FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION
HIGHER EDUCATION NATIONAL RESEARCH UNIVERSITY
«HIGHER SCHOOL OF ECONOMICS»

Faculty of Informatics, Mathematics, and Computer Science
bachelor programm «Applied Mathematics and Informatics»

Miakov Timofey Ilich

DYNAMIC TOPIC MODELING FOR VOICE DIALOGUES

Graduated Qualification Thesis

Reviewer
PhD in Physical and Mathematical Sciences, professor
V.A. Kalyagin

Scientific advisor
Associate professor NRU HSE - Nizhniy Novgorod
L.V. Savchenko

Nizhny Novgorod, 2024

Contents

Introduction

Natural Language Processing (NLP) has emerged as a crucial field in computer science, enabling machines to comprehend and respond to human language in a meaningful way. Among the various tasks within NLP, dialogue understanding, which involves analyzing and interpreting natural language conversations, has gained significant attention. One of the key challenges in dialogue understanding is identifying and tracking different topics discussed throughout the conversation.

Topic modeling is a statistical technique that helps uncover topic structures within textual data. Traditional topic modeling approaches, however, suffer from limitations such as topic drift, where the topics change over time, and the inability to capture dynamic relationships between topics. In order to analyse time-dependent text corpora, the Dynamic Topic Modeling (DTM) approach is utilised, which will be subjected to further investigation in the forthcoming work.

In this paper, we explore current approaches to (DTM) and apply them to text and audio dialogue datasets. Furthermore, we will endeavour to combine several of the most effective open-source methodologies into pipeline from text clustering to topic evolution over time. A significant aspect of our research will be to investigate the potential for integrating Large Language Models into our DTM pipeline, with a view to comparing it with other existing solutions.

The quality and performance of the pipeline will be evaluated using specific metrics at both the overall and stage-specific levels. Following this, a comparison will be made with other existing solutions and models.

0.1. Relevance of the topic

Dynamic Topic Modeling is a powerful tool for voice dialogue understanding, offering a dynamic and comprehensive representation of the conversation structure. It has the potential to significantly enhance the performance of voice dialogue systems and improve the overall user experience. In addition, exploring the possibilities of using large language models, which are now showing significant results in many NLP tasks, can further improve the model.

0.2. Research Objectives

The main objectives of the research work is to create and apply Dynamic Topic Modeling pipeline based on LLM for topic extraction for audio and text dialogues. To solve this problem, you will need to complete the following tasks

1. The identification of suitable datasets for experiments;
2. The investigation of the best suitable open-source solutions for speaker diarisation, dialogue clustering, topic extraction and topic evolution;
3. The exploration of the possibilities of using Large Language Models for some stages and the selection of models suitable for training in limited resources;
4. The selection of appropriate metrics and measurement of the quality of the approaches used at each stage;

Chapter 1. Literature review

1.1. Transformer architecture

The main architecture for working with text was Recurrent Neural Networks. However, this approach has several well-known disadvantages:

First of all, RN processes tokens and updates the hidden state sequentially, with this approach, the model begins to forget what happened before or it has to have a huge hidden state. Secondly, it is very difficult to start distributed training of recurrent networks, because the hidden state at each step must be considered separately.

In order to avoid the issues previously outlined, the article "Attention is all you need" [1] presented the Transformer architecture and its principal component, the Attention Mechanism. The objective of this methodology is to permit the elements of the sequence to be accessed at any time, thereby reducing the loss of information.

1.1.1. Self-attention mechanism

Self-attention represents a key component of the model in which tokens' information interact with each other. At each stage of the encoder, tokens exchange information with one another, gather contextual data, and refine their previous representations (Figure 2.3).

For each input token trainable projections W_Q, W_K, W_V obtained three representations interpreted by humans as a query (q_i), a key (k_i), and a value (v_i).

- q_i - query for other tokens about the information;
- k_i - keys that will be used to extract information about the token;
- v_i - values of information;

Next are the attention weights that will be applied to the values from which the important information will be extracted:

$$Attn_Weights_i = softmax(\frac{q_i \cdot k_1^T}{C}, \frac{q_i \cdot k_2^T}{C}, \dots),$$

where C is some normalization constant.

The Self-Attention result is a weighted sum of Values with coefficients in the form of attention weights. In vectorized form, you can write:

$$Attention(q, k, v) = softmax(\frac{Q \cdot K^T}{\sqrt{d_k}}) \cdot V$$

where$\sqrt{d}$ constant of the dimension of K and V.

The figure below shows the complete algorithm for the interaction of Q, K, V.

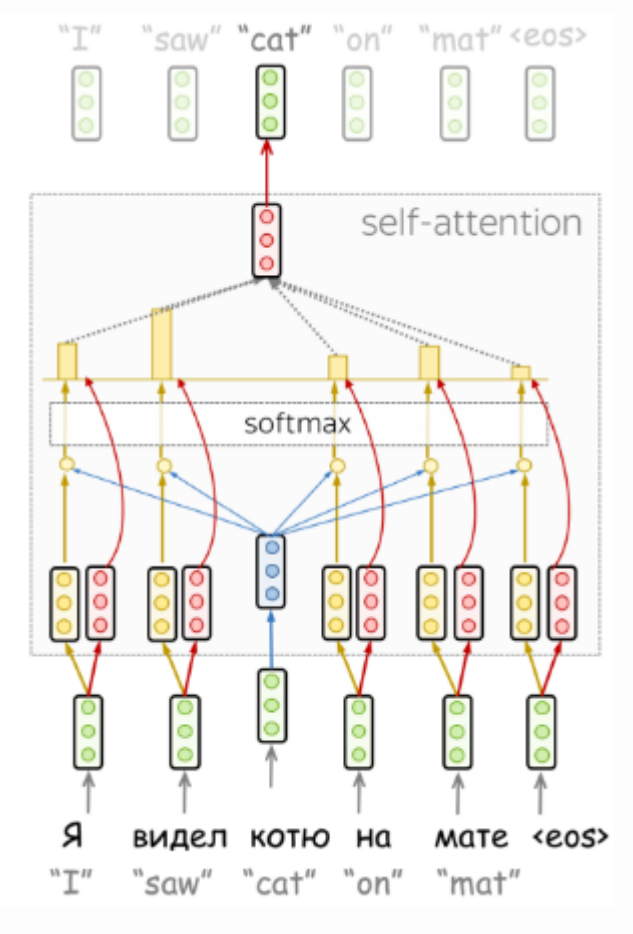

Figure 1.1. Query, Key, Value in Self-Attention mechanism

The decoder uses a modified Self-attention mechanism called Masked Self-attention. Because a new sequence is generated in the decoder, the tokens that come after the current one are masked so that the model does not see future tokens and does not rely on information about them. Masking leads to the fact that after softmax in the attention mechanism, the probability of future tokens is zero.

