Korean ”Airbnb” Script Deobfuscation Using Sequence-to-Sequence Models.

Abdul Rafay[1], Mehmet Bora Sarioglu[2]

Boston University, Boston, USA[1], Boston University, Boston, USA[2], CS505 : Intro. NLP

Introduction

Korean online communities have developed a unique form of phonetic text coding to communicate candidly while maintaining a degree of privacy. By subtly altering word sounds, users obscure meaning in ways that are intuitive for native speakers but challenging for automated systems and non-Korean readers. This coded language appears frequently in social media, forums, and online reviews.

This project aims to systematically decode such phonetically transformed Korean text. We will construct a dataset of coded and standard Korean text, train a model to recognize transformation patterns, and deploy a real-time decoding tool. By bridging this linguistic gap, our tool enhances accessibility and understanding of modern Korean online discourse.

About Text Obfuscation

Text obfuscation—the deliberate alteration of text to make it difficult for automated systems to process while remaining comprehensible to humans—poses significant challenges across multiple domains. Although we are interested in specific case of Korean ”airbnb script”, for better understanding of the concept, we can try to understand it through the context of English first. Common examples include replacing characters with visually similar ones (e.g., ”noon” with ”nouan”), substituting words with phonetically similar alternatives (e.g., ”meet me later” → ”meat mi letter”), or other transformations that preserve human readability while confusing automated systems. These obfuscation techniques are frequently employed to 1) Evade content moderation systems, 2) Bypass plagiarism detection, 3) Circumvent keyword filtering, and 4) Avoid automated content analysis.

Korean Wiki Text by Korpora

We used the **KoWiki** dataset as our primary Korean text source. It consists of the headers and text of Wikipedia in Korean. The size is as follows:

Dataset	Lines	Size
Train	26,794,425	1.7G
Dev	130,419	7.7M
Test	134,340	8.4M

Table 1: Dataset Information

We used it to create the encoded dataset, for which we applied phonetic transformations, forming a parallel corpus of: 1) Standard Korean text, and 2) Corresponding encoded text. The data set serves as input (encoded text) and output (decoded Korean text) for our model.

Methods

How can Koreans understand encoded text, without non native speaker knowing? Answer to this was the pronunciation. Koreans automatically pronounce those encoded text in their head which happens to sound similar to regular phrases. From this realization, we tried to look for ways to convert encoded text into phonetic representations that could be easily parsed or trained.

g2pK Module

Hangul, the main script for Korean, is phonetic, but the pronunciation rules are notoriously complicated. That's why we employed the g2pK (Grapheme-to-Phoneme) library specialized for Korean to convert both original and obfuscated texts into phonetic representations that matches the actual sound of the phrases. This conversion maps visually distinct but phonetically similar characters and character sequences to the nearly similar phonetic representation, providing our model with a more consistent input space.

```
Original:  어제는 날씨가 맑았는데...
Obfuscated:  었제눈 냘쒸카 맑앗눈뎨...
g2pK conversion:  얻쩨눈 냘쒸카 말간눈뎨...
```

Figure 1: Example of g2pK Conversion.

So, this would convert obfuscated text to a phonetic representation that captures their sound patterns, regardless of the specific characters used to create those sounds, making it more similar to original text.

Character-Level Tokenization

Rather than using subword or word-level tokenization, we implement a character-level tokenizer that treats each character as a distinct token. Our tokenizer includes special tokens for padding (¡pad¿), unknown characters (¡unk¿), sequence start (¡s¿), and sequence end (¡/s¿).

Training Details

We trained our model using cross-entropy loss with the following hyperparameters:

- Learning rate: 3e-5
- Batch size: 16 per device
- Weight decay: 0.01
- Training epochs: 10
- FP16 precision (when GPU is available)

Training was done on shared cloud computing cluster, and took 14 hours to complete.

Model Details

We used a BART-based (Lewis et al., 2020) sequence-to-sequence architecture with the following specifications:

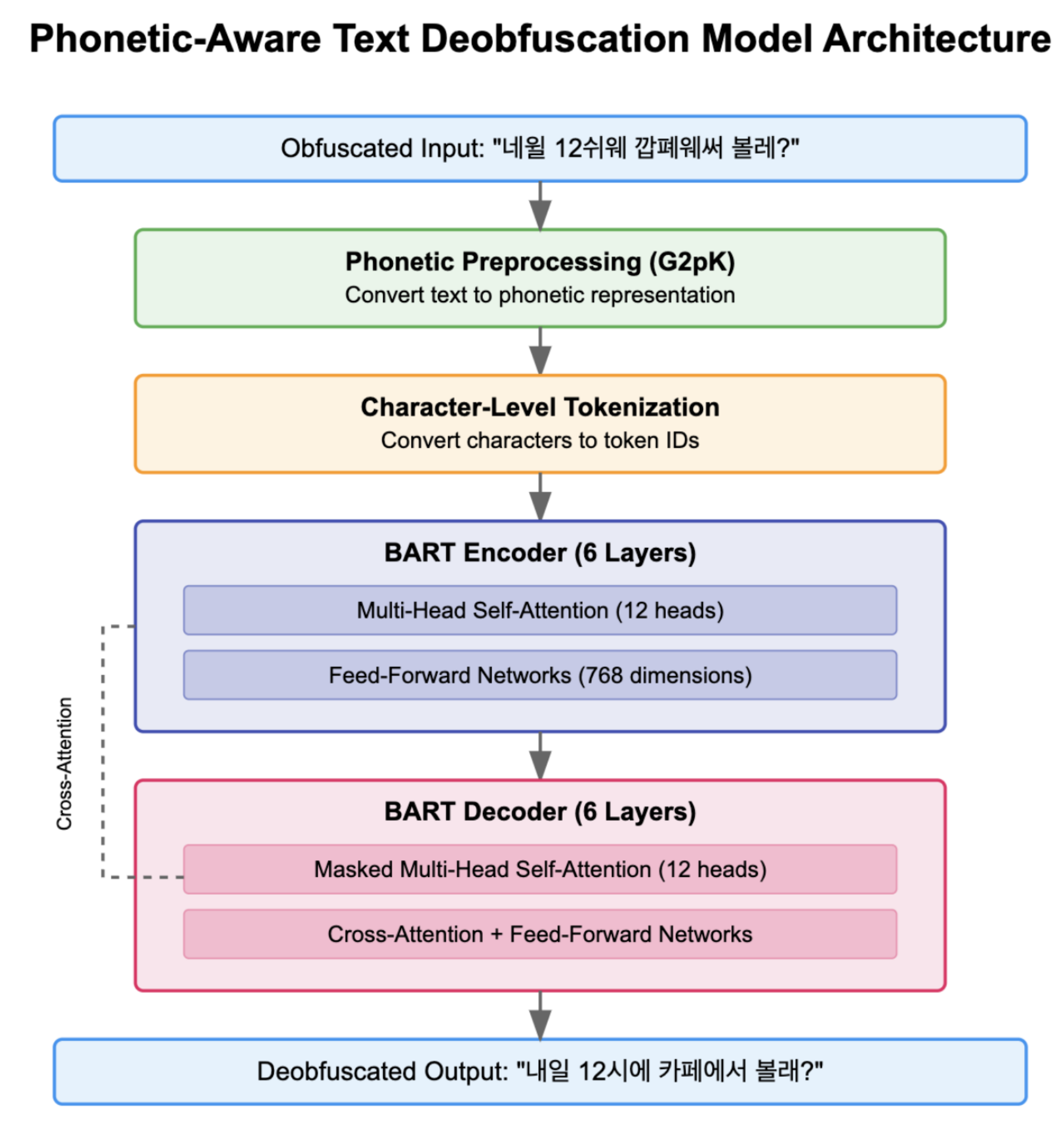
Phonetic-Aware Text Deobfuscation Model Architecture
Obfuscated Input: "네윌 12쉬웨 깝폐웨써 볼레?"
Phonetic Preprocessing (G2pK)
Convert text to phonetic representation
Character-Level Tokenization
Convert characters to token IDs
BART Encoder (6 Layers)
Multi-Head Self-Attention (12 heads)
Feed-Forward Networks (768 dimensions)
Cross-Attention
BART Decoder (6 Layers)
Masked Multi-Head Self-Attention (12 heads)
Cross-Attention + Feed-Forward Networks
Deobfuscated Output: "내일 12시에 카페에서 볼래?"

Figure 2: Model Architecture Diagram

- Vocabulary size based on unique characters in our dataset
- 6 encoder layers and 6 decoder layers
- 768-dimensional hidden representations
- 12 attention heads in both encoder and decoder
- Maximum position embeddings of 1024 tokens

The model inputs are the phonetic representations of obfuscated text, and it is trained to output the phonetic representations of the corresponding original text.

Web Application

We've developed an website that bridges the communication gap for non-Korean speakers encountering obfuscated Korean text. Our intuitive three-step platform first allows users to experiment with Korean text obfuscation, giving them insight into how this process works. The core functionality lies in our transformer model that we have trained which deobfuscate 'Korean AirBnB Script'. For non-Korean users, we've integrated the Helsinki translation model, enabling translation of the deobfuscated text. This way we can help users not only decode obfuscated Korean text but also understand its meaning, making digital Korean content more accessible regardless of obfuscation techniques. The website functionality can be accessed at our GitHub repository : `https://github.com/Abdul03Rafay/KoreanDeobfuscator`.

Experiments & Results

We evaluate our model using the following metrics:

Character Error Rate (CER): The Levenshtein distance between predicted and original texts, normalized by the length of the original text.

Word Error Rate (WER): The WER is derived from the Levenshtein distance, working at the word level instead of the phoneme level.

BLEU Score: To measure the general similarity between predicted and original texts.

Before settling with our phonetic-aware approach, we tried following baselines:

- **LSTM:** We set up a character-level sequence to sequence where we created our of Encoder and Decoder architecture with LSTMs that we wrote ourself using pytorch.
- **Rule-based Normalization:** A system using handcrafted rules for common obfuscation patterns (e.g. ”ㅈ” → ”ㅊ”, ”ㅂ” → ”ㅃ”).
- **Character-level JAMO Transformer:** We used character-level tokenization on decomposed Hangul characters (Jamos) to enhance phonetic representation. We also introduced Jamo embeddings to improve context understanding. Special tokens handle sequence padding and start/end markers.

Our phonetic-aware model outperforms all baselines, with a particularly significant improvement in BLEU score .

Model Variant	WER	CER	BLEU
Latest Model	**0.013**	**0.393**	**0.82**
LSTM	-	0.23	0.17

Table 2: The huge performance increase we got fine tuning transformers

Based on common errors made by our model we found out that:

1. The model struggles with severe obfuscations.
2. Foreign noun based hangul are error prone.
3. Domain-specific terms and proper nouns are more challenging to deobfuscate correctly.

Conclusion

Our project demonstrates that a phonetic-aware sequence-to-sequence approach significantly improves text deobfuscation performance compared to traditional methods in the case of Korean ”airbnb script”. The use of phonetic representations helps bridge the gap between visually distinct but phonetically similar text, allowing our model to more effectively recover original text from obfuscated versions.