Extracting Semantic Relationships from Unstructured Textual Data in E-Learning Video Script

George Elfayoumi, Alejandro Bravo de la Serna, Parag Bamel, Jinyu Lee, Mohamed Hesham Ibrahim Abdalla

29.04.2024

Final Presentation: Application Project with FAST AI Movies

Chair of Software Engineering for Business Information Systems (sebis)
Department of Computer Science
School of Computation, Information and Technology (CIT)
Technical University of Munich (TUM)
wwwmatthes.in.tum.de

Outline

Introduction

Motivation

Recap: Chapter Segmentation

Recap: Keyphrase Extraction

Semantic Relation Classification

Conclusion

Appendix

Introduction

Team Members (Data Engineering and Analytics Master Students)

George Elfayoumi

Task :
- Chapterization (evaluation & cluster algorithms, Llama 2)
- Semantic Relation classification for higher-level concepts

Alejandro Bravo de la Serna

Task:
- Keyphrase Extraction (generative)
- Generative approach for Semantic Relation Classification

Parag Bamel

Task:
- Chapterization (data, tokenizer techniques)

Jinyu Lee

Task:
- Chapterization (own model architectures, BERT)
- Generative Approach for Semantic Relation Classification

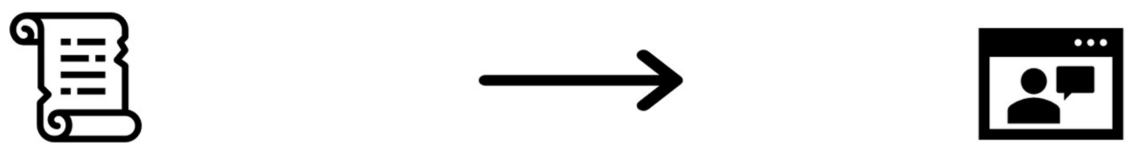
Mohamed Hesham Abdalla

Task:
- Keyphrase Extraction (classification)
- Semantic Relation Classification

Introduction

NLP Application Project @ FAST AI Movies

- ✓ Young, TUM-affiliated Start-Up FAST AI Movies
- ✓ Software-as-a-Service (SaaS) for e-Learning Video Generation

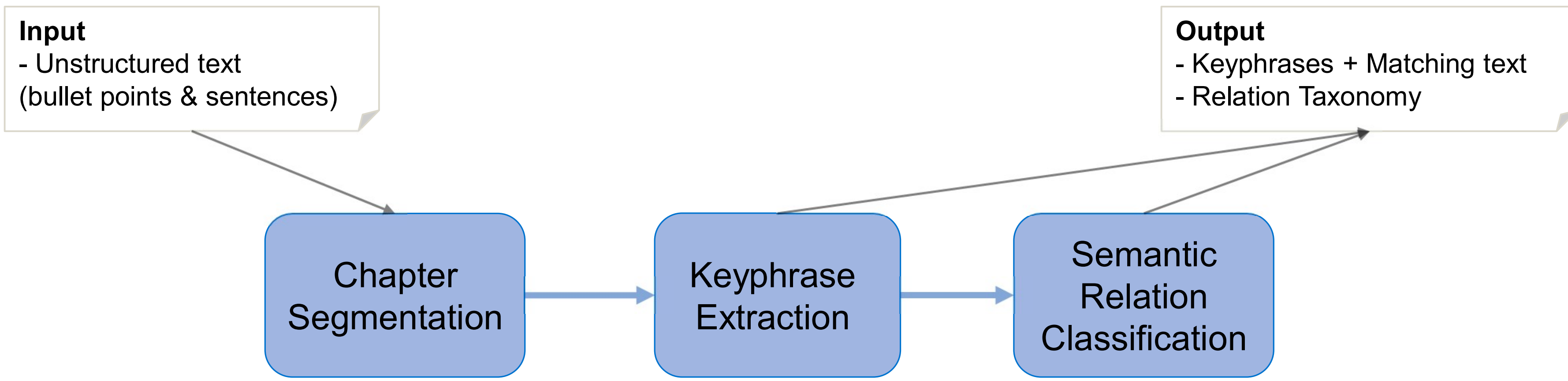


Script of an e-learning video

Animated training video

Motivation

Project Overview



Main Goal:
Create a **comprehensive nested hierarchy** to capture **semantic relationships among keyphrases within a text**, which can be used for visual information representation of e-learning videos.
→ Reduce/Remove "human-in-the-loop" in this process.

Motivation

Example

Generative Semantic Relation Classification
Extractive Semantic Relation Classification

Chapter Segmentation

Input Text (15-50 Sentences)

[0] Welcome to a beginner's guide to software testing.
[1] Software testing is a process, where a software system is evaluated for quality, performance, security, and other attributes.
[2] This process helps to identify any errors or issues in the software before it is launched.
[3] It reduces the risk of software failure and ensures that the end product meets the user requirements.
[4] Let's talk about the key concepts in Software Testing.

Ch. 1

[5] Functional Testing is related to how well the software meets the specified requirements.
[6] Functional Testing techniques include unit testing, integration testing, system testing, and acceptance testing.
[7] Performance Testing checks how the software performs under different workloads.
[8] This includes testing the speed, response time, reliability, resource usage, and scalability of the software.
[9] Another important aspect of software testing is Security Testing.
[10] This evaluates the system's ability to protect data, withstand attacks, and avoid unauthorized access.
[11] User Acceptance Testing (UAT) involves real-world users testing the software to make sure it can handle required tasks in real-world scenarios.
[12] Unit Testing is the first level of testing and is carried out during the development phase - it tests individual components of the software.

Ch. 2

[13] To conclude, effective software testing allows any bugs, errors, or issues to be identified and handled prior to product release.
[14] It is a critical part of ensuring the software system's reliability and performance.

Ch. 3

Keyphrase Extraction

'Effective Software Testing', 'Bugs', 'Errors', 'Identified & Handled', 'Prior to Product Release', 'Critical Part', 'Software System', 'Reliability', 'Performance'

Semantic Relation Classification

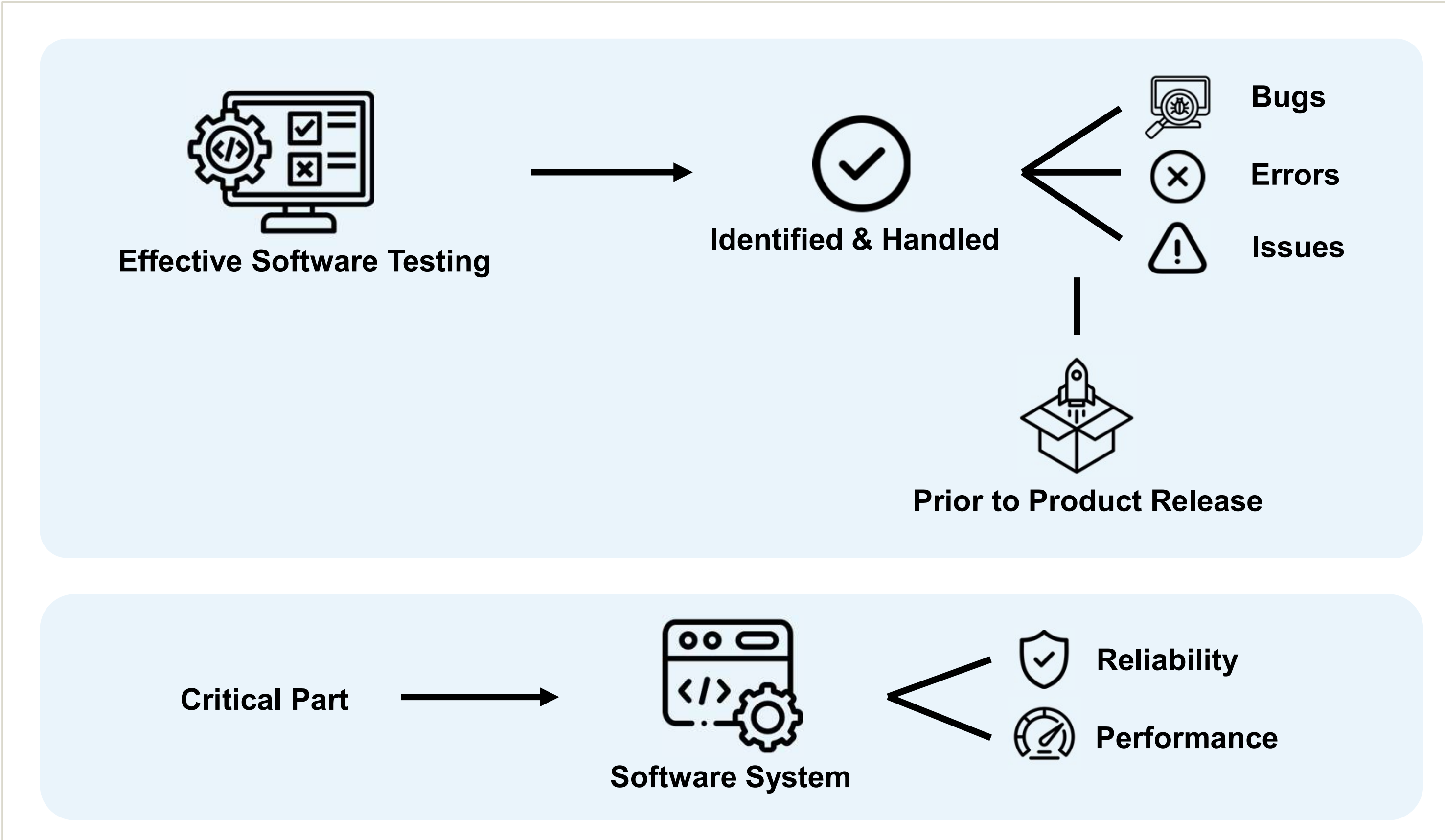
```
causality('Effective Software Testing',
        attribution(
                attribution(listing('Bugs', 'Errors', 'Issues'),
                        'Identified & Handled'
                ),
                'Prior to Product Release'))
causality('Critical Part',
        attribution('Software System',
                listing('Reliability', 'Performance')))
```

