T2R2 東京科学大学 リサーチリポジトリ
Science Tokyo Research Repository

論文 / 著書情報
Article / Book Information

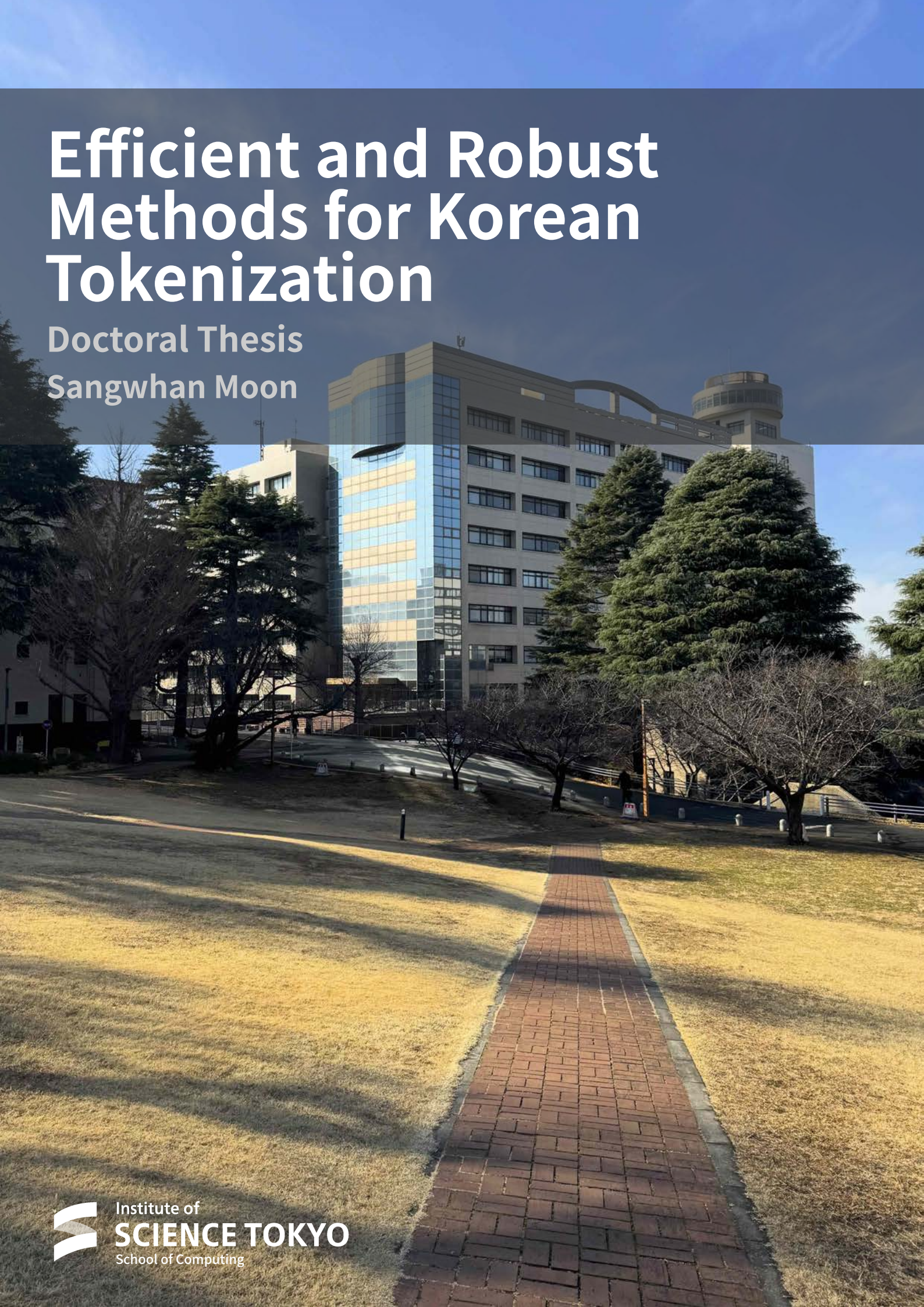
題目(和文)	
Title(English)	Efficient and Robust Methods for Korean Tokenization
著者(和文)	文翔煥
Author(English)	Moon Sangwhan
出典(和文)	学位:博士(学術), 学位授与機関:東京科学大学, 報告番号:甲第395号, 授与年月日:2025年3月26日, 学位の種別:課程博士, 審査員:岡崎 直観,徳永 健伸,宮崎 純,村田 剛志,井上 中順
Citation(English)	Degree:Doctor (Academic), Conferring organization: Institute of Science Tokyo, Report number:甲第395号, Conferred date:2025/3/26, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Powered by T2R2 (Science Tokyo Research Repository)

