Extracting Semantic Relationships from Unstructured Textual Data in eLearning Video Scripts

Alejandro Bravo de la Serna
TU Munich
alejandro.bravo@tum.de

George Elfayoumi
TU Munich
george.elfayoumi@mytum.de

Parag Bamel
TU Munich
parag.bamel@tum.de

Jinyu Lee
TU Munich
jinyu.lee@tum.de

Mohamed Hesham Ibrahim Abdalla
TU Munich
mohamed.abdalla@tum.de

Abstract

FAST AI Movies offers an AI solution for automatically generating eLearning videos based on provided eLearning scripts. Effective representation of visual information in these videos relies on the extraction of meaningful semantic relations from complex textual data. In this project, we explore both extractive and generative approaches to address three main tasks. Firstly, we tackle chapter segmentation, dividing lengthy scripts into coherent chapters, by deploying clustering, sentence-level architectures, encoder and decoder based architectures. Second, we extract keyphrases within each chapter by fine-tuning Llama2 and BERT to identify keyphrase positions. Finally, we retrieve semantic relations between extracted keyphrases through classification of relations among keyphrase embeddings and prompting or fine-tuning with Large Language Models (LLMs). The experimental results demonstrate that BERT excels in chapter segmentation, while Llama2 performs relatively on par, the use of Llama2 and BERT proves beneficial in keyphrase extraction, and our fine-tuned Mixtral outperforms the few-shot prompting with GPT-4 in semantic relation classification.

1 Introduction

This project delves into the challenging task of extracting semantic relationships from eLearning video scripts. As technology permeates into the field of education, eLearning platforms have emerged as powerful tools for learning and instruction. A critical component of these platforms are their video scripts, which contain valuable information articulated through detailed lessons. However, these scripts are often presented as unstructured data, making it challenging to extract meaningful information. Our research primarily focuses on three aspects: Chapter Segmentation (CS), Keyphrase Extraction (KE), and Semantic Relation Classification (SRC).

CS involves the process of splitting unstructured input text that consists of short sentences and bullet points from customers into smaller chapters such that each chapter typically focuses on a single topic. CS is essential for facilitating subsequent tasks such as keyphrase extraction and semantic relation classification. CS is particularly needed to overcome context length limitations in LLMs as it is easier to deal with small coherent chapters when extracting semantic relations.

KE, in the context of our project, refers to the process of extracting or creating summary bullet-points from sections of the text. These keyphrases serve as an input to the next stage of the project, SRC.

SRC involves identifying semantic relationships between different fragments of the text represented as keyphrases. It provides the basis for understanding and representing the underlying message or knowledge encoded in the script.

Overall, the main goal of this project is to break down lengthy video script descriptions into smaller components called keyphrases. Subsequently, we aim to establish relationships between these keyphrases through SRC, allowing for their representation as icons and diagrams [1]. These visual elements are then integrated into slides to create an eLearning video. Ultimately, our aim is to automate the process of generating eLearning videos from text.

2 Related Work

2.1 Chapter Segmentation

CS consist of segmenting lengthy eLearning scripts into coherent and manageable chapters, facilitating subsequent tasks, such as KE and SRC, to be explored with LLMs with limited context length. This task can be regarded as text segmentation, in which

[1]Icons and diagram creation is out of the scope of the project.

a document is divided into coherent paragraphs. There are generally two approaches to this task: unsupervised learning and supervised learning. In the realm of unsupervised learning, the GRAPHSEG model (Glavaš et al., 2016) proposes the use of semantic relatedness graphs and the model in Solbiati et al. (2021) focuses on extracting semantic similarity from sentence embeddings, such as Sentence-BERT (Reimers and Gurevych, 2019). These approaches can reduce the need for labeling large datasets since unsupervised learning does not require labeled input. On the other hand, supervised learning-based methods primarily utilize sentence embeddings and their corresponding labels as input, indicating whether to split paragraphs or not. For supervised learning approaches, it is crucial to deploy models capable of understanding contextual meaning from input embeddings, such as bi-directional LSTM (Koshorek et al., 2018) or Transformer (Somasundaran et al., 2020). Additionally, text segmentation can be implemented through fine-tuning LLMs like BERT (Pethe et al., 2020). We fine-tune LLMs such as Llama2 (Touvron et al., 2023) and BERT (Devlin et al., 2019) on token embeddings, leveraging the benefits of pre-trained models with richer contextual information.

An important aspect to consider is how to evaluate the output of the methods used for CS. CS is about classifying each sentence in the input text to determine whether it is a boundary sentence or not. This can be treated as a binary classification problem where each sentence is classified whether it is a boundary sentence or not. Since this is a binary classification problem, it might make sense to utilize precision and recall as an evaluation metric. However, the problem with precision and recall is that they are not sensitive to near misses.

A more sophisticated evaluation approach used to evaluate text segmentation is PK (Beeferman et al., 1999) which is a sliding window-based method. While sliding the window, the algorithm determines whether the two ends of the window are in the same or different segments in the ground truth segmentation, and increases a counter if there is a mismatch. The final score is calculated by scaling the penalty between 0 and 1 and dividing the number of measurements. There are multiple challenges with the PK method which are: a) false negatives are penalized more than false positives. b) it does not take the number of boundaries into consideration. If there are multiple boundaries inside the window, PK does not consider that. c) PK is sensitive to the variation in segment size. d) Near-miss errors are penalized too much.