(Go to Next Slide)

Motivation

Example

Visualization Example in E-learning Video



Example Slide

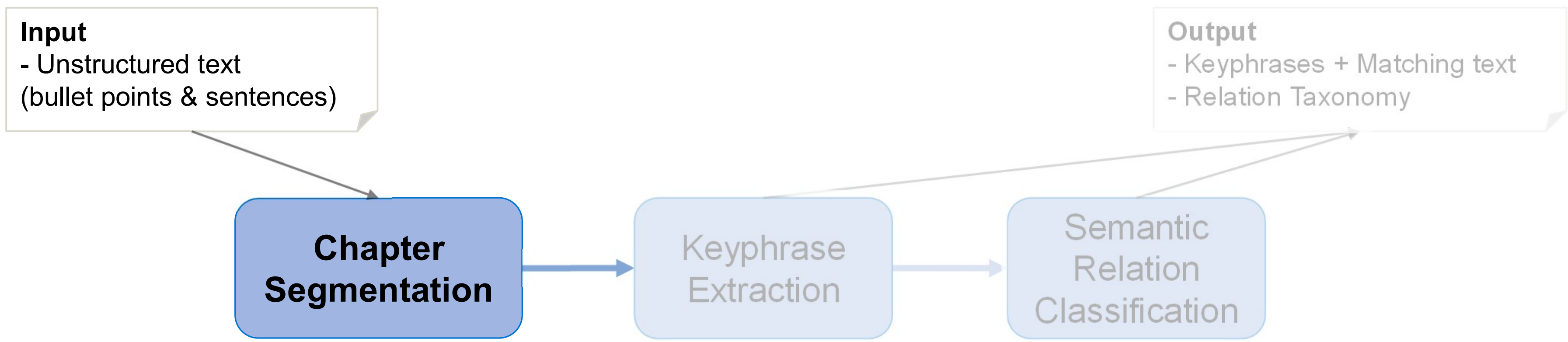
Effective Software Testing → **Prior to product release** — **Identified&Handled:**

- Bugs
- Errors
- Issues

Critical Part → **Software System:**

- Reliability
- Performance

Recap: Chapter Segmentation



Recap: Chapter Segmentation

Example

Input Text (15-50 Sentences)

Chapter 1 — Chap 1-1

[0] Welcome to a beginner's guide to software testing.
[1] Software testing is a process, where a software system is evaluated for quality, performance, security, and other attributes.
[2] This process helps to identify any errors or issues in the software before it is launched.
[3] It reduces the risk of software failure and ensures that the end product meets the user requirements.
[4] Let's talk about the key concepts in Software Testing.

Chapter 2 — Chap 2-1

[5] Functional Testing is related to how well the software meets the specified requirements.
[6] Functional Testing techniques include unit testing, integration testing, system testing, and acceptance testing.

Chap 2-2

[7] Performance Testing checks how the software performs under different workloads.
[8] This includes testing the speed, response time, reliability, resource usage, and scalability of the software.

Chap 2-3

[9] Another important aspect of software testing is Security Testing.
[10] This evaluates the system's ability to protect data, withstand attacks, and avoid unauthorized access.

Chap 2-4

[11] User Acceptance Testing (UAT) involves real-world users testing the software to make sure it can handle required tasks in real-world scenarios.
[12] Unit Testing is the first level of testing and is carried out during the development phase - it tests individual components of the software.

Chapter 3 — Chap 3-1

[13] To conclude, effective software testing allows any bugs, errors, or issues to be identified and handled prior to product release.
[14] It is a critical part of ensuring the software system's reliability and performance.

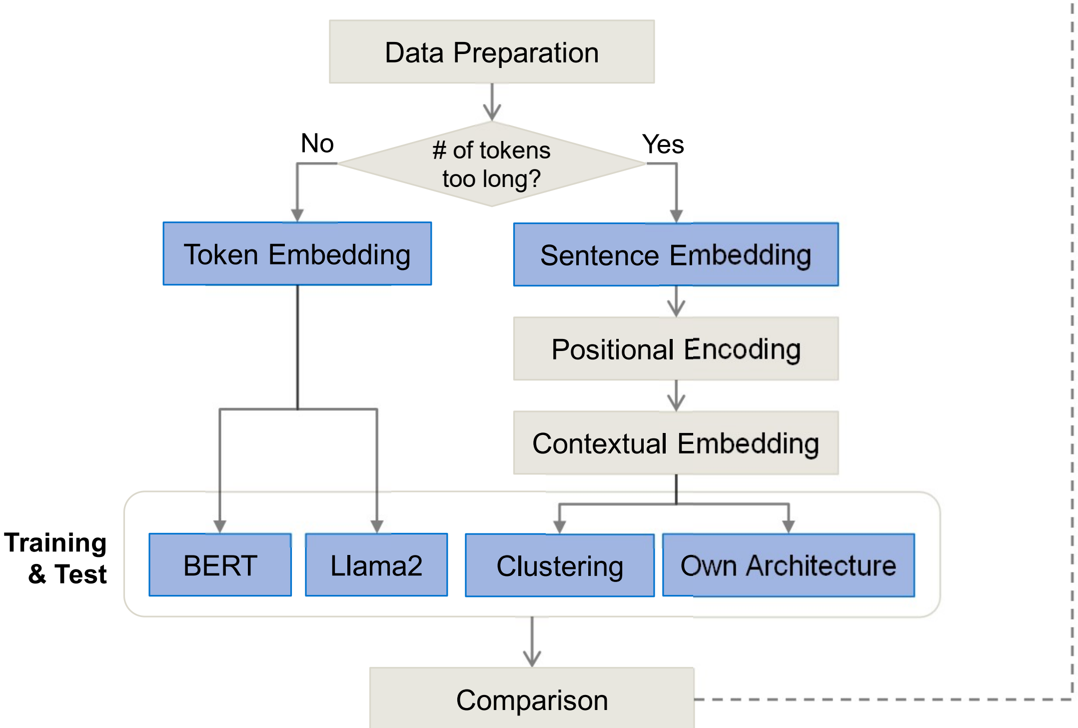
Output (1: End of Chapters)

Main Chapters:
[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1]
Main & Sub-chapters:
[0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]

IMPORTANCE
Well-segmented chapters can enhance the effectiveness of subsequent tasks (Keyphrase Extraction, Semantic Relation Classification).

Recap: Chapter Segmentation

Overview

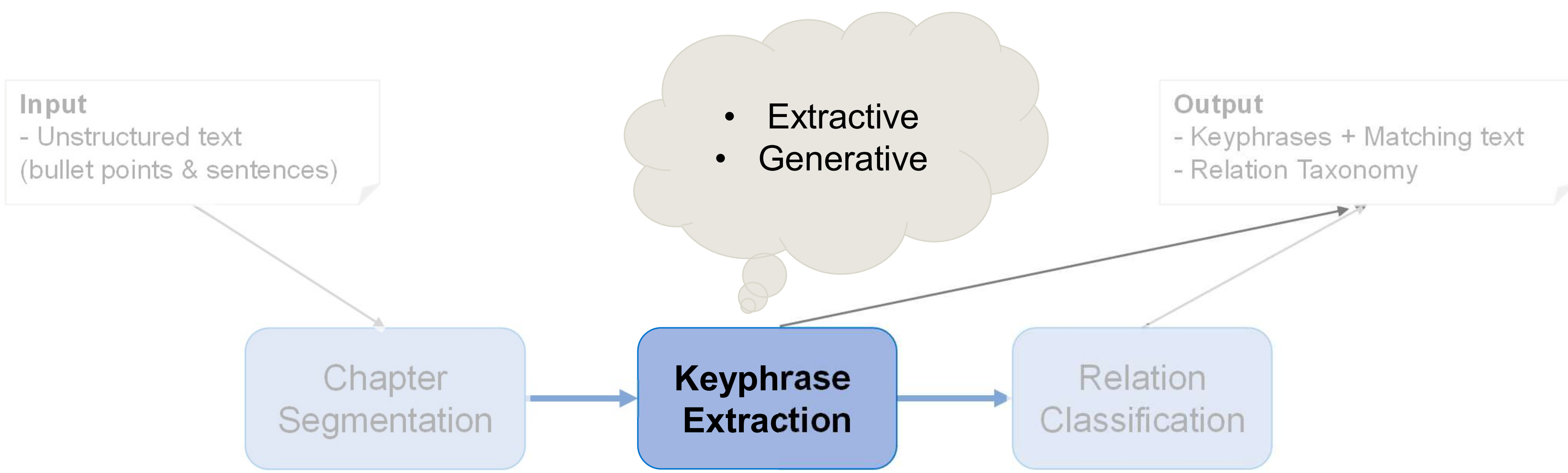


Result Table

Methods		Average Score (%)
Clustering	Louvain	54.62
	DBSCAN	51.52
	MeanShift	51.25
	Aggolmerative	54.06
	HDBSCAN	51.10
	OPTICS	51.77
Sentence-level Architecture	FF-net	52.22
	Transformer Encoder	63.10
	Bi-directional LSTM*	66.36
Token-level Architecture	**BERT**	**75.90**
	Llama2-13B	74.81

*Benchmarking: Koshorek, Omri, et al. (2018)

Recap: Keyphrase Extraction



Recap: Keyphrase Extraction

- **What and Where are the Keyphrases** in the text?
- Keyphrases are **not only words found in the text** but can also **rephrasings of parts of senteces**.

