Member-only story

Topic Modelling with BERTtopic in Python

Hands-on tutorial on modeling political statements with a state-of-the-art transformer-based topic model

Petr Korab · Follow

Published in Towards Data Science · 5 min read · Apr 1, 2024

Photo by Harryarts on Freepik

Topic modeling (i.e., topic identification in a corpus of text data) has developed quickly since the *Latent Dirichlet Allocation (LDA)* model was published. This classic topic model, however, does not well capture the relationships between words because it is based on the statistical concept of a bag of words. Recent embedding-based *Top2Vec* and *BERTopic* models address its drawbacks by exploiting pre-trained language models to generate topics.

In this article, we'll use Maarten Grootendorst's (2022) **BERTopic** to identify the terms representing topics in political speech transcripts. It outperforms most traditional and modern topic models in topic modeling metrics on various corpora and has been used in companies, academia (Chagnon, 2024), and the public sector. We'll explore in Python code:

- how to effectively preprocess data
- how to create a Bigram topic model
- how to explore the most frequent terms over time.

1. Example data

As an example dataset, we'll use the *Empoliticon: Political Speeches-Context & Emotion dataset*, released under the *Attribution 4.0 International license,* as part of the Efat et al. (2023) paper. It contains 2010 transcripts of political speeches from the presidents/ prime ministers of the USA, UK, China, and Russia. To make the topic model more focused, the subset only includes the 556 speeches of leaders from Russia:

Date	Speech
2021-07-16	Madam Chair, Colleagues, First of all, I woul...
2021-09-05	Citizens of Russia, Dear veterans, Comrades so...
2021-08-04	Good afternoon, colleagues. Let's start. As yo...
2020-11-21	Colleagues, The scope of problems humanity has...
2020-11-20	Colleagues, friends, First of all, I would lik...
2020-11-16	Colleagues, good afternoon. Today we will disc...
2020-10-11	Today, on November 9, President of the Republi...
2020-10-26	The Russian Federation continues to believe th...
2020-09-25	One of today's major strategic challenges is t...
2020-09-22	Mr. President, Mr. Secretary-General, colleagu...
2020-08-14	Debates around the Iranian issue within the UN...

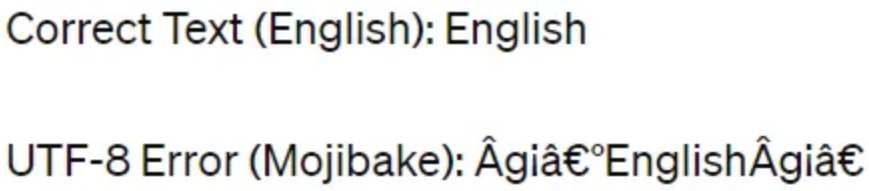
Source: Emopoliticon: Political Speeches-Context & Emotion dataset

2. Data pre-processing

Working with text datasets is complex. Just cleaning involves several steps that should systematically remove all unnecessary information from the dataset. Check all requirements for this project here.

2.1. Fixing mojibake errors

Mojibake is a Japanese word for the confusing text that results from character-encoding errors. Here is an example:



Mojibake example

It is useful to include this step right at the beginning of the cleaning. Correcting encoding-related errors is simple:

```
#  fix mojibake errors
from ftfy import fix_encoding

data['Speech'] = data['Speech'].apply(fix_encoding)
```

mojibake.py hosted with ❤️ by GitHub view raw

2.2. Cleaning special characters, punctuation, and numbers

This step should come right after fixing the encoding errors. The simplest way is to use the cleantext library. Also, consider lower-casing. Does *"labor"* mean the same as *"Labor"* in the dataset? In case it does, add a `lowercase` parameter and apply the cleaning function:

```python
from cleantext import clean

def cleaning(text):
  text = clean(str(text), punct=True, numbers = True, # remove punctuation, numbers
               extra_spaces=True)                     # and extra spaces
  text = clean(text, lowercase=True)                  # lowercase text

  return text.lower()

data['Speech'] = data['Speech'].apply(cleaning)
```

clean_bertopic.py hosted with ❤️ by GitHub view raw

2.3. Define the stopwords removal strategy

Removing the standard list of stopwords is generally a necessary step. Depending on the project focus, it might also be useful to clean data from an additional list of stopwords that don't bring any value. As written in the BERTopic's documentation:

> Removing stop words as a preprocessing step is not advised as the transformer-based embedding models that we use need the full context to create accurate embeddings.

