CLEF 2024 SimpleText Tasks 1-3: Use of Llama-2 for Text Simplification*

Notebook for the SimpleText Lab at CLEF 2024

Rowan Mann[1], Tomislav Mikulandric[2]

[1]Christian-Albrechts-Universität zu Kiel (CAU), Christian-Albrechts-Platz 4, 24118 Kiel
[2]The University of Split, Ul. Ruđera Boškovića 31, 21000, Split, Croatia

Abstract

In an era defined by the vast availability of information, the challenge of discerning reliable information is more pressing than ever. Our paper presents findings from Tasks 1, 2, and 3 of the SimpleText track at the 15th Conference and Labs of the Evaluation Forum (CLEF) 2024, aimed at advancing research in automatic simplification of scientific texts using LLaMA-2.

Task 1 involved selecting relevant passages for simplified summaries, leveraging ElasticSearch and TF-IDF with cosine similarity for evaluating relevance. We achieved an average Flesch-Kincaid grade level of 0.6, indicating a moderate complexity suitable for further simplification.

Task 2 focused on identifying and explaining difficult concepts. Using the LLaMA-2 13B model, we extracted and rated the difficulty of scientific terms, generating explanations for the most challenging ones. However, reliance on Wikipedia for definitions proved inconsistent, highlighting a limitation in our methodology.

Task 3 addressed the simplification of scientific abstracts and sentences. We utilized LLaMA-2 to generate simplified versions, effectively maintaining the original meaning while reducing complexity and length. Human validation confirmed the preservation of essential content in the simplified texts.

Our research demonstrates the efficacy of LLaMA-2 for text simplification tasks, albeit with noted challenges in obtaining reliable definitions from external sources like Wikipedia. These findings contribute to the broader goal of enhancing scientific literacy through accessible information.

Keywords

LLMs, text simplification, LLaMA-2

1. Introduction

We live in an era characterised by an abundance of information available to all, almost instantaneously. However, far from creating a world defined by truth and understanding, our era seems to be accurately defined by misinformation and polarity. Fake news and algorithmically determined “echo-chambers” have helped spread conspiracy and division across the world, with consequences that reverberate far beyond their origins in cyberspace.

For the average person, it’s more difficult than ever to know what information to believe. We all need to be able to understand our world, so scientific literacy is a more important skill than ever.

This paper presents the results of our analysis of Tasks 1, 2 and 3 of the SimpleText track as part of the 15th Conference and Labs of the Evaluation Forum 2024. The main goal of SimpleText is to advance research in the area of automatic simplification of scientific texts [1].

The paper deals with:

- Task 1: What is in (or out)? Selecting passages to include in a simplified summary.
- Task 2: What is unclear? Difficult concept identification and explanation
- Task 3: Rewrite this! Given a query, simplify passages from scientific abstracts.

CLEF 2024: Conference and Labs of the Evaluation Forum, September 09–12, 2024, Grenoble, France

*CLEF 2024 SimpleText Tasks 1-3: Use of LLaMA-2 for Text Simplification
*Corresponding author.
†These authors contributed equally.

rowanmann93@gmail.com (R. Mann); tomislav.mikulandric@gmail.com (T. Mikulandric)

2. Task 1: Experimental Setup

2.1. Data Description

The data provided to us by CLEF consisted of 2 folders, "corpus" and "topics qrels". The corpus includes a new vector database with sentence embedding scores and retains the previously released ElasticSearch Index for field-specific document searches. The ElasticSearch Index allows querying fields such as id, abstract, authors, title, year, and doi from the DBLP dump and is suitable for various applications like passage retrieval, Latent Dirichlet Allocation models, and training Graph Neural Networks. The Vector Database stores each article's id and sentence-embedding vectors from their title and abstract, excluding articles with empty or very short abstracts, supporting longer queries enabled by sentence embedding.