An improvement over PK is the windowDiff (Fournier, 2013a) method which is also sliding window-based method. For each position of the window of size k, it compares how many boundaries are in the ground truth, and how many boundaries are predicted by the Topic Segmentation model. It solves some of the PK problems such that penalizing FPs and FNs more equally and penalizing near misses less harshly. However, it still has some problems as it is biased towards favouring automatic segmentations with either small number of clusters.

More advanced approaches attempt to overcome the sliding-window limitations by adopting a new metric called boundary edit distance, which differentiates between full and near misses. The usage of an edit distance that supports transpositions to compare segmentations is an advancement over window-based methods. The boundary edit distance (Fournier, 2013a) models full misses as the addition/deletion of a boundary, and near misses as n-wise transpositions.

2.2 Keyphrase Extraction

KE refers to the task of automatically extracting specific keyphrases or special terms from a piece of text. This task can be divided into two categories: extractive and generative. The extractive approach involves locating potential keyphrases in the text through span classification, where each span of the text is classified based on whether it is a keyphrase or not (Sun et al., 2021; Alzaidy et al., 2019). On the other hand, the generative approach focuses on generating keyphrases given a piece of text using a sequence-to-sequence (seq2seq) model (Yuan et al., 2020; Chen et al., 2020). Both of these methods produce the list of keyphrases all at once.

In the generative line, a recent approach is to fine-tune transformer-based language models that were pretrained for abstractive text summarization (Glazkova and Morozov, 2023). Another interesting approach that combines both generation and extraction of keyphrases is given in Chen et al. (2019). They use a pure extractive model, and pass the retrieved keyphrases to the decoder part of a generative model to help the generative model correctly generate keyphrases.

Using a pure extractive approach have limitations regarding keyphrases that are not explicitly mentioned in the text. This could pose challenges for our task, particularly since keyphrases in our data may not always be explicitly stated within the text. Therefore, our approach will look into a pure generative model to generate keyphrases and locate them, and an extractive approach that given generated keyphrases, it can locate the text referenced by those keyphrase. We further discuss our methodology in Section 3.2.

2.3 Semantic Relation Classification

The SRC task involves automatically recognizing semantic relation between two or more entities in the text. In contrast with the classical approach where the entities are mentioned explicitly in the text, we are interested in the semantic relations between entities that might be absent in the text. In our setting, entities are generated keyphrases.

In the task of identifying different semantic relations between parts of texts, relevant literature is the one related to **SemEval-2010 Task 8** (Hendrickx et al., 2010). The task consist on identifying mutually exclusive semantic relations between pairs of nominals, as for example the relations Cause-Effect or Entity-Destination. Besides our entities being generated, with some not found in the text, our relations are not limited to be between nominals, as keyphrases can, for example, include verbs. Another limitation is that the ten used relations in the paper are relations between exactly two elements, while we are interested in more expressive relations that can connect more than two keyphrases.

Similar to the Dataset Creation Chapter outlined in **SemEval-2010 Task 8** (Hendrickx et al., 2010), we establish specific annotation guidelines (refer to Appendix Section **??**). The annotation samples are divided among the team members and subsequently reviewed by another annotator.

Another interesting dataset related to our task is **SciERC** (Luan et al., 2018). **SciERC** is a collection of 500 scientific abstract annotated with scientific entities, their relations, and coreference clusters which shows a comprehensive approach to identify scientific entities and cross-sentence relations within academic articles. Given that our task involves identifying semantic relationships that connect more than two keyphrases, which can include but are not limited to nominals, methodologies used in SciERC dataset do not perfectly fit our purpouse.

3 Methods

3.1 Chapter Segmentation

Currently, the employed CS technique is a graph-based strategy that groups sentences into chapters based on similarity between them. The graph structure is formed by treating each sentence as a node in the graph and the edges between different nodes represents the cosine similarity measure between these two sentences (nodes). Finally, the louvain method for community detection is used to cluster the graph into chapters such that pairs of sentences in the same chapter would have high similarity measure, while pairs of sentences in different chapters would have a low measure.

3.1.1 Clustering

The objective of this experiment is to apply unsupervised clustering techniques to segment a given text into chapters where each sentence belongs to only one chapter. Two relatively small datasets are employed for this clustering task: the first one is generated by the **Fast AI Movies** team and it has around 100 samples, while the second one is a subset of 350 samples taken from the **Wiki-727k** dataset (700k samples). The process of CS using clustering begins with taking text as input, extracting all sentences from the input text, converting them into a compatible format for the algorithm, and finally apply the clustering method.

Clustering algorithms only work with numerical values, hence, it is essential to convert sentences into a vector format. We use the following sentence embedder models in our experiments: all-mpnet-base-v2[2], distilbert-base-nli-stsb-mean-tokens[3], roberta-base-nli-stsb-mean-tokens[4], and sentence-transformers/paraphrase-albert-small-v2[5].

For the clustering experiments, various techniques are employed to cluster similar sentences together. The key idea for choosing these clustering techniques is that they don't require specifying the number of clusters before training. The clustering algorithms used in this experiment are DBSCAN, MeanShift, AgglomerativeClustering, HDBSCAN

[2]https://huggingface.co/sentence-transformers/all-mpnet-base-v2

[3]https://huggingface.co/sentence-transformers/distilbert-base-nli-stsb-mean-tokens

[4]https://huggingface.co/sentence-transformers/roberta-base-nli-stsb-mean-tokens

[5]https://huggingface.co/sentence-transformers/paraphrase-albert-small-v2

and OPTICS. Different strategies were explored to improve efficacy of clustering algorithms. The first approach is to leverage the sentence context prior to embedding. Sentence context is represented by the two sentences surrounding the target sentence. Two variations of this sentence context were employed which are concatenating the surrounding sentences or averaging them.