Instead, we use `CountVectorizer` to preprocess our documents **after** having generated embeddings **during** the topic **model generation.**

3. Topic generation

Having a cleaner dataset, it is now possible to remove *English stopwords* along with a list of *additional stopwords*, generate a topic bigram model, and apply it to the data.

```python
# topic generation
from bertopic import BERTopic
from umap import UMAP
from sklearn.feature_extraction.text import CountVectorizer
from nltk.corpus import stopwords

# create a list of speeches
docs = data['Speech'].tolist()

# remove english stopwords with a vectorizer
standard_stopwords = list(stopwords.words('english'))
additional_stopwords = ['let', 'us','like','say','would','also',
                        'th','need','afternoon','ladies','gentleman',
                        'foremost','colleagues','friends','years',
                        'ago','last','year']
full_stopwords = standard_stopwords + additional_stopwords

vectorizer_model = CountVectorizer(ngram_range=(2, 2),
                                   stop_words=full_stopwords)

# generate a bigram topic model with 10 top terms and 8 topics
bertopic_model = BERTopic(top_n_words=10,
                          n_gram_range=(2,2),
                          nr_topics=7,
                          vectorizer_model=vectorizer_model,
                          umap_model = UMAP(random_state=1))  # setting seed topics repr

# fit the model to data
topics, probabilities = bertopic_model.fit_transform(docs)
```

bertopic_model.py hosted with ❤️ by GitHub view raw

Note that the `nr_topics` parameter is set to 7 for generating 6 topics. The remaining topic is used to keep the outliers.

4. Topic visualization

In the next step, let's visualize the data in a heatmap to present the results better. Here is the outcome:

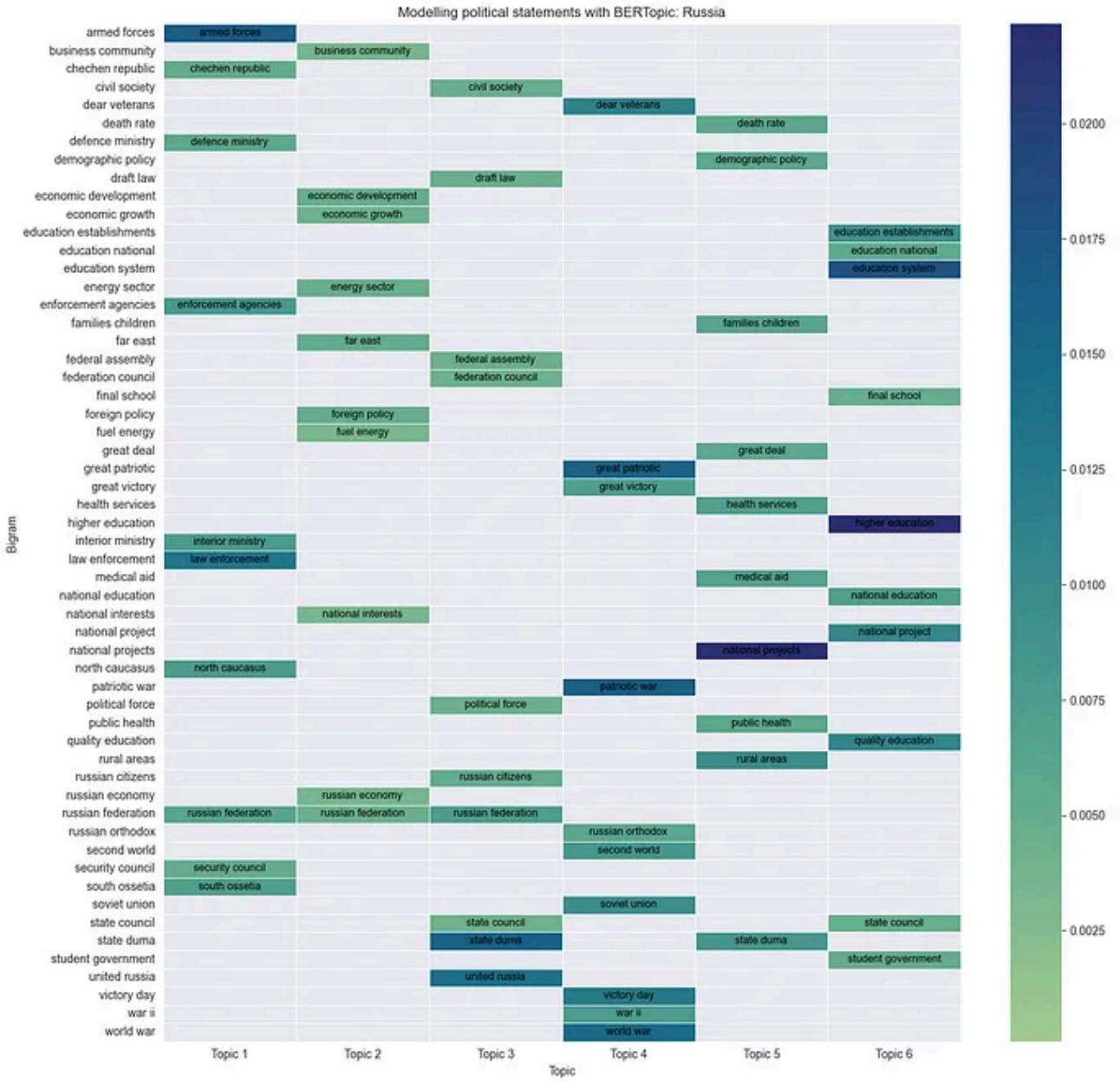


Figure 1: Heatmaps with bigrams and their probabilities, Image by Author

To do so, we'll extract bigrams and their probabilities from the topic model and create a data frame for each of the 6 topics:

```
topic_1 = pd.DataFrame(bertopic_model.get_topic(0), columns=["Topic_1_word", "Topic_1_pro
topic_2 = pd.DataFrame(bertopic_model.get_topic(1), columns=["Topic_2_word", "Topic_2_pro
topic_3 = pd.DataFrame(bertopic_model.get_topic(2), columns=["Topic_3_word", "Topic_3_pro
topic_4 = pd.DataFrame(bertopic_model.get_topic(3), columns=["Topic_4_word", "Topic_4_pro
topic_5 = pd.DataFrame(bertopic_model.get_topic(4), columns=["Topic_5_word", "Topic_5_pro
topic_6 = pd.DataFrame(bertopic_model.get_topic(5), columns=["Topic_6_word", "Topic_6_pro
topics_df = pd.concat([topic_1, topic_2, topic_3, topic_4, topic_5, topic_6], axis=1)
```

topics_df.py hosted with ❤️ by **GitHub** view raw

Next, this code creates a heatmap in Figure 1.

```python
import seaborn as sns
import matplotlib.pyplot as plt

# Initialize DataFrame to store reshaped data
reshaped_data = pd.DataFrame(columns=['Bigram', 'Topic', 'Prob'])

# Reshape data from original DataFrame
for i in range(1, 7):
    topic_word_col = f'Topic_{i}_word'
    topic_prob_col = f'Topic_{i}_prob'
    temp_df = pd.DataFrame({
        'Bigram': topics_df[topic_word_col],
        'Topic': f'Topic {i}',
        'Prob': topics_df[topic_prob_col]
    })
    reshaped_data = pd.concat([reshaped_data, temp_df])

# Set 'Bigram' as index
reshaped_data.set_index('Bigram', inplace=True)

# Create pivot table for heatmap
pivot_table = reshaped_data.pivot(columns='Topic', values='Prob').fillna(0)

# Plot heatmap
plt.figure(figsize=(14, 14))
cmap = sns.color_palette("crest", as_cmap=True)
cmap.set_under(color='white')

# Generate heatmap
sns.heatmap(pivot_table, cmap=cmap, linewidths=0.5, annot=False,
            cbar=True, mask=(pivot_table == 0), vmin=0.0001)

# Add bigrams with non-zero probability on top of the heatmap
for i, bigram in enumerate(pivot_table.index):
    for j, topic in enumerate(pivot_table.columns):
        if pivot_table.loc[bigram, topic] > 0:
            plt.text(j + 0.5, i + 0.5, bigram, ha='center', va='center',
                     color='black' if pivot_table.loc[bigram, topic] > 0 else 'white', f

# Customize plot
plt.title('Modelling political statements with BERTopic: Russia')
plt.xlabel('Topic')
plt.ylabel('Bigram')
plt.tight_layout()
plt.show()
```