1.1.2. Multi-head attention

To cover more information and dependencies, the authors propose Multi-head attention, that use of multiple levels of attention in parallel and the summarisation of the values obtained. The figure below illustrates what a multi-head approach looks like:

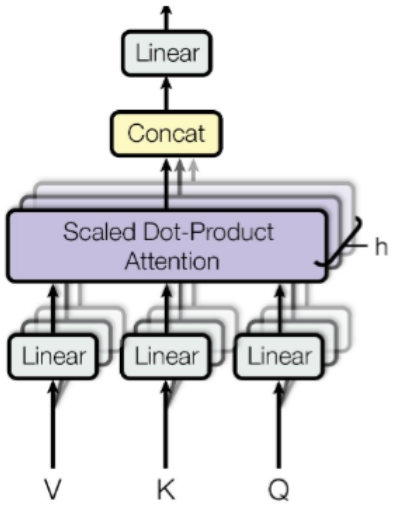

Figure 1.2. Multi-head Attention mechanism

1.1.3. Feed Forward Network

Transformer block also contains feed-forward network consisting of a linear layer and a activation function. The FFN layer can be expressed as follows:

$$FFN(x) = act(x\dot{W}_1 + b_1)\dot{W}_2 + b_2,$$

where *act* is the activation function such as $\text{ReLU} = \max(0, x)$ or more contemporary $\text{GELU} = x\Phi(x)$.

1.1.4. LayerNorm

In terms of LayerNorm (Figure 2.3), it is normalizes the vector representation of each sequence in the batch. In addition, LayerNorm has trainable parameters, *scale* and *bias*, which are used after normalization to scale the output of the layer.

The application of layer normalisation ensures that all parameters within a given layer exhibit a uniform distribution across all features for a specific input. The LayerNorm can be expressed as:

$$y = \frac{x-\mu}{\sqrt{\sigma^2+\epsilon}} \odot \gamma + \beta,$$

where μ - mean of last D dimension, σ^2 - variance of last D dimension, γ and β - learnable params;

1.1.5. Positional encodings and Rotary embedding

Model itself does not understand in which order the input tokens tokens are located. Therefore, positional embeddings are added to the tokens, which are de-

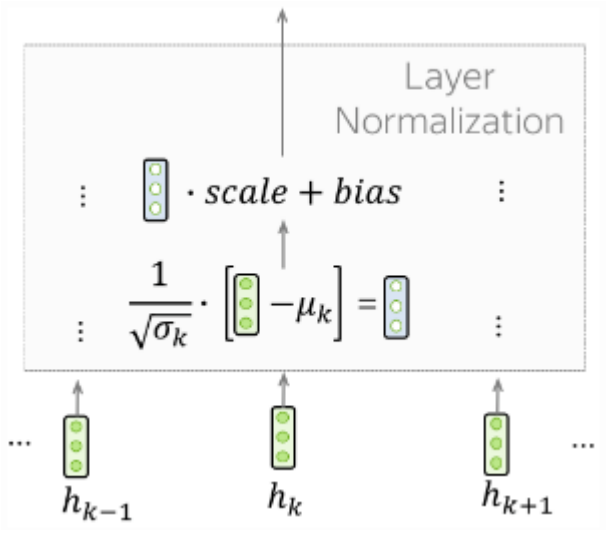

Figure 1.3. LayerNorm слой

signed to show the model the input order of the elements. Positional embeddings are given by formulas:

$$PE_{position,2i} = \sin(\frac{position}{10000^{2i/d_{model}}}),$$
$$PE_{position,2i+1} = \cos(\frac{position}{10000^{2i/d_{model}}}),$$

Each measure of position encoding corresponds to a sin wave, and the wavelengths form a geometric progression from 2π to $10{,}000 \cdot 2\pi$.

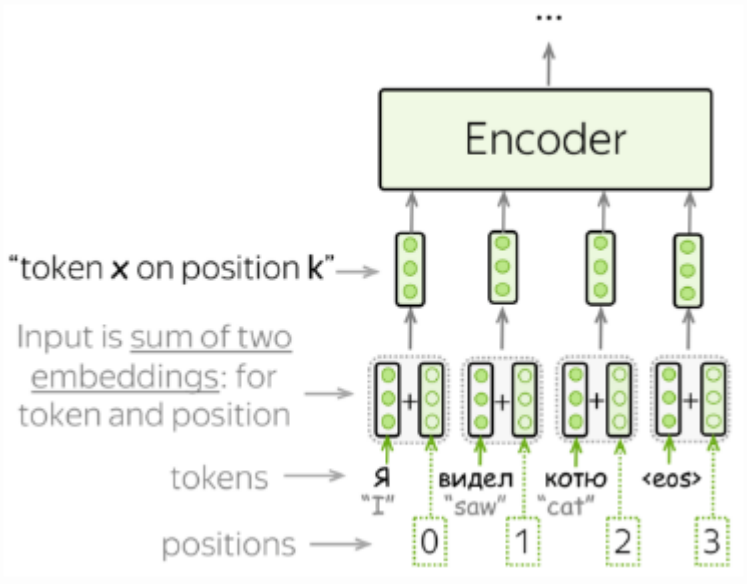

Figure 1.4. Positional encoding

A newer version of positional encoding is Rotary Positional Embeddings [2]. Rotary position embedding represents a distinct approach to incorporating relative

position information into the attention matrix. This approach differs from other approaches in that it first multiplies queries and keys with a rotation matrix. This rotation matrix is a function of absolute position. Calculating the inner products of rotated queries W_Q and keys W_K results in an attention matrix that is a function of relative position information only.

In order to illustrate the concept, we will consider an examples with two features at position m.

$$RoPE(x_1, x_2) = \begin{pmatrix} \cos m\theta & -\sin m\theta \\ \sin \theta & \cos m\theta \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \begin{pmatrix} x_1 \cos m\theta - x_2 \sin m\theta \\ x_2 \cos m\theta + x_1 \sin m\theta \end{pmatrix},$$

where θ is a constant angle.

1.2. Transformer based models

Several models based on the Transformer architecture will be required in the work at different stages. More specifically, text clustering will require an encoder for text information based on BERT, and LLM will be used for topic processing.

1.2.1. BERT

The BERT is a language model based on the transformer architecture, that was first introduced in the article BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding[3]. From an architectural point of view, BERT consists of 12 encoder blocks, including a Self-Attention mechanism, Normalization and Feed-forward layers.

