

Image by the author.

✦ Member-only story

NLP — GETTING STARTED

Interactive Topic Modeling with BERTopic

An in-depth guide to topic modeling with BERTopic



Maarten Grootendorst · Follow

Published in Towards Data Science · 7 min read · Jan 6, 2021



1.3K



18



Every day, businesses deal with large volumes of unstructured text. From customer interactions in emails to online feedback and reviews. To deal with this large amount of text, we look towards **topic modeling**. A technique to automatically extract meaning from documents by identifying recurrent topics.

A few months ago, I wrote an article on leveraging **BERT** for topic modeling. It blew up unexpectedly and I was surprised by the positive feedback I had gotten!

Topic Modeling with BERT

Leveraging BERT and TF-IDF to create easily interpretable topics.

towardsdatascience.com

I decided to focus on further developing the topic modeling technique the article was based on, namely BERTopic.

BERTopic is a topic modeling technique that leverages BERT embeddings and a class-based TF-IDF to create dense clusters allowing for easily interpretable topics whilst keeping important words in the topic descriptions.

I am now at a point where BERTopic has gotten enough traction and development that I feel confident it can replace or complement other topic modeling techniques, such as LDA.

BERTopic

BERTopic is a topic modeling technique that leverages 🤗 transformers and c-TF-IDF to create dense clusters allowing...

github.com

The main purpose of this article is to give you an in-depth overview of BERTopic's features and tutorials on how to best apply this for your own projects.

1. Installation

As always, we start by installing the package via pypi:

```
pip install bertopic
```

To use the visualization options, install BERTopic as follows:

```
pip install bertopic[visualization]
```

2. Basic Usage

Using BERTopic out-of-the-box is quite straightforward. You load in your documents as a list of strings and simply pass it to the `fit_transform`

method.

To give you an example, below we will be using the 20 newsgroups dataset:

```
1 from bertopic import BERTopic
2 from sklearn.datasets import fetch_20newsgroups
3
4 docs = fetch_20newsgroups(subset='all', remove=('headers', 'footers', 'quotes'))['data']
5
6 model = BERTopic()
7 topics, probabilities = model.fit_transform(docs)
```

bertopic.py hosted with ❤ by GitHub

[view raw](#)

There are two outputs generated, `topics` and `probabilities`. A value in `topics` simply represents the topic it is assigned to. Probabilities on the other hand demonstrate the likelihood of a document falling into any of the possible topics.

Next, we can access the topics that were generated by their relative frequency:

```
1 >>> model.get_topic_freq().head()
2 Topic    Count
3 -1        7288
4 49        3992
5 30        701
6 27        684
7 11        568
```

get_topic_freq.py hosted with ❤ by GitHub

[view raw](#)

In the output above, it seems that Topic -1 is the largest. -1 refers to all outliers which do not have a topic assigned. Forcing documents in a topic

could lead to poor performance. Thus, we ignore Topic -1.

Instead, let us take a look at the second most frequent topic that was generated, namely Topic 49:

```
1 >>>> model.get_topic(49)
2 [('windows', 0.006152228076250982),
3  ('drive', 0.004982897610645755),
4  ('dos', 0.004845038866360651),
5  ('file', 0.004140142872194834),
6  ('disk', 0.004131678774810884),
7  ('mac', 0.003624848635985097),
8  ('memory', 0.0034840976976789903),
9  ('software', 0.0034415334250699077),
10 ('email', 0.0034239554442333257),
11 ('pc', 0.003047105930670237)]
```

get_topic_49.py hosted with ❤ by GitHub

[view raw](#)

Since I created this model, I am obviously biased, but to me, this does seem like a coherent and easily interpretable topic!

Languages

Under the hood, BERTopic is using `sentence-transformers` to create embeddings for the documents you pass it. As a default, BERTopic is set to using an English model but it supports any language for which an embedding model exists.

You can choose the language by simply setting the `language` parameter in BERTopic:

```
1 from bertopic import BERTopic
2 model = BERTopic(language="Dutch")
```

language.py hosted with ❤️ by GitHub

[view raw](#)

When you select a language, the corresponding `sentence-transformers` model will be loaded. This is typically a multilingual model that supports many languages.

Having said that, if you have a mixture of language in your documents you can use `BERTopic(language="multilingual")` to select a model that supports over 50 languages!

Embedding model

To chose a different pre-trained embedding model, we simply pass it through BERTopic by pointing the variable `embedding_model` towards the corresponding sentence-transformers model:

```
1 from bertopic import BERTopic
2 model = BERTopic(embedding_model="xlm-r-bert-base-nli-stsb-mean-tokens")
```

embedding_model.py hosted with ❤️ by GitHub

[view raw](#)

Click [here](#) for a list of supported sentence transformers models.