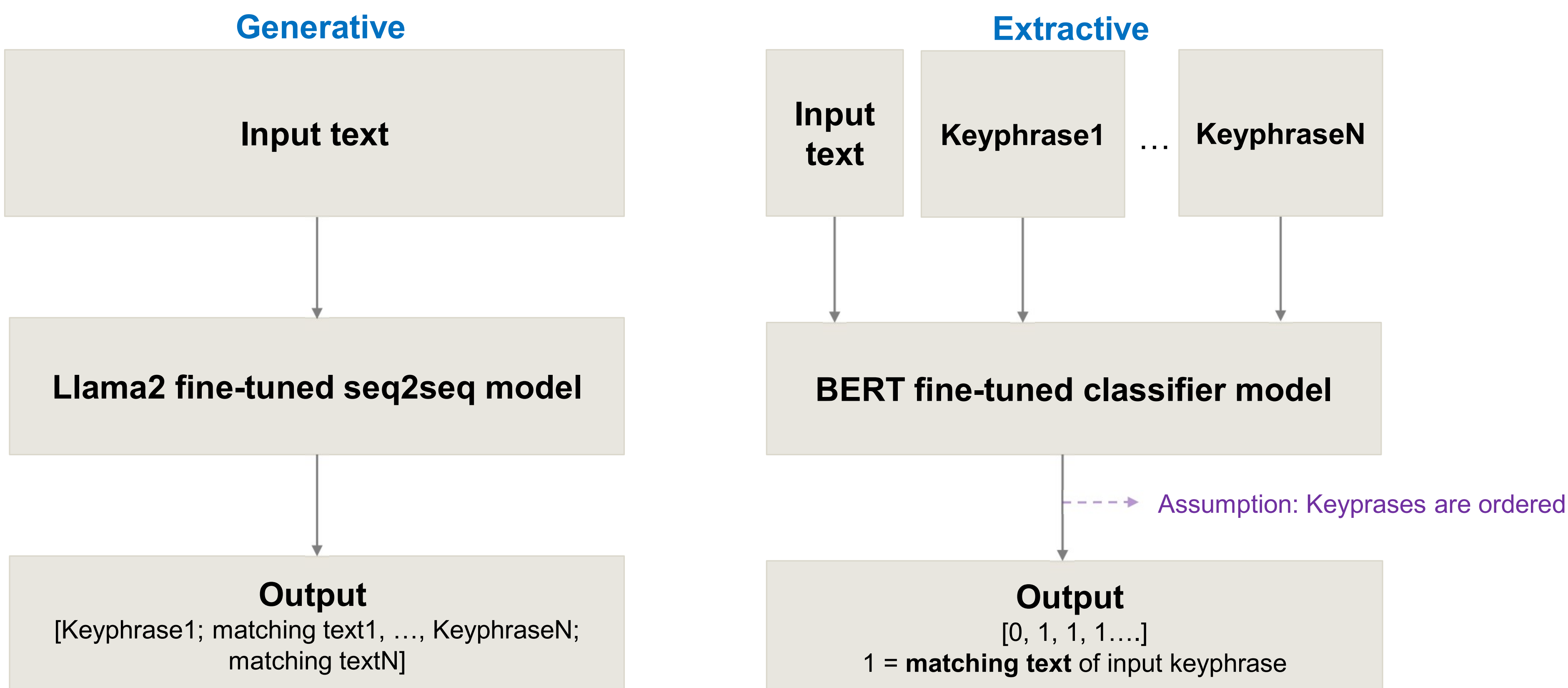
To conclude, effective software testing allows any bugs, errors, or issues to be identified and handled prior to product release. It is a critical part of ensuring the software system's reliability and performance.

Keyphrase: Effective Software Testing
Matching Text: effective software testing

Keyphrase: Bugs
Matching Text: any bugs, errors, or issues

Keyphrase: Identified & Handled
Matching Text: to be identified and handled

Recap: Keyphrase Extraction



Recap: Keyphrase Extraction: Results

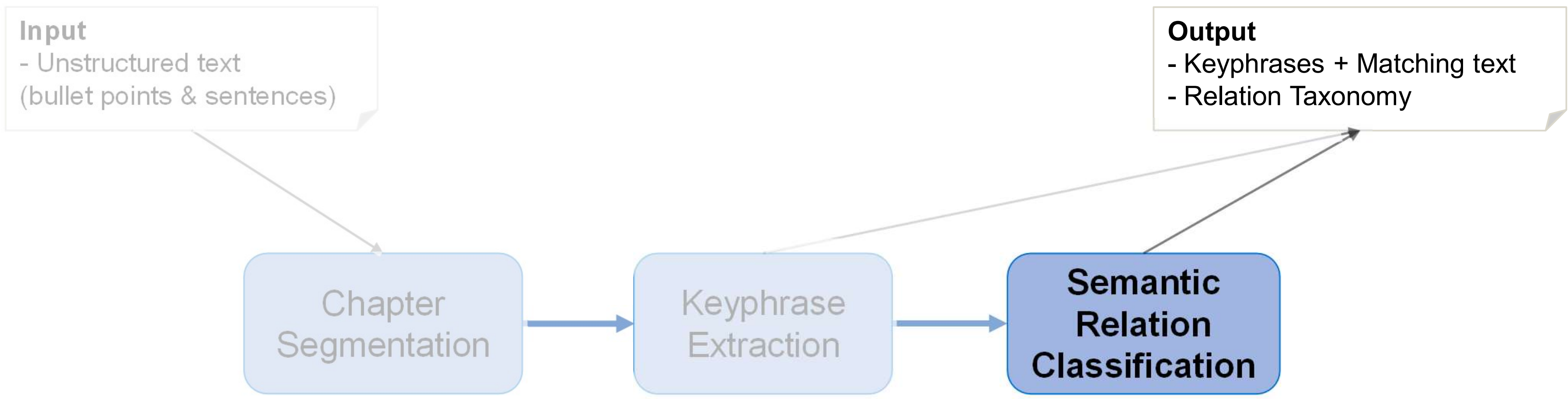
- Extractive:

Prediction off by	Accuracy	Precision	Recall	**F1**
0 (base)	98%	96%	86%	**91%**
1	99%	98%	91%	**94%**
2	99%	98%	94%	**96%**

- Generative:

Matching text score	Keyphrase score	Matching and keyphrase related	Matching text found on original text
66%	49%	98%	100%

Semantic Relation Classification



Semantic Relation Classification

Example

Chapter Segmentation

[Segmented Chapter]

To conclude, effective software testing allows any bugs, errors, or issues to be identified and handled prior to product release. It is a critical part of ensuring the software system's reliability and performance.

Keyphrase Extraction

[Keyphrases]

'Effective Software Testing', 'Bugs', 'Errors', ‘Identified & Handled’, ‘Prior to Product Release', 'Critical Part’, 'Software System', 'Reliability', 'Performance'

Our Work

Ideal

Semantic Relation Classification

```
causality('Effective Software Testing’,
          attribution(
                      attribution(listing('Bugs’, 'Errors’, 'Issues’),
                                  ‘Identified & Handled’
                                  ),
                      ‘Prior to Product Release’
                     )
          )
causality('Critical Part’,
          attribution(‘Software System’,
                      listing('Reliability’, 'Performance’)
                     )
          )
```

IMPORTANCE
Remove/reduce “human-in-the-loop” when automatically generating e-learning video from unstructured script.