Efficient and Robust Methods for Korean Tokenization

Doctoral Thesis

Sangwhan Moon



Institute of
SCIENCE TOKYO
School of Computing

Efficient and Robust Methods for Korean Tokenization

by

Sangwhan Moon

to obtain the degree of Doctor of Philosophy
at the Institute of Science Tokyo,
to be defended on Wednesday, February 5, 2025 at 9:30 AM (GMT+9).

Thesis committee:	Professor Naoaki Okazaki,	Supervisor
	Professor Nakamasa Inoue,	Co-supervisor
	Professor Jun Miyazaki,	Co-supervisor
	Professor Tsuyoshi Murata,	Co-supervisor
	Professor Takenobu Tokunaga,	Co-supervisor

An electronic version of this thesis is available at `https://t2r2.star.titech.ac.jp`.

Institute of
SCIENCE TOKYO

Preface

Dedicated to my parents and family.
Sangwhan Moon
Institute of Science Tokyo, February 2025

Acknowledgements

While a thesis bears a single author's name, the journey to its completion is shaped by many remarkable individuals whose contributions, though often unseen, have been truly invaluable. It is with profound gratitude and appreciation that I acknowledge those who have supported, guided, and inspired me throughout this academic journey.

I owe my deepest and most heartfelt thanks to my family - Sujung and Theo[1] - whose boundless love and unwavering support made this work not just possible, but meaningful. Their extraordinary patience during the challenging final months, particularly during those countless late nights of writing until dawn, demonstrated a level of understanding and sacrifice that I can never adequately repay. I am equally grateful to my extended family, whose constant encouragement and steadfast belief in my capabilities pushed me to reach higher and persevere through challenges.

The invaluable administrative guidance from our laboratory staff, particularly Yukiko Konishi and Naoko Furuya, proved absolutely essential in navigating the complex landscape of university procedures. Their expertise and tireless assistance helped me avoid numerous potential bureaucratic complications that could have significantly impeded my progress. Their dedication to supporting researchers like myself often goes unrecognized, but their impact cannot be overstated.

I am especially indebted to Angela Smiley, whose exceptional editorial expertise and unwavering commitment transformed this work from its initial draft into a coherent scholarly publication. Her astute insights, meticulous attention to detail, and generous investment of time have elevated the quality of this thesis beyond what I could have achieved alone. Her contribution has been truly transformative.

My research journey has been immeasurably enriched by numerous brilliant colleagues and collaborators. I extend my deepest appreciation to Youmi Ma, Tatsuya Hiraoka, and Vijay Daultani for their thorough review of draft materials and their remarkably insightful feedback, which has significantly strengthened this work. The collaborative work on tokenization with Marco Cognetta and Tatsuya Hiraoka has been not only formative but genuinely inspiring. I am profoundly grateful for the productive and enlightening research partnerships with Hwichan Kim, Won-ik Cho, Youngsook Song, and Hyejoo Han. Special thanks go to Sakae Mizuki and Zhishen Yang, whose thoughtful

[1] As promised; my homework is now done. Although by the time you can comprehend this, you'd be the one busy with homework. If you ever end up reading this, come talk to me - I might have a present for you.

guidance and mentorship helped me establish my footing in the laboratory and set me on a path to success.