The SimpleText 2024 Task 1 Corpus includes topics defined by articles from The Guardian's tech section (G01 to G20) and Tech Xplore (T01 to T20), with URLs and textual content provided for participant use. Queries associated with each topic, manually verified for relevance, enable retrieval of relevant DBLP passages. This edition introduces new queries for The Guardian articles, generated by ChatGPT 4.0, focusing on specific sub-topics and provided in CSV and JSON formats. The Simpletext 2024 task1 train.qrels file offers quality relevance judgments on a 0-2 scale for abstracts, incorporating data from previous editions and new judgments for topics G01-G15, excluding articles with nearly empty abstracts to ensure consistency with the new vector database.

2.2. Method

We created an ElasticSearch function "query elasticsearch" to query the ElasticSearch database. It took two parameters: query (the search query) and size (the number of results to return, defaulting to 100). The function sent a GET request to the ElasticSearch URL with the specified query and size and returned the search results in JSON format.

```
# Function to query the ElasticSearch
def query_elasticsearch(query, size=100):
    response = requests.get(f"{ES_URL}?q={query}&size={size}", auth
        =('inex', 'qatc2011'))
    if response.status_code == 200:
        return response.json()['hits']['hits']
    else:
        print("Failed to fetch data:", response.status_code)
        return []
```

We used the first five examples from the "simpletext 2024 task1 queries.json" file and went over every index. (Appendix A)

We created a function to calculate how relevant the abstracts retrieved were to our search. The function created a Text Frequency Inverse Document Frequency (TF IDF), for vectorizing the texts, which assessed relevancy of words with regards to our corpus, then calculated the cosine similarity of our vectorised words. This function could then return a relevance score (rel score)

To create the combined score, we calculated word difficulty based on the Flesch Kincaid grade level. The Flesch–Kincaid grade level is one of the formulas used for assessing reading-ease, scores indicate the grade a person would have to be in US education system to understand the text.

```
def flesch_kincaid_grade_level(text):
    # Constants for the formula
    ASL = average_sentence_length(text)
    ASW = average_syllables_per_word(text)


    # Calculating the score
    score = 0.39 * ASL + 11.8 * ASW - 15.59


    # Normalize score to range from 0 to 1
    normalized_score = normalize(score, min_score=0, max_score=25)
       # Adjust max_score as needed


    return normalized_score
```

3. Task 1: Experimental Results

We analysed our success by using elastic search to select passages and calculated scores using FKGL and normalisation. The mean of these scores was close to 0.6 which meant that the texts were more complex than everyday speech and appropriate to be used for the next tasks.

Table 1
Official results for Task 1

	MMR	Precision 10	Precision 20	NDCG 10	NDCG 20	Bpref	MAP
$T1_1$	0.217	0,0233	0,0150	0,0121	0,0106	0,0062	0,0025
$T1_2$	0,5444	0,3733	0,2750	0,2443	0,2183	0,0963	0,0601

4. Task 2: Experimental Setup

4.1. Data Description

The dataset for "Task 2: Identifying and Explaining Difficult Concepts" in the SimpleText Lab is divided into training and validation folders, each containing several tab-separated files. The training folder includes documents.tsv (576 rows, 115 documents), documents users.tsv (145 rows, document and expert IDs), terms.tsv (1,910 rows, terms, difficulty, expert ID), definitions explanations.tsv (1,046 rows, definitions, explanations, expert ID), and definitions generated.tsv (589 rows, automatically generated definitions). The validation folder contains definitions explanations.tsv (960 rows, definitions without explanations), definitions generated.tsv (932 rows, automatically generated definitions), and terms.tsv (680 rows, terms, difficulty). Initial annotations were performed by multiple experts, with a second round of validation by an external expert to identify additional terms and definitions. The dataset will later include test files for the evaluation phase.

The test dataset for "Task 2: Identifying and Explaining Difficult Concepts" in the SimpleText Lab includes several tab-separated files. The documents.tsv file contains 501 rows across 55 documents with columns for document ID, sentence ID, and sentence text. The terms.tsv and definitions explanations.tsv files, available after the evaluation phase, provide annotated sentence IDs, extracted terms, difficulty levels (easy, medium, difficult), user-provided definitions, and explanations. Finally, the definitions generated.tsv file contains 3,816 rows with unique definition IDs and the corresponding definitions to be ranked.