The second technique explored is positional encoding which is used to provide a relative position for each token or word in a sequence because clustering algorithms do not take into account the position of sentence in the input text. Consequently, the resulted clusters would contain sentences that are not consecutive in the original text. To indicate where one chapter ends and another begins, it is established that when two consecutive sentences belong to different clusters, it marks the conclusion of a chapter and the start of a new one. If a subsequent sentence is assigned to a previous cluster, it is treated as a different chapter. Cosine similarity is another approach utilized to measure the distance between sentences rather than euclidean distance.

3.1.2 Sentence-level Architectures

We frame the CS task as a binary classification problem, aiming to predict chapter-ending sentences within a given text. Our methodology encompasses the development of models based on the Feed-Forward network (FF-net) and Transformer Encoder, with a focus on constructing lightweight models and assessing their feasibility compared to the established bi-directional LSTM model (Koshorek et al., 2018).

The dataset includes 10,000 texts extracted from the Wiki-727K (Koshorek et al., 2018). The dataset is split into training, validation, and test sets with a ratio of 9:0.5:0.5, respectively. The input for each model consists of sentence embeddings generated from the text using *all-mpnet-base-v2* [6]. The target variable, t, can be defined as a list $\{t_i | t_i \in [0, 1], 0 \leq i < n-1\}$, indicating whether the i^{th} sentence marks the end of a chapter, with n denoting the total number of sentences. The final sentence of each text's last chapter is excluded from the target list, as it invariably occupies the last position. To accommodate for the lack of sentence-level context in embeddings, enhancements such as positional encoding and contextual embeddings are considered. Positional encoding utilizes a combination of sine and cosine functions as proposed by Vaswani et al. (2017), while the approach for contextual embedding mirrors that used in Clustering methodologies.

FF-net: The model is deliberately designed with simplicity, consisting of a single hidden layer with a ReLU activation function and an output layer followed by a Sigmoid function. The model trains with a batch size of 32 for training and 16 for validation, over 50 epochs, employing early stopping to curb overfitting.

Transformer Encoder: The model comprises a stack of 6 transformer encoder layers, each featuring eight heads for multi-head attention. Training is carried out using a batch size of 8 for both the training and validation sets, over 25 epochs, with early stopping applied.

LSTM: The bi-directional LSTM model, serving as the benchmark (Koshorek et al., 2018), validates the efficacy of the FF-net and Transformer Encoder models. Training is conducted with a batch size of 8 for training and 5 for validation, continuing over 25 epochs with early stopping implemented.

3.1.3 BERT

In contrast to previous methodologies relying on sentence embeddings, we leverage the BERT (Devlin et al., 2019) model to implement CS at the token level. Given our expectation of superior performance from LLMs, we have extended the CS task to predict both main chapters and sub-chapters. This task can be approached in two ways: 1) training separate models to predict main chapters and sub-chapters individually, and then combining predictions during inference; 2) training a single model to classify three classes (main chapter, sub-chapter, none). While the second approach is straightforward, the first offers flexibility and can be implemented in various ways. For the first approach, we compare two methods: the top-down approach, which predicts the main chapter first then the sub-chapter; and the bottom-up approach, which proceeds in the opposite direction.

Our objective is to roughly assess the potential of utilizing the pre-trained BERT model for this task. To this end, we fine-tune the model on an extremely limited dataset from FAST AI Movies. We pre-process 165 texts, each containing 15 to 50 sentences, into pairs of sequences. This pre-processing involves tokenizing the texts with

[6] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

bert-base-uncased [7] and labeling each sequence to indicate whether the first [SEP] token splits any main or sub-chapters. The resulting pre-processed dataset consists of 2,791 sequences. An example of a pair of sequence is as below:

$$\texttt{[CLS]}\ token_{-tlen} \cdots token_{-1}\ \texttt{[SEP]}\ token_1 \cdots token_{tlen}\ \texttt{[SEP]}$$

Here, $tlen$ represents half the maximum token length of the tokenizer except 3 (tokens for [CLS] and [SEP]).

For fine-tuning, we split the pre-processed training dataset in a ratio of 9:1 for training and testing, then feed the data into the pre-trained model, *BertForSequenceClassification*. We fine-tune for 5 epochs with a batch size of 16, optimizing with *BertAdam* with a learning rate of 2e-5 and a linear warmup of 0.1.

3.1.4 Llama2

Continuing with the most recent advancement in the natural language processing domain, the aim of this experiment is to employ the Llama2 model to achieve CS. Llama2 (Touvron et al., 2023), released by Meta in 2023 , is a collection of fine-tuned large language models (LLMs) that are used to handle natural language processing tasks. The Llama2 model utilized in this experiment is the 13B chat model **meta-llama/Llama-2-13b-chat-hf** from hugging face and it is fine-tuned on high quality dataset of **3333** samples to segment input english text into chapters/sub-chapters.

To make the training more efficient, Parameter-Efficient Fine-Tuning (PEFT) (Liu et al., 2022), is a machine learning method used to enhance the performance of a pre-trained model on a particular task. Instead of training the model from scratch, which can be time-consuming and computationally expensive, PEFT works by fine-tuning a small number of model parameters instead of all the model's parameters. In this experiment, Low-Rank Adaptation (Lora) (Yu et al., 2023), which is a PEFT method, is employed to save memory and speed up finetuning of large models by reducing the number of training parameters. To further speed up fine-tuning, we utilize QLoRA (Dettmers et al., 2023) which uses 4-bit quantization to compress a pre-trained language model and further reduce memory usage during fine-tuning. The optimizer used in fine-tuning is **paged_adamw_32bit**.

