004.2 - Тематичне моделювання. Сучасні методи

State-of-the-art topic modeling approaches have evolved significantly from traditional methods like Latent Dirichlet Allocation (LDA). These newer approaches leverage deep learning, contextual word embeddings, and variations of probabilistic models to improve topic discovery, coherence, and scalability. Below are some of the prominent modern approaches in topic modeling:

1. BERTopic (Bidirectional Encoder Representations from Transformers for Topic Modeling)

BERTopic is one of the most recent advancements in topic modeling. It leverages **BERT embeddings** (or any transformer-based embeddings), combined with clustering techniques and dimensionality reduction, to produce more semantically coherent topics. BERTopic offers several advantages over traditional approaches:

- **Contextual Word Embeddings**: BERTopic uses transformer-based embeddings like BERT, which capture word meanings in context. This helps in better topic coherence and semantic understanding.
- **HDBSCAN Clustering**: After transforming the document embeddings, BERTopic uses **Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN)** to cluster documents into topics.
- **Dimensionality Reduction**: It uses techniques like **UMAP** (Uniform Manifold Approximation and Projection) for dimensionality reduction, allowing high-dimensional embeddings to be visualized and processed efficiently.
- **Dynamic Topic Modeling**: BERTopic supports dynamic topic modeling, where topics evolve over time, making it useful for time-series data.

2. Top2Vec (Topological Data Analysis for Topic Modeling)

Top2Vec is a novel topic modeling approach that directly learns document and topic embeddings using **word embeddings** and clustering. Key features include:

- **Semantic Embeddings**: It uses **pretrained word embeddings** (e.g., GloVe, Word2Vec, or BERT embeddings) to capture the semantic meaning of words and documents.
- **Direct Topic Discovery**: Unlike traditional methods that first represent documents in a bag-of-words format and then extract topics, Top2Vec learns topics by clustering document embeddings. This method is more natural and provides better-quality topics.
- **Topic Hierarchy**: It creates a hierarchical structure of topics, providing more granular topics or merging them into broader themes.

Top2Vec is highly efficient and can discover topics without requiring the number of topics to be specified upfront.

3. CTM (Combined Topic Modeling with Contextualized Embeddings)

The **Combined Topic Modeling (CTM)** approach integrates traditional topic models with modern neural embeddings. It was proposed to address the challenges of poor semantic coherence and sparsity in traditional topic models like LDA.

- **Contextualized Word Embeddings**: CTM uses **BERT embeddings** to capture the contextual meaning of words, making the generated topics more semantically rich.
- **Neural Variational Inference**: CTM uses neural networks to optimize the variational inference process in topic modeling, improving the modeling of complex data distributions.

CTM combines both generative (probabilistic) models and pre-trained embeddings to generate high-quality topics.

4. NVDM (Neural Variational Document Model)

NVDM is a neural probabilistic model that extends LDA using **variational autoencoders (VAEs)** to represent documents in a continuous latent space. NVDM uses neural networks to approximate posterior distributions, improving topic discovery for larger and more complex datasets. Key aspects include:

- **Continuous Latent Variables**: NVDM represents documents in continuous latent spaces, which is an extension of the discrete latent variables used in LDA.
- **Neural Networks**: It leverages neural networks to learn better representations for documents, making it more flexible in dealing with complex datasets.
- **Scalability**: NVDM scales well for large corpora and generates more coherent topics.

NVDM has been shown to outperform LDA on various text modeling tasks in terms of topic coherence and document representation.

5. ETM (Embedded Topic Model)

ETM is a more advanced neural topic model that integrates **word embeddings** into the topic modeling process. It aims to combine the benefits of topic models and embedding models to generate semantically rich topics. ETM's key advantages include:

- **Word Embeddings**: Instead of generating topics purely from document-term matrices, ETM uses pretrained word embeddings like GloVe or Word2Vec, allowing the model to better capture semantic similarities between words.
- **Neural Networks**: ETM uses neural networks to parameterize the topic distributions, offering greater flexibility in learning word-topic and document-topic distributions.
- **Efficient Inference**: By combining embeddings with efficient inference techniques, ETM is scalable and produces more coherent topics compared to traditional models.