Save/Load BERTopic model

We can easily save a trained BERTopic model by calling `save` :

```
1 from bertopic import BERTopic
2 model = BERTopic()
3 model.save("my_model")
```

save.py hosted with ❤️ by GitHub

[view raw](#)

Then, we can load the model in one line:

```
1 loaded_model = BERTopic.load("my_model")
```

load.py hosted with ❤️ by GitHub

[view raw](#)

3. Visualization

Now that we have covered the basics and generated our topics, we visualize them! Having an overall picture of the topics allows us to generate an internal perception of the topic model's quality.

Visualize Topics

To visualize our topics I took inspiration from [LDAvis](#) which is a framework for visualizing LDA topic models. It allows you to interactively explore topics and the words that describe them.

To achieve a similar effect in BERTopic, I embedded our class-based TF-IDF representation of the topics in 2D using Umap. Then, it was a simple matter of visualizing the dimensions using Plotly to create an interactive view.

To do this, simply call `model.visualize_topics()` in order to visualize our topics:

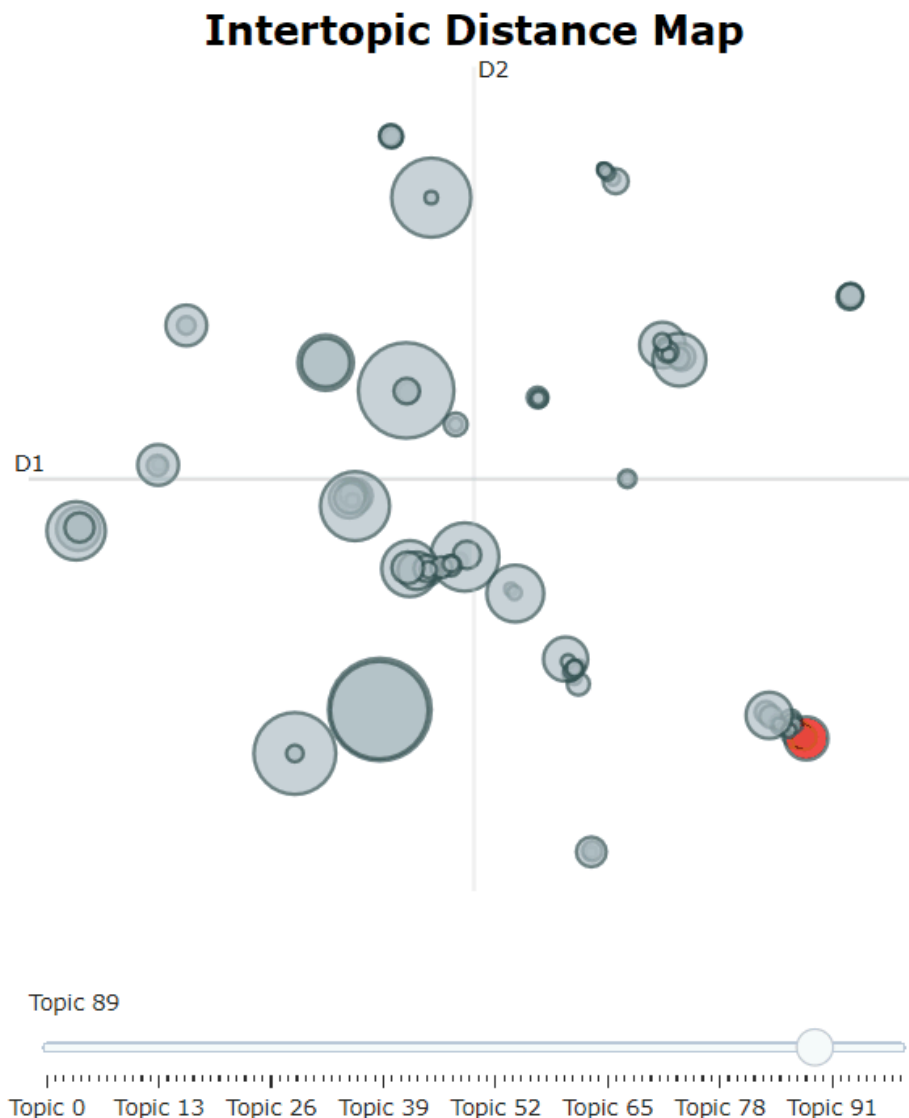


Image by the author.