[7] https://huggingface.co/google-bert/bert-base-uncased

3.2 Keyphrase Extraction

Our KE tasks deviates from the classical extraction approach as we are interested not only in extracting, but generating these keyphrases. For example, in the phrase *"While we strive to uphold our diversity and inclusion values"*, a potential keyphrase could be *"Upholding Diversity & Inclusion Values"*. We can clearly see that the generated keyphrase is not part of the input text. Moreover, since the SRC task relies on the positions of the text segment referenced by each keyphrase, locating this text is necessary. Thus, our approach involves two models: a generative model for keyphrase generation, which is also capable of locating the keyphrases, as discussed in Section 3.2.1, and an extractive model for locating the text referenced by generated keyphrases, as detailed in Section 3.2.2.

3.2.1 Generative Approach

For the generative approach, we use the Llama2 13B parameter model (Touvron et al., 2023) from HuggingFace (**meta-llama/Llama-2-13b-hf**). As in Section 3.1.4, to make the training more efficient we use PEFT and 4 bit quantization with QLoRA. The modules finetuned by PEFT were the Query, Key, Value and O matrix, the latter being used in multi-head attention before adding the output of the different heads. As an optimizer we used *adamw_bnb_8bit*.

The dataset used for fine-tuning consists of 4.8k samples. Each sample has the text we want to extract the keyphrase from and a list of keyphrases with the section of the text the keyphrase was extracted from. The gold labels used for fine-tuning are of the form: [$keyphrase_1$:$matching\_text_1$; $keyphrase_2$...]. After running an analysis on a tokenization of the prompt and corresponding output, we decide to use a maximum token length of 2000.

In line with Glazkova and Morozov (2023), we also fine-tune two models, an extractive model and a generative one. The pure extractive model used is *keyphrase-extraction-kbir-kpcrowd*[8], based on KBIR (Kulkarni et al., 2022). For the generative model, we fine-tune and try different models which were pre-trained on summarization, as done in Chen et al. (2019), such as *facebook/bart-large-xsum* [9] model. Training on our dataset, we use the extractive model to find the matching texts and we

[8] https://huggingface.co/ml6team/keyphrase-extraction-kbir-kpcrowd

[9] https://huggingface.co/facebook/bart-large-xsum

pass the matching texts to the summarizer to extract the desired generative keyphrases.

3.2.2 Keyphrase Position Matching (Extractive Approach)

Unlike the generative approach, this method requires a list of keyphrases and a piece of text as input, aiming to locate these keyphrases within the given text. The task involves training a BERT large (Devlin et al., 2019) model to identify the specified keyphrases within the text through binary token-level classification. This is done by concatenating the input text with all the keyphrases of the text and surrounding the current keyphrase with special tokens, namely `[KEYPHRASE_START]` and `[KEYPHRASE_END]`. Essentially, this creates an input sample for each keyphrase, with the current keyphrase marked by the special tokens. A token assigned a prediction value of 1 indicates that it corresponds to the input keyphrase. Including all keyphrases as input ensures that the model grasps sufficient contextual information, allowing it to enforce an order and handle instances where duplicate keyphrases refer to distinct parts of the text. Since the prediction is given to each token of the input including the list of keyphrases and the special tokens, we ignore this part of the prediction and only consider the text tokens (see Figure **??** in the appendix for an example).

A keyphrase can refer to only one span of the text and they are arranged in a sequential order, where the text referenced by $keyphrase_i$ precedes that of $keyphrase_{i+1}$. Therefore, we developed two postprocesing methods, one to fix predictions with more than one span of ones and a second method to fix overlaps between two predictions. Here, a prediction consists of a list of zeros and ones assigned to each token. The first method merges any two spans of ones that are separated by two tokens labeled as 0. If multiple spans persist, we choose the span associated with the token showing the highest confidence score for class 1 (see Figure **??** in the appendix for an example). The second approach cuts between two predictions by zeroing out intersected indices with values of 1. Additionally, to maintain order, predictions are adjusted if a later keyphrase is predicted before an earlier one. For instance, if $keyphrase_{i+1}$ is predicted to be between indices 2 and 3, while $keyphrase_i$ is predicted to be between indices 4 and 5, manual enforcement of order is necessary in the post-processing step. In such cases where predicted indices do not intersect, the end index of the second keyphrase is assigned to the first one, and the start index of the first keyphrase is assigned to the second. For example, given the input [(4, 5), (2, 3)], it will be adjusted to [(3, 3), (4, 4)].

To map predictions on tokens back to words, we predicted a word as class 1 if any of its tokens are predicted as class 1.

Our dataset originally comprised 4.8k text samples, which, when concatenated with their corresponding keyphrases, totaled 26k samples. Through augmentation techniques such as randomly removing keyphrases, discarding the longest ones randomly, incorporating three different text styles, and shuffling the order of keyphrases. These augmentations, combined with the original samples, resulted in a dataset of 65k samples. Upon concatenating these with their keyphrases, the dataset grew to 330k samples, which were used for training. A dataset with the same amount of samples was created for training the model for German language. In this case we use German BERT Large model (Chan et al., 2020). Table **??** in the appendix shows the training parameters for both the English and the German models.

