AgenTopic: Autotuned Topic Modeling with Agent

Yanlan Kang

1 Overview

Autotuned Topic Modeling enhances traditional neural topic modeling by integrating an iterative feedback loop with a language model (e.g., GPT-4) and a caching memory module to optimize and compare different parameters and topic outcomes, selecting the best parameters for modeling.

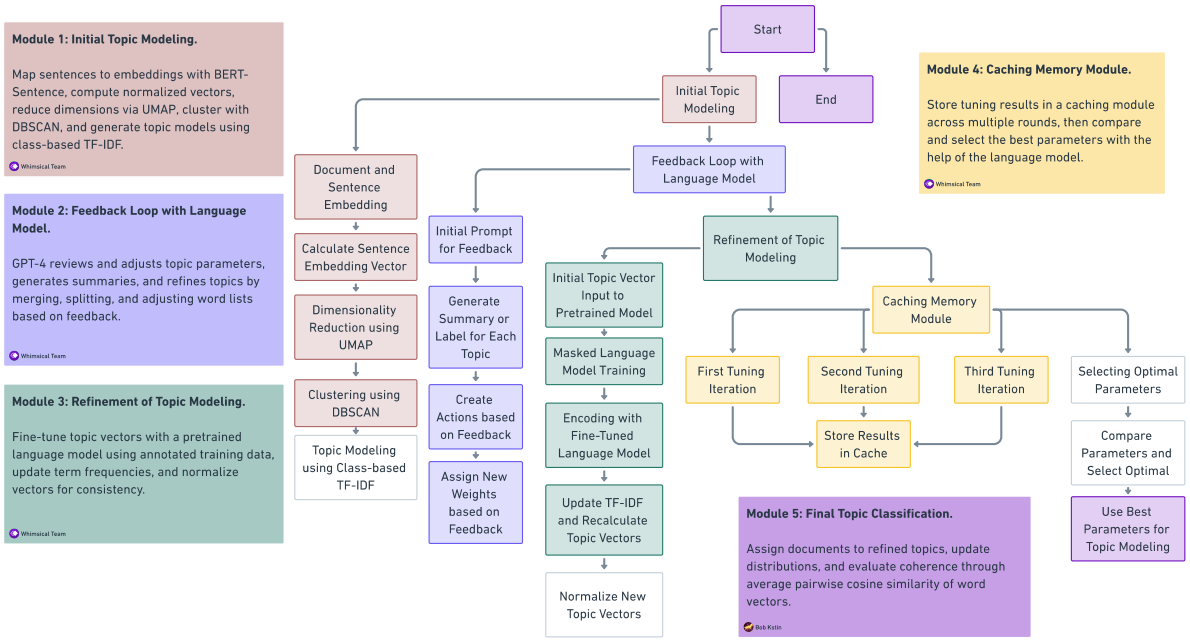

Figure 1: Workflow of the AgenTopic algorithm.

2 Methods

2.1 Initial Topic Modeling

2.1.1 Document and Sentence Embedding

Assume a document set D where each document $d_i \in D$ consists of n_i sentences s_i^j, where $j \in [1, n_i]$. Each sentence s_i^j consists of m_i^j words w_i^{jk}, where $k \in [1, m_i^j]$.

Word Embedding:

Use the RoBERTa-base model to obtain word embeddings for each word w_i^{jk}. RoBERTa-base has an embedding dimension of $d_w = 768$. Thus, each word is represented as a vector $e_i^{jk} \in \mathbb{R}^{768}$.

2.1.2 Sentence Embedding Vector

Attention Mechanism:

Apply an attention mechanism to the word embeddings e_i^{jk} to compute attention weights α_k:

$$\alpha_k = \frac{\exp(\text{score}(e_i^{jk}, c_i^j))}{\sum_{k'=1}^{m_i^j} \exp(\text{score}(e_i^{jk'}, c_i^j))}$$

Here, $\text{score}(e_i^{jk}, c_i^j)$ is a scoring function measuring the relevance of word w_i^{jk} to the context c_i^j of sentence s_i^j. The context vector c_i^j can be computed as the average of the word embeddings in the sentence:

$$c_i^j = \frac{1}{m_i^j} \sum_{k=1}^{m_i^j} e_i^{jk}$$

The scoring function can be defined as the dot product:

$$\text{score}(e_i^{jk}, c_i^j) = e_i^{jk} \cdot c_i^j$$

Sentence Embedding Calculation:

Compute the sentence embedding vector v_i^j as the weighted sum of the word embeddings:

$$v_i^j = \sum_{k=1}^{m_i^j} \alpha_k e_i^{jk}$$

Normalization:

Normalize the sentence embedding vector:

$$v_i^j = \frac{v_i^j}{\|v_i^j\|}$$

2.1.3 Dimensionality Reduction

Use UMAP (Uniform Manifold Approximation and Projection) to reduce the dimensionality of the sentence embeddings v_i^j from $\mathbb{R}^{768}$ to $\mathbb{R}^{d_s}$, where d_s is the reduced dimension (e.g., $d_s = 50$). The reduced sentence embeddings are $u_i^j \in \mathbb{R}^{50}$.