I would be remiss not to acknowledge my former colleagues at Odd Concepts and their exceptional academic advisor, Yannis Avrithis. Their mentorship in academic research methodology and their remarkable patience in sharing both the rewards and challenges of scholarly work were instrumental in guiding me toward this academic pursuit. Their influence on my academic development has been profound and lasting.

Lastly, I extend my sincere appreciation to my colleagues and team at Google, particularly my immediate team members who have shown extraordinary understanding and support during my period of focused thesis work. Their patience and encouragement, despite my growing backlog of correspondence, has been not just appreciated but truly humbling.

Abstract

Systems capable of understanding human language have been a holy grail amongst scientists and engineers for many decades. With recent advancements in natural language processing (NLP) methods and large-scale training, the field has never been closer to making systems capable of understanding language a reality.

This research investigates a critical but often overlooked aspect of modern multilingual language processing methods: vocabulary allocation in tokenizers - which significantly impacts model performance and efficiency. This impact is particularly pronounced in multilingual pre-trained models, where systematic biases inherited from training data and the inherent complexity of building generic vocabularies create substantial disadvantages for languages with diverse character sets, leading to both computational inefficiencies and degraded performance.

My work demonstrates that these vocabulary-related limitations, particularly evident in Korean language processing, represent addressable engineering challenges rather than fundamental constraints. I present two complementary approaches that target different aspects of the vocabulary challenge: first, an out-of-vocabulary (OOV) mitigation strategy that enables recovery of model performance through intelligent vocabulary mapping without retraining, and second, Jamo Pair Encoding, a sub-character tokenization method that dramatically reduces vocabulary requirements while ensuring maximum fidelity.

The OOV mitigation approach I developed maps OOV tokens to in-vocabulary surrogates through character distance, masked language model prediction, and unseen subword assignment. Combined with continued pre-training, this method improved accuracy by 1-2% across multiple downstream tasks including sentiment analysis and question answering. Notably, it achieves 86.04% accuracy on the NSMC dataset even under synthetic 50% OOV conditions, significantly outperforming the 85.50% FastText baseline.

The proposed Jamo Pair Encoding decomposes Korean syllable blocks into constituent phonetic elements, reducing vocabulary requirements from 11,172 characters to approximately 100 subcharacters while maintaining complete coverage. This approach achieved a 90% reduction in vocabulary size while improving sequence length efficiency, as measured by corpus token count and subword fertility metrics. Through state machine-based reconstruction, it guarantees lossless round-trip conversion between original text and tokenized forms.

Through comprehensive evaluation across multiple frameworks, I demonstrate these improvements using traditional downstream metrics, information theoretic measures, and novel tokenization quality metrics. The methods presented maintain relevance despite field evolution - while byte-level approaches have largely solved OOV challenges in large models, these techniques offer particular value for resource-constrained scenarios and improving efficiency in multilingual model vocabulary allocation.

In summary, this thesis makes four key contributions:

1. Demonstrates feasibility of post-training recovery for models with suboptimal tokenization,
2. Establishes new approaches for efficient processing of character-diverse languages,
3. Contributes early evaluation frameworks for assessing tokenization quality,
4. Opens new research directions in sub-character processing for other writing systems.

Most importantly, it shows that the seemingly competing goals of tokenization robustness and efficiency can be simultaneously improved through principled approaches to writing system analysis and vocabulary management.