3.3 Semantic Relation Classification

3.3.1 Data Generation

Inspired by the Stanford Alpaca project (Taori et al., 2023), we aim to generate synthetic data for SRC task. Initially, we manually prepare input and output samples, which are used as few-shot for few-shot prompting. Using these manually created samples, we generate 5,812 input and output samples using GPT-4 (Achiam et al., 2023). These samples are then used as the training dataset for the following extractive and generative tasks.

3.3.2 Extractive Approach

In this approach, we first embed each keyphrase by extracting the corresponding hidden states from the text. This is achieved by feeding the complete input text into the model and selecting the hidden states of the tokens of the text referenced by each keyphrase. Leveraging the keyphrase position matching model detailed in Section 3.2.2, we determine which segments of the text correspond to each keyphrase. Subsequently, we represent the relations between these keyphrases by computing the difference between their hidden state vectors, and then train a basic feedforward network to classify these relations.

It is noteworthy that as each token possesses its own hidden state, therefore we experiment with four aggregation methods to consolidate these representations: taking the mean between the first and last token, considering only the first token, considering only the last token, and computing the mean across all tokens.

3.3.3 Generative Approach

Contrary to the extractive approach, we utilize LLMs as a generative approach for the SRC task. Our methodology begins with few-shot prompting, aimed at identifying the most effective few-shot prompts for LLMs. The few-shot comprises 4 samples of input and manually labeled output. With the refined few-shot prompt, we proceed to two-step fine-tuning with Mixtral-8x7B (Jiang et al., 2024) and Llama2-70B (Touvron et al., 2023). In the first step, the model is fine-tuned on the synthetic dataset generated by GPT-4 (Achiam et al., 2023) (see Section 3.3.1), which includes 5,812 input and output samples. Then, the model is further fine-tuned on a high-quality manually labeled dataset, which has 275 samples. There are three main goals for SRC: extracting key elements from the input text, identifying semantic relations among these elements, and understanding logical concepts that encompass these relations.

Few-Shot Prompting We assume that the optimal few-shot prompt for GPT-4 (Achiam et al., 2023) is also effective for other LLMs such as Mixtral-8x7B and Llama2-70B. We explore two types of prompts: an *all-in-one* prompt and a *step-by-step* prompt. The system role in the *all-in-one* prompt includes a task description, definitions of elements, relations, and concepts, followed by input and output examples as few-shot. In contrast, the *step-by-step* prompt, inspired by Chain-of-Thought Prompting (Wei et al., 2022), introduces three sequential prompts for elements, relations, and concepts, respectively, while preserving the preceding conversation. We evaluate few-shot prompts using the OpenAI API's *gpt-4-1106-preview* model, which is capable of handling a context window of 128,000 tokens.

Unlike GPT-based models, Mistral-based models provide only user and assistant roles within the prompt structure. Therefore, we integrate the prompts from the system and user roles, evaluated with GPT-4, and then test this unified prompt with the Mixtral-8x7B Instruct model [10] available on Hugging Face. While detailed contexts can be delivered to the prompts for Mixtral-8x7B, which has a maximum sequence length of 4,096x32 tokens, the Llama2-70B model, which supports a maximum sequence length of 4,096 tokens, requires an alternative approach to prompting. Due to this limited setting, Llama2-70B *all-in-one* prompting had to be done using *one-shot* and with a smaller description of the problem. Because of the huge Llama2-70B size, we had to run the prompts on *Azure AI Machine Learning Studio*.

We also explored the difference of performance between providing the key elements and letting the LLM find them. For the first case, we constructed a simple quantitative metric to check the performance. First, the metric finds for each relation (in the gold label and predicted label) all its key elements inside (nested elements included). Second, for each relation in the predicted label we compute an intersection-based similarity with all gold labels of the same relation category. Note we can use intersection-based similarities as key elements are (almost always) unique and follow the same order as the input (otherwise metric is zero). Third, we take the higher similarity score and remove its associated gold label in order that we do not consume it again for a similarity calculation. Lastly, the final metric is the average of the similarities found and one zero for each relation missing.

Fine-tuning Mixtral-8x7B To ensure efficient fine-tuning of the Mixtral-8x7B Instruct model [10] on 4 NVIDIA V100 GPUs, we lighten the model by setting its maximum sequence length to 1,024, resulting in dataset filtering. Initially, the model is fine-tuned on 5,811 samples from the synthetic dataset, with each sample starting with a shortened instruction prompt, followed by input text and its corresponding output. Then, we fine-tune the fine-tuned model from its 1st and 5th epochs, using high-quality 267 samples. For both fine-tuning steps, we use a batch size of 4 and a total epoch of 5 and employ 4-bit quantization along with LoRA (Hu et al., 2021) as a PEFT (Mangrulkar et al., 2022) method.

Fine-tuning Llama2-70B As for Fine-tuning Mixtral-8x7B, we use 4-bit quantization, LoRA as PEFT, and a maximum token length of 1024. Due to model's size, we use the best available *Azure* compute instance we had acceess to: 4xA100,

[10]https://huggingface.co/mistralai/Mixtral-8x7B-Instruct-v0.1

880GB RAM, and 256GB storage. We run experiments using both Data Parallel Distribution (DDP) and Data Parallel (DP) [11]. Unfortunately, DP runtime was estimated to be 140 hours, costing around $2800, which was out of the budget. On the other hand, when using DDP we got decent runtimes of around 40 hours. Due to the high hardware requirements for running our model, even when using batch size 1, we got time-outs that systematically kill the training process.