An interactive Plotly figure will be generated which can be used as indicated in the animation above. Each circle indicates a topic and its size is the frequency of the topic across all documents.

To try it out yourself, take a look at the documentation [here](#) where you can find an interactive version!

Visualize Probabilities

For each document, we can also visualize the probability of that document belong to each possible topic. To do so, we use the variable `probabilities`

after running BERTopic to understand how confident the model is for that instance.

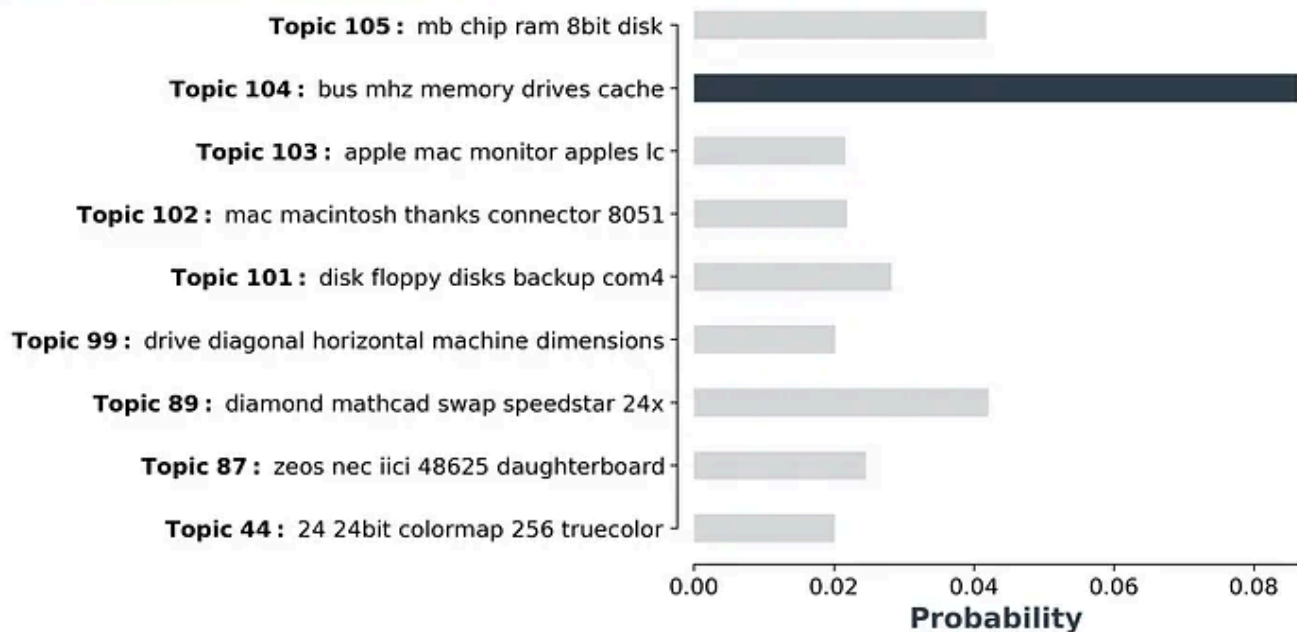
Since there are too many topics to visualize, we visualize the probability distribution of the most probable topics:

```
1 model.visualize_distribution(probabilities[0])
```

visualize_distribution.py hosted with ❤ by GitHub

[view raw](#)

Topic Probability Distribution



It seems that for this document, the model had some more difficulty choosing the correct topic as they were multiple topics very similar to each other.

4. Topic Reduction

As we have seen before, hundreds of topics could be generated. At times, this might simply be too much for you to explore or a too fine-grained solution.

Fortunately, we can reduce the number of topics by merging pairs of topics that are most similar to each other, as indicated by the cosine similarity between c-TF-IDF vectors.

Below, I will go into three methods for reducing the number of topics that result from BERTopic.

Manual Topic Reduction

When initiating your BERTopic model, you might already have a feeling of the number of topics that could reside in your documents.

By setting the `nr_topics` variable, BERTopic will find the most similar pairs of topics and merge them, starting from the least frequent topic, until we reach the value of `nr_topics`:

```
1 from bertopic import BERTopic
2 model = BERTopic(nr_topics=20)
```

manual_topic_reduction.py hosted with ❤ by GitHub

[view raw](#)

It is advised, however, to keep a decently high value, such as 50 to prevent topics from being merged that should not.

Automatic Topic Reduction

As indicated above, if you merge topics to a low `nr_topics` topics will be forced to merge even though they might not actually be that similar to each other.

Instead, we can reduce the number of topics iteratively as long as a pair of topics is found that exceeds a minimum similarity of 0.9.