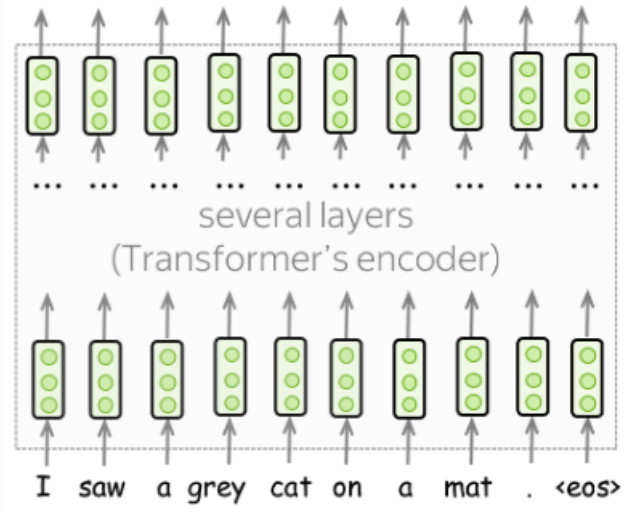

Figure 1.5. BERT architecture

BERT has two pretrain objective described in article:
The first objective of the training process is the masked language modelling (MLM). During the training phase of MLM, the following occurs:

- A number of tokens are selected with a probability of 15% for each token.
- The selected tokens are replaced by [MASK] with a probability of 80%, a random token with a probability of 15% and remain unchanged with a probability of 10%.
- The model must predict the original token.

Secondly, Next Sentence Prediction (NSP) pretraining task is a binary classification task. In order to understand relationship between two sentences, BERT training process also uses next sentence prediction. During training the model gets as input pairs of sentences and it learns to predict if the second sentence is the next sentence in the original text as well. The training dataset presented in the original paper states that when trained, 50% of the examples contain related sentences extracted from the training texts, while the remaining 50% contain a random pair of sentences.

LLM is a type of artificial intelligence(AI) algorithm that use deep learning techniques and massively large data sets to understand, summarize, generate, and predict new content. LLMs are trained with immense amounts of data and use self-supervised learning to predict the next token in a sentence, given the surrounding context. Once an LLM has been trained, it can be fine-tuned for a wide range of NLP tasks, including generating and classifying text, answering questions. The following section will examine the main LLMs used in the experiments: LLaMA, Mistral, Qwen.

1.2.2. LLaMA

In July 2023, an open-source collection of models, called LLaMA2 [4], was released by Mata AI Team. The collection consists of models with 7B and 70B parameters.

In order to create a collection of LLaMA 2 models, the authors employed the architecture of an optimised auto-regressive transformer (Touvron et al.(2023)) with

RMSNorm normalisation, Rotary Embeddings and SwiGLU activation function in FFN, but implemented a number of alterations with the intention of enhancing the quality of the model and its performance, such as:

- Meticulous data preparation
- Larger dataset
- Increased the length of the context in compare with earlier approaches
- Incorporation of grouped-query attention (GQA)

Grouped-query attention can be thought of as a way to optimize the attention mechanism in transformer-based models. Instead of computing attention for each query independently, GQA groups queries together and computes their attention jointly. This reduces the number of attention computations, leading to faster inference times.

The GQA method has been applied to speed up inference on large language models without significantly sacrificing quality. It's a promising technique for improving the efficiency of transformer models, particularly in the context of generative AI.

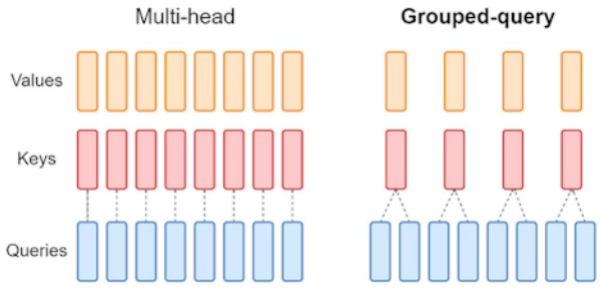

Figure 1.6. Self-attention and Grouped-query attention

In april 2024 was released LLaMA3 8B and 70B models with some new features: larger tokenizer vocabulary 128k agains 32k, the dataset for pre-training contains 8 times more sampels on over 15 trillion tokens on a new mix of publicly available online data.

1.2.3. Mistral

In September 2023, the MistralAI [5] team introduced the Mistral 7B model, which has been designed for high performance and efficiency. In comparison to the 13B Llama 2, the Mistral 7B model has demonstrated superior performance in all benchmarks related to reasoning, mathematics, and code generation. The Mistral 7B model employs grouped-query attention (GQA) for rapid inference and sliding window attention (SWA) for handling long sequences at a reduced cost.

As other models, Mistral is a transformer based model. However, it differs from the Llama model in several key respects.

- Sliding Window Attention (SWA): This method allows the model to attend to information beyond a set window size by leveraging the stacked layers of a transformer. With a window size of 4096, it can attend to around 131K tokens.

- Rolling Buffer Cache: It is limits the size of the cache using a rolling buffer. The cache has a fixed size, and as new data is added, older data is overwritten once the cache exceeds its size. This method reduces the memory usage of the cache by 8x for sequences of 32k tokens without compromising the quality of the model.

- In the context of sequence generation, the model employs a process of pre-filling and chunking. This involves the prediction of tokens in a sequential manner, given that each token is contingent upon the previous one. However, given that the prompts are known in advance, the cache can be pre-filled with them. In the event that a prompt is of a considerable length, it can be divided into smaller units, with the cache being filled in this manner. The attention mask operates over both the cache and the chunk

1.2.4. Qwen

The Qwen LLMs have been extensively trained on up to 3 trillion tokens of diverse text and code examples from a wide range of domains. Qwen demonstrate superior performance in various downstream tasks.

In terms of architecture: model based on LLaMA, activation function is SwiGLU, Rotary Embeddings, Pre-Norm and RMSNorm instead LayerNorm.

The pre-training phase involves learning vast amounts of data to gain a comprehensive understanding of the world and its various complexities. This includes not only basic language skills, but also advanced skills such as arithmetic, coding and reasoning. In this section we introduce the data, model design and scaling, and the extensive evaluation results on benchmark datasets.

The Qwen Series encompasses models that employ SFT and RLHF to align Qwen with human preference. Consequently, the Qwen-Chat and its enhanced variant, the Qwen-Chat-RLHF, have been developed. Furthermore, we have created bespoke models for coding and mathematics.

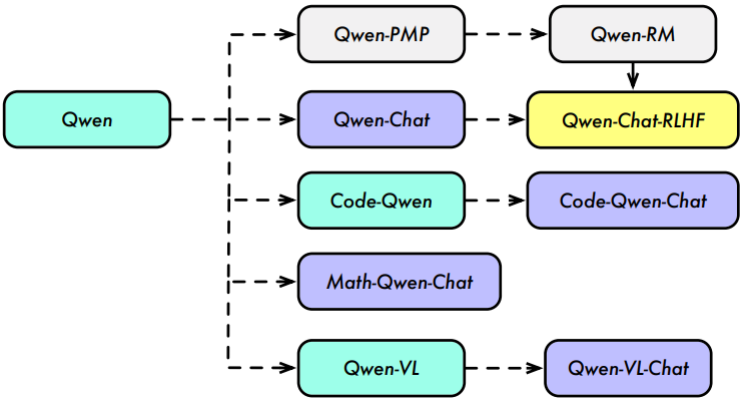

Figure 1.7. Qwen Series

For now, there are 5 models of different sizes, 4 of which are opensourced. Specially, Qwen-1.8B, Qwen-7B, Qwen-14B, and Qwen-72B.

