Professional Portfolio

Jason Heesang Lee

List of Contents

Timeline

GLION INSTITUTE OF HIGHER EDUCATION
2015.08 ~ 2021.02

FOUR SEASONS HOTEL SEOUL
2021.04 ~ 2021.09

Atheneum
2021.09 ~ 2023.02

fast campus × KOSME
이어드림스쿨
YEAR-DREAM SCHOOL
2023.04 ~ 2024.02

Bachelor of Business Administration
Hospitality Management
With Luxury Specialization

People & Culture Coordinator
Apr

Associate
Sep

Sr. Associate
Jun

Team Lead
Aug

My First Hello World!
Mar

My First Kaggle Submit
Apr

Achieved Notebook Expert
Aug

Achieved Notebook Master
Oct

Released DanGam
Dec

2020 2021 2022 2023

Previous Jobs & Career Transition

After graduating from Glion Institute of Higher Education, a Hotel School located in Switzerland, I joined at Four Seasons Hotel Seoul as a People & Culture (P&C) Coordinator.

I was mostly responsible for the Human Resources tasks :
General Administration, Enrollment & Termination Interviews,
Organizing Employee Events and Job Fair.

I joined Atheneum Partners after being scouted by a director I met previously.

I was responsible for about 15 projects per month as a Project Manager.
Exceeded target every month by around 250%, best achievement being 425%.
Thankfully, I was able to be promoted as a Team Lead,
which was the fastest promotion in all global Atheneum offices.

Main Industries Covered : Artificial Intelligence, Semiconductor and Digital Transformation.

After being being responsible for number of Artificial-Intelligence-related projects at Atheneum Partners,
I got more interested in the developing AI itself than developing business strategies using AI.

Coincidently, I happened to bump into a news article that Korea SMEs and Startups Agency is collaborating with Fast Campus,
one of the top computer education corporate in Korea, to launch a government-funded AI education course : Year-Dream School.

At Year-Dream School, I learned the basics required for Machine Learning and Deep Learning.
Also, I was introduced to Kaggle, which I later found my enthusiasm in.

Competitions

AI CONNECT

🥈 **Generating Seamless En-Ko Translation**

- **Date :** 2023-10-27 ~ 2023-11-08
- **Type of Data :** NLP / LLM / Machine Translation
- **Rank :** 2nd
- **Used Models :** Mistral & LLaMa2 based pretrained models
- **Focus :** Utilizing LLM models.

AIFactory

- **Predicting Particular Matter Pollution Degree**
  - **Date :** 2023-04-24 ~ 2023-05-11
  - **Type of Data :** Tabular / Time-Series / Regression
  - **Rank :** Unranked
  - **Used Models :** None (EDA only)
  - **Focus :** Getting Familiar with Pandas

DACON

- **Judicial Precedent Prediction**
  - **Date :** 2023-06-05 ~ 2023-07-03
  - **Type :** NLP / Classification
  - **Rank :** Public : 15% / Private 18%
  - **Used Models :** Sentence-BERT / Legal-BERT
  - **Focus :** Getting familiar with Text data
- **Sound Emotion Recognition**
  - **Date :** 2023-05-07 ~ 2023-06-05
  - **Type :** Acoustic / Emotion Recognition / Classification
  - **Rank :** Public : 43% / Private 40%
  - **Used Models :** Librosa / RandomForest / DecisionTree / XGBoost / LightGBM
  - **Focus :** Getting familiar with Acoustic data

kaggle

🥈 **CAFA 5 Protein Function Prediction**

- **Date :** 2023-04-18 ~ 2023-12-21
- **Type of Data :** Tabular / Biology
- **Rank :** Public 4% / Private 4% / Top 63rd
- **Used Models : ProtBERT, Prot-T5, ESM2**.
- **Focus :** Solving given problem with Protein Language Models.