2.1.4 Clustering

Apply the DBSCAN algorithm to cluster the reduced sentence embeddings u_i^j, resulting in K clusters C_k, where $k \in [1, K]$.

2.1.5 Topic Modeling

Use the class-based TF-IDF algorithm to model each cluster C_k and obtain the topic model M_k:

$$M_k = \sum_i \sum_j \sum_{k=1}^{m_i^j} \text{TF}_{ijk} \cdot \text{IDF}_{ijk} \cdot e_i^{jk} \cdot \text{cohesion}(w_i^{jk}, C_k)$$

Here:

- TF_{ijk} is the term frequency of word w_i^{jk} in cluster C_k. - IDF_{ijk} is the inverse document frequency of word w_i^{jk} in cluster C_k. - e_i^{jk} is the word embedding vector of w_i^{jk}. - $\text{cohesion}(w_i^{jk}, C_k)$ is the cohesion score of the word w_i^{jk} within cluster C_k.

Cohesion Score Definition:

The cohesion score $\text{cohesion}(w_i^{jk}, C_k)$ is defined as the average cosine similarity between the word embedding e_i^{jk} and the embeddings of other words in cluster C_k:

$$\text{cohesion}(w_i^{jk}, C_k) = \frac{1}{|W_{C_k}| - 1} \sum_{\substack{e \in W_{C_k} \\ e \neq e_i^{jk}}} \cos(e_i^{jk}, e)$$

where W_{C_k} is the set of word embeddings in cluster C_k, and $\cos(e_i^{jk}, e)$ is the cosine similarity between vectors e_i^{jk} and e.

2.2 Feedback Loop with Language Model

2.2.1 Initial Prompt for Feedback

Present GPT-4 with the initial prompt:

"You are an expert assistant in tuning topic modeling parameters. Please review the current topic modeling results and provide suggestions for improvement."

2.2.2 Generate Topic Summaries and Labels

For each topic M_k, generate a summary or label L_k using the top-ranked words based on their TF-IDF and cohesion scores. Present these summaries to the language model to receive feedback.

2.2.3 Create Actions Based on Feedback

Based on the language model's feedback, create a set of actions A_k for each topic, such as:

- Merging similar topics. - Splitting topics that cover multiple themes. - Adding or removing specific words. - Refining the topic summaries L_k.

2.2.4 Assign New Weights to Word Embeddings

Update the weights $w_i^{jk(\text{new})}$ of each word embedding e_i^{jk} based on the feedback, indicating the relevance of each word to the adjusted topic. The new weight can incorporate the cohesion score and feedback adjustments:

$$w_i^{jk(\text{new})} = w_i^{jk} + \delta w_i^{jk}$$

where δw_i^{jk} represents the weight adjustment derived from the feedback.

2.3 Refinement of Topic Modeling

2.3.1 Fine-Tuning the Language Model

Fine-Tuned Components:

Fine-tune the encoder layers of the RoBERTa-base model to capture topic-specific semantics.

2.3.2 Fine-Tuning Task

Masked Language Modeling (MLM):

- **Construct Training Data**:

- Extract sentences related to each topic M_k from the original document set D to form a local corpus C_k. - Annotate the sentences with topic labels or keywords as additional inputs if necessary.

- **Generate Masked Sequences**:

- Randomly mask 15% of the tokens in the corpus C_k. - The model's task is to predict these masked tokens, focusing on the context of the specific topic.

Training Objective:

Minimize the negative log-likelihood of the masked words:

$$\mathcal{L}_{\text{MLM}} = -\sum_{\text{masked } w} \log P(w|C_k)$$

2.3.3 Encoding with Fine-Tuned Language Model

Use the fine-tuned RoBERTa model to re-encode the word embeddings:

$$e_i^{jk(\text{new})} = \text{RoBERTa}_{\text{fine-tuned}}(w_i^{jk})$$

2.3.4 Update Term Frequency and Inverse Document Frequency

Update $\text{TF}_{ijk}^{(\text{new})}$ and $\text{IDF}_{ijk}^{(\text{new})}$ based on the refined word list and actions A_k. This may involve recalculating frequencies after adding or removing words.

2.3.5 Recalculate Topic Vectors

Recalculate the topic vectors using the updated weights, term frequencies, inverse document frequencies, and word embeddings:

$$M_k^{(\text{tuned})} = \sum_i \sum_j \sum_{k=1}^{m_i^j} w_i^{jk(\text{new})} \cdot \text{TF}_{ijk}^{(\text{new})} \cdot \text{IDF}_{ijk}^{(\text{new})} \cdot e_i^{jk(\text{new})}$$

2.3.6 Normalization

Normalize the new topic vectors:

$$M_k^{(\text{normalized})} = \frac{M_k^{(\text{tuned})}}{\|M_k^{(\text{tuned})}\|}$$

2.4 Caching Memory Module

2.4.1 Multiple Tuning Iterations

Perform three iterations of parameter tuning, each time:
- Applying the feedback to adjust the model. - Storing the results and parameters in the cache.