Semantic Relation Classification

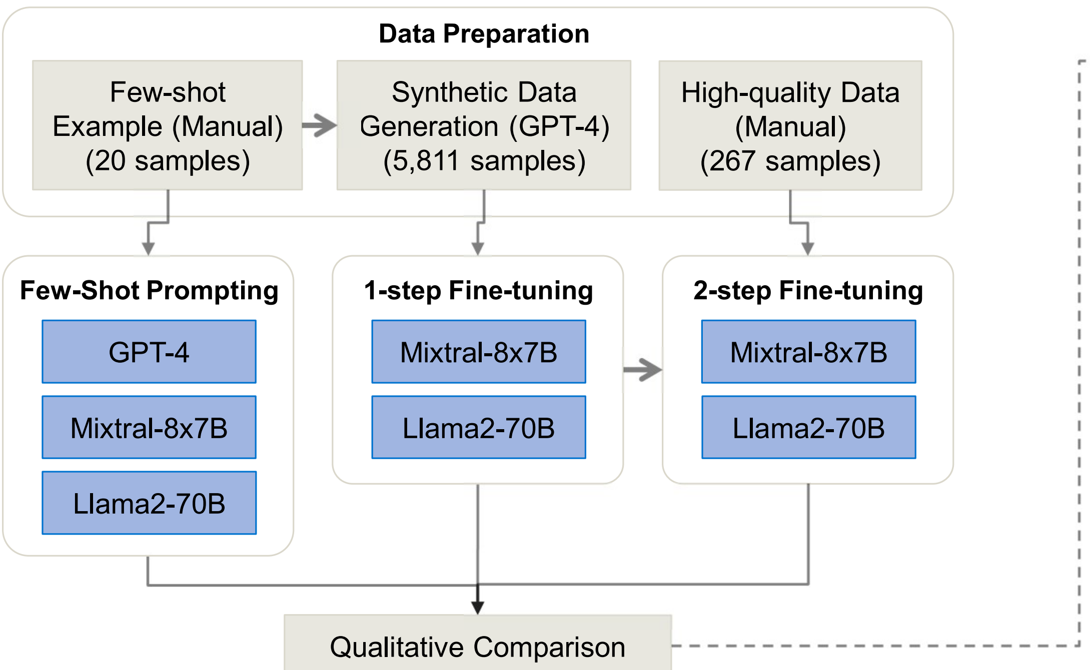
- The dataset SciERC (*) defines 6 relation categories.
- SciERC only takes into account binary relations.

SciERC	Ours	Example	Visualization (without Keyphrase Icons)
Compare	Comparison	Unlike the quantitative prior, the qualitative prior is often ignored...	Quantitative Prior ⇄ Qualitative Prior
Conjuction	Listing	NLP applications such as machine translation and language generation.	NLP Applications < Machine Translation, Language Generation
Used-for	Causality	The TISPER system has been designed to enable many text applications.	TISPER System → Many Text Applications
Feature-of	Attribution	Prior knowledge of the model.	Prior Knowledge — Model
Hyponoym-of			
Part-of			

(*) [Multi-Task Identification of Entities, Relations, and Coreference for Scientific Knowledge Graph Construction](https://aclanthology.org/D18-1360) (Luan et al., EMNLP 2018)

Semantic Relation Classification

Overview



Result Table

Methods		Top-1 (%)	Top-3 (%)
Few-Shot Prompting	GPT-4	22.22	53.33
	Mixtral-8x7B	0.00	22.22
	Llama2-70B	0.00	4.44
1-step Fine-tuning	Mixtral-8x7B (after 1 epoch)	44.44	84.44
	Mixtral-8x7B (after 5 epoch)	13.33	60.00
2-step Fine-tuning	Mixtral-8x7B (from 1 epoch)	11.11	40.00
	Mixtral-8x7B (from 5 epoch)	8.88	35.55

Semantic Relation Classification

Qualitative Comparison

Segmented Chapter

To conclude, effective software testing allows any bugs, errors, or issues to be identified and handled prior to product release. It is a critical part of ensuring the software system's reliability and performance.

1-step Fine-tuning with Mixtral 8x7B (after 1 epoch)

```
causality('Effective Software Testing',
          listing('Identify Bugs',
                  'Handle Errors',
                  'Handle Issues')
         )
causality('Critical Part',
          listing('Software System Reliability',
                  'Software System Performance')
         )
```

Few-Shot Prompting with GPT-4

```
causality('Effective Software Testing',
          listing('Identify Bugs, Errors, Issues',
                  'Handle Before Release',
                  'Ensure Reliability',
                  'Ensure Performance')
         )
```

Conclusion

- **Token-based models typically outperform sentence-level models** in **Chapter Segmentation**, where established methods such as BERT remain crucial and achieve best performance.
- **Traditional methods like BERT** remain **important** when tackling **well-defined supervised tasks** such as Keyphrase Extraction.
- **LLMs such as Llama2 and Mixtral** proves **good performance in few shot generation tasks** as shown in the generative approach of Keyphrase Extraction and Semantic Relation Classification.
- Our **fine-tuned Mixtral** model for Semantic Relation Classification **surpasses** the few-shot prompting with **GPT-4**.
- **Keyphrase Extraction** can be used in the **future** for **extractive relation classification.**

Prof. Dr.

Florian Matthes

Technical University of Munich (TUM)
TUM School of CIT
Department of Computer Science (CS)
Chair of Software Engineering for Business Information Systems (sebis)

Boltzmannstraße 3
85748 Garching bei München

+49.89.289.17132
matthes@in.tum.de
wwwmatthes.in.tum.de

Keyphrase Extraction: Generative Results

Matching text score	Keyphrase score	Matching and keyphrase related	Matching text found on original text
66%	49%	**98%**	**100%**

Example:

Text: issues to be identified and handled prior to product release.
Expected Keyphrase: Identified & Handled

Keyphrase Extracted: Identified and Handled
Matching Text Extracted: be identified and handled

Metric	Score	Explanation
Matching text	75%	Text matches except for the preposition *to.*
Keyphrase	100%	Keyphrase has same meaning.
Matching and Keyphrase related	100%	Keyphrase meaning extracted from the matching text found.
Matching text found on original text	100%	Matching text exists in the original text.

Keyphrase Extraction: Position Matching (Extractive)

Given Input:

TEXT: "The E-Learning platform is specially developed to support our employees. It is user-friendly and accessible, so employees can learn quickly and retain information. The available topics include safety measures, operational guidelines, customer service, and more."

KEYPHRASE: "Specialized E-Learning Platform"

OTHER KEYPHRASES: ["Accessible Learning", "Safety and Operations Training"]

Model Input:

"[CLS] The E-Learning platform is specially developed to support our employees. It is user-friendly and accessible, so employees can learn quickly and retain information. The available topics include safety measures, operational guidelines, customer service, and more. [SEP] Accessible Learning [KEYPHRASE_START] Specialized E-Learning Platform [KEYPHRASE_END] Safety and Operations Training [SEP]"

Model Output:

[0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

ignored

Chapter Segmentation

Overview

["Welcome to SmartHome's guide on the use of our suite of smart devices and how to integrate them into your home automation setup.", "revolution of Internet of Things (IoT) seen rise in smart home technology", "At SmartHome, our vision: every home connected - more efficient, safer, & more convenient", "Perhaps you\u2019re interested in the idea of automation, but not sure where to start", "This guide: step by step instructions and key points to consider", "Broadly, our smart products can be categorized into the following categories:", …]

Input Text

Chapter 1
- Sub Chapter 1
- Sub Chapter 2

Chapter 2
- Sub Chapter 1
- Sub Chapter 2

…

Target

Data Gen

Sentence Embedding
- Current: all-mpnet-base-v2
- New: distilbert-base-nli-stsb-mean-tokens, roberta-base-nli-stsb-mean-tokens, sentence-transformers/paraphrase-albert-small-v2

Positional Encoding

Contextual Embedding

Sentence level

Clustering
- Current: Louvain method
- New: DBSCAN, Mean Shift, Agglomerative, HDBSCAN, OPTICS

Own Architecture
- Simple feed-forward NN
- Simple Transformers

Token level

BERT (bert-base-uncased, BertForSequenceClassification)

Llama2 (13b)

Goal:

Find the most effective chapterization architecture using the average of scoring functions.

Importance:

Effective segmentation of the input text into paragraphs (or chapters) is essential, as we progress to the next tasks (Keyphrase Extraction and Relation Classification) which are performed within these chapters.

Scoring Functions

PK:

- 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.
- Challenges with the Pk Evaluation Metric:
  - False negatives are penalized more than false positives.
  - Does not take the number of boundaries into consideration. If there are multiple boundaries inside the window, Pk doesn't consider that.
  - Sensitive to the variation in segment size.
  - Near-miss errors are penalized too much.