1.3. Speaker diarization

Since one of the significant tasks is to use DTM for audio dialogues, the first, but optional, stage of our pipeline will be a model for speaker diarization.

Speaker diarisation is the process of automatically segmenting and identifying different speakers in an audio recording. The goal of speaker diarisation is to divide

the audio stream into non-homogeneous segments, each segment corresponding to a specific speaker or speaker turn.

The fundamental process of building a speaker diarization system involves the following steps:

- Segmentation: The audio recording is divided into differente segments based on conditions like silence, pauses, and changes in speaker turns.
- Speaker Embedding: Embeddings are extracted for each segment to represent the speech features of unique speaker.
- Clustering: The extracted embeddings are clustered based on similarity, with segments from the same speaker forming a single cluster.
- Labeling: Once the clustering is complete, each segment is assigned a label representing the speaker's identity. Segments in the same cluster are assigned the same label, indicating they belong to the same speaker.

As ASR model for this stage, we will look at the Whisper [6] model from OpenAI.

1.3.1. Whisper

Whisper is an automatic speech recognition (ASR) system trained on 680,000 hours of multilingual and multitask supervised data collected from the web. We show that the use of such a large and diverse dataset leads to improved robustness to accents, background noise and technical language. Moreover, it enables transcription in multiple languages, as well as translation from those languages into English. We are open-sourcing models and inference code to serve as a foundation for building useful applications and for further research on robust speech processing.

The Whisper architecture is a simple end-to-end approach, implemented as an encoder-decoder Transformer. Input audio is split into 30-second chunks, converted into a log-Mel spectrogram, and then passed into an encoder. A decoder is trained to predict the corresponding text caption, intermixed with special tokens that direct the single model to perform tasks such as language identification, phrase-level timestamps, multilingual speech transcription, and to-English speech translation.

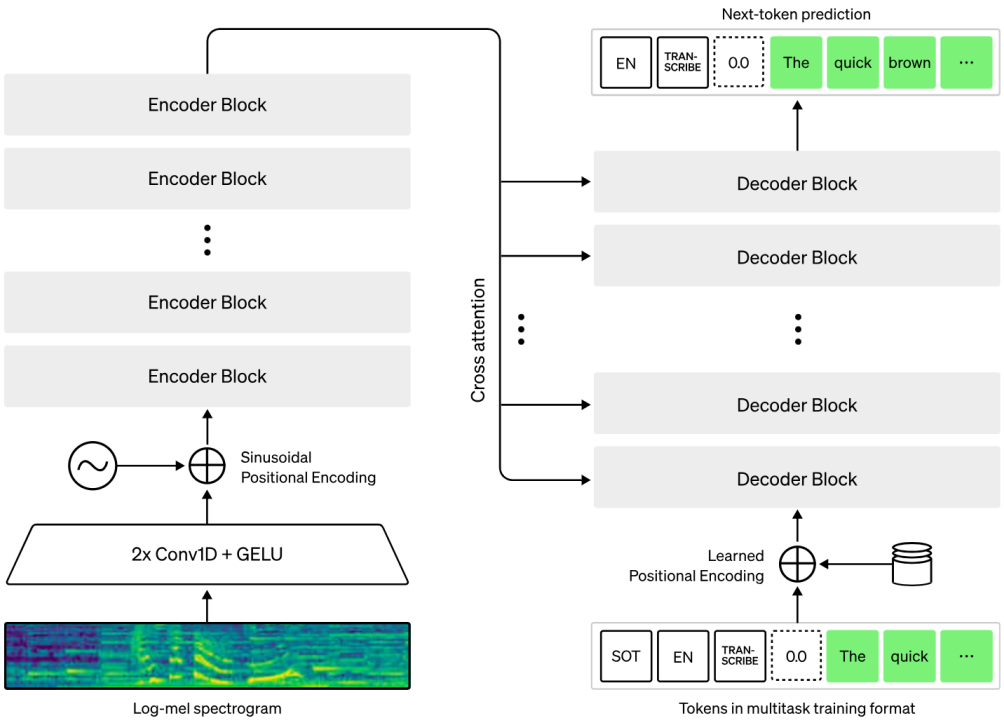

Figure 1.8. Whisper architecture

1.4. Improving Utterance-Pair Coherence Scoring

The dialogue segmentation stage represents the initial stage for textual type of data. The result is a set of utterance indices that indicate the beginning of a thematically related group.

In accordance with the majority of previous studies, we have utilise the best open-source solution based on the presented in papers metrics. It is algorithm from Improving Unsupervised Dialogue Topic Segmentation with Utterance-Pair Coherence Scoring [7] Let us now examine the algorithm in greater detail.

1.4.1. BERT for coherence scoring

In terms of our approach to topic segmentation, authors utilise Next Sentence Prediction (NSP) BERT as the encoder, due to the similar type of task. Input of both tasks an utterances, with the objective of predicting the appropriate next sentence, which should be topically related.

In more detail, the positive instances (s_i, s_{t+i}) and negative instances (s_i, s_{t-i}) use in the model's fine-tuning stage in the form described below:

$$[CLS]s_i[SEP]s_{t+/-i}[SEP],$$

where the symbols $[CLS]$ and $[SEP]$ represent special tokens in BERT.

In order to indicate the thematic similarity of two statements in a dialogue, the $[CLS]$ token is employed, which performs a similar role in the usual BERT. The resulting value is subjected to a linear dictionary and a final similarity value is obtained, which is then utilised to calculate Coherence scores.

1.4.2. Depth Scores

With the help of a fine-tuned BERT (the training algorithm is described above) coherence scores c_i, where $i \in [0, n-1]$ are calculate. According to texttiling algorithm description c_i indicates the topical relatedness of the two adjacent utterances. The model uses the obtained values as base values, but processes them for final predictions and gets Depth scores ds_i, where $i \in [0, n-1]$ - measuring the sharpness of a valley.

The depth score is calculated as shows below:

$$D_{pi} = \frac{h_l(i)+h_r(i)-2 \cdot c_i}{2},$$

where h_l, h_r - the highest cs for i pair in choosen range.

The greater the depth score between two statements, the less likely it is that they share a thematic relationship. To identify the pairs along which the boundaries of the thematic segments lie, the threshold value is calculated as follows: $threshold = \mu - \frac{\sigma}{2}$, where μ is the average value and σ is the standard deviation. Pairs with a depth score above the $threshold$ value will be choosen as boundaries.