2.4.2 Selecting Optimal Parameters

Retrieve the cached results and prompt the language model:
"Based on the results of the three tuning iterations, please help identify the optimal parameters for topic modeling."

Compare the parameters using clustering metrics and coherence scores (as defined in the Evaluation Metrics section). Select the parameters that yield the best performance.

3 Evaluation Metrics

To evaluate the performance and coherence of the topic models, calculate the following metrics:

1. **V-measure**: - The harmonic mean of homogeneity and completeness:

$$V = 2 \cdot \frac{\text{homogeneity} \times \text{completeness}}{\text{homogeneity} + \text{completeness}}$$

2. **Completeness**: - Measures whether all members of a given class are assigned to the same cluster:

$$\text{completeness} = 1 - \frac{H(C|K)}{H(C)}$$

3. **Adjusted Rand Index (ARI)**: - Measures the similarity between the predicted clusters and the true clusters, adjusted for chance:

$$\text{ARI} = \frac{\text{RI} - \mathbb{E}[\text{RI}]}{\max(\text{RI}) - \mathbb{E}[\text{RI}]}$$

4. **Silhouette Score**: - Measures how similar an object is to its own cluster compared to other clusters:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

where $a(i)$ is the average intra-cluster distance, and $b(i)$ is the average nearest-cluster distance.

5. **Calinski-Harabasz Index**: - Evaluates cluster validity based on the average between- and within-cluster sum of squares:

$$\text{CH} = \frac{\text{tr}(B_k)/(k-1)}{\text{tr}(W_k)/(n-k)}$$

where B_k is the between-cluster dispersion matrix, W_k is the within-cluster dispersion matrix.

6. **Davies-Bouldin Index**: - Measures the average similarity between each cluster and its most similar one:

$$\text{DB} = \frac{1}{k}\sum_{i=1}^{k}\max_{j\neq i}\left(\frac{s_i + s_j}{d_{ij}}\right)$$

where s_i is the average distance between each point in cluster i and the centroid of cluster i, d_{ij} is the distance between centroids of clusters i and j.

7. **Homogeneity**: - Measures whether each cluster contains only members of a single class:

$$\text{homogeneity} = 1 - \frac{H(K|C)}{H(K)}$$

4 Final Topic Classification

Assign Documents to Topics:

- Assign documents to the newly refined topics based on the updated topic vectors $M_k^{(\text{normalized})}$. - Update the document-topic distribution accordingly.

Evaluate Coherence:

- Evaluate the coherence of the new topics by calculating the average pairwise cosine similarity between word embeddings within each topic:

$$\text{Coherence}(C_k) = \frac{2}{|W_{C_k}|(|W_{C_k}|-1)}\sum_{i=1}^{|W_{C_k}|}\sum_{j=i+1}^{|W_{C_k}|}\cos(e_i, e_j)$$

where W_{C_k} is the set of word embeddings in cluster C_k.

5 Pseudocode

Algorithm 1 Autotuned Topic Modeling with Agent

Require: Document set D
Ensure: Optimized topic models $M_k^{\text{optimized}}$

Step 1: Initial Topic Modeling
for all document $d_i \in D$ **do**
  for all sentence $s_i^j \in d_i$ **do**
    for all word $w_i^{jk} \in s_i^j$ **do**
      $e_i^{jk} \leftarrow \text{RoBERTa-base}(w_i^{jk})$ {Word Embedding}
    end for
    Compute context vector c_i^j
    Compute attention weights α_k
    $v_i^j \leftarrow \sum_{k=1}^{m_i^j} \alpha_k e_i^{jk}$ {Sentence Embedding}
    $v_i^j \leftarrow v_i^j / \|v_i^j\|$ {Normalize}
  end for
end for
$u_i^j \leftarrow \text{UMAP}(v_i^j)$ {Dimensionality Reduction}
$\{C_k\} \leftarrow \text{DBSCAN}(\{u_i^j\})$ {Clustering}
for all cluster C_k **do**
  Compute M_k using TF-IDF and cohesion scores
end for
Step 2: Feedback Loop with Language Model
Present initial topics to GPT-4 and receive feedback
for all topic M_k **do**
  Generate summary L_k
  Create actions A_k based on feedback
  Update weights $w_i^{jk(\text{new})}$
end for
Step 3: Refinement of Topic Modeling
Fine-tune RoBERTa on local corpora C_k
for all word w_i^{jk} **do**
  $e_i^{jk(\text{new})} \leftarrow \text{RoBERTa}_{\text{fine-tuned}}(w_i^{jk})$
end for
Update TF and IDF based on A_k
for all topic M_k **do**
  Recalculate $M_k^{(\text{tuned})}$
  Normalize $M_k^{(\text{tuned})}$
end for
Step 4: Caching Memory Module
for iteration = 1 **to** 3 **do**
  Perform parameter tuning and store results in cache
end for
Select optimal parameters based on cached results
Step 5: Final Topic Classification
Assign documents to refined topics
Evaluate topic coherence

Figure 2: Topic Modeling Cluster Evaluation Radar Chart.