4.2. Method

We created a prompt for LLAMA-2 13B model that asked the LLM to iterate over each of our source sentences and extract three scientific terms from the phrase.

```
prompt_terms="""
    You are a robot that ONLY outputs JSON.
    You reply in JSON format with the field 'terms'.
    You provide ONLY semicolon-separated  list of MAXIMUM 3
        scientific terms of a source sentence ONLY.
    You DO NOT add 'Sure, Here are the scientific terms of your
        sentence:'.
    Example source sentence: In the modern era of automation and
        robotics, \
    autonomous vehicles are currently the focus of academic and
        industrial research.? \
    Example answer: {'terms': 'robotics; autonomous vehicles'}
    Now here is my sentence:
"""
```

We used Regex to help us deal with regular expressions, removing unnecessary content in the outputs. (Appendix B)

The terms were then sorted into three rows, with duplicates removed, one term per row and we prompted Llama to give us a difficulty rating of easy, medium, or difficult for our terms. (Appendix C)

We used wikipedia to return definitions for the difficult terms, with limited success. (Appendix D)

We also asked the LLM to provide an explanation. When creating our prompt for our LLM, we gave it a few examples of correct return phrases, that were taken from the document provided. This was to improve the ability to achieve "few-shot" results. (Appendix E)

We then created a function to remove unnecessary text. (Appendix F)

Finally, we compiled our results in a JSON file, with those terms considered "d" for difficult, generating definitions. (Appendix G)

5. Task 2: Experimental Results

The LLM was successful in generating definitions for our difficult terms, but an issue we encountered was that Wikipedia was unsuccessful in generating definitions for our terms. Therefore, this certainly harms the appropriateness of this method as many of our definitions are missing.

Table 2
Official results for task 2

	Recall overall	Recall avg
Task 2.2	0,0069	0,0040
Task 2.2_1	0,0083	0,0084

6. Task 3: Experimental Setup

6.1. Method

We used LLAMA-2 13B once more, creating a larger context window of 4096. We gave the sentences to the LLM, asking it to simplify the texts. Again, we instructed the LLM to remove fluff words like "Sure!" etc. This gave us an additional column for our simplified sentences, simplified snt. (Appendix H)

Once again, it was important to remove unnecessary text therefore we created a function to carry out this task. (Appendix I)

7. Task 3: Experimental Results

Our results from the LLAMA 13B model for simplifying both the source abstracts and source sentences seems promising. Based on human validation of the simplified phrases, it's seems clear that the meaning has been preserved while reducing the complexity of words and the length of the sentences.

8. Conclusion

Our research has shown LLAMA-2 18B to be an effective model for selecting and simplifying passages from scientific texts. However, we've also highlighted the unreliability of relying on wikipedia for the provision of definitions in this context.

Acknowledgments

We'd like to extend our gratitude to the University of Brest for organising the Blended Intensive Programme (BIP) AI For Humanities. We would also like to thank Liana Ermakova for her teaching of the course and Caroline L'haridon for her support during our stay in Brest.

References

[1] L. Ermakova, et al., Overview of CLEF 2024 SimpleText track on improving access to scientific texts, in: L. Goeuriot, et al. (Eds.), Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Fifteenth International Conference of the CLEF Association (CLEF 2024), Lecture Notes in Computer Science, Springer, 2024.

[2] E. SanJuan, et al., Overview of the CLEF 2024 SimpleText task 1: Retrieve passages to include in a simplified summary, in: G. Faggioli, et al. (Eds.), Working Notes of the Conference and Labs of the Evaluation Forum (CLEF 2024), CEUR Workshop Proceedings, CEUR-WS.org, 2024.

[3] G. M. D. Nunzio, et al., Overview of the CLEF 2024 SimpleText task 2: Identify and explain difficult concepts, in: G. Faggioli, et al. (Eds.), Working Notes of the Conference and Labs of the Evaluation Forum (CLEF 2024), CEUR Workshop Proceedings, CEUR-WS.org, 2024.