Contents

Preface i

Acknowledgements ii

Abstract iv

List of Figures x

List of Tables xii

Nomenclature xiv

1 Introduction 1

1.1 Motivation 1

1.2 Contributions 2

1.3 Outline 3

1.4 Publications 5

2 Background 6

2.1 Traditional Methods 6

2.2 Statistical Methods 8

2.3 Tokenization 11

2.4 Out-of-Vocabulary Handling 13

2.5 Complexities of CJK Languages 15

2.5.1 Chinese 16

2.5.2 Japanese 17

2.5.3 Korean 17

2.5.4 Implications for Text Processing 18

2.6 Embeddings and Neural Networks 19

2.7 Transfer Learning . . . 22
2.8 Language Models and Foundation Models in NLP . . . 24
2.9 Robustness and Efficiency, and its Trade-offs . . . 26
2.9.1 Robustness . . . 26
2.9.2 Efficiency . . . 27
2.10 Trade-offs . . . 29
2.11 Summary . . . 30

3 Related Work 31
3.1 The Bounds of Relation . . . 31
3.2 Foundation Models . . . 31
3.2.1 Sequence Modeling with Neural Networks . . . 31
3.2.2 GPT: Generative Pre-trained Transformer . . . 35
3.2.3 BERT: Bi-directional Encoder Representations from Transformers . . . 36
3.3 Tokenization . . . 38
3.3.1 Segmentation and Subword Tokenization . . . 38
3.3.2 WordPiece . . . 39
3.3.3 BPE: Byte Pair Encoding . . . 40
3.3.4 SentencePiece . . . 41
3.4 Related Methods . . . 42
3.4.1 Vocabulary-free Methods . . . 42
3.4.2 Vocabulary Adaptation . . . 43
3.4.3 OOV Mitigations . . . 44
3.4.4 Normalizing Methods . . . 45
3.4.5 Sub-character Architectures . . . 46

4 The Effects and Mitigation of Out-of-vocabulary in Universal Language Models 48
4.1 Motivation . . . 48
4.2 Terminology Evolution . . . 50
4.3 Preliminaries . . . 51
4.3.1 Foundation Model Tokenizer Training in Practice . . . 51
4.3.2 BERT Tokenizer . . . 52
4.3.3 OOV Mitigation . . . 53

4.4 Method . . . 53
4.4.1 Surrogated Tokens . . . 55
4.4.2 Additional Tokens . . . 58
4.4.3 Post-mitigation . . . 58
4.5 Tasks . . . 59
4.5.1 Classification: Naver Sentiment Movie Corpus . . . 60
4.5.2 Classification: Japanese Twitter Sentiment Analysis . . . 60
4.5.3 Classification: Chinese News Sentiment Analysis . . . 60
4.5.4 Question Answering: KorQuAD 1.0 . . . 60
4.6 Experiments . . . 61
4.6.1 Hyperparameters . . . 62
4.6.2 Results on Task Datasets . . . 62
4.6.3 Effects of OOV on Task Performance . . . 65
4.6.4 Task Performance and OOV . . . 67
4.6.5 Recovery with Proposed Method . . . 69
4.7 Applicability to Other Models . . . 71
4.7.1 Multilingual Models . . . 71
4.7.2 Comparison with Monolingual Models . . . 71
4.8 Conclusion . . . 72

5 Jamo Pair Encoding: Subcharacter-tokenization of Korean 74
5.1 Motivation . . . 74
5.2 Method . . . 77
5.2.1 General Decomposition . . . 78
5.2.2 Aligned Processing . . . 79
5.2.3 Automaton (Unaligned) Processing . . . 80
5.3 Experiments . . . 85
5.3.1 Datasets . . . 85
5.3.2 Hyperparameters . . . 86
5.3.3 Results . . . 87
5.4 Limitations and Future Work . . . 91
5.5 Conclusion . . . 91