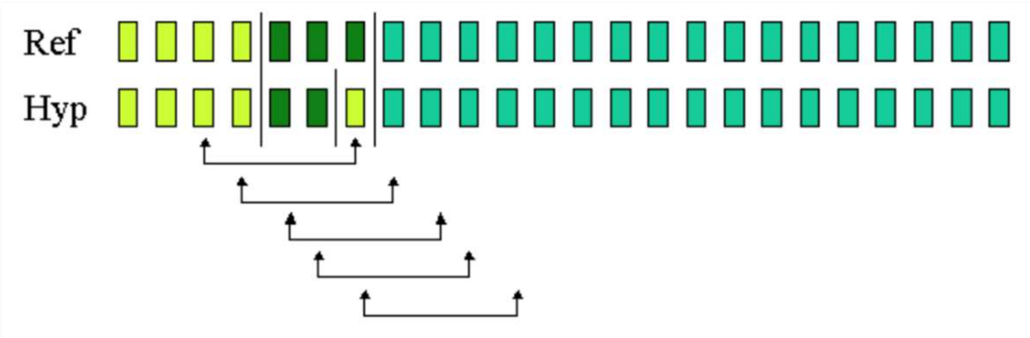


Figure: Sliding window over reference and predictions (Pevzner and Hearst - 2002)

https://www.assemblyai.com/blog/text-segmentation-approaches-datasets-and-evaluation-metrics/

Scoring Functions

WindowDiff:

- sliding window-based method
- for each position of the window of size k, we simply compare how many boundaries are in the ground truth, and how many boundaries are predicted by the Topic Segmentation model.
- How it solves PK problems:
  - penalize FPs and FNs more equally
  - Not skip full misses
  - Less harshly penalize near misses
  - Reduce its sensitivity to internal segment size variance.
- Challenges with the WindowDiff Evaluation Metric:
  - Penalize errors less at the beginning and end of segmentations
  - Are biased towards favouring automatic segmentations with either few or tightly-clustered boundaries
  - Are not symmetric (meaning that they cannot be used to produce a pairwise mean of multiple manual segmentations)

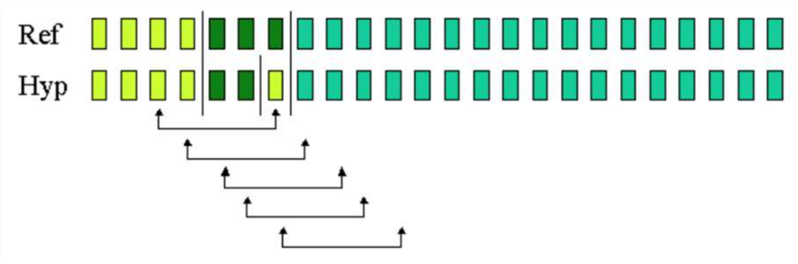


Figure: Sliding window over reference and predictions (Pevzner and Hearst - 2002)

https://www.assemblyai.com/blog/text-segmentation-approaches

Scoring Functions

Segmentation Similarity:

- Instead of using windows, a new metric edit distance called **boundary edit distance** which differentiates between full and near misses.
- Boundary edit distance models full misses as the addition/deletion of a boundary, and near misses as n-wise transpositions.
- The usage of an edit distance that supported transpositions to compare segmentations was an advancement over window-based methods
- **Challenges with Segmentation Similarity:**
  - S is sensitive to variations in the total size of a segmentation, leading it to favour **very sparse segmentations with few boundaries**.
  - S produces **overly optimistic values** due to its normalization. This can make it challenging to interpret the results accurately.
  - S uses **string reversals** (e.g., turning "ABCD" into "DCBA") to handle transpositions. This method makes it difficult to analyze individual pairs of boundaries between segmentations.

Recap: Chapter Segmentation

w/ = with
w/o = without
pos = positional encoding
cont(avg) = contextual embedding (average)
cont(conc) = contextual embedding (concatenation)

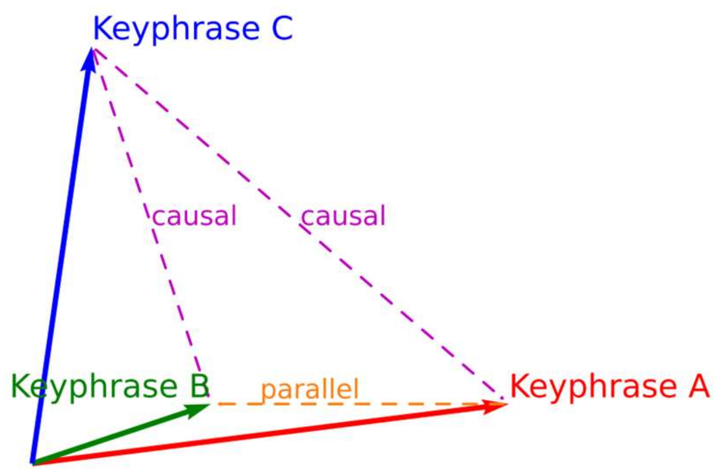
Result

METHODS		Sentence-level									Token-level	
		Clustering						Own Architecture				
		Louvain [Current] w/o pos w/ cont(conc) w/ wiki-350	DBSCAN w/o pos w/ cont(conc) w/ wiki-350	Mean Shift w/o pos w/ cont(conc) w/ wiki-350	Agglomerativ w/o pos w/ cont(conc) w/ wiki-350	HDBSCAN w/o pos w/ cont(conc) w/ wiki-350	OPTICS w/o pos w/ cont(conc) w/ wiki-350	FF-Net w/o pos w/ cont(conc) w/ wiki-10k	Transformers w/o pos w/ cont(conc) w/ wiki-10k	Bi-directional LSTM (*paper) w/ wiki-10k	BERT w/ our data (165)	Llama2-13B
SCORES	tiling_score (FAST AI Movies)	0.795	0.457	0.548	0.613	0.654	0.513	0.12674	0.60331	0.60015	0.84508	0.7649
	Boundary Similarity (segeval)	0.1402	0.0002	0.084	0.121	0.053	0.023	0.25545	0.34269	0.41317	0.65000	0.6384
	Segmentation Similarity (segeval)	0.8257	0.891	0.847	0.862	0.812	0.8754	0.90832	0.89813	0.90990	0.83750	0.8292
	WindowDiff (segeval)	0.48	0.6136	0.524	0.542	0.4875	0.582	0.64649	0.62682	0.68031	0.67384	0.7186
	Pk (segeval)	0.49	0.614	0.5595	0.565	0.54858	0.595	0.67417	0.68392	0.71422	0.78853	0.7894
	Average Score	**0.54618**	**0.51516**	**0.5125**	**0.5406**	**0.511**	**0.51768**	**0.52223**	**0.63097**	**0.66355**	**0.75899**	**0.7481**

*paper: Text Segmentation as a Supervised Learning Task (Koshorek, Omri, et al.)

Relation Classification: Extractive Approach

$keyphrase_i, keyphrase_j$ —LLM Embeddings→ $\overrightarrow{embed(keyphrase_i) embed(keyphrase_j)}$ → Classifier




```
causal(
    "Keyphrase C"
    parallel(
        "Keyphrase A",
        "Keyphrase B"
    )
)
```

BEFORE THIS PAGE, WE KEEP IT AS APPENDIX.
AFTER THIS, WE WILL DELETE.

Motivation

Current Output

```
heading("Data protection",
      causal("Not merely a legal obligation",
            attribution("Commitment",
                  parallel("Privacy", "Trust", "Security of our clients")
                  ),
            ),
      causal( "Adhering", "Principles", "Practices outlined today")
      attribution("Contribute",
            "Safer more secure data environment"
)
```

Unstructured text

To conclude, data protection isn't merely a legal obligation; it is a testament to our commitment to ensuring the privacy, trust, and security of our clients.
By adhering to the principles and practices outlined today, we can contribute to a safer, more secure data environment at MunichInsure.

Ideal Output

```
conclusion(
   heading("Data protection",
         parallel("Not merely a legal obligation",
               attribution("Commitment",
                     parallel("Privacy", "Trust", "Clients Security")
causal(
   attribution("Adhering",
         parallel("Principles", "Practices outlined")
   ),
   "Contribute",
   "Safer data environment"
)
```

Main Goal

Create a **comprehensive nested hierarchy** to capture **semantic relationships among keyphrases in a text**, facilitating a more nuanced and structured representation of their contextual connections.

Motivation

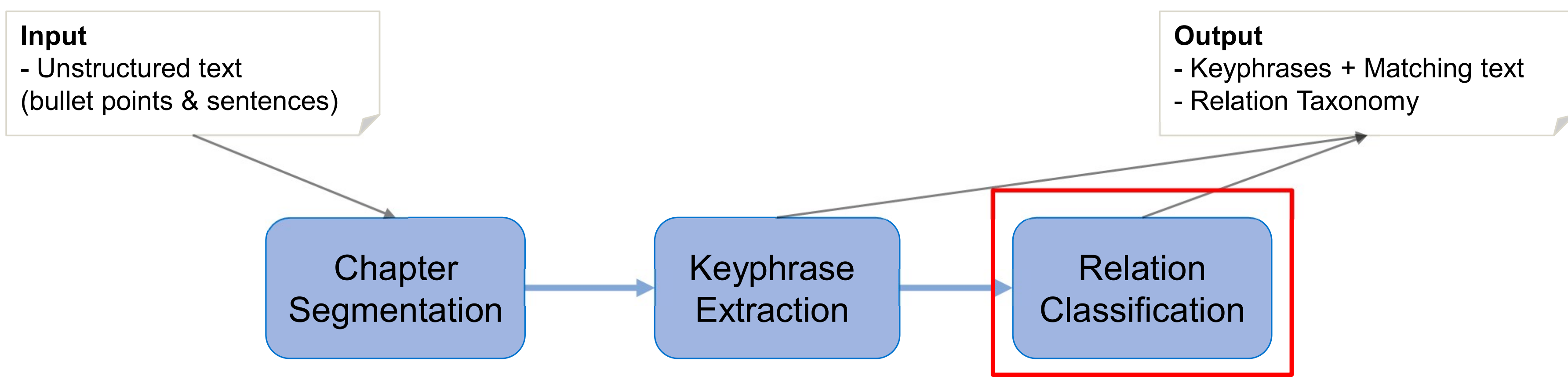
Challenges

Difficult to capture semantic relationships among keyphrases in a text.
→ “human-in-the-loop” is necessary in the current process.

Main Goal

Create a **comprehensive nested hierarchy** to capture **semantic relationships among keyphrases in a text**, facilitating a more nuanced and structured representation of their contextual connections.