6. Neural LDA (Deep Learning-based Extensions of LDA)

Neural extensions of LDA combine probabilistic topic models with neural networks to improve the quality and flexibility of topics generated. Some notable approaches include:

- **ProdLDA**: This is a neural version of LDA where the Dirichlet distribution is replaced with a logistic normal distribution, allowing for more efficient variational inference and better topic discovery.
- **NVLDA**: Neural Variational LDA incorporates neural variational inference to improve LDA's scalability and accuracy on large datasets. It learns topic distributions more efficiently by using deep learning methods for optimization.

7. Gibbs Sampling-based Variants of LDA

While not new, advanced **Gibbs sampling methods** are still used to improve LDA performance:

- **Collapsed Gibbs Sampling**: This is a popular method for faster and more accurate inference in LDA.
- **Sparse LDA**: Incorporates sparsity constraints in the LDA model to focus only on the most relevant topics for each document, making it more interpretable and efficient.

8. Variational Autoencoders (VAEs) for Topic Modeling

VAEs are used for unsupervised learning and topic modeling due to their ability to capture complex data distributions. Variational autoencoders can be applied to model the latent structure of text data for topic modeling. For instance:

- **Dirichlet VAEs**: These models use Dirichlet distributions in a VAE framework to perform topic modeling, allowing for more flexible posterior approximations and improved performance.

9. Dynamic Topic Models (DTM)

Dynamic Topic Models are an extension of traditional LDA designed for temporal or time-series data. DTM captures how topics evolve over time, making it ideal for applications like:

- **Tracking trends in news** or research papers over time.
- **Modeling temporal changes** in public opinion or social media discussions.

DTM can be combined with embeddings or neural methods to improve scalability and topic coherence.

10. Topic Models with Pretrained Language Models

Using pretrained language models such as **GPT**, **BERT**, or **RoBERTa** for topic modeling is gaining popularity due to their ability to capture context and meaning. These models are

used in combination with clustering methods like k-means or HDBSCAN to group documents into topics after extracting embeddings.

Example: Clustering with BERT embeddings

- **Generate document embeddings** using BERT or any other transformer model.
- **Cluster embeddings** using a clustering algorithm (e.g., k-means, HDBSCAN) to discover topics.
- **Label topics** by examining the top words in the clusters.

This approach allows for more flexible and accurate topic discovery than traditional methods.

Conclusion

The state-of-the-art topic modeling methods, such as **BERTopic**, **Top2Vec**, **ETM**, and **Neural Variational Models**, outperform traditional models like LDA by leveraging deep learning, word embeddings, and advanced clustering techniques. These methods are able to produce more coherent, interpretable, and scalable topics from large text corpora.

Examples

BERTopic

To apply **BERTopic** in Python, you can use the `BERTopic` library, which is built on top of `transformers` and `HDBSCAN`. Here's a step-by-step guide on how to install and apply BERTopic for topic modeling on a collection of documents.

Step 1: Install BERTopic and Dependencies

First, you'll need to install BERTopic and its dependencies.

```
pip install bertopic
pip install sentence-transformers
```

Step 2: Apply BERTopic

Once installed, you can proceed with using BERTopic on a sample dataset. Here's a Python example that shows how to use BERTopic to discover topics in a collection of documents.

Python Code Example:

```
# Import necessary libraries
from bertopic import BERTopic
from sklearn.datasets import fetch_20newsgroups
```

```
# Step 1: Load a sample dataset
# For this example, we'll use the 20 newsgroups dataset
newsgroups = fetch_20newsgroups(subset='all')['data']

# Step 2: Create a BERTopic model
# By default, it uses the "all-MiniLM-L6-v2" sentence transformer
model
topic_model = BERTopic()

# Step 3: Fit the model on the documents
topics, probabilities = topic_model.fit_transform(newsgroups)

# Step 4: View the topics
# Display the top n words for each topic
print(topic_model.get_topic_info())

# Step 5: Get topic details
# For example, get the most relevant words for a specific topic
(e.g., topic 1)
print(topic_model.get_topic(1))

# Step 6: Visualize the topics
# You can visualize the topic clusters
topic_model.visualize_topics()

# Step 7: Visualize the documents and topic probabilities
# This allows you to see how documents relate to topics
topic_model.visualize_documents(newsgroups, probabilities)
```