- **Kaggle – LLM Science Exam**
  - **Date :** 2023-07-12 ~ 2023-10-11
  - **Type of Data :** NLP / LLM / Question Answering
  - **Rank :** Public : 15% / Private 15%
  - **Used Models :** T5, DeBERTa, LLaMA2, Platypus2, Alpaca
  - **Focus :** Getting familiar with Large Language Models.
- **CommonLit – Evaluate Student Summaries**
  - **Date :** 2023-07-13 ~ 2023-10-12
  - **Type of Data :** NLP / LLM / Text Summary Evaluation
  - **Rank :** Public : 7% / Private 30%
  - **Used Models :** MobileBERT, DeBERTa, Numerous BERT family models
  - **Focus :** Transformer-based Deep Learning model compression
- **ICR – Identifying Age-Related Conditions**
  - **Date :** 2023-05-12 ~ 2023-08-11
  - **Type :** Tabular / Classification
  - **Rank :** Public : 7% / Private 48%
  - **Used Models :** Rule-Based / TabPFN / XGBoost / LightGBM
  - **Focus :** Finding relations between each column and the meta data

DANGAM

Word-Level Sentiment Analysis

Purpose

The final project at YearDream School with Avocadoland required us to analyze the daily records of users, find keywords (nouns) in each prompt, and then separate them into positive and negative emotions.

The baseline code suggested using TweetNLP for sentiment extraction.
However, we realized that it was only segmenting the sentences, not the words.
The company have told us that word-level sentiment analysis is not a must, if there is no such model that can perform the task, we can simply convert to sentence-level sentiment analysis.

It could have been a simple task if we gave up on word-level sentiment analysis.
But I felt more inclined to challenge myself.
Consequently, DanGam was created.

DanGam Overview

Compared with other existing research

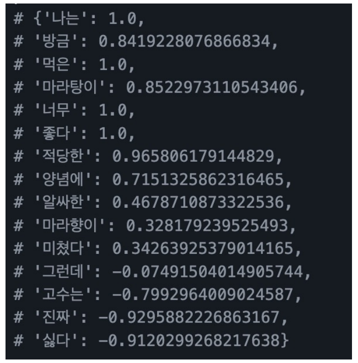
- **TweetNLP**
  - TweetNLP supports multiple languages including Korean and works well at the sentence level. However, it does not support word-level sentiment analysis.
- **HuggingFace Text Classification Models**
  - Similar to TweetNLP, they support sentence-level sentiment analysis, but not word-level.
- Word-Level Sentiment Analysis with Reinforcement Learning
  - This research is similar to DanGam, but DanGam offers sentiment analysis for all the words in each sentence.
- Word-Level Contextual Sentiment Analysis with Interpretability
  - The result of this research research is similar to those of DanGam.
    However, DanGam is an inference tool, in contrast to a Deep-Learning Model that requires training..

How does it work

- DanGam takes a sentence as an input and identifies the overall emotion (positive, negative, neutral) as well as specific emotions (happy, sad, rage, calm, etc.) within that sentence.
- DanGam calculates the cosine similarity between the sentence and the emotion, and between the sentence and the specific emotion.
- It combines sentence embedding, emotion embedding and specific emotion embedding with weights based on the calculated similarities.
- Then it calculates the cosine similarity between word embedding and the combined embedding.
- If the similarity is high, it suggests that the word has an emotion similar to that of the combined embedding.