System Components (Our Tasks Only)

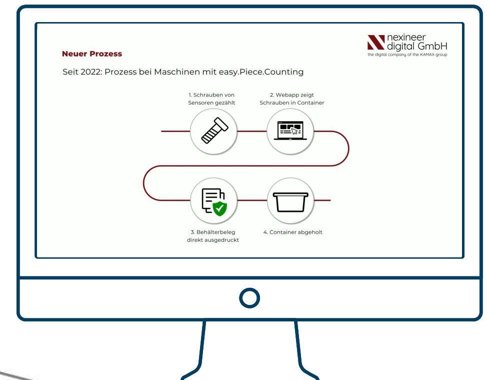


System Components

Main NLP Models at FAST AI Movies

Input
- Unstructured text (bullet points & sentences)
- Images
- Corporate Design

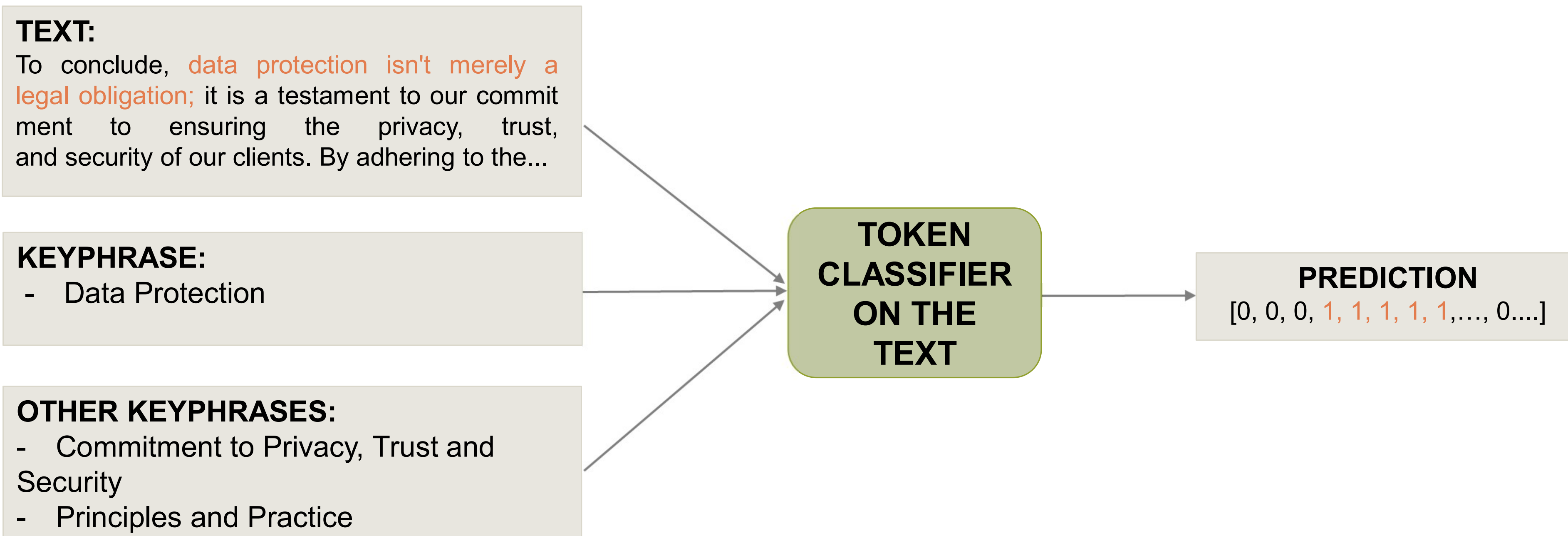
Output
- Training Video Script (TTS)
- Keyphrases + Icons
- Relation Taxonomy
- MC Quizzes



Chapter Segmentation	Text Generation	Keyphrase Extraction	Relation Classification	Icon Database	MC Quiz Generation
Information Retrieval (high-dimensional vector embeddings)	**Generative AI** (fine-tuned state-of-the-art transformers)	Generative AI & **Information Retrieval**	**Information Retrieval** (high-dimensional vector embeddings)	Generative AI & **Information Retrieval**	**Generative AI** (fine-tuned state-of-the-art transformers)

Keyphrase Extraction: Position Matching (Extractive)

- Predict the **matching text** of a given keyphrase.
- Expects Keyphrases are **already there**. (fall back for generetive)



Keyphrase Extraction: Generation with Matching text (Generative)

Input text

↓

Llama2 fine-tuned model

↓

Output
[Keyphrase1; matching text1, …, KeyphraseN; matching textN]

It is a testament to our **commitment to ensuring the privacy, trust, and security** of our clients.

↓

Llama2 fine-tuned model

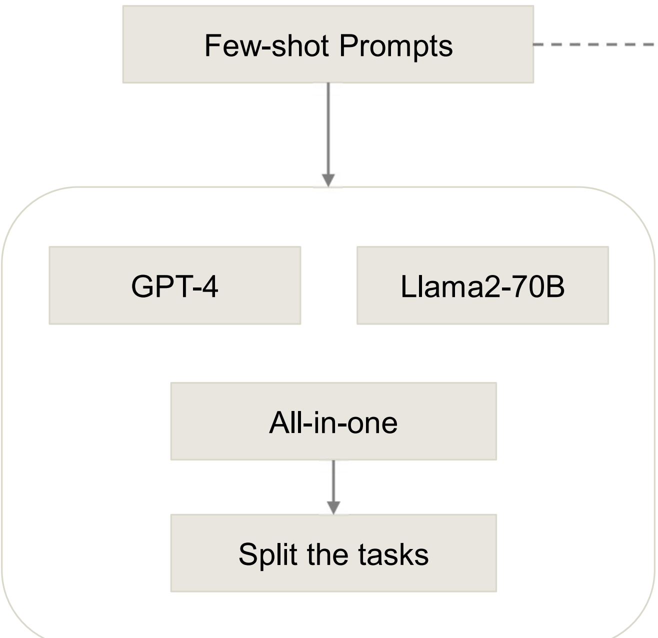
↓

[Commitment to Privacy, Trust, and Security; **commitment to ensuring the privacy, trust, and security**]

Relation Classification: Generative Approach

- Chapterization (Topic Segmentation) as a **binary classification problem**, where each sentence is classified as boundary sentence or not.

Chapterization Evaluation Metrics



Few-shot Example
Input:
- Rules
- Example 1
- Example 2

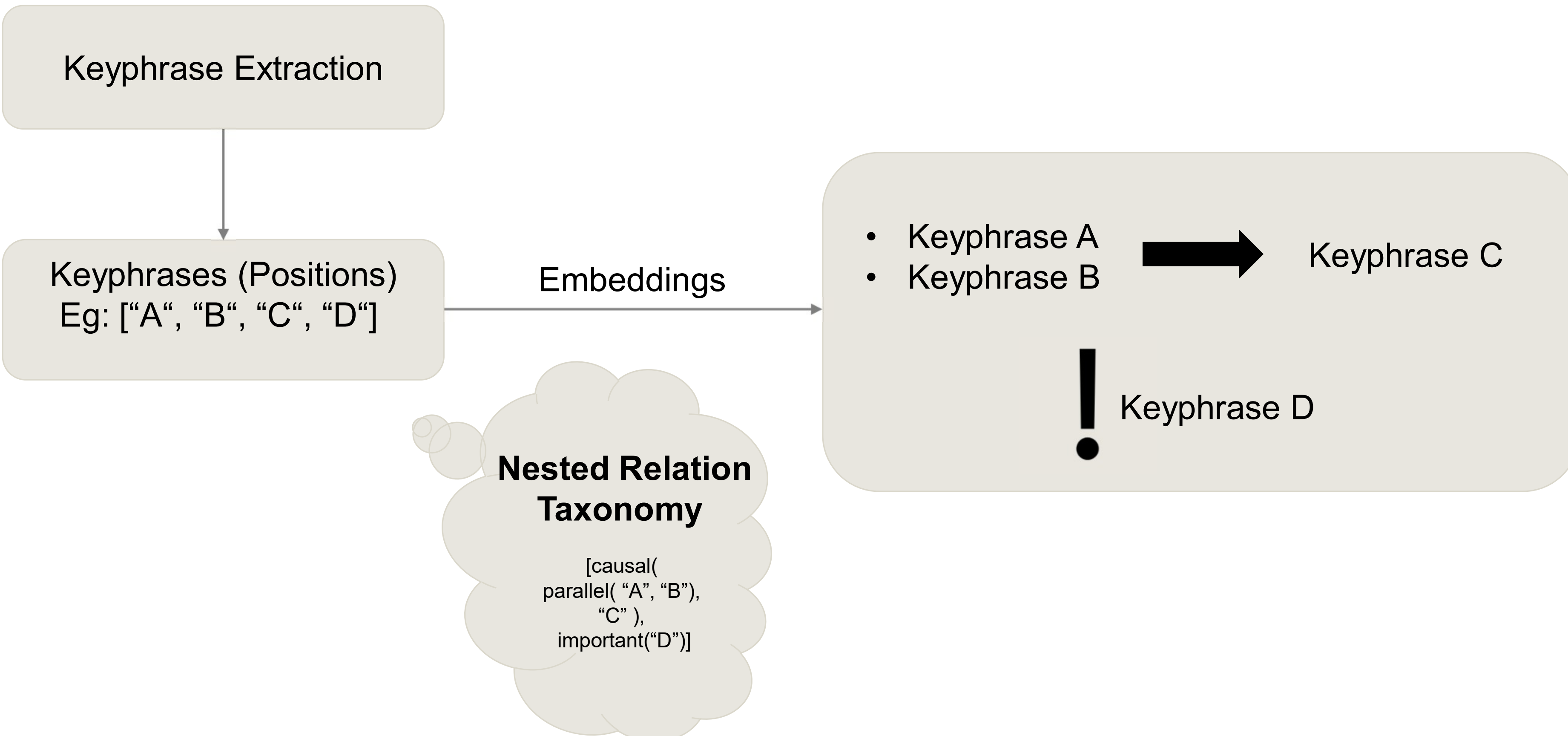
The machine works on the rotary vane principle. The oil seals the gaps, lubricates the vanes and takes away compression heat. In order to avoid reverse rotation after switching off, the machine is equipped with a non-return valve, abbreviated as NRV. In order to avoid solids from entering, the machine is equipped with an inlet screen. Exhaust filters separate the oil from the discharged gas.