Explanation of the Code:

1. **Dataset**: In this example, we use the 20 Newsgroups dataset from `scikit-learn` which contains thousands of text documents categorized into different newsgroups. You can replace it with your own dataset if needed.
2. **BERTopic Model**: We create a `BERTopic` object. By default, BERTopic uses the `"all-MiniLM-L6-v2"` sentence transformer model from `sentence-transformers` to generate document embeddings. You can specify another embedding model if needed.
3. **Fit the Model**: We fit the model on the documents using `fit_transform()`. This method returns two things:
    - `topics`: The topic assigned to each document.
    - `probabilities`: The probability that each document belongs to each topic (useful for visualizations).

4. **Get Topic Info**: The `get_topic_info()` method gives an overview of the discovered topics, including the most important words associated with each topic and the size of the topic (i.e., how many documents belong to each topic).
5. **View Specific Topic**: You can inspect a particular topic with `get_topic()`, which will show you the top words associated with that topic.
6. **Visualization**: BERTopic comes with several built-in visualizations:
    - `visualize_topics()`: Creates an interactive 2D plot where each topic is represented by a cluster.
    - `visualize_documents()`: Displays documents in the topic space based on their probabilities of belonging to different topics.

Sample Output

- **Topic Info**: The output will look something like this for the top topics:

```
   Topic  Count                                                Name
0     -1   6422                            -1_topic_name
(unclassified)
1     0    1711         0_topic_name (topic most words in cluster)
2     1     876         1_topic_name (some other set of words)
3     2     592         2_topic_name (...)
...
```

Topic Words: For each topic, BERTopic will show you the most representative words. For example:

```
Topic 1: [('computer', 0.03), ('system', 0.02), ('file', 0.02),
('software', 0.02)]
```

Advanced Options

- **Custom Embeddings**: You can pass your own sentence transformer models or embeddings to BERTopic by specifying the `embedding_model` parameter. For example:

```
from sentence_transformers import SentenceTransformer
embedding_model = SentenceTransformer("paraphrase-MiniLM-L6-v2")
topic_model = BERTopic(embedding_model=embedding_model)
```

Tuning HDBSCAN: You can tune the clustering method (HDBSCAN) by passing custom parameters to the model:

```
from hdbscan import HDBSCAN
hdbscan_model = HDBSCAN(min_cluster_size=10)
topic_model = BERTopic(hdbscan_model=hdbscan_model)
```

This allows you to control how tightly or loosely topics are grouped based on the clustering method.

Conclusion

BERTopic is a powerful and flexible topic modeling tool that leverages state-of-the-art transformer-based embeddings and density-based clustering techniques. It offers high interpretability, interactive visualizations, and works out-of-the-box for many applications. You can experiment with your own datasets, embedding models, and clustering parameters to tailor it to your specific use case.

Top2Vec

To apply **Top2Vec** in Python, follow these steps. **Top2Vec** uses pre-trained word embeddings (e.g., `Doc2Vec`, `Word2Vec`, or `BERT`) to identify clusters of semantically similar documents and then identifies the most representative words from each cluster as the topics.

Step 1: Install Top2Vec and its Dependencies

```
pip install top2vec
pip install umap-learn  # Optional for dimensionality reduction
```

