euclidean as, after reducing the dimensionality, we have low dimensional data and not much optimization is necessary. However, if you increase n\_components in UMAP, then it would be advised to look into metrics that work with high dimensional data.

**prediction\_data**[¶](https://maartengr.github.io/BERTopic/getting_started/parameter%20tuning/parametertuning.html#prediction_data)

Make sure you always set this value to True as it is needed to predict new points later on. You can set this to False if you do not wish to predict any unseen data points.