Output:
information#["Rotary Vane Principle"]#{attribution(["Oil"], listing(["Sealing Gaps"], ["Lubricating Vanes"], ["Reducing Compression Heat"]))}
concept{causality(["Avoid Reverse Rotation"], ["Non-Return Value NRV"])}
concept{causality(["Avoid Solids Entering\"], ["Inlet Screen"])}
concept{causality(["Exhaust Filters"], ["Separate Oil from Gas"])}

Tasks

1. Retrieve Elements (Keyphrases)
2. Relations Classification
   : Causality, Attribution, Comparision, Parallelism, Enumeration
3. Concepts Classification
   : Information, Important, Conclusion, Process, Concept, Heading

Relation Classification: Extractive Approach



Motivation

Motivation

Unstructured text

Ensuring the protection of customer data is not just a legal obligation but a cornerstone of trust.

Current output example

```
important(
   causal(
      attribution(
         "Securing customer data",
         "Beyond legal requirements"
      ),
      "Essential for trust"
)
```

Ideal output example

```
important(
   causal(
      "Securing customer data",
      parallel(
         "Beyond legal requirements",
         "Essential for trust")
      )
)
```

Main Goal

Create a **comprehensive nested hierarchy** to capture **semantic relationships among keyphrases in a text**, facilitating a more nuanced and structured representation of their contextual connections.

Motivation

Motivation

Unstructured text

By adhering to the principles and practices outlined today, we can contribute to a safer, more secure data environment at MunichInsure.

Current output example

```
causal(
   parallel(
      "Adhering",
      "Principles",
      "Practices outlined today")
      ),
)
attribution(
   "Contribute",
   "Safer more secure data environment”
)
```

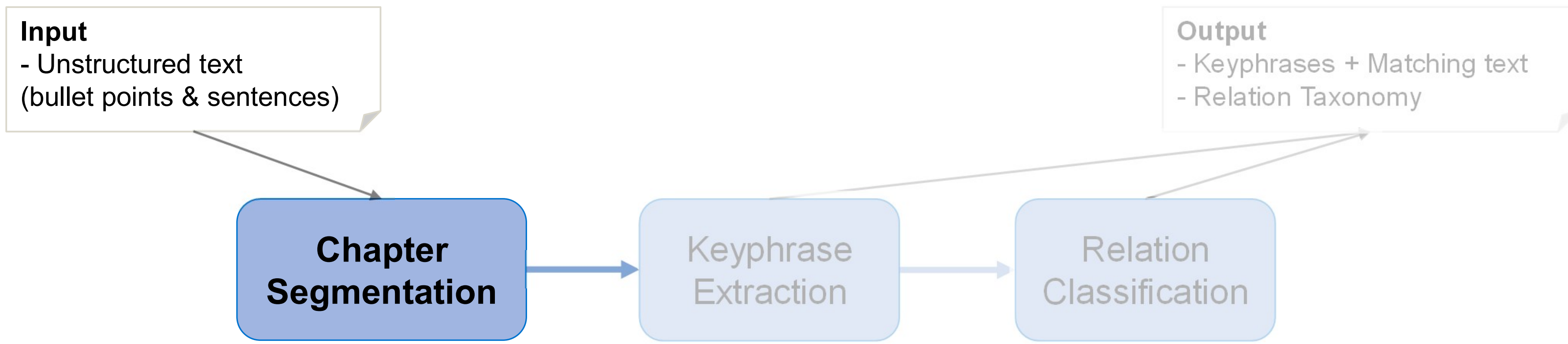
Ideal output example

```
causal(
   attribution(
      "Adhering",
      parallel(
         "Principles",
         "Practices outlined today")
      ),
   "Contribute",
   "Safer more secure data environment”
   )
)
```

Main Goal

Create a **comprehensive nested hierarchy** to capture **semantic relationships among keyphrases in a text**, facilitating a more nuanced and structured representation of their contextual connections.

Chapter Segmentation



Input & Output Example
for Chapter Segmentation

Input text (in a sentence list format):
[sentence1, sentence2, sentence3, sentence4, sentence5, sentence6, sentence7, sentence8, sentence9, sentence10, sentence11, sentence12, sentence13, …]

Target (1: start of the chapter):
[1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, …]

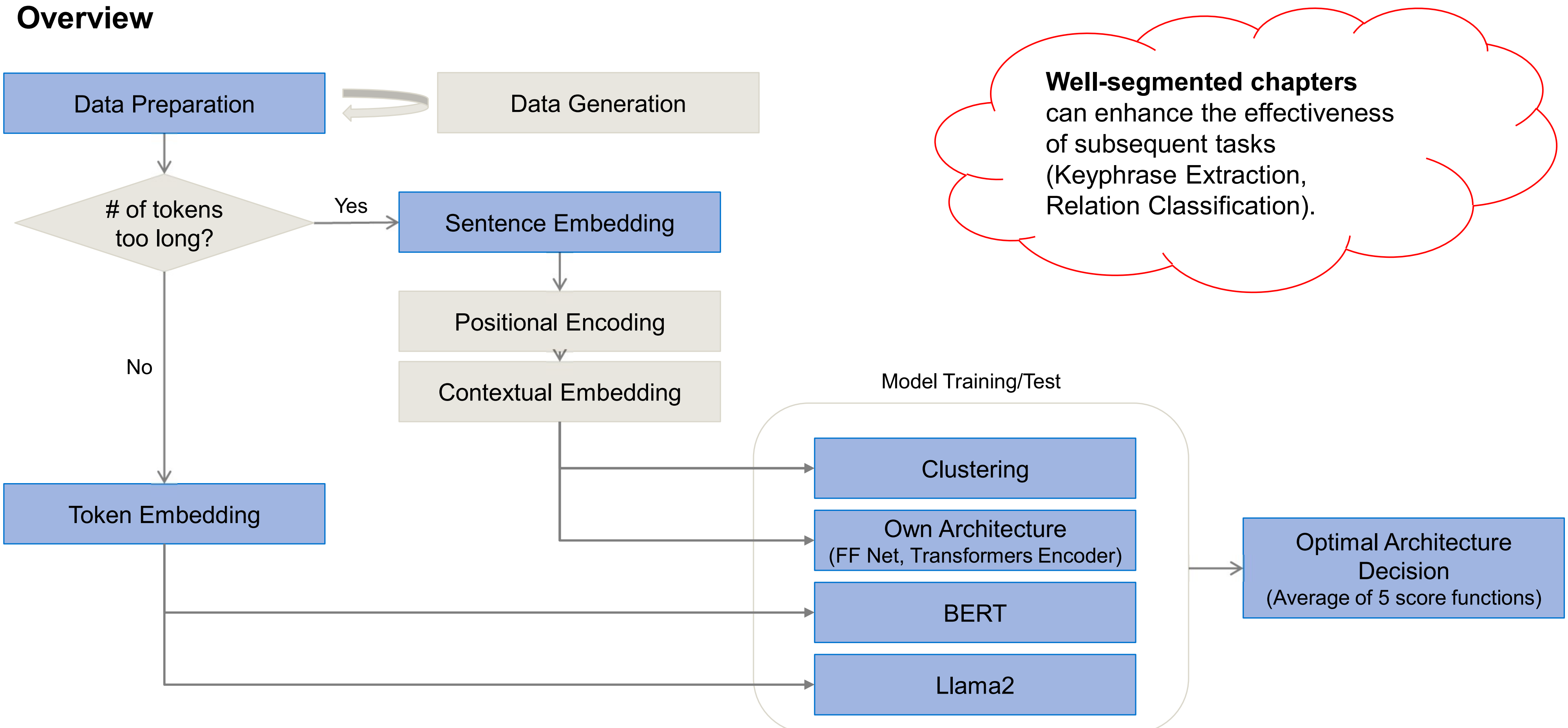
Chapter 1
- Sub Chapter 1
- Sub Chapter 2