Output Example

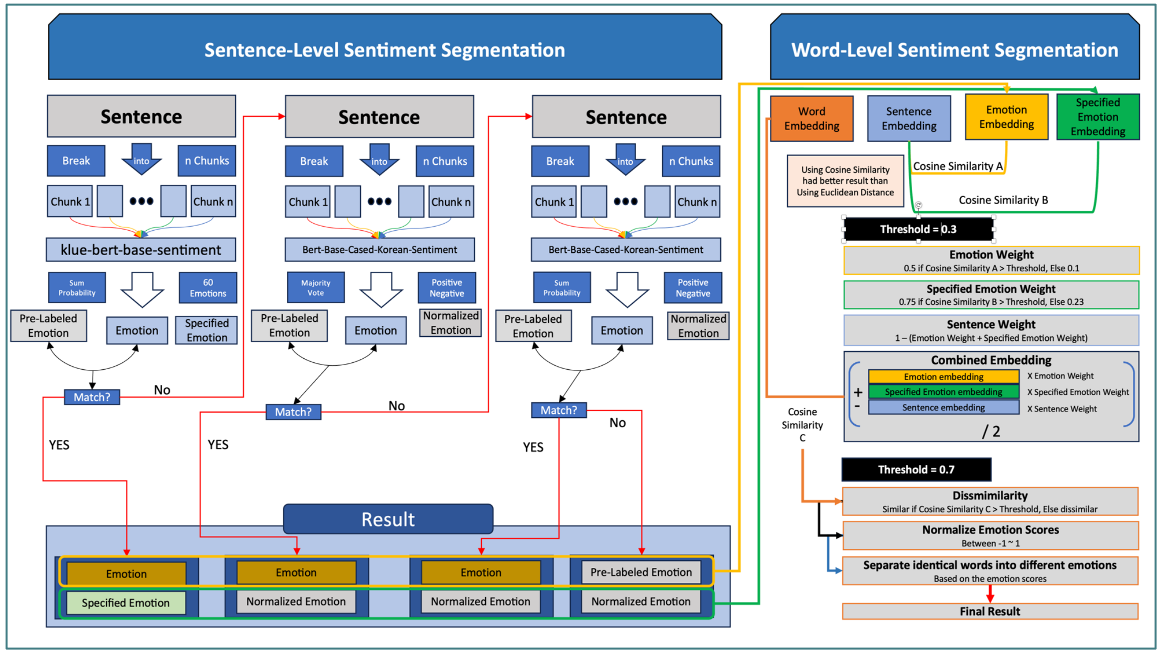
- Example Sentence :
  “나는 방금 먹은 마라탕이 너무 좋다.
  적당한 양념에 알싸한 마라향이 미쳤다.
  그런데 고수는 진짜 싫다”
- The resulted output is in a range as below.
  Positive Emotion (1) ~ Negative Emotion (-1)



```
# {'나는': 1.0,
# '방금': 0.8419228076866834,
# '먹은': 1.0,
# '마라탕이': 0.8522973110543406,
# '너무': 1.0,
# '좋다': 1.0,
# '적당한': 0.965806179144829,
# '양념에': 0.7151325862316465,
# '알싸한': 0.4678710873322536,
# '마라향이': 0.328179239525493,
# '미쳤다': 0.34263925379014165,
# '그런데': -0.07491504014905744,
# '고수는': -0.7992964009024587,
# '진짜': -0.9295882226863167,
# '싫다': -0.9120299268217638}
```

Logic and Key Features

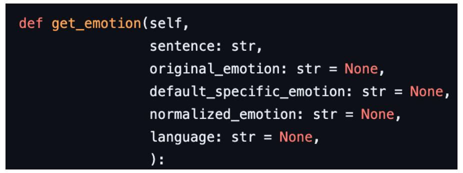
Visualized Diagram



Key Features

get_emotion

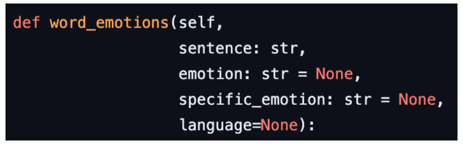
- Determines the overall emotion of a given sentence by analyzing it in chunks. Considers both the general and specific emotions to enhance accuracy.
- It returns strings of overall emotion and specific emotion of the sentence.



```
def get_emotion(self,
                sentence: str,
                original_emotion: str = None,
                default_specific_emotion: str = None,
                normalized_emotion: str = None,
                language: str = None,
                ):
```

word_emotions

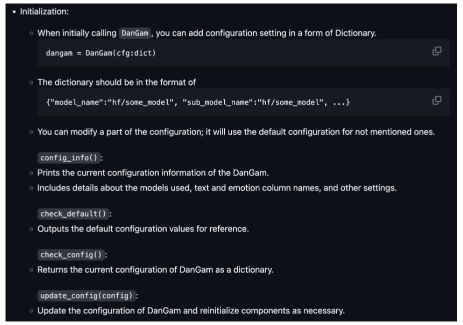
- Segments a sentence and assigns emotions to each word based on the overall sentence emotion and specific emotion.
- It returns a dictionary mapping each word in the sentence to its assigned emotion.



```
def word_emotions(self,
                  sentence: str,
                  emotion: str = None,
                  specific_emotion: str = None,
                  language=None):
```

DanGamConfig

- DanGam offers extensive customization options.
- Users can adjust settings according to their requirements.



- Initialization:
  - When initially calling `DanGam`, you can add configuration setting in a form of Dictionary.

    ```
    dangam = DanGam(cfg:dict)
    ```

  - The dictionary should be in the format of

    ```
    {"model_name":"hf/some_model", "sub_model_name":"hf/some_model", ...}
    ```

  - You can modify a part of the configuration; it will use the default configuration for not mentioned ones.

