# CAS2105 Homework 6: Mini AI Pipeline Project 🤗

**Chaeyeon Kim(2021127027)**

## 1 Introduction

The task of this project is binary hate speech classification in Korean text. Given a short Korean sentence, the system predicts whether the sentence contains hate speech or not. This task is interesting because hate speech detection is an important real-world problem for online content moderation and social media analysis. Compared to English, Korean hate speech detection has fewer publicly available resources, which makes the task more challenging. By designing a simple AI pipeline for this task, this project aims to understand how different approaches, from naïve rules to pre-trained language models, handle linguistic nuance and contextual meaning in Korean text.

- Choose a small, concrete problem solvable with an AI pipeline (e.g., text classification, retrieval, simple QA, image classification).
- Choose or collect a small dataset (e.g., from `datasets`).
- Implement a simple *naïve baseline* (e.g., rule-based or heuristic).
- Build an improved pipeline using existing pre-trained models.
- Evaluate both approaches using appropriate metrics.
- Reflect on what worked, what failed, and what you learned.

The emphasis is on the *process* of AI work: problem definition, pipeline design, evaluation, iteration, and writing. Your problem should be small enough to run comfortably on a single GPU (e.g., RTX 3090) or CPU.

**Your tasks.** Please replace all the sections to complete your report, starting from adding your project title, your name, and student ID above! Feel free to reorganize the sections. Then, submit a **public** github repository link that includes your code (including Jupyter notebook with results) and report **in PDF**, via LearnUs.

You can use tables and figures, and cite references using `\citep` ([1]) or `\citet` (Wei et al. [1]).

## 2 Task Definition

- **Task description:** The task is binary text classification: given a Korean sentence, the system predicts whether it contains hate speech or not.
- **Motivation:** Hate speech detection is a practically important NLP task for content moderation and social media analysis. Korean-language resources for hate speech detection are more limited than English ones, making this task both challenging and meaningful.
- **Success criteria:** The system is considered successful if it correctly classifies unseen text and achieves higher evaluation scores than the naïve baseline, measured using accuracy, weighted F1-score, macro F1-score, and micro F1-score.

# 3 Methods

This section includes both the naïve baseline and the improved AI pipeline.

## 3.1 Naïve Baseline

Implement a simple method that does not rely on heavy models. Examples include:

- Keyword-based text classification,
- Simple color/shape heuristics for image tasks,
- String-overlap–based retrieval.

In your report, explain:

- How the baseline works,
- Why it is considered naïve,
- Expected failure cases.

### Your Baseline (TODO)

- **Method description:** The naïve baseline uses a keyword-based rule system. A predefined list of offensive or hate-related keywords is created, and a sentence is classified as hate speech if it contains any of these keywords.
- **Why naïve:** This method ignores context, syntax, and semantic meaning. It treats all keyword occurrences equally and cannot distinguish between quoting, negation, or sarcastic usage.
- **Likely failure modes:** False positives when hate-related words appear in neutral or explanatory contexts

  False negatives when hate speech is expressed indirectly without explicit keywords

## 3.2 AI Pipeline

Design a small pipeline using one or more pre-trained models. Examples include:

- **Text:** embedding encoder + classifier, or a small transformer model,
- **Retrieval:** embedding model + nearest-neighbor search,
- **Vision:** pre-trained classifier (e.g., ViT-tiny).

A typical pipeline contains:

1. Preprocessing,
2. Embedding or representation,
3. Decision/ranking component,
4. Optional post-processing.

Fine-tuning large models is not required; inference-only usage is sufficient.

### Your Pipeline (TODO)

- **Models used:** A pre-trained Korean Transformer sequence classification model (`monologg/koelectra-base-v3-discriminator`).

- **Pipeline stages:**

  (1) data loading of the `train/validation/test` splits from Hugging Face, (2) binary label processing and tokenization with truncation and padding to a fixed length (`max_length=64`), (3) fine-tuning and inference using a pre-trained Korean Transformer sequence classification model (`monologg/koelectra-base-v3-discriminator`), and (4) evaluation on the test split using accuracy and F1-score variants (macro, micro, and weighted).