To use this option, we simply set `nr_topics` to `"auto"` before training our model:

```
1 from bertopic import BERTopic
2 model = BERTopic(nr_topics="auto")
```

auto_topic_reduction.py hosted with ❤️ by GitHub

[view raw](#)

[Open in app](#) ↗

Medium

 Search

 Write





with the number of topics.

Fortunately, we can reduce the number of topics after having trained a BERTopic model. Another advantage of doing so is that you can decide the number of topics after knowing how many are actually created:

```
1 from bertopic import BERTopic
2
3 model = BERTopic()
4 topics, probs = model.fit_transform(docs)
5
6 # Further reduce topics
7 new_topics, new_probs = model.reduce_topics(docs, topics, probs, nr_topics=30)
```

new_topic_reduction.py hosted with ❤️ by GitHub

[view raw](#)

Using the code above, we reduce the number of topics to 30 after having trained the model. This allows you to play around with the number of topics that suit your use-case!

5. Topic Representation

Topics are typically represented by a set of words. In BERTopic, these words are extracted from the documents using a class-based TF-IDF.

At times, you might not be happy with the representation of the topics that were created. This is possible when you selected to have only 1-gram words as representation. Perhaps you want to try out a different n-gram range or you have a custom vectorizer that you want to use.

To update the topic representation after training, we can use the function `update_topics` to update the topic representation with new parameters for c-TF-IDF:

```
1 # Update topic representation by increasing n-gram range and removing english stopwords
2 model.update_topics(docs, topics, n_gram_range=(1, 3), stop_words="english")
```

`update_topic_representation.py` hosted with ❤️ by GitHub

[view raw](#)

We can also use a custom `CountVectorizer` instead:

```
1 from sklearn.feature_extraction.text import CountVectorizer
2
3 cv = CountVectorizer(ngram_range=(1, 3), stop_words="english")
4 model.update_topics(docs, topics, vectorizer=cv)
```

`cv_topic_update.py` hosted with ❤️ by GitHub

[view raw](#)

6. Custom Embeddings

Why limit ourselves to publicly available pre-trained embeddings? You might have very specific data for which you have created an embedding model that you could not have found pre-trained available.

Transformer Models

To use any embedding model that you trained yourself, you will only have to embed your documents with that model. You can then pass in the embeddings and BERTopic will do the rest:

As you can see above, we used a SentenceTransformer model to create the embedding. You could also have used 🧠 transformers , Doc2Vec , or any other embedding method.

TF-IDF

While we are at it, why limit ourselves to transformer models? There is a reason why statistical models, such as TF-IDF, are still around. They work great out-of-the-box without much tuning!

As you might have guessed, it is also possible to use TF-IDF on the documents and use them as input for BERTopic. We simply create a TF-IDF matrix and pass it through `fit_transform`:

Here, you will probably notice that creating the embeddings is quite fast whereas `fit_transform` is quite slow. This is to be expected as reducing the dimensionality of a large sparse matrix takes some time. The inverse of using transformer embeddings is true: creating the embeddings is slow whereas `fit_transform` is quite fast.

Thank you for reading!

If you are, like me, passionate about AI, Data Science, or Psychology, please feel free to add me on [LinkedIn](#) or follow me on [Twitter](#).

You can find BERTopic, as well as its documentation, below:

BERTopic

BERTopic is a topic modeling technique that leverages 🧠 transformers and c-TF-IDF to create dense clusters allowing...

[github.com](#)

Artificial Intelligence

Machine Learning

Python

Data Science

Getting Started



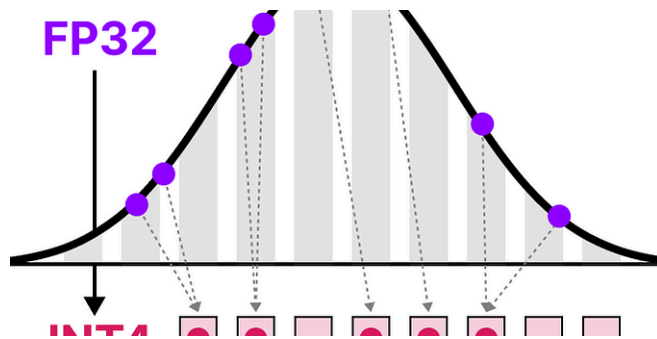
Written by Maarten Grootendorst 

Follow

7K Followers · Writer for Towards Data Science

Data Scientist | Psychologist. Passionate about anything AI-related! Get in touch:
www.linkedin.com/in/mgrootendorst/