  `config_info()`:
  - Prints the current configuration information of the DanGam.
  - Includes details about the models used, text and emotion column names, and other settings.

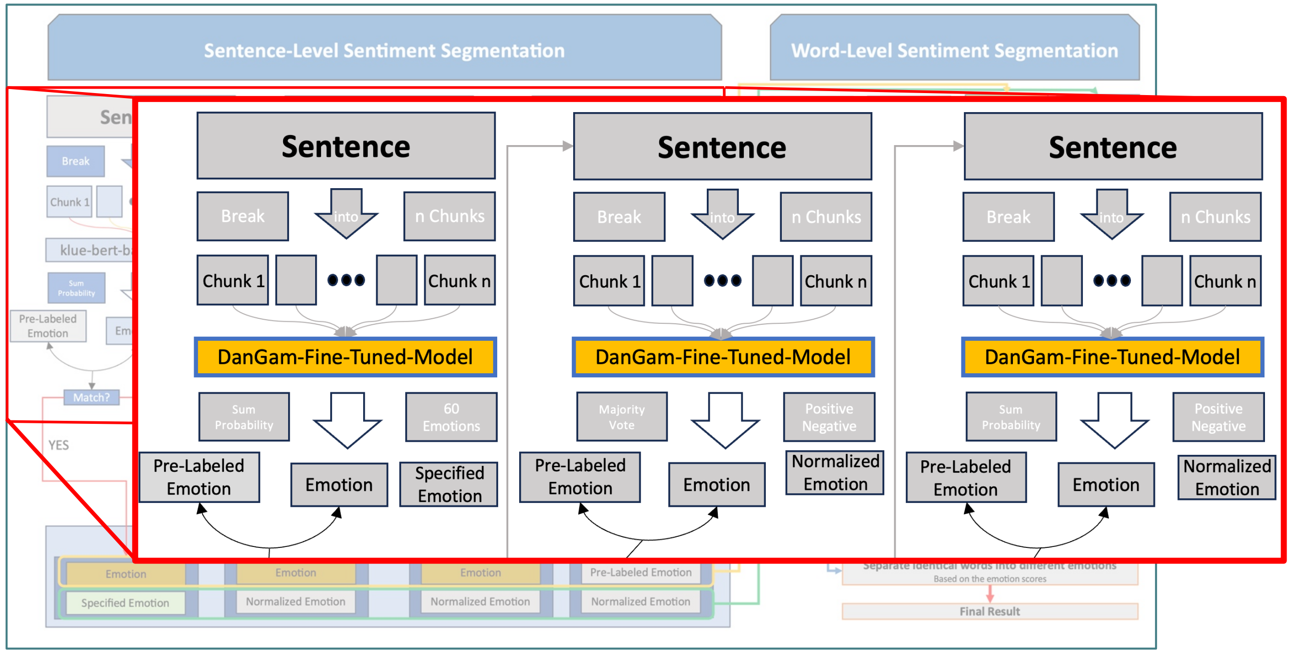
  `check_default()`:
  - Outputs the default configuration values for reference.

  `check_config()`:
  - Returns the current configuration of DanGam as a dictionary.

  `update_config(config)`:
  - Update the configuration of DanGam and reinitialize components as necessary.

Reflections

What could be improved



- **Develop Fine-Tuned model explicitly for DanGam.**
  - DanGam currently utilizes fine-tuned models retrieved from HuggingFace. These will be replaced once the fine-tuned model is developed.
- **Implementing reliable evaluation metric.**
  - Currently, the only way to evaluate DanGam is Human Evaluation. To enhance the effectiveness of input resource, reliable evaluation metric will be introduced along with DanGam.
- **Applying feedback from peers.**
  - There should be some unrealized errors or misuse of logics. After a careful review of the feedback, the logic will be modified accordingly.

COMMONLIT

Evaluate Student Summaries

Competition Review

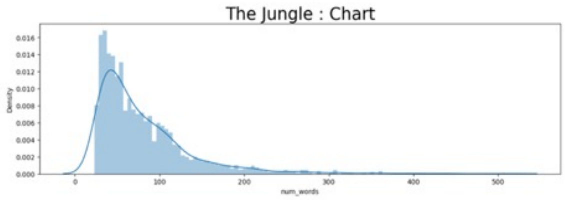
Purpose

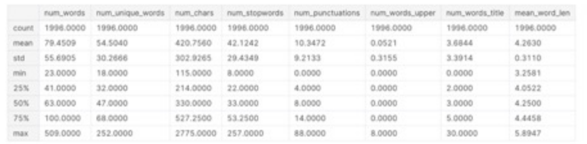
I participated in the CommonLit – Evaluating Student Summary Competition, where the objective was to develop an automated system for evaluating the quality of summaries written by students in grades 3-12.