heatmap_bertopic.py hosted with ❤️ by GitHub view raw

5. Token frequencies over time

Now, we'll add a perspective on the development of bigrams over time. The goal is to look at which years the bigrams in the Russian leader(s) speeches were most frequently spoken out. The heatmap in Figure 2 displays the frequencies of the 5 most frequent bigrams for each year.

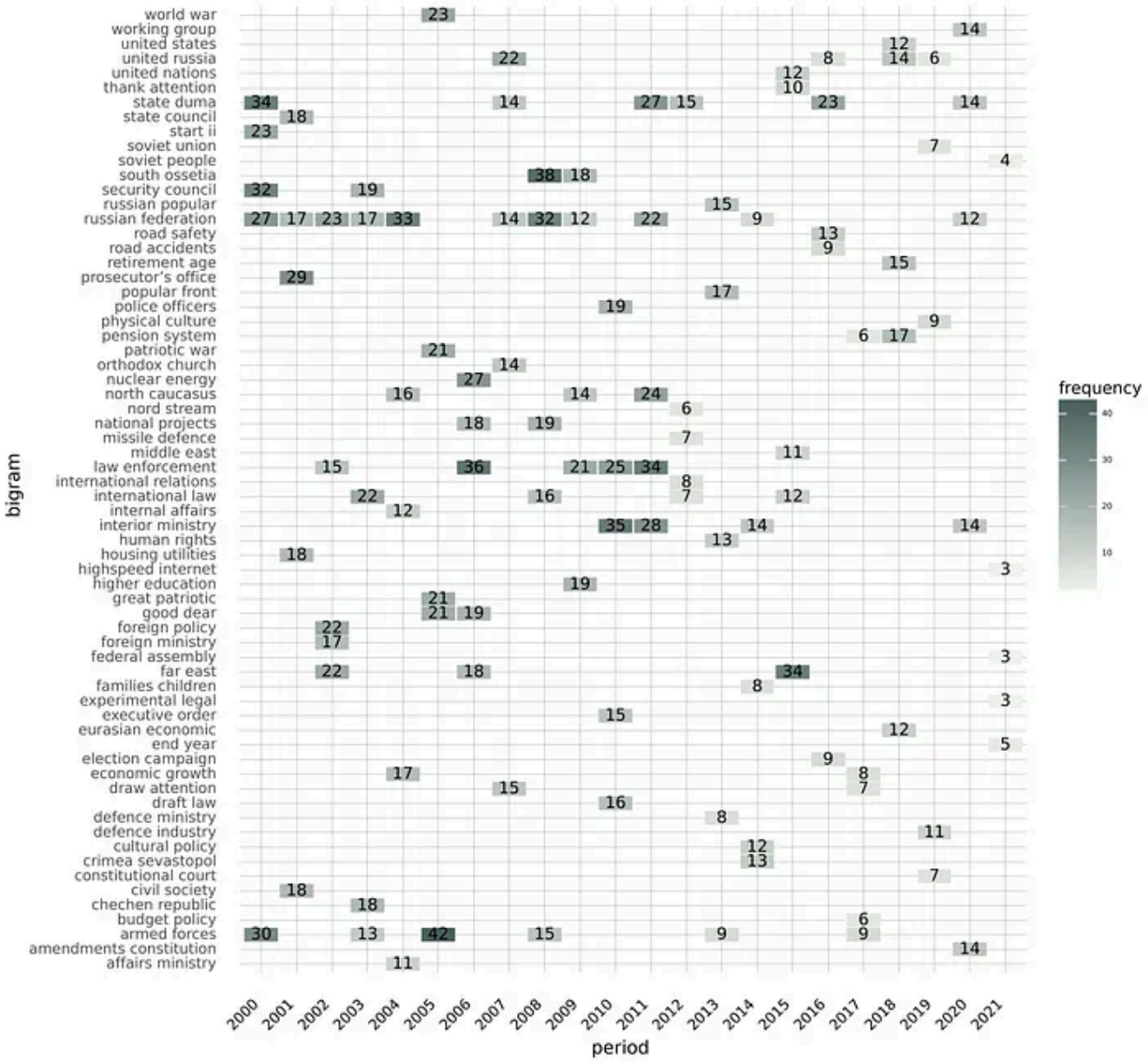


Figure 2: Heatmaps with bigrams and their frequencies by year, Image by Author

The **arabica** library, which is now forthcoming in the **Journal of Open Source Software** (Koráb & Poměnková, 2024), was developed for this purpose.

EDIT Jul 2024: *Arabica has been updated. Check the* ***documentation*** *for the full list of parameters.*

Here is the code generating the heatmap in Figure 2:

```
from arabica import cappuccino

# extended list of additional stopwords
additional_stopwords_ext = ['let us',
                            'like thank',
                            'would like',
                            'th anniversary',
                            'afternoon',
                            'ladies',
                            'gentleman',
                            'foremost',
                            'colleagues',
                            'friends',
                            'percent',
                            'also need',
                            'last year',
                            'great deal',
                            'good dear',
                            'years ago',
                            'good dear']

cappuccino(text = data['Speech'],
           time = data['Date'],
           date_format = 'us',             # Use US-style date format to parse dates
           time_freq = 'Y',                # Calculate bigram frequencies in yearly fre
           plot = 'heatmap',               # Plot type
           ngram = 2,
           max_words = 5,                  # Display 5 most frequent bigrams in each pe
           stopwords = ['english'],        # Remove English set of stopwords
           skip = additional_stopwords_ext, # Remove additional stopwords
           numbers = True,                 # Remove numbers
           lower_case = True)              # Lowercase text
```