More from Maarten Grootendorst and Towards Data Science



Maarten Grootendorst in Towards Data Science

A Visual Guide to Quantization

Demystifying the compression of large language models

Jul 24 🖱️ 733 💬 2



Shaw Talebi in Towards Data Science

5 AI Projects You Can Build This Weekend (with Python)

From beginner-friendly to advanced

★ Oct 8 🖱️ 2.9K 💬 45

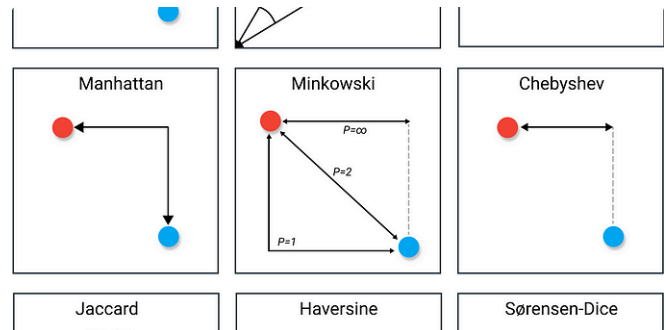


Mauro Di Pietro in Towards Data Science

GenAI with Python: Build Agents from Scratch (Complete Tutorial)

with Ollama, LangChain, LangGraph (No GPU, No APIKEY)

★ Sep 29 🖱️ 1.6K 💬 23



Maarten Grootendorst in Towards Data Science

9 Distance Measures in Data Science

The advantages and pitfalls of common distance measures

★ Feb 1, 2021 🖱️ 4.2K 💬 26



[See all from Maarten Grootendorst](#)[See all from Towards Data Science](#)

Recommended from Medium

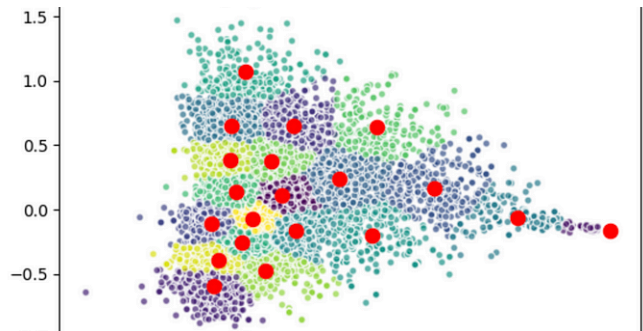


 Mariya Mansurova in Towards Data Science

Topics per Class Using BERTopic

How to understand the differences in texts by categories

Sep 8, 2023  648  4



 Kartheepan G

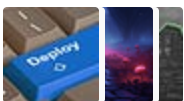
Unveiling Text Clustering: Exploring Algorithms and Text...

Introduction In today's data-driven world, the ability to effectively analyze and organize...

Apr 22  7  1



Lists



Predictive Modeling w/ Python

20 stories · 1599 saves



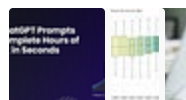
Practical Guides to Machine Learning

10 stories · 1947 saves



Natural Language Processing


1759 stories · 1358 saves



ChatGPT prompts

50 stories · 2096 saves

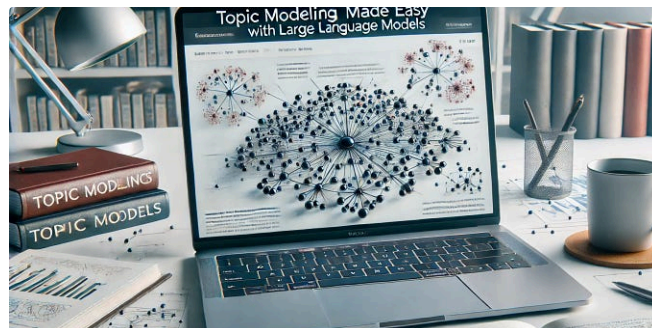



 DhanushKumar

Topic Modelling with BERTopic

BERTopic is a topic modeling technique that leverages BERT (Bidirectional Encoder...

Jul 1  13




 Nakano Kappei

Topic Modeling Made Easy with Large Language Models

Topic modeling has long been a complex and time-consuming task, requiring significant...

Jun 22  2  1




 Emmanuel Ikogho

Data Science is dying; here's why

Why 85% of data science projects fail

Sep 3  1.5K  72



 Ebad Sayed

Scikit-LLM: Scikit-Learn Meets Large Language Models

As a beginner in Python and ML, I frequently relied on scikit-learn for almost all of my...

Sep 9  70  1



See more recommendations