4 Results

4.1 Chapter Segmentation

For evaluation, five different scoring functions are used to evaluate all conducted experiments. These five scoring functions are: scoring function provided by **FAST AI Movies**, Boundary Similarity, Segmentation Similarity, WindowDiff, and PK. The last four functions are provided by the **SegEval** python library. An average of the five functions is calculated and used to compare results of different strategies. Results are in Table 1.

For the clustering approaches, extensive experimentation is conducted utilizing all possible combinations of the strategies and algorithms used. This resulted in 520 different experiements applied to each input text from our 100-sample dataset and a subset of 350 samples taken from the **Wiki-727k** dataset. Clustering functions did not result in better performance than the previously implemented louvain graph-based method. Generally, positional encoding did not improve the efficacy of clustering algorithms. Although, the use concatenated sentences in sentence context would increase the input size for the clustering algorithms, it proved to be effective and achieved the highest scores. The highest score (54.6%) is achieved by the louvain method without positional encoding and with sentence context as concatenation. A very similar score (54%) is achieved by agglomerative clustering without positional encoding and with the use of concatenation sentence context.

In the sentence-level architecture approach, the models demonstrate average scores either as good as or better than those achieved by the clustering approach. Interestingly, positional encoding at the sentence level does not enhance performance as observed in the clustering approach. The application of contextual embedding to the simply layered FF-net yields a decent average score of 52.22%, surpassing most clustering algorithms except for the Louvain method and Agglomerative clustering. The Transformer Encoder, known for providing more contextual information, outperforms all clustering algorithms with an average score of 63.10%, although the benchmarked bi-directional LSTM slightly surpasses it with an average score of 66.36%. We anticipate that hyperparameter tuning with the Transformer Encoder, combined with densely contextualized sentence embeddings, will lead to even better performance.

Fine-tuning the pre-trained BERT model surpasses all previous methods as expected, achieving an average score of 75.90%. This underscores the power of understanding both left and right contexts at the token level and fine-tuning a pre-trained model.

Finally, fine-tuning Llama2-13B chat model has second highest average score with (74.81%) and it is quite close to the bert model. The results are reasonable in most cases but the resulted CS need not match the ground truth because there might be multiple sensible chapter splits. This shows that the process of fine tuning the huge 13 billion Llama2 model with only few high quality dataset of 3333 samples can achieve high results and avoid re-training the whole model or investigating a lot of time to generate a huge dataset.

4.2 Keyphrase Extraction

4.2.1 Generative Approach

For evaluation, as we are generating the keyphrases instead of extracting them, a quantitative metric such as edit distance does not precisely measure how good the predictions are. For example, the gold keyphrase *"Phising Attempt Awareness"* and the predicted one *"Data Phising Suspicion"* should have a high value and quantitative metrics would fail at this task. For this reason, we opted to do a qualitative evaluation of 50 samples based on four metrics:

- Matching Text Score (MTS): For each keyphrase and its corresponding matching text generated we give score 1 if the gold matching text is the same (up to extra connectors), 0.5 if it was splitted or consists of two different keyphrases matching text, and zero if it is not found in the gold labels. We additionally add zeroes for each matching text we did not predict and calculate the average.

[11] https://huggingface.co/docs/transformers/perf_train_gpu_many

Table 1: CS results. FAM is a scoring function from FAST AI Movies. BS is Boundary Similarity, SS is Segmentation Similarity, and WD is WindowDiff from SegEval (Fournier, 2013b).

Score (%)			**FAM**	**BS**	**SS**	**WD**	**PK**	**Average**
Method	Clustering	Louvain	79.50	14.02	82.57	48.00	49.00	54.62
		DBSCAN	45.70	0.00	89.10	61.36	61.40	51.52
		MeanShift	54.80	8.40	84.70	52.40	55.95	51.25
		Agglomerative	61.30	12.10	86.20	54.20	56.50	54.06
		HDBSCAN	65.40	5.30	81.20	48.75	54.86	51.10
		OPTICS	51.30	2.30	87.54	58.20	59.50	51.77
	Sentence-level Architecture	FF-net	12.67	25.55	90.83	64.65	67.42	52.22
		Transformer Encoder	60.33	34.27	89.81	62.68	68.39	63.10
		LSTM	60.02	41.32	**91.00**	68.03	71.42	66.36
	Token-level Architecture	BERT	**84.51**	**65.00**	83.75	67.38	78.85	**75.90**
		Llama2-13B	76.49	63.84	82.92	**71.86**	**78.94**	74.81

- Keyphrase Score (KS): For each keyphrase generated, we give score 1 if its corresponding gold label has the same semantic meaning, 0.5 if it close but not quite the same, and zero when carries a different meaning. We compute the average after it.

- Matching and Keyphrase Related Score (MKRS): We give score 1 if the predicted keyphrase is related to its predicted matching text and 0 otherwise. Compute the average after it.

- Matchint Text Found in original Text Score (MTFS): We give score 1 if the matching text predicted is a substring of the original text and 0 otherwise.

For each metric, we computed afterwards the total average over the 50 samples. The results can be seen in Table 2. The fine-tuned model outputed text that is found in the original text for each keyphrase of the 50 samples and its corresponding keyphrase was related in the 98% of the cases. However, only an average of 0.5 in the Keyphrase Score is achieved, deviating from the expected keyphrases. This is not so surprising since even professional annotators do not always agree on the same annotations.