heatmap_time.py hosted with ❤️ by GitHub view raw

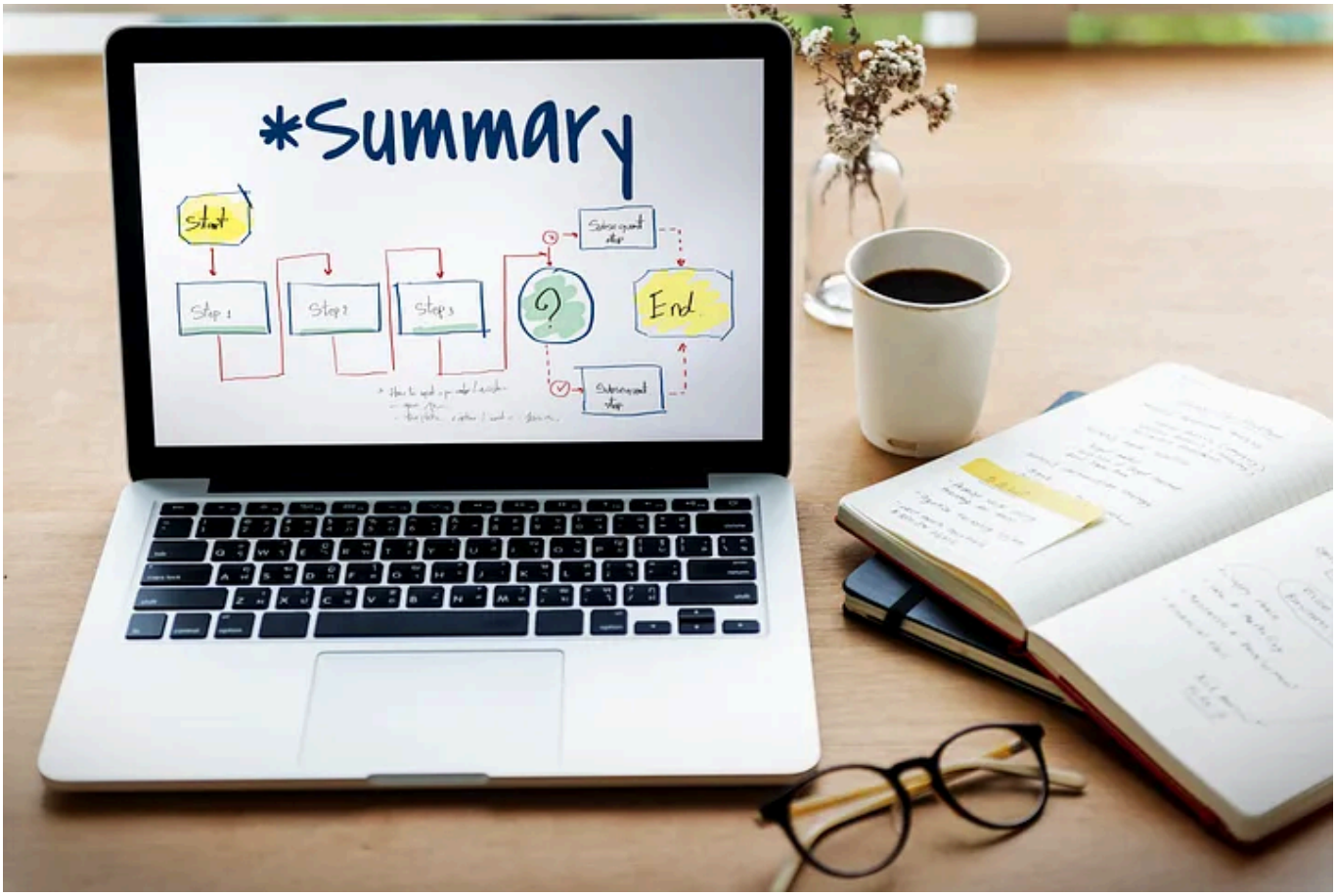


Image by rawpixel on Freepik

Conclusions

This article briefly introduced topic modeling with BERTopic. The model's framework offers many extensions, fine-tuning, and visualization methods (see the documentation). Let's summarize the key findings:

- topic models show 6 distinct topics for ***defense policy*** (topic 1), ***economic development*** (topic 2), ***WW2*** (topic 3), ***internal policies*** (topic 4), ***healthcare and demographics*** (topic 5), and ***education*** (topic 6).
- combining BERTopic with Arabica, we can see that the **foreign** and **defense policy** topics (*"armed forces", "russian federation", "law enforcement"*) were more frequently discussed before 2012, while there is a shallow frequency of topics discussed related to **education** and **healthcare**, especially after 2010.

- The dataset contains more **WW2, foreign,** and **defense policy** terms because Arabica returns absolute frequencies. However, it's difficult to interpret the results well without knowing the regional context.

My previous article briefly explains a simpler approach to topic modeling with **LDA**. The complete code for this tutorial is on my GitHub.

If you enjoy my work, you can invite me for coffee and support my writing. You can also subscribe to my email list to get notified about my new articles. Thanks!