- **Design choices and justification:** The model `monologg/koelectra-base-v3-discriminator` was chosen because it is a Korean-specific Transformer model pre-trained on large-scale Korean text. Compared to keyword-based baselines, ELECTRA-style models capture contextual meaning and linguistic nuance through self-attention, which is essential for detecting implicit or context-dependent hate speech.

# 4 Experiments

## 4.1 Datasets

- **Source::** Hugging Face dataset `jeanlee/kmhas_korean_hate_speech`.
- **Total examples::** The original dataset consists of approximately 79k training samples, 8.8k validation samples, and 21.9k test samples. Due to computational constraints, we randomly sampled 15k/1.5k/3k instances for train/validation/test, respectively.`train`, `validation`, and `test` splits, which were loaded separately.
- **Splits**:The dataset provides predefined `train`, `validation`, and `test` splits, which were loaded separately.
- **Preprocessing Steps**: Text inputs are tokenized using the chosen model's tokenizer, and labels are mapped to a binary format (hate vs. non-hate). Token length is controlled via a maximum sequence length setting, 64 in this case.

## 4.2 Metrics

**Metrics.** I evaluate both methods using Accuracy, F1-score variants (macro F1, micro F1, and weighted F1), Macro-averaged precision and recall. Although only two metrics are reported in the table below, all of the metrics mentioned above were evaluated in the actual code implementation.

Accuracy measures the overall proportion of correctly classified instances and provides a general view of model performance. However, hate speech datasets are often imbalanced, so accuracy alone can be misleading.

F1-score, which combines precision and recall, is more informative for hate speech detection. It balances false positives and false negatives, making it suitable for evaluating how well the model detects hateful content without over-predicting it.

## 4.3 Results

Report:

- Metric values for baseline vs. pipeline,
- A results table,
- At least three qualitative examples.

| Method | F1-macro | Accuracy |
|---|---|---|
| Baseline | 0.4773 | 0.6067 |
| AI Pipeline | 0.8889 | 0.8896 |

### 4.4 Qualitative Analysis: Cases Where Methods Differ

**Example Case 1** **Text:** "쪽바리니들은 이미 우리 발아래에 있다 그걸 이제 알았냐"
**Baseline:** Non-hate
**Model:** Hate

**Discussion:** The baseline model fails to capture implicit ethnic hostility, while the proposed model correctly recognizes a derogatory group-based expression.

**Example Case 2** **Text:** "사나는 건드리지 마라!!!! 시ㅂ ㅜ랄 넘들아!!!"
**Baseline:** Non-hate
**Model:** Hate

**Discussion:** The baseline struggles with aggressive informal profanity, whereas the model successfully captures hostile intent.

**Example Case 3** **Text:** "위에화 짱깨 소속 연습생들 다 빼자"
**Baseline:** Non-hate
**Model:** Hate

**Discussion:** The baseline overlooks a subtle ethnic slur, while the model detects discriminatory and exclusionary intent.

If you want to visualize results with any other than tables, refer to below links

- Matplotlib official tutorial: Introduction to `pyplot`
- Matplotlib example gallery (many bar/line/scatter plots with source code)
- Kaggle notebook: Matplotlib tutorial for beginners (interactive code)

## 5 Reflection and Limitations

Write approximately 6–10 sentences reflecting on:

- What worked better than expected,
- What failed or was difficult,
- How well your metric captured "quality",
- What you would try next with more time or compute.

### Your Reflection (TODO)

**Reflection and limitations.** Through this project, it became clear that the Transformer-based AI pipeline was better at capturing implicit and context-dependent hate speech than the keyword-based baseline. Even in the absence of explicit slurs, the model was often able to infer hateful intent from contextual cues, which was particularly encouraging. This difference was more clearly reflected by the macro F1-score, which highlights performance on the minority hate class rather than being dominated by the majority class. However, comparing the two approaches was not always straightforward, as the baseline could achieve relatively high accuracy while still missing subtle forms of hate speech. Some errors were also difficult to interpret, since certain expressions were ambiguous even from a human perspective. With more time and computational resources,

future work could explore larger models, richer qualitative evaluation methods, and the use of larger datasets.

## References

[1] Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations* (*ICLR*), 2022. URL https://openreview.net/forum?id=gEZrGCozdqR.