[4] L. Ermakova, et al., Overview of the CLEF 2024 SimpleText task 3: Simplify scientific text, in: G. Faggioli, et al. (Eds.), Working Notes of the Conference and Labs of the Evaluation Forum (CLEF 2024), CEUR Workshop Proceedings, CEUR-WS.org, 2024.

[5] J. D'Souza, et al., Overview of the CLEF 2024 SimpleText task 4: Track the state-of-the-art in scholarly publications, in: G. Faggioli, et al. (Eds.), Working Notes of the Conference and Labs of the Evaluation Forum (CLEF 2024), CEUR Workshop Proceedings, CEUR-WS.org, 2024.

.1. Appendix A

```
def main():
    # Read queries from JSON file into a dataframe
    queries = pd.read_json('/content/drive/MyDrive/BIP/SimpleText/
        task 1/task 1/topics_qrels/simpletext_2024_task1_queries.json
        ')
    queries = queries.head(5)


    all_results = []
    for index, query_row in queries.iterrows():
        query_text = query_row['query_text']
        topic_id = query_row['topic_id']
        query_id = query_row['query_id']


        docs = query_elasticsearch(query_text)
        scores = calculate_relevance(docs, query_text)
        results = format_results(docs, scores,topic_id, query_id)
        all_results.extend(results)


    # Output results to a JSON file
    with open('results.json', 'w') as f:
        json.dump(all_results, f, indent=4)
```

.2. Appendix B

```
def extract_value_inside_curly_braces(text):
    # Use regex to find the value inside curly braces
    match = re.search(r"\{([^{}]*)\}", text)


    if match:
        return match.group(1)
    else:
        return None
```

.3. Appendix C

```
prompt_difficulty="""
    You are a robot that rates the difficulty of different terms.
    You provide ONE LEVEL o difficulty for scientific terms.
    You need to consider two words as one term.
    Provide ONE rating for the understablity difficulty of term
       provided.
    There are 3 levels. You need to use: e for easy, m for medium
       and d for difficult.
    Give the rating inside of curly braces like this {e}
    You can reply with ONLY one word.
    Example source: autonomous vehicles
    Example answer: {'m'}
    Now here is my sentence:
"""
```

.4. Appendix D

```
import wikipedia


def get_wikipedia_definition(term):
    try:
        # Fetch Wikipedia summary for the term
        summary = wikipedia.summary(term)
        return summary
    except wikipedia.exceptions.DisambiguationError as e:
        # If there's a disambiguation error, handle it as needed
        return "DisambiguationError: Ambiguous term"
    except wikipedia.exceptions.PageError as e:
        # If the page doesn't exist, handle it as needed
        return "PageError: Term not found"
    except Exception as e:
        # Handle other exceptions
        return str(e)


# Assuming test['difficulty'] contains terms for which you want
   Wikipedia definitions
#test['wiki'] = test['term'].apply(get_wikipedia_definition)
test.loc[test['difficulty'] == 'd', 'wiki'] = test.loc[test['
   difficulty'] == 'd', 'term'].apply(get_wikipedia_definition)
test
```

.5. Appendix E

```
prompt_explanation="""
    You are a robot that explains difficult scientific terms.
    DO NOT add intro like "Sure, I'd be happy to help!"
    Use only once sentance and wrap the sentance in curly braces.
     Dont justify your answers. Dont give information not
        mentioned in the CONTEXT INFORMATION.
    Example source: wireless network environment
    Example answer: {'a system in which devices makes use of Radio
        Frequency connections between nodes in the network a system
        in which devices are connected to a network without the need
        for physical cables or wires'}
    Example source: Bluetooth wireless technology
    Example answer: {'short-range wireless communication technology
        that allows devices to connect and exchange data. It
        facilitates data exchange between devices like smartphones,
        computers, and peripherals such as headphones or medical
        devices. Bluetooth technology eliminates the need for
        physical cables, providing convenience and versatility in
        device connectivity.'}
    Example source: application
    Example answer: {'software program or tool designed to perform
        specific tasks or functions on electronic devices. It can
        range from productivity tools and games to utilities and
        communication platforms on electronic devices such as
        computers, smartphones, or tablets.'}
    Example source: PDA
    Example answer: {'PDA is the acronym for personal digital
        assistant, which is a handheld electronic device designed for
         personal organization, communication, and information access
        . PDAs may include features such as calendars, contact lists,
         and note-taking capabilities, serving as portable tools for
        managing daily tasks. PDA is the acronym for personal digital
```

```
         assistant, which is a handheld electronic device crafted for
         personal organization, communication, and information
        retrieval. PDAs often incorporate features like calendars,
        contact lists, and note-taking capabilities, functioning as
        portable tools for managing daily tasks and staying connected
        . While modern smartphones have largely replaced traditional
        PDAs, the concept influenced the development of contemporary
        mobile devices.'}
    Example source: pilot study
    Example answer: {'a preliminary research investigation conducted
         on a small scale to assess the feasibility, and potential
        challenges of a larger research project. an initial and
        smaller-scale research investigation undertaken to evaluate
        the feasibility, methodology, and potential obstacles of a
        larger research project. It serves as a testing ground to
        refine the study design, identify logistical issues, and
        enhance the overall robustness and effectiveness of the
        planned full-scale research endeavor.'}
    Now here is my ONE sentence explanation:
"""
```

.6. Appendix F

```
def remove_redundant_text(text):
    # Define patterns to search for
    patterns = [
        r'^Hey there!',
        r'^Sure!',
        r'^As a scientific journalist,',
        r'I\'m here to break down a complex study into simple terms
            for you\.',
        r'Here\'s a simplified version of the text',
        r'Let me break it down for you:',
        r'I\'m here to break down a complex study into simple terms
            for you\.',
        r'I\'m here to break down complex scientific concepts into
            simple, easy-to-understand language.',
        r'I\'m here to break down a complex topic into simpler terms
             for you. So, let\'s talk about',
        r'Here is my one sentence explanation of'


    ]
```

.7. Appendix G

```
    # Add definition and explanation if they are not empty
    if row["difficulty"] == "d":
        definition = row.get("definition", None)
        explanation = row.get("explanation", None)
        if definition:
            json_obj["definition"] = definition
        if explanation:
            json_obj["explanation"] = explanation


    return json_obj
```

.8. Appendix H

```
# Example usage
def simplify(snt):
    c = model.create_chat_completion(
        messages=[
            {"role": "system", "content": "You are a scientific
                journalist who popularizes scientific results."},
            {"role": "user", "content": "Simplify the following text
                :\n" + snt}
        ]
    )
    return c['choices'][0]['message']['content'].strip()


def simplify(snt):
  c=model.create_chat_completion(
      messages = [
          {"role": "system", "content": "You are a scientific
             journalist who popularizes scientific results."},
          {
              "role": "user",
              "content": "Simplify the following text:\n"+snt
          }
      ]
  )
  return c['choices'][0]['message']['content'].strip()


simplify("With the ever increasing number of unmanned aerial
   vehicles getting involved in activities in the civilian and
   commercial domain, there is an increased need for autonomy in
   these systems too.")
```

.9. Appendix I

```
def remove_redundant_text(text):
    # Define patterns to search for
    patterns = [
        r'^Hey there!',
        r'^Sure!',
        r'^As a scientific journalist,',
        r'I\'m here to break down a complex study into simple terms
           for you\.',
        r'Here\'s a simplified version of the text',
        r'Let me break it down for you:',
        r'I\'m here to break down a complex study into simple terms
           for you\.',
        r'I\'m here to break down complex scientific concepts into
           simple, easy-to-understand language.',
        r'I\'m here to break down a complex topic into simpler terms
            for you. So, let\'s talk about',
        r'Sure, I\'d be happy to help!',
        r'Here\'s a simplified explanation of',
        r'In other words,',
        r'In simple terms,'


    ]
    # Compile regular expressions
    regex_patterns = [re.compile(pattern) for pattern in patterns]


    # Remove patterns from text
    for pattern in regex_patterns:
        text = re.sub(pattern, '', text).strip()


    return text
```