1.5. Evaluation Metrics

1.5.1. Speaker diarization

The standard metric for speaker diarization problem is the Diarization error rate This metric computes the error in diarization in terms of a duration ratio, which means that a perfectly aligned hypothesis and reference diarization would result in DER=0. It sums up 3 error:

- False Alarm - speech segment predicted where there is no speaker. It is error from ASR models.
- Missed Detection - no speech detected where there is a speaker. It is error from ASR models.
- Confusion Speech - the wrong cluster. It is error from the clustering model

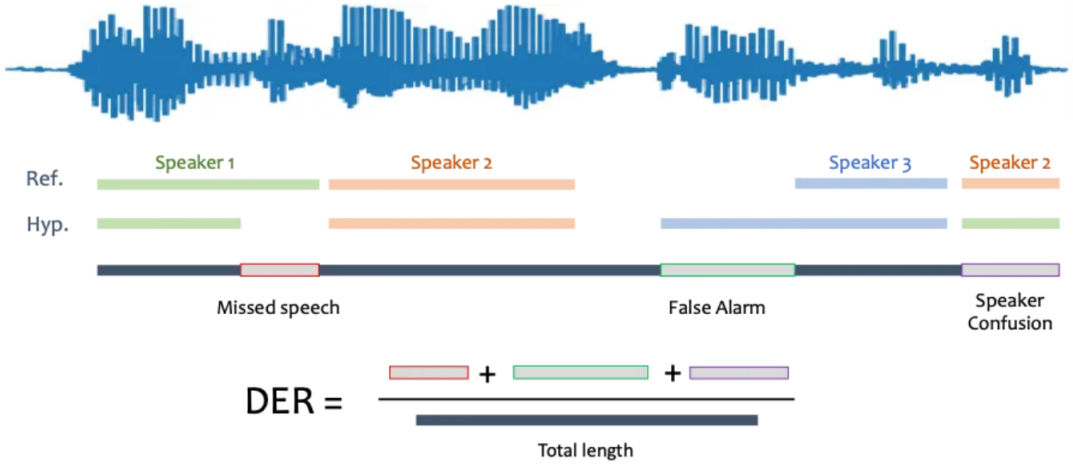

Figure 1.9. Diarization error rate

Mathematically, the metric can be written:

$$DER = \frac{FalseAlarm + MissedDetection + Confusion}{Total}$$

1.5.2. Dialogue segmentation

After defining the basic segmentation algorithm, it is worth talking about evaluation metrics for this stage:

- F1
  Given that this is a binary classification problem, it is tempting to use Precision and Recall, and their combination F_β score as an evaluation metric.
  - Metric is defined as the boundaries ratio identified as true boundaries;
  $Precision = \frac{true\ positive}{true\ positive + false\ positive}$
  - Recall is defined as the true boundaries ratio identified by the model.
  $Recall = \frac{true\ positive}{true\ positive + false\ negative}$
  F_β score is defined as the weighted harmonic mean of precision and recall:
  $F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall}$

- Pk score [8] is uses the sliding window method and counts the number of discrepancies between the ends of the predicted segments and the true segments. The final score is the counter divided by the number of measurements. The lower the value, the better.

- In WindowDiff [9] this algorithm, for each position of a sliding window of size k, comparing the number of boundaries in the ground truth with the number

of boundaries predicted by the Topic Segmentation model. The lower the value, the better.
$WD(pred, true) = \frac{1}{N-k} \sum_{i=1}^{N-k}(|bounds(true_i, true_{i+k}) - bounds(pred_i, pred_{i+k})| > 0)$,
where $bounds(s, s+i)$ is the number of boundaries between i and j positions,
N - number of sentences in the text.

1.5.3. Topic extraction

As for the quality evaluation of the extracted topics, we will use two metrics: Topic coherence and Topic diversity

- Topic diversity [10] is a metric that assesses the ratio of unique keywords across all topic representations. This metric is used to measure the capabilities of a topic model covering a wide range of topics. The diversity score is a value between 0 and 1, with a score of 0 indicating repetitive topics and a score of 1 indicating diverse topics.

- Topic coherence [11] is a measure of the degree of interconnection of words within a topic. In our experiments, we employed normalized point-to-point mutual information (PMI) as a metric for thematic consistency. A higher score for a given name indicates greater consistency, with an absolute correlation estimated at 1 point.

Chapter 2. Methodology

2.1. Pipeline configuration

The pipeline of dynamic thematic modelling of audio dialogues, which will be considered in this paper, is divided into several main stages. These include ASR, dialogue segmentation, topic extraction and topic evolution.

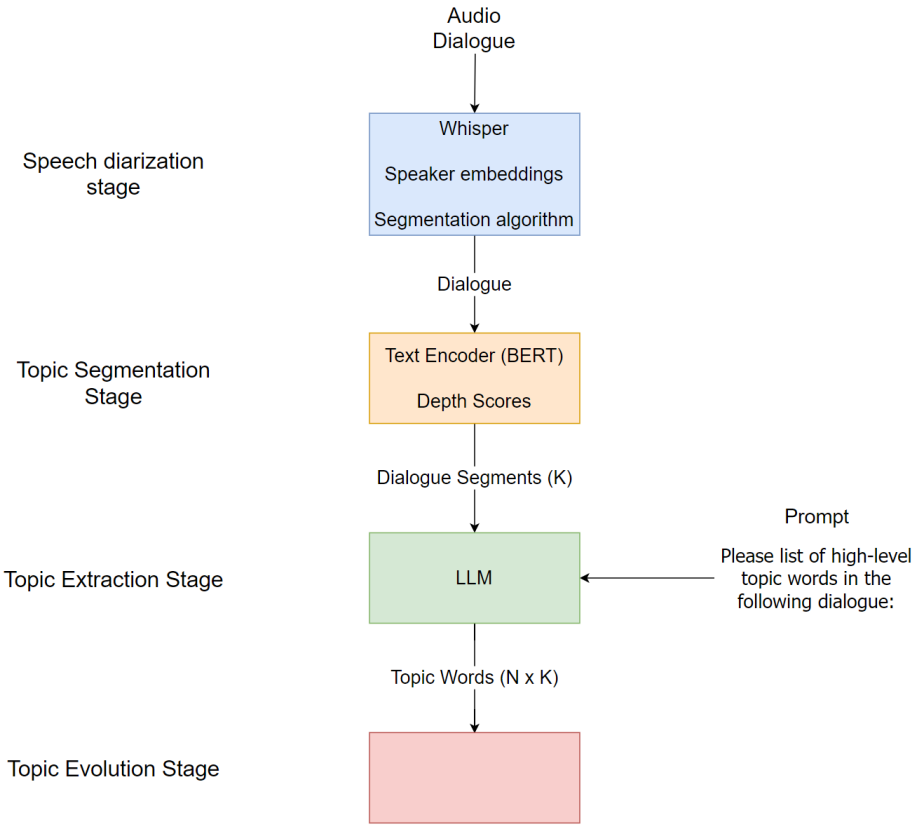

Figure 2.1. DTM pipeline

All the models in the pipeline were taken from the HuggingFace hub. Tools from transformers, which is also part of HuggingFace, will be used to create the pipeline.

2.2. Speaker Diarization

The first optional stage is the Speaker Diarization. For audio dialogues, text is extracted from the audio track before topic segmentation. For this stage we use:

- OpenAI Whisper
  Link to model: whisper-large-v3;

- pyannote - speaker diarization library [12]
  Link to model: [pyannote];

2.3. Dialogue Segmentation

The algorithm 1.4 will be used to dialogues segmentation. The implementation of the algorithm will be taken from the official github repository. Part of the code with the model inference will be adapted to the pipeline.

2.4. Topic Extraction

The next stage in the pipeline is topic extraction. At this stage, n topics will be extracted from each segment in the dialogues. At this stage, past approaches have used probabilistic or embedding approaches, like BertTopic [13]. In our work, we will apply a generative approach using Large Language models, as in PromptTopic [14].

We will try several Large Language Models:

- MetaAI LLaMA3 8b
  Link to model: meta-llama3-8b

- Qwen1.5 7b
  Link to model: Qwen1.5 7B

- Mistral 7b
  Link to model: Mistral 7B

  The prompt for LLM will look like this:

```
<|user|>
<dialogue>
Give me N topic words
from the following dialogue:
<|assistant|>
```

where N is number of topic words, the appearance of the keywords <|user|>, <|assistant|> dependes on type of LLM.

Moreover, in experiments, models are tested not only in zero-hour mode, but also in few-shot. A few-short is a technique in which there are examples of a query with an answer before the main question. In this case the prompt for LLM will look like this:

```
<|user|>
<example_dialogue1>
Give me N topic words
from the following dialogue:
<example_answer1>

<example_dialogue2>
Give me N topic words
from the following dialogue:
<example_answer2>

<dialogue>
Give me N topic words
from the following dialogue:
<|assistant|>
```

2.5. Topic Evolution

In the pipeline, topic evolution can be used in two formats. The first format allows you to analyze the change of topics in a single dialog. In this format, graphs of changes in the set of topics in the order of the segments will be plotted.

The second format analyzes the entire dataset. In this case, either a special mark will be used for timestamp analysis along with each dialog, or the analysis will be performed in the order in which the samples are located in the dataset.

With a large number of topic words, the N most common ones will be selected to visualize their changes.

Chapter 3. Experiments

The experiments will be divided into stages, with each stage accompanied by a description of the stage and the algorithms used. This description is included in the theoretical part of the work.

3.1. Experiments settings

Datasets For the experiments we will use two types of datasets: text dialogues and audio dialogues. For text dialogues, we will use tiage, compiled from the personal chat corpus and manually annotated, and superdialseg, compiled from several smaller datasets: doc2dial and multidoc2dial. As for the audio dialogues, we will use QMSum, which includes the audio recording corpuses - AMI is product meetings and ICSI is academic meetings.

dataset	train/val/test	Avg Turns	Avg tokens
SuperDialseg	6984/1322/1322	13	239
Tiage	7939/-/1000	14.7	-
Product Meetings (AMI)	137	535.6	-
Academic Meetings (ICSI)	59	819.0	-

Table 3.1. Datasets

At the text clustering stage, comparisons are made between models of non-alternative baselines, because at this stage we are using a ready-made algorithm and making a minimum of changes. As for topic extraction, our LLM approach will be compared with BertTopic and LSA.

3.2. Speaker diarization

model	AMI	ICSI
pyannote-speaker-diarization-2.1	18.9	16.4
pyannote-speaker-diarization-3.1	18.8	16.7
whisper + pretrain embeds + agglomerative cluter	14.7	13.9

Table 3.2. Speaker diarization evaluation

The finished models from pyannote showed better results than the assembled pipeline from whisper and pretraining embeddings. Therefore, it was decided to include pyannote models in the final pipeline.

3.3. Dialogue Segmenentation

The input data at this stage are dialogues divided into separate utterances of the participants $ut_1, ut_2, \ldots, ut_n$. At this stage, experiments will be conducted using the encoder model for the Utterance-Coherence Scoring algorithm.

models	Tiage			SuperDialseg		
	Pk ↓	wd ↓	f1	Pk ↓	wd ↓	f1
GreedySeg	0.492	0.508	0.181	0.497	0.501	0.35
GraphSeg	0.462	0.466	0.239	0.496	0.515	0.23
TextTilling	0.469	0.488	0.204	0.446	0.458	0.3
TeT + BERT	0.425	0.439	0.285	0.522	0.531	0.20
CohScoring+BERT-NSP	0.4465	0.4575	0.267	0.5071	0.5219	0.20
CohScoring+BERT-NSP fine-tune	0.3981	0.4018	0.32	0.4361	0.44510	0.28
models	AMI			ICSI		
	Pk ↓	wd ↓	f1	Pk ↓	wd ↓	f1
GreedySeg	0.885	0.792	0.3641	0.762	0.730	0.30
GraphSeg	0.875	0.750	0.422	0.810	0.756	0.27
TextTilling	Pk	wd	f1	Pk	wd	f1
TeT + BERT	Pk	wd	0.443	Pk	wd	0.15
CohScoring+BERT-NSP	0.6447	0.7394	0.551	0.6148	0.6731	0.7
CohScoring+BERT-NSP fine-tune	0.5854	0.6032	0.587	0.5842	0.5932	0.8

Table 3.3. Dialogue segmentation experiments

The selected solution shows the best metrics on the datasets we selected. It is important to understand that we have researched open source solutions.

3.4. Topic extraction

3.4.1. Zero-shot prompting

models	Tiage		SuperDialseg	
	TD	TC	TD	TC
LSA	0.25824	0.00951	0.420026	-0.01929
BertTopic	0.34823	0.02751	0.420026	0.01929
LLaMA3 8B	0.80707	0.01621	0.49525	0.005887
Qwen1.5 7B	0.72169	0.015453	0.47777	-0.07682
Mistral 7B	0.63344	0.002248	0.34215	-0.01689
models	AMI		ICSI	
	TD	TC	TD	TC
LSA	0.009648	0.001375	0.013490	-0.00172
BertTopic	0.013996	0.02485	0.029090	0.27210
LLaMA3 8B	0.50863	0.11409	0.76137	0.14734
Qwen1.5	0.38641	0.10328	0.7185	0.00237
Mistral 7B	0.29852	0.084103	0.5761	-0.11318

Table 3.4. Dialogue topics extraction experiments

The result of experiments in zero-shot that LlaMA 3 8B demonstrate the best performance, it is far ahead of everyone in Topic Diversity and competes in Topic Coherence with BertTopic. A strong drawdown in quality relative to extractive model was also noticed in the article PromptTopic [14], where the models also lost to BertTopic in Topic Coherence.

The main problem at this stage is the problems with the format of the topic words themselves and the formats of the generated output of the model. Firstly, the model sometimes gives out not only the word itself but also an explanation of why, there are also options when the model outputs whole sentences or phrases of 3 or more words, which does not suit our task.
For example:

```
good answer for dialogue:
car, summer, camp.

possible bad variants:
---------------------
car (because they travel by car)
summer (because all dialogue about summer)
camp (because the spent a lot of time in camp)
---------------------
we take a car
spent the whole summer together
camp was an important place
---------------------
```

Moreover, the zero-shot format has problems with the format of the output generated sequence. The model with the same prompt sometimes produces a different format and parsing topic words becomes much more difficult.

```
possible bad variants:
---------------------
1. car
2. summer
3. camp
---------------------
* car
* summer
* camp
---------------------
```

Consequently, it was important to use different techniques "to embrace" model the correct formats for convenient operation in the whole pipeline.

3.4.2. Few-shot prompting

The few-short approach was explained in the chapter of Methodology.

	AMI		ICSI	
model	TD	TC	TD	TC
LLaMA3 zero-shot	0.50863	0.11409	0.76137	0.14734
LLaMA3 few-shot	0.51291	0.12129	0.78201	0.13985
Qwen1.5 zero-shot	0.38641	0.10328	0.7185	0.00237
Qwen1.5 few-shot	0.38875	0.10190	0.7196	0.00239
Mistral zero-shot	0.29852	0.084103	0.5761	-0.11318
Mistral few-shot	0.31920	0.065381	0.5761	-0.008673

Table 3.5. Comparisons of models in zero-shot and few-shot

Experiments with few-short prompts did not show a significant increase in metrics, but it became much easier to get topic words from the model output because the model understood the format in which we wanted to get a list of keywords. As a result, it doesn not mean much on its own, but as part of a pipeline where everything is working together, it's a significant improvement.

3.4.3. Fine-tuning

It was also decided to carry out fine-tuning of the models on the selected train part of the datasets (tiage, superseg). The main point of this stage is to train the model for a given topic words format and their output format, as in few-shot. In addition, it is possible to improve the quality metrics of the models.

	AMI		ICSI	
model	TD	TC	TD	TC
LLaMA3 few-shot	0.51291	0.12129	0.78201	0.13985
LLaMA3 fine-tuning	0.53812	0.13329	0.79854	0.15636
Qwen1.5 few-shot	0.38875	0.10190	0.71960	0.00239
Qwen1.5 fine-tuning	0.39471	0.11004	0.72041	0.00241
Mistral few-shot	0.31920	0.065381	0.57617	-0.008673
Mistral fine-tuning	0.31998	0.065421	0.57531	0.00031

Table 3.6. Comparisons of models in zero-shot and after fine-tuning

As with few-shot, the metrics have increased, but now it is convenient to work with the model in the whole pipeline, because now it is easy to understand the "behavior" of the model during response generation. In addition, the increase in the

metric comes from the fact that it was possible to parse all samples, unlike some in zero-shot and few-shot.

3.5. Topic Evolution

In this section, we will show examples of the results of analyzing changes in topics over time. All examples will be presented on the AMI dataset.

Topic evolution over time on one dialogue:

```
dialogue_topics_analyze(dialogue,
                        n_topic_words)
```

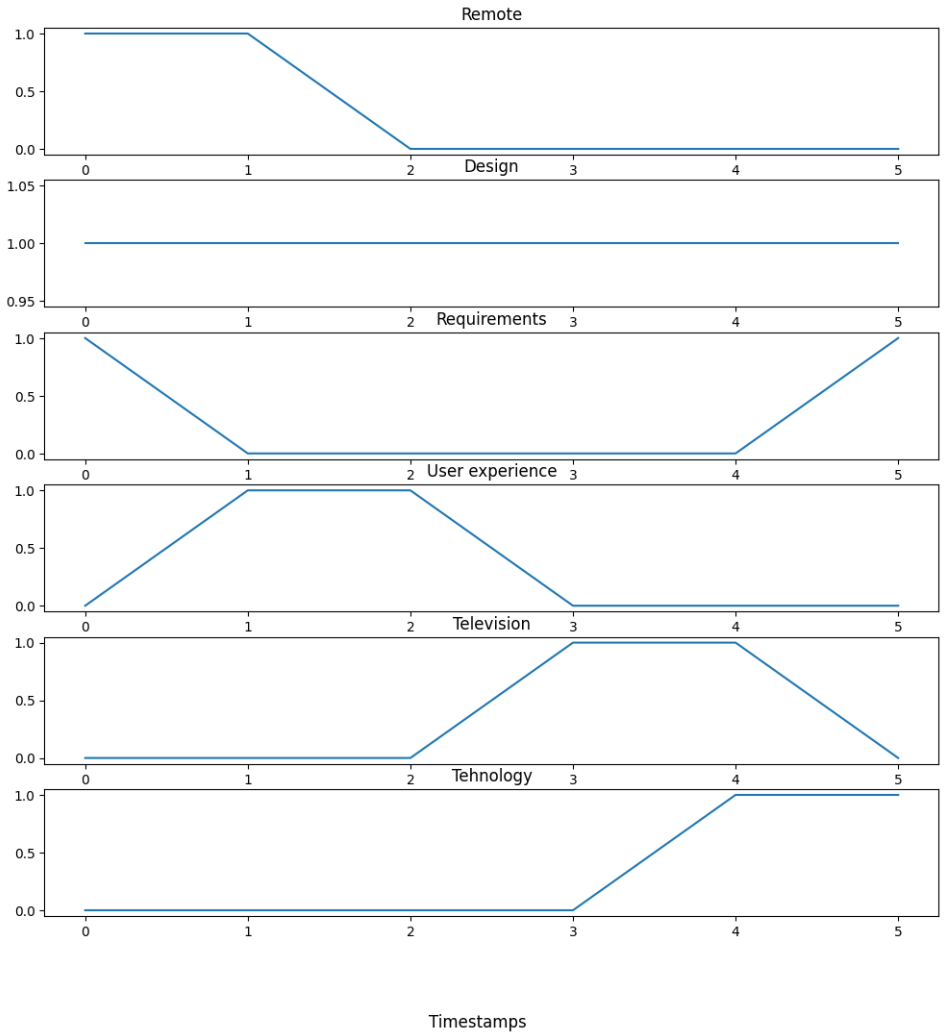

Figure 3.1. Topic evolution over time on one dialogue

The picture shows the change of themes over time. There were 5 segments in this dialog, so 5 timestamps.