6 Discussion 94
6.1 Evaluation of Tokenization Quality . . . 94
6.1.1 Corpus Token Count, Fertility, Compression, and Information Theory . . . 95
6.1.2 Parity . . . 97
6.2 Retrospective: The Effects and Mitigation of OOV . . . 98
6.2.1 Why does it work? . . . 98
6.2.2 Validity as of Today . . . 99
6.2.3 Revisiting with Current Frameworks . . . 100
6.3 Retrospective: Jamo Pair Encoding . . . 101
6.3.1 Why does it work? . . . 101
6.3.2 Validity as of Today . . . 102
6.3.3 Revisiting with Current Frameworks . . . 103
6.4 The Significance of Tokenization . . . 105
6.5 Future Work . . . 107
6.5.1 Opportunities for Enhanced Evaluation . . . 107
6.5.2 Subword Architectures for Other Languages . . . 108
6.5.3 Integration with Byte-Level BPE . . . 108
6.6 Disclosure and Broader Implications . . . 109
6.6.1 Usage of AI Assistants . . . 109
6.6.2 Impact Assessment: Ethical and Societal . . . 109
6.6.3 Compute Costs . . . 110
6.7 Summary . . . 111

7 Conclusion 112
7.1 Summary . . . 112
7.1.1 Contributions . . . 112
7.1.2 Conclusion . . . 113

8 References 115

List of Figures

2.1 Character-level and character bi-gram vocabulary size bounds for different languages. 16
2.2 Examples of two shallow artificial neural network architectures. 19

3.1 RNN compared with LSTM. 32
3.2 The Transformer architecture, separated by the encoder and decoder. 34
3.3 BPE compared to Unigram LM. 39

4.1 Multilingual BERT vocabulary distribution for CJK languages. 49
4.2 OOV surface contrasted to subword OOV, illustrated with mitigation. 53
4.3 The hierarchy of OOV and the high level process explained with a simplified example. 54
4.4 Different experiment variants, illustrated as a pipeline. 54
4.5 Intuition of character distance method, showing radical sharing and phonetic similarities. 56
4.6 Masked language-model OOV surrogate candidate selection. 57
4.7 Qualitative inspection of positive, positive (with bad patches) and negative vocabulary patch cases. 64
4.8 Performance degradation trends of artificial OOV with different pruning strategies. . 67
4.9 Performance recovery under different OOV rates. (Note: Plot range for the y-axis differs between frequencies.) 70

5.1 Two samples of transition from ideograph to syllable, syllable to phonemes (jamo). 75
5.2 Common Jamo surface morpheme under different verb conjugations. 76
5.3 Filler (<f>) interfering a morpheme surface merge. 79
5.4 State machine explaining the automaton method, demonstrating consonant carry-over state transition. 81
5.5 OOV Rate (OR) and OOV Length Coverage (LC) compared at 1K vocabulary size. . . . 88
5.6 OOV rates (OR) across different vocabulary sizes. 89

5.7 OOV rates (OR) on larger vocabulary baseline contrasted with proposed using a smaller vocabulary. 89
5.8 OOV rates (OR) comparing proposed method with other multilingual and monolingual models. 91

6.1 Comparison between OOV surface forms and subword OOV, with mitigation strategies illustrated. 100
6.2 Comparison of multilingual BERT and our 2.5K model. 103
6.3 Comparison of XLM and our 10K model. 104
6.4 Example of tokenization-induced model misbehavior in ChatGPT. 105
6.5 Proposed evolution of character-level processing. 107

List of Tables

2.1 A minimal bag-of-words (BOW) vocabulary. 7

2.2 Bag-of-words (BOW) encoding of "The weather makes the dogs sick, but I am happy". In a multiple-Bernoulli representation, the value for "*the*" is 1. 8

2.3 Our previous example vocabulary with stopwords removed and stemmed with the Porter stemmer algorithm. 12

2.4 A bag-of-words (BOW) encoding using the trimmed vocabulary,with OOV handling (<*unknown*>). 14

4.1 Examples of OOV in the task datasets, demonstrating different patterns. 59

4.2 OOV rates on the task datasets with multilingual BERT. 59

4.3 Hyperparameters used to train each of the downstream task models. 61

4.4 Experiment results, with statistically significant p-values underlined (< 0.05). +CPT indicates usage of CPT, Acc is *accuracy* and Std is *standard deviation*. 63