Step 2: Python Code to Apply Top2Vec

Here's an example of how to apply **Top2Vec** for topic modeling:

```
# Import necessary libraries
from top2vec import Top2Vec

# Step 1: Prepare a list of documents
# This can be your own dataset of documents, or use a predefined
list like below
documents = [
      "The cat sat on the mat.",
      "The dog ate the food.",
      "Cats and dogs are common pets.",
      "I love programming in Python.",
      "Python is great for data science.",
      "The cat and the dog are friends.",
      "Data science is a growing field.",
      "I enjoy hiking in the mountains."
]

# Step 2: Create a Top2Vec model
```

```
# By default, Top2Vec uses "universal-sentence-encoder" for
embedding, but you can change it
model = Top2Vec(documents,
embedding_model='universal-sentence-encoder')

# Step 3: Get topic information
# Print the topics with the top n words
topics, topic_sizes = model.get_topics()
for i, topic in enumerate(topics):
     print(f"Topic {i}: {topic}")

# Step 4: Find similar documents for a given topic
# Retrieve documents that are most representative of a specific
topic (e.g., Topic 0)
topic_number = 0
docs, doc_scores =
model.search_documents_by_topic(topic_num=topic_number,
num_docs=5)
print(f"\nDocuments in Topic {topic_number}:")
for doc, score in zip(docs, doc_scores):
     print(f"Score: {score:.4f} | Document: {doc}")

# Step 5: Search for topics by keyword
# You can search for topics by providing a word related to the
topic
similar_topics, similarity_scores =
model.search_topics(keywords=["data"], num_topics=2)
print(f"\nTopics similar to 'data':")
for i, score in zip(similar_topics, similarity_scores):
     print(f"Topic {i} | Similarity: {score:.4f} | Words:
{topics[i]}")

# Step 6: Visualize the topics (optional, requires `umap-learn`)
# Visualize the topics in 2D space
model.visualize_topics()
```

Explanation of the Code:

1. **Documents**: We define a list of text documents (`documents`). You can replace this with your own dataset or a collection of documents you'd like to analyze.
2. **Top2Vec Model**: The `Top2Vec` model is initialized using the `documents` list. You can specify different pre-trained embedding models:
    - `'universal-sentence-encoder'` (default)
    - `'doc2vec'`
    - `'distiluse-base-multilingual-cased'`

    - `'BERT'` or `'sentence-transformers'` embeddings if you want higher quality embeddings for multilingual tasks.
3. **Get Topics**: `get_topics()` retrieves the discovered topics, along with their most representative words.
4. **Search Documents by Topic**: `search_documents_by_topic()` retrieves the most representative documents for a given topic, allowing you to see examples of documents clustered into each topic.
5. **Search Topics by Keyword**: `search_topics()` finds topics that are semantically related to a given keyword.
6. **Visualization**: The `visualize_topics()` method allows you to create a 2D visualization of topics using dimensionality reduction (requires `umap-learn`).

Sample Output:

```
Topic 0: ['cat', 'mat', 'dog', 'sat', 'pets']
Topic 1: ['python', 'programming', 'data', 'science', 'love']
Topic 2: ['hiking', 'mountains', 'enjoy', 'love', 'growing']

Documents in Topic 0:
Score: 0.9912 | Document: The cat sat on the mat.
Score: 0.9874 | Document: Cats and dogs are common pets.
Score: 0.9756 | Document: The cat and the dog are friends.

Topics similar to 'data':
Topic 1 | Similarity: 0.8764 | Words: ['python', 'programming',
'data', 'science', 'love']
```

Key Features of Top2Vec:

- **No Predefined Number of Topics**: Unlike models like LDA, Top2Vec does not require you to specify the number of topics beforehand.
- **Word Embeddings**: Top2Vec uses word embeddings (e.g., Doc2Vec, Universal Sentence Encoder, BERT) to capture semantic information.
- **Topic Search**: You can search for topics related to specific words, which makes it easier to explore large datasets.
- **Interactive Visualization**: Top2Vec allows you to visualize topics and their relationships using `UMAP` for dimensionality reduction.

Advanced Options:

- **Custom Embeddings**: If you want to use custom word embeddings, you can pass your own pre-trained embeddings to Top2Vec using the `embedding_model_path` parameter.
- **Preprocessing**: Top2Vec has built-in preprocessing options like stopword removal and lowercasing, but you can also preprocess the documents yourself before passing them to the model.

Conclusion:

Top2Vec is a state-of-the-art method for discovering topics in text data without requiring the number of topics to be specified in advance. It's fast, easy to use, and provides high-quality topic representations using modern word embeddings. You can apply it to your dataset to quickly uncover and explore the key themes in your text corpus.