Regarding the model that combined the usage of KBIR and a summarizer, the results had a low quality, which led to stopping the research in this direction.

4.2.2 Keyphrase Position Matching

For evaluation, we constructed a validation dataset with 487 text samples, resulting in 2.9k samples when combined with their keyphrases, constituting 10% of the full dataset before augmentation. In addition to standard metrics like F1 score, accuracy, precision, and recall, we introduced a new metric called "word offset by x." This metric evaluates predictions within a window of x words, instead of strictly one-to-one mapping, aiming to capture the model's ability to identify keyphrases with slight offsets.

Table 3 shows that the model achieves a 91% F1 score for exact predictions (Offset=0), and this increases to 96% when allowing for a word's prediction to be off by 2. These metrics are reported on the English dataset. Table **??** in the appendix displays the performance metrics for the German dataset.

Scores	**MTS**	**KS**	**MKRS**	**MTFS**
Values (%)	66	49	98	100

Table 2: Keyphrase generation model scores over 50 test samples: Matching text score (MTS), Keyphrase score (KS), Matching and Keyphrase related score (MKRS), and Matchint text found on original text score (MTFS).

Metric	**Offset=0**	**Offset=1**	**Offset=2**
Precision (%)	96.80	97.88	98.38
Recall (%)	86.65	90.66	93.77
F1 Score (%)	91.44	94.13	96.02
Accuracy (%)	98.19	98.74	99.13

Table 3: Keyphrase position prediction model: performance metrics with different word offsets for the English dataset.

4.3 Semantic Relation Classification

4.3.1 Extractive Approach

Despite enhancements introduced in the extractive approach, the produced results are not comparable to the generative approach and therefore not reported.

4.3.2 Generative Approach

For evaluating the results of SRC, we compare seven different approaches:

- Few-Shots Prompting with GPT-4, Mixtral-8x7B, and Llama2-70B.
- MixtralFT1_Epoch_i: Two models resulted from fine-tuning Mixtral-8x7B on the dataset with 5,811 samples.
- MixtralFT2_Epoch_i: Two models resulted from further fine-tuning Mixtral-8x7B with a high quality dataset with 267 samples.

Inspired by A/B test, we anonymized the output of the seven models for eleven different prompts, randomized the order the outputs appear for each sample, and asked the five members of the team and the company supervisor to give a ranking of the three best outputs. We give as scores *Top 1*, the percentage of times a model is ranked as first model, and *Top 3*, the percentage of times a model is selected as being one of the three best models. Results can be seen in Table 4. Interestingly, the best model that performs the best in the qualitative evaluation is the first fine-tuned Mixtral model after just one epoch.

Score (%)	Top 1	Top 3
Few-Shots GPT-4	22.22	53.33
Few-Shots Mixtral	0.00	22.22
Few-Shots Llama2-70B	0.00	4.44
MixtralFT1_Epoch_1	**44.44**	**84.44**
MixtralFT1_Epoch_5	13.33	60.00
MixtralFT2_Epoch_1	11.11	40.00
MixtralFT2_Epoch_5	8.88	35.55

Table 4: SRE results: qualitative metrics Top 1 and Top 3 for seven different LLMs.

5 Conclusion

We notice that the traditional method like BERT still remain important when tackling well-defined supervised tasks such as CS and KE. On the other hand, the latest techniques proves good performance in generating unknown or new output as shown in generative approach of KE and SRC. It is also noteworthy that matching the position of keyphrases with BERT and generating the meaningful keyphrases with LLama2 can be effective in the KE task. Specifically, the BERT model reliably locates the generated keyphrase, achieving an F1 score of 96% when allowing the predictions to be within a window of two words. Lastly, our fine-tuned Mixtral model for SRC surpasses the few-shot prompting with GPT-4.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Rabah Alzaidy, Cornelia Caragea, and C. Lee Giles. 2019. Bi-lstm-crf sequence labeling for keyphrase extraction from scholarly documents. In *The World Wide Web Conference*, WWW '19, page 2551–2557, New York, NY, USA. Association for Computing Machinery.

Doug Beeferman, Adam Berger, and John Lafferty. 1999. Statistical models for text segmentation. *Machine Learning*, 34(1-3):177–210.

Branden Chan, Stefan Schweter, and Timo Möller. 2020. German's next language model. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6788–6796, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Wang Chen, Hou Pong Chan, Piji Li, Lidong Bing, and Irwin King. 2019. An integrated approach for keyphrase generation via exploring the power of retrieval and extraction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2846–2856, Minneapolis, Minnesota. Association for Computational Linguistics.

Wang Chen, Hou Pong Chan, Piji Li, and Irwin King. 2020. Exclusive hierarchical decoding for deep keyphrase generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1095–1105, Online. Association for Computational Linguistics.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Chris Fournier. 2013a. Evaluating text segmentation using boundary edit distance. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1702–1712, Sofia, Bulgaria. Association for Computational Linguistics.

Chris Fournier. 2013b. Evaluating Text Segmentation using Boundary Edit Distance. In *Proceedings of 51st Annual Meeting of the Association for Computational Linguistics*, page to appear, Stroudsburg, PA, USA. Association for Computational Linguistics.

Goran Glavaš, Federico Nanni, and Simone Paolo Ponzetto. 2016. Unsupervised text segmentation using semantic relatedness graphs. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 125–130, Berlin, Germany. Association for Computational Linguistics.