References

[1] Blei, Ng, Jordan (2003). Latent Dirichlet Allocation. *Journal Of Machine Learning Research* 3, pp. 993–1022.

[2] Chagnon, Pandolfi, Donatelli, Ushizima (2024). Benchmarking topic models on scientific articles using BERTeley. *Natural Language Processing Journal 6.*

[3] Efat, Atiq, Abeed, Momin, Alam (2023). Empoliticon: NLP And MLBased Approach For Context And Emotion Classification Of Political Speeches From Transcripts. *IEEE Access,* vol. 11.

[4] Grootendorst (2022). Bertopic: Neural Topic Modeling With A Class-Based TF-IDF Procedure. *Computer Science.*

[5] Koráb, Poměnková (2024). Arabica: A Python package for exploratory analysis of text data. In The Journal of Open Source Software. Journal of Open Source Software. https://doi.org/10.5281/zenodo.10866697.

Topic Modeling | Bertopic | Text Mining | Python | Data Science

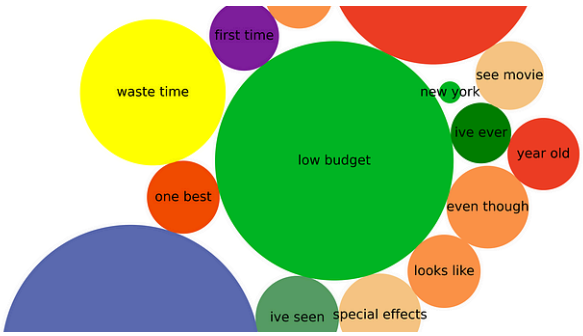
Written by Petr Korab

Follow

909 Followers · Writer for Towards Data Science

Post-doc at Zeppelin University / Python engineer / text mining / data Viz

More from Petr Korab and Towards Data Science



Petr Korab in Towards Data Science

Advanced Visualisations for Text Data Analysis

Explore n-gram word cloud, chord diagram, and a bubble chart, and their implementatio...