Chapter 2
- Sub Chapter 1
- Sub Chapter 2

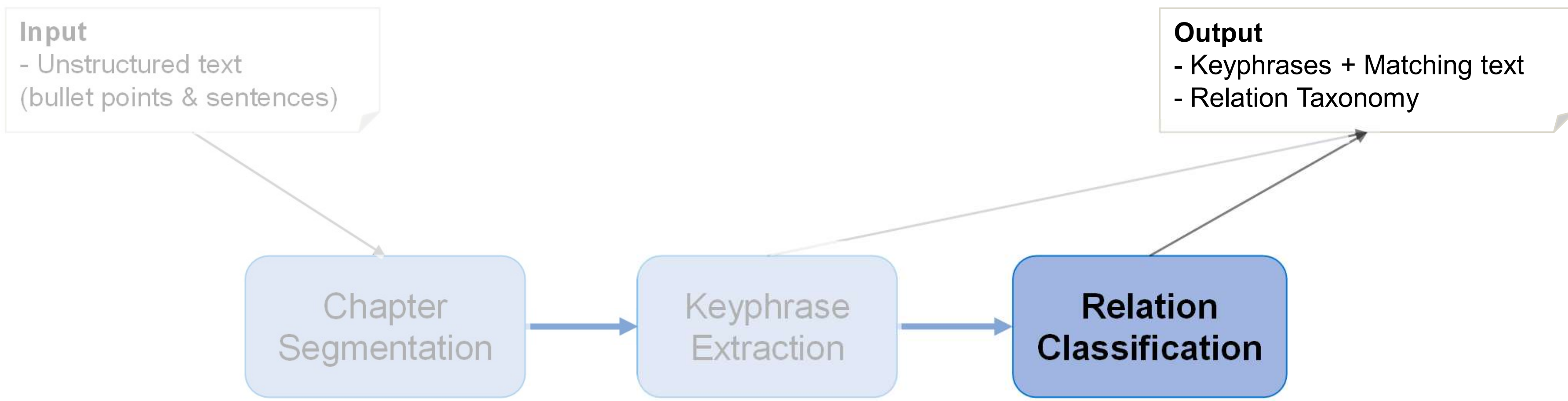
…

Chapter Segmentation

Overview



Relation Classification



Semantic Relation Classification

Overview

Text, Keyphrases

Tasks

1. **Relation** Classification

- n (≥2) keyphrases
  Parallel, Causal, Enumeration
- 2 keyphrases
  Contrary, Positive Correlation, Negative Correlation, Heading, Comparision, Attribution

2. **Concept** Classification

Instruction, Important, Information, Target, Final Conclusion

Approaches

Generative

- GPT-3.5, GPT-4
- Llama2-70B

VS

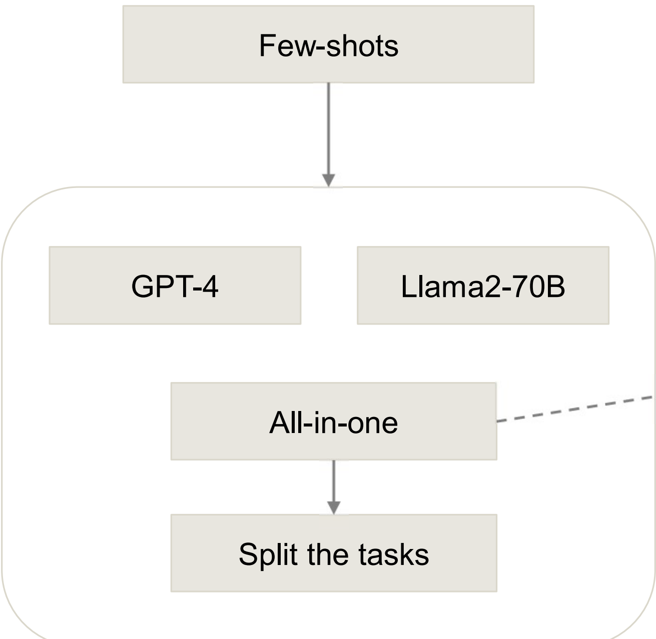
Extractive

- LLM Embeddings

Importance

Remove/reduce "human-in-the-loop"
→ Improve SaaS Capabilities

Semantic Relation Classification: Generative Approach



All-in-one Example

Input Prompt:

- **Explain rules for Relations and Concepts**
- **Few-shot Examples**
- Generate a high-quality output for the following input text and the keyphrases.
    - Input text: "it is a testament to our **commitment** to ensuring the **privacy**, **trust**, **and security** of our clients."
    - Keyphrases: ["Commitment", "Privacy", "Trust", "Security"]

Output:

```
[
   important(
              attribution("Commitment",
                          parallel( "Privacy",
                                    "Trust",
                                    "Security"
                                  )
                          )
              )
]
```

Input Text

[0] Welcome to a beginner's guide to software testing.
[1] Software testing is a process, where a software system is evaluated for quality, performance, security, and other attributes.
[2] This process helps to identify any errors or issues in the software before it is launched.
[3] It reduces the risk of software failure and ensures that the end product meets the user requirements.
[4] Let's talk about the key concepts in Software Testing.
[5] Functional Testing is related to how well the software meets the specified requirements.
[6] Functional Testing techniques include unit testing, integration testing, system testing, and acceptance testing.
[7] Performance Testing checks how the software performs under different workloads.
[8] This includes testing the speed, response time, reliability, resource usage, and scalability of the software.
[9] Another important aspect of software testing is Security Testing.
[10] This evaluates the system's ability to protect data, withstand attacks, and avoid unauthorized access.
[11] User Acceptance Testing (UAT) involves real-world users testing the software to make sure it can handle required tasks in real-world scenarios.
[12] Unit Testing is the first level of testing and is carried out during the development phase - it tests individual components of the software.
[13] To conclude, effective software testing allows any bugs, errors, or issues to be identified and handled prior to product release.
[14] It is a critical part of ensuring the software system's reliability and performance.

Output

Main Chapters: [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1]
Main & Sub-chapters: [0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]

<table>
<tr><th>Our Model (Fine-tuned Mixtral-8x7B Instruct Model)</th><th>Few-shot Prompting with GPT-4</th></tr>
<tr><td colspan="2">Main Chapter 2 - Input Text: Functional Testing is related to how well the software meets the specified requirements. Functional Testing techniques include unit testing, integration testing, system testing, and acceptance testing. Performance Testing checks how the software performs under different workloads. This includes testing the speed, response time, reliability, resource usage, and scalability of the software. Another important aspect of software testing is Security Testing. This evaluates the system's ability to protect data, withstand attacks, and avoid unauthorized access. User Acceptance Testing (UAT) involves real-world users testing the software to make sure it can handle required tasks in real-world scenarios. Unit Testing is the first level of testing and is carried out during the development phase - it tests individual components of the software.</td></tr>
<tr><td>Output:
concept{'Software Testing'}[attribution('Functional Testing', 'Specified Requirements')]
concept[listing('Unit Testing', 'Integration Testing', 'System Testing', 'Acceptance Testing')]
concept[attribution('Performance Testing', listing('Speed', 'Response Time', 'Reliability', 'Resource Usage', 'Scalability'))]
concept[attribution('Security Testing', listing('Protect Data', 'Withstand Attacks', 'Avoid Unauthorized Access'))]
concept[attribution('User Acceptance Testing (UAT)', 'Real-World Scenarios')]
concept[attribution('Unit Testing', 'Development Phase')]</td><td>Output:
concept{'Functional Testing'}[attribution('Meets Specified Requirements', listing('Unit Testing', 'Integration Testing', 'System Testing', 'Acceptance Testing’))]
concept{'Performance Testing'}[listing('Speed', 'Response Time', 'Reliability', 'Resource Usage', 'Scalability’)]
concept{'Security Testing'}[listing('Data Protection', 'Withstand Attacks', 'Avoid Unauthorized Access’)]
concept{'User Acceptance Testing (UAT)'}[causality('Real-World Users', 'Real-World Scenarios’)]
concept{'Unit Testing'}[attribution('First Level of Testing', 'Individual Components'), causality('During Development Phase', 'Individual Components')]</td></tr>
<tr><td colspan="2">Main Chapter 3 - Input Text: To conclude, effective software testing allows any bugs, errors, or issues to be identified and handled prior to product release. It is a critical part of ensuring the software system's reliability and performance.</td></tr>
<tr><td>Output:
conclusion[causality('Effective Software Testing', listing('Identify Bugs', 'Handle Errors', 'Handle Issues'))]
concept[causality('Critical Part', listing('Software System Reliability', 'Software System Performance'))]</td><td>Output:
conclusion[causality('Effective Software Testing', listing('Identify Bugs, Errors, Issues', 'Handle Before Release', 'Ensure Reliability', 'Ensure Performance'))]</td></tr>
</table>

Conclusion

Chapter Segmentation

- While the clustering methods align with the existing Louvain method, **BERT classifier** achieves a notable **ca. 21% improvement in scores**.

→ Will finalize the method after evaluating the performance with Llama-2.

Keyphrase Extraction

- Generetive approach produces **meaningful keyphrases** but **not always identical to the ground truth**. However, matching text is always in the text.
- Position matching through **simple token classification** achieves up to **95% F1 score**.

→ **Combining both approaches** is the way forward.

Our project is **mainly practical and giving good empirical results**, however it is a challenge to compare the performance based on standardized metrics.