This involved building a Deep Learning model capable of assessing how effectively a student captures the main idea and details of a prompt text, along with the clarity, precision, and fluency of their written summary. A comprehensive dataset of real student summaries were given to the participants to train and refine the model.

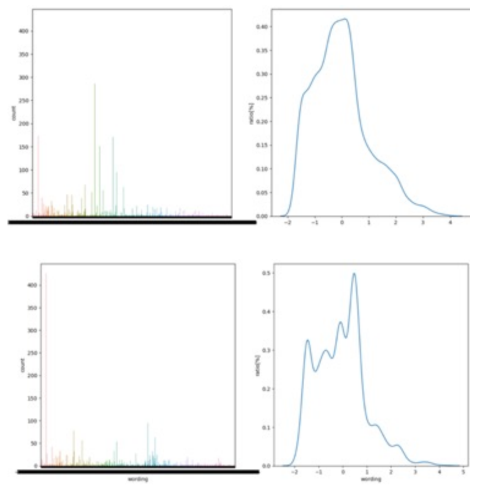
Data Overview

Exploratory Data Analysis





	num_words	num_unique_words	num_chars	num_stopwords	num_punctuations	num_words_upper	num_words_title	mean_word_len
count	1996.0000	1996.0000	1996.0000	1996.0000	1996.0000	1996.0000	1996.0000	1996.0000
mean	79.4509	54.5040	420.7560	42.1242	10.3472	0.0521	3.6844	4.2630
std	55.6905	30.2666	302.9265	29.4349	9.2133	0.3155	3.3914	0.3110
min	23.0000	18.0000	115.0000	8.0000	0.0000	0.0000	0.0000	3.2581
25%	41.0000	32.0000	214.0000	22.0000	4.0000	0.0000	2.0000	4.0522
50%	63.0000	47.0000	330.0000	33.0000	8.0000	0.0000	3.0000	4.2500
75%	100.0000	68.0000	527.2500	53.2500	14.0000	0.0000	5.0000	4.4458
max	509.0000	252.0000	2775.0000	257.0000	88.0000	8.0000	30.0000	5.8947



social structure
slave
structure
vizier
priest
god
pharoah
pharaoh
social classe
pyramid
government
farmer
people
noble
pharoh
workpower
bottom
top
war
read

```
Train content description:        Train wording description:
count    7165.000000              count    7165.000000
mean       -0.014853              mean       -0.063072
std         1.043569              std         1.036048
min        -1.729859              min        -1.962614
25%        -0.799545              25%        -0.872720
50%        -0.093814              50%        -0.081769
75%         0.499660              75%         0.503833
max         3.900326              max         4.310693
Name: content, dtype: float64     Name: wording, dtype: float64
```

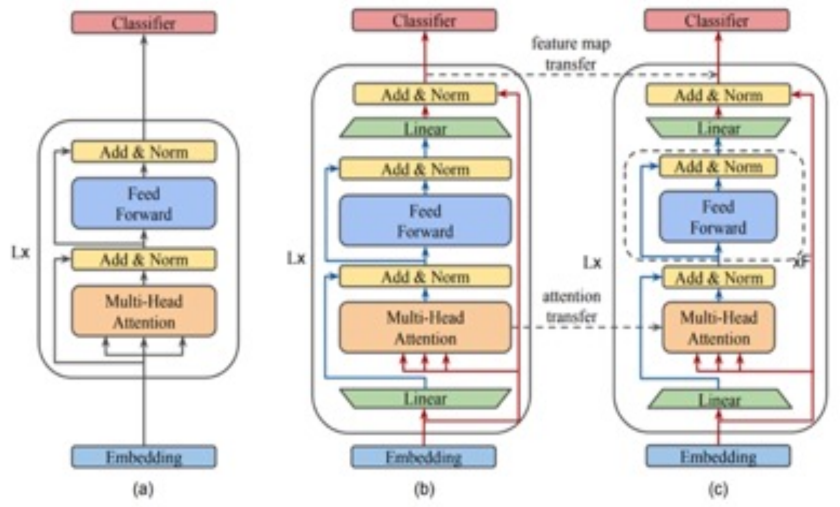
Preprocessing & Spellcheck

- Cleansing the text is a must to run the NLP model effectively
- Many misspelled words were in text:
  ex) "Pharaoh" -> "Pharoh" or "Pharoah".
- Accuracy : SymSpellPy > PySpellChecker > TextBlob > AutoCorrect
- Speed : AutoCorrect > PySpellChecker > SymSpellPy > TextBlob