A. V. Glazkova and D. A. Morozov. 2023. Applying transformer-based text summarization for keyphrase generation. *Lobachevskii Journal of Mathematics*, 44(1):123–136.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. SemEval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 33–38, Uppsala, Sweden. Association for Computational Linguistics.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Omri Koshorek, Adir Cohen, Noam Mor, Michael Rotman, and Jonathan Berant. 2018. Text segmentation as a supervised learning task. *arXiv preprint arXiv:1803.09337*.

Mayank Kulkarni, Debanjan Mahata, Ravneet Arora, and Rajarshi Bhowmik. 2022. Learning rich representation of keyphrases from text. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 891–906, Seattle, United States. Association for Computational Linguistics.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`.

Charuta Pethe, Allen Kim, and Steven Skiena. 2020. Chapter captor: Text segmentation in novels. *arXiv preprint arXiv:2011.04163*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Alessandro Solbiati, Kevin Heffernan, Georgios Damaskinos, Shivani Poddar, Shubham Modi, and Jacques Cali. 2021. Unsupervised topic segmentation of meetings with bert embeddings. *arXiv preprint arXiv:2106.12978*.

Swapna Somasundaran et al. 2020. Two-level transformer and auxiliary coherence modeling for improved text segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7797–7804.

Si Sun, Zhenghao Liu, Chenyan Xiong, Zhiyuan Liu, and Jie Bao. 2021. Capturing global informativeness in open domain keyphrase extraction.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yu Yu, Chao-Han Huck Yang, Jari Kolehmainen, Prashanth G. Shivakumar, Yile Gu, Sungho Ryu Roger Ren, Qi Luo, Aditya Gourav, I-Fan Chen, Yi-Chieh Liu, Tuan Dinh, Ankur Gandhe Denis Filimonov, Shalini Ghosh, Andreas Stolcke, Ariya Rastow, and Ivan Bulyko. 2023. Low-rank adaptation of large language model rescoring for parameter-efficient speech recognition. In *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE.

Xingdi Yuan, Tong Wang, Rui Meng, Khushboo Thaker, Peter Brusilovsky, Daqing He, and Adam Trischler. 2020. One size does not fit all: Generating and evaluating variable number of keyphrases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7961–7975, Online. Association for Computational Linguistics.

Contributions

Table 5 shows how each member of the team contributed to the project and the report.

Name	Report Contribution	Project Contribution
Alejandro Bravo de la Serna	• Section 2.2: Generative approach. • Section 2.3 • Section 3.2.1 • Section 3.3.3: Llama2 related sections and last paragraph from Few-Shot Prompting. • Section 4.2.1 & Table 2 • Section 4.3.2	• **Keyphrase Extraction:** contributed to the *Keyphrase Extraction Generative Approach*, fine-tuning different models for the task, such as a Llama2 model or a combination of KBIR and a summarizer. Additionaly, designed and carried the qualitative approach for assessing performance, and integrated the model with the code base. • **Semantic Relation Classification:** contributed to Few-Shot Prompting with GPT-4 and different Llama2 models. Worked on the fine-tuning of Llama2-70B model for the task. Designed a score function to quantitative evaluate Semantic Relation Classification when keyphrases are an input of the LLM.
George Elfayoumi	• Rest of Section 2.1: Scoring Functions • Section Intro 3.1, Section 3.1.1 • Section 3.1.4 • 1st and 2nd paragraphs in Section 4.1 • Conclusion	• **Chapter Segmentation:** contributed to exploring different unsupervised clustering algorithms and contributed to fine tuning Llama2 model on a curated high quality dataset for splitting text into chapter and subchapters. • **Scoring Functions:** contributed to researching and adding scoring functions: boundary similarity, segmentation similarity, windowDiff and Pk to evaluate different methods for chapter segmentation.
Parag Bamel	• Introduction • Section 3.3.1 • Section 3.3.2 • Section 4.3.1	• **Chapter Segmentation:** Tested and implemented different sentencec embedder for the clustering + sentence-based Chapter Segmentation. • **Keyphrase Extraction:** Tested different representation modes of embeddings for token spans of arbitrary length.
Jinyu Lee	• Abstract • 1st paragraph in Section 2.1 • Section 3.1.2 & 3.1.3 & • All but the metric and Llama2-70B related content in Section 3.3.3 • 3rd and 4th paragraph in Section 4.1 & Table 1 • Appendix Section **??**, **??**	• **Chapter Segmentation:** contributed to Sentence-level Architectures by pre-processing the dataset, building our own models like FF-net and Transformer Encoder, and benchmarking the LSTM-based model. Additionally, pre-processed the dataset, fine-tuned the BERT model, and evaluated its performance. • **Semantic Relation Classification:** contributed to Few-Shot Prompting with the GPT-4 and Mixtral-8x7B Instruct model. Furthermore, fine-tuned the Mixtral-8x7B Instruct model in two steps and evaluated its performance.
Mohamed Hesham Ibrahim Abdalla	• Section 2.2: extractive approach and intro. • Section 3.2.2 • Section 3.3.2 in the paragraph explaining how the keyphrases are embedded. • Section 4.2.2 • Appendix Section **??** and **??**	• **Keyphrase Extraction:** contributed to the *keyphrase position matching* model by training the model on a German and an English dataset, implementing post processing algorithms and integrating the model with the code base. • **Semantic Relation Classification:** created a dataset of keyphrases along with their corresponding text positions. In addition, enhanced the extractive approach to accommodate these keyphrases by enabling the processing of positional information.

Table 5: Contributions

A Appendix

removed due to NDA