Topic evolution over time on dataset:

```
corpus_topics_analyze(dataset, n_topic_words,
                      top_popular=8, time_range=50)
```

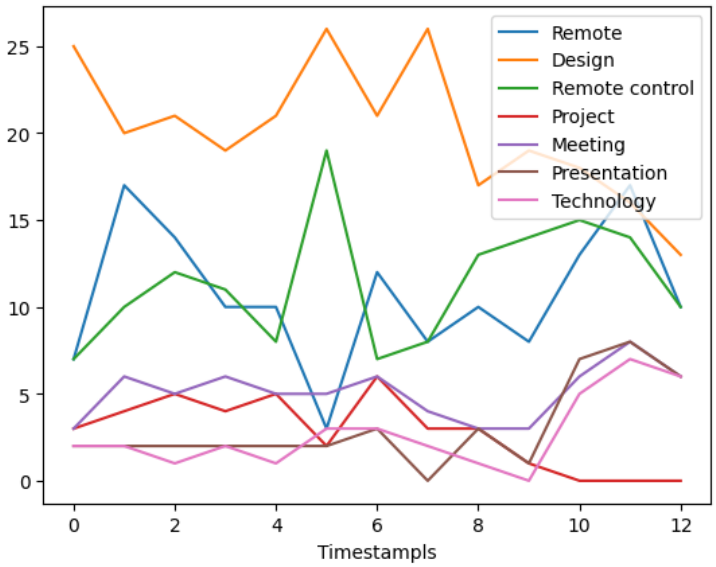

Figure 3.2. Topic evolution in AMI dataset

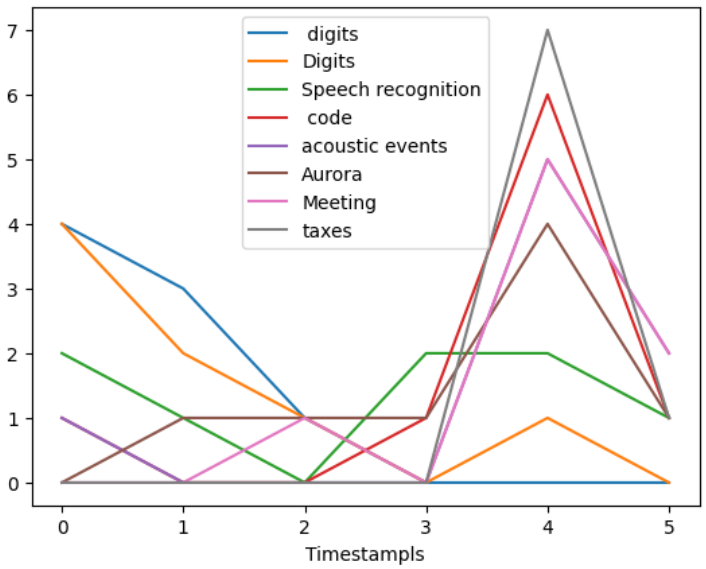

Figure 3.3. Topic evolution in ICSI dataset

In analyzing the evolution of topics on the dataset, top_popular is also selected, because there are many more topics on one dialog and it is impossible to analyze everything at once. In addition, the time_range parameter is responsible for reducing time intervals for more convenient analysis.

3.6. Limitations

The LLM and other models require a significant amount of computing resources, including GPUs. Due to the limited availability of resources, the models were not larger than 8 billion parameters and did not exceed 32 GB during training and inference at all stages. Additionally, because of the absence of financial opportunities, open-source models were employed in the work. Should it be possible to reproduce all stages on more advanced models (such as GPT4 or LlaMA3 80b), the experimental results would be considerably more favourable.

Conclusion

Summarizing all the results of the article:

- Suitable datasets were selected, experiments were carried out on them at all stages and quality metrics of the selected algorithms were obtained;
- Among the available solutions, the best available metrics were selected. Experiments were carried out with each of the algorithms at each stage on ready-made models, pre-trained and trained from scratch. The best models were added to the final pipeline;;
- At the topic extraction stage, a study was conducted on the use of LLM. The best open-source LLMs were selected for experiments with zero-shot, feat-shot and fine-tuning. The generative approach using LLM shows the results of the best available extractive solutions for topic modeling;
- A suitable set of metrics was identified and selected to evaluate the quality of each stage. In order to utilise the metrics, a library with an implementation was Investigated and integrated into pipeline.

References

Vaswani, A. Attention Is All You Need / A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin // arXiv:1607.06450. — 2017.

Jianlin, S. RoFormer: Enhanced Transformer with Rotary Position Embedding / S. Jianlin, L. Yu, P. Shengfeng, W. Bo, L. Yunfeng // CoRR. — 2021.

Devlin, J. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / J. Devlin, M.-W. Chang, K. Lee, K. Toutanova // arXiv:1810.04805. — 2018.

Touvron, H. Llama 2: Open Foundation and Fine-Tuned Chat Models / H. Touvron, L. Martin, S. Kevin // arXiv:2307.09288. — 2023.

Jiang, A. Mistral 7B / A. Jiang, A. Sablayrolles, A. Mensch, C. Bamford // arXiv:2310.06825. — 2023.

Radford, A. Robust Speech Recognition via Large-Scale Weak Supervision / A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, I. Sutskever // arXiv:2212.04356. — 2022.

Linzi, X. Improving Unsupervised Dialogue Topic Segmentation with Utterance-Pair Coherence Scoring / X. Linzi, G. Carenini // arXiv:2106.06719. — 2021.

Doug, B. Statistical Models for Text Segmentation / B. Doug, B. Adam, L. John // Machine Learning 34. — 1999.

Pevzner, L. A Critique and Improvement of an Evaluation Metric for Text Segmentation / L. Pevzner, M. A. Hearst // Computational Linguistics. — 2002.

Dieng, A. B. Topic Modeling in Embedding Spaces / A. B. Dieng, F. J. R. Ruiz, D. M. Blei // arXiv:1907.04907. — 2019.

Bouma, G. Normalized (Pointwise) Mutual Information in Collocation Extraction / G. Bouma // Proceedings of GSCL. — 2009.

Bredin, H. pyannote.audio 2.1 speaker diarization pipeline: principle, benchmark, and recipe / H. Bredin // Proc. INTERSPEECH 2023. — 2023.

Grootendorst, M. BERTopic: Neural topic modeling with a class-based TF-IDF procedure / M. Grootendorst // arXiv:2203.05794. — 2022.

Wang, H. Prompting Large Language Models for Topic Modeling / H. Wang, N. Prakash, N. Hoang, M. Hee // arXiv:2312.09693. — 2023.