Tool	Detected	Correctly Replaced	Misjudged	Time
Grammarly	16 words	16 words	Less 1 word	-
TextBlob	20 words	14 words	2 words 2 names	4 sec
PySpellChecker	18 words	16 words	2 words	1 sec
SymSpellPy	18 words	17 words	1 name	2 sec
AutoCorrect	15 words	12 words	2 words 1 name	0 sec

Model Application

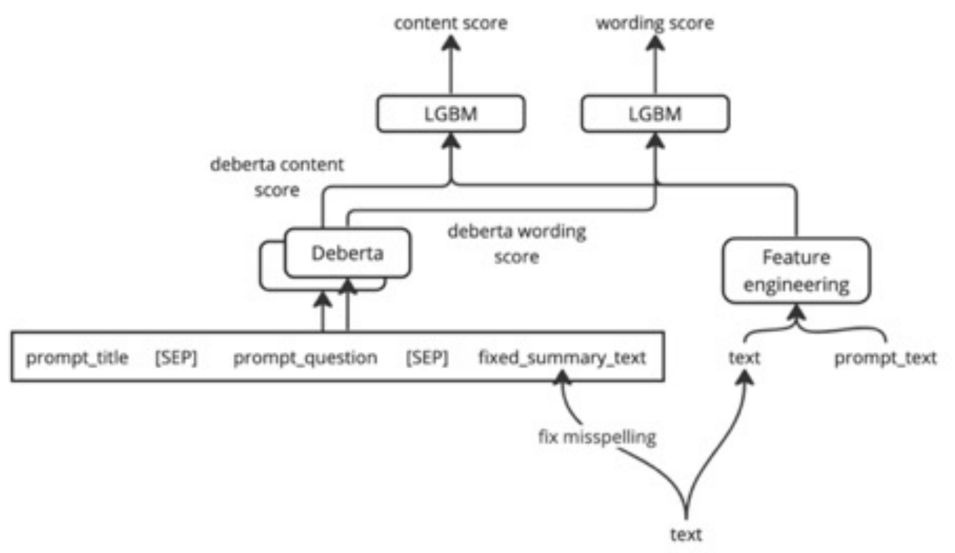
Solving a Real-World Problem with MobileBERT



MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices (Sun et al., 2020)

- As the purpose of this competition is to enhance the teaching and scoring experience of the teachers, I believed making this model lightweight is the key.
- By implementing MobileBERT, the model would run on individual mobile devices, making it accessible to teachers in environments where computer access is difficult.

LGBM – Post-Processing

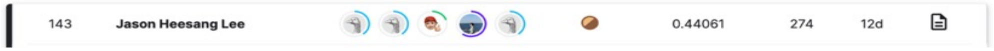


- After getting the result with NLP Deep Learning models, I used LightGBM to improve the score by feature engineering, as introduced in this notebook.
- The original author of the notebook found the hyperparameters through experimentation but added Optuna, a hyperparameter optimization module, as a last step, thinking that it would find better-performing hyperparameters.

Result & Reflection

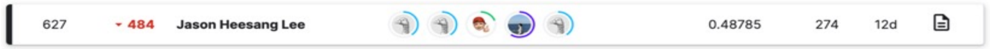
Result

- We were placed at the top 143 in the Public Leaderboard, so we believed that we could make it to the Bronze medal when the Private Leaderboard was published.
- Therefore, we tried our best until the very end to make minor changes to our existing solution.



143	Jason Heesang Lee		0.44061	274	12d	

- When the Private Leaderboard was published, we were pretty shocked with the result.