4.5 Improvements and regressions with OOV samples, best Character Distance models compared to best baseline models. 64

4.6 Performance degradation data of artificial OOV with different pruning strategies. . . 67

4.7 Recovery experiment results. Patched is with mitigation, Patched+CPT is with mitigation and CPT. 69

4.8 OOV rates on the datasets, comparing multilingual BERT with monolingual. 72

5.1 Example of 1:n relationship between syllable and ideographs. 75

5.2 Computational and written (or typed) Jamo alphabet contrasted, along with their roles for composition. 76

5.3 An example of the decomposition in aligned mode, with orphan hints (<o>) and fillers (<f>). 79

5.4 Decomposition in automaton mode, with orphan hints (<o>) but without fillers (<f>). 80

5.5 Comparison of Korean capabilities across different models. 85

5.6 Complete experiment results comparing different tokenizers and datasets with our method. 90
5.7 Count of Korean subwords, minimum, maximum, average, and variance of the token length. Jamo methods scaled down by 2.5 for a fair comparison. 92

6.1 OOV Recovery Performance Comparison 99
6.2 Translation performance using byte-level BPE across language pairs. 99
6.3 Raw data comparing our method (2.5K/10.K) with recently proposed metrics on NSMC. 104

Nomenclature

If a nomenclature is required, a simple template can be found below for convenience. Feel free to use, adapt or completely remove.

Notation

Notation	Definition
cf.	confer/conferatur (en: compare)
e.g.	exemplum gratia (en: for example)
et al.	et alia (en: and others)
i.e.	id est (en: that is)

Abbreviations

Abbreviation	Definition
BERT	Bi-Directional Encoder Representations from Transformers
BiLSTM	Bi-directional Long-Short Term Memory
BOW	Bag-of-Words
BPTT	Back-Propagation Through Time
CBOW	Continuous Bag-of-Words
CRF	Conditional Random Field
CPT	Continual Pre-Training
CTC	Corpus Token Count
EM	Expectation Maximization
FFN	Feed-forward Network
GPT	Generative Pre-trained Transformer
HMM	Hidden Markov Model
LM	Language Model
LLM	Large Language Model

Abbreviation	Definition
LSTM	Long Short-Term Memory (a type of RNN)
mBERT	Multilingual BERT
ML	Machine Learning
MT	Machine Translation
MLP	Multi-Layer Perceptron
NLP	Natural Language Processing
NMT	Neural Machine Translation
OOV	Out-of-Vocabulary
PDF	Probability Density Function
PMF	Probability Mass Function
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SFT	Supervised Fine-Tuning
tf-idf	term frequency-inverse document frequency

Symbols

Symbol	Definition	Unit
P	Probability mass function	
p	Probability density function	
δ	Standard deviation	
δ^2	Variance	
η	Learning rate	
$\boldsymbol{\theta}$	Model parameters	
$\mathcal{V}$	Vocabulary (in corpus)	
$\hat{\mathcal{V}}$	Vocabulary (exhaustive)	
$\mathcal{V}'$	Vocabulary (in trained tokenizer)	
$\mathcal{V}^*$	Subword vocabulary (in corpus)	
$\hat{\mathcal{V}}^*$	Subword vocabulary (oversampled)	
$\mathcal{V}'^*$	Subword vocabulary (in trained tokenizer)	

Symbol	Definition	Unit
$\sqsubseteq$	Word	
$\sqsubseteq^*$	Subword	
$\lvert\mathcal{V}\rvert$	Vocabulary size	
$\lvert\mathcal{V}^*\rvert$	Subword vocabulary size	
Σ	Characters (in corpus)	
$\hat{\Sigma}$	Characters (exhaustive)	
Σ'	Characters (in trained tokenizer)	
σ	Bytes (0x00-0xFF)	
μ	Subword merge	

1

Introduction

1.1. Motivation

Communication, and by extension language, is widely recognized as one of the fundamental characteristics that distinguish intelligent beings from their non-intelligent counterparts. Since the advent of computers