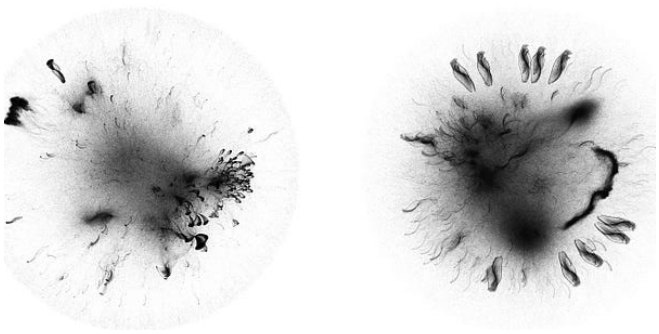
Shaw Talebi in Towards Data Science

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

From beginner-friendly to advanced

May 14, 2022 258 4

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

Petr Korab in Towards AI

Dynamic Visualization of Blockchain Network

Introduction to dynamic network visualization with an example of 0.3 million Bitcoin...

Sep 1 101 1

See all from Petr Korab

See all from Towards Data Science

Recommended from Medium

DhanushKumar

Topic Modelling with BERTopic

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

Jul 1 13

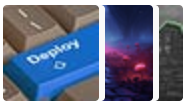
Mariya Mansurova in Towards Data Science

Topic Modelling in production

Leveraging LangChain to move from ad-hoc Jupyter Notebooks to production modular...

Oct 30, 2023 539 3

Lists

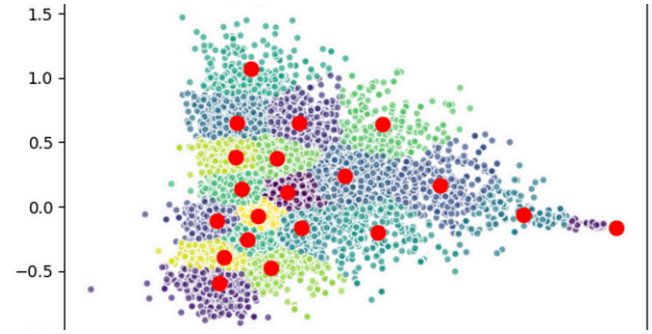


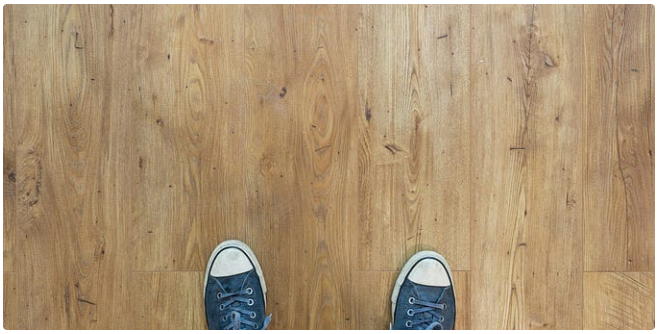
Predictive Modeling w/ Python
20 stories · 1599 saves

Coding & Development
11 stories · 854 saves

Practical Guides to Machine Learning
10 stories · 1947 saves

ChatGPT prompts
50 stories · 2096 saves





Kartheepan G

Ednalyn C. De Dios in Data Science Nerd

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

Topic modeling and network analysis using BERTopic and...

A quick and dirty guide to uncovering themes in domestic violence studies using machine...

Jun 29 151

Emmanuel Ikogho

Data Science is dying; here's why

Why 85% of data science projects fail

Sep 3 1.5K 72

Siddharth Kshirsagar in Stackademic

Advanced Semantic Analysis Integrating BERTopic and LLMs fo...

"Unlock industry insights with AI: BERTopic and Gemini integration revolutionizes...

Jul 21 6

See more recommendations