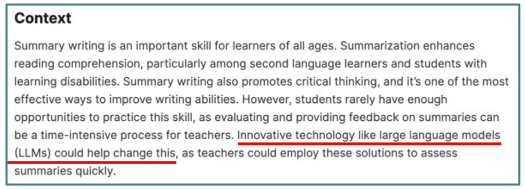
627	▾ 484	Jason Heesang Lee	0.48785	274	12d	

What could be improved

- **First** : The proportion of the Public Data.
  The organizer stated that the test data provided to the competitors is only 13% of the whole test data.
  Data augmentation could have been done as the 1st place team have.
- **Second** : Using SymSpellPy for spellchecking.
  Most competitors used PySpellCheck and AutoCorrect.
- **Third** : Focused too much on MobileBERT.
  If it was to aim for higher rank, MobileBERT should have been neglected earlier.
  However, as it was done on purpose, I have no regret.
- **Fourth** : Need more and better modeling techniques.
  It would have been better if we could also try T5 and BART, as no one else was trying that measure.
  Only if we had a better modeling technique, we could have also tried these two models.

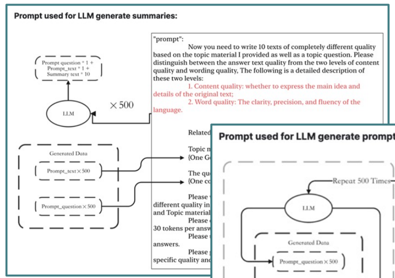
1st Place Solution

- Surprisingly or not, the first-place holder turned out to be using the same baseline that we were using.
- The differences between our solution and the first-place solution are:
  1. They have barely modified the logic of the model.
  2. They not only have used the provided four prompts and generated even more by effectively using Large Language Model.
  3. They have excluded using LightGBM and solved the problem solely with DeBERTa-v3-Large.
- Per their discussion, they have spent more time generating new prompts and summaries with LLM than tuning the DeBERTa model.
- I believe this was quite a smart move, as the competition overview – context has already mentioned utilizing LLM.
  - Even though I don't think this is not the way the organizer expected the LLM to be used, they were the only team that actually used LLM for this competition,



Context

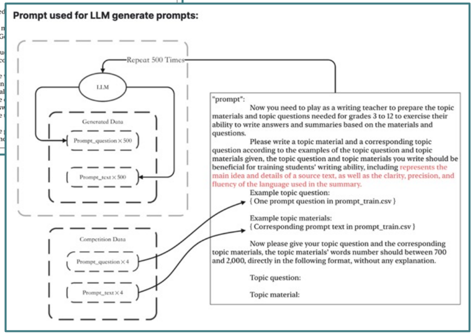
Summary writing is an important skill for learners of all ages. Summarization enhances reading comprehension, particularly among second language learners and students with learning disabilities. Summary writing also promotes critical thinking, and it's one of the most effective ways to improve writing abilities. However, students rarely have enough opportunities to practice this skill, as evaluating and providing feedback on summaries can be a time-intensive process for teachers. Innovative technology like large language models (LLMs) could help change this, as teachers could employ these solutions to assess summaries quickly.



Prompt used for LLM generate summaries:
Prompt question * 1 + Prompt_text * 1 + Summary text * 10
LLM
× 500
Generated Data
Prompt_text × 500
Prompt_question × 500

"prompt":
Now you need to write 10 texts of completely different quality based on the topic material I provided as well as a topic question. Please distinguish between the answer text quality from the two levels of content quality and wording quality, The following is a detailed description of these two levels:
1. Content quality: whether to express the main idea and details of the original text;
2. Word quality: The clarity, precision, and fluency of the language.

Related
Topic m
{One G
The qu
{One c
Please
different quality in
and Topic material
Please
30 tokens per answ
Please
answers.
Please
specific quality and



"prompt":
Now you need to play as a writing teacher to prepare the topic materials and topic questions needed for grades 3 to 12 to exercise their ability to write answers and summaries based on the materials and questions.
Please write a topic material and a corresponding topic question according to the examples of the topic question and topic materials given, the topic question and topic materials you write should be beneficial for training students' writing ability, including represents the main idea and details of a source text, as well as the clarity, precision, and fluency of the language used in the summary.
Example topic question:
{ One prompt question in prompt_train.csv }
Example topic materials:
{ Corresponding prompt text in prompt_train.csv }
Now please give your topic question and the corresponding topic materials, the topic materials' words number should between 700 and 2,000, directly in the following format, without any explanation.
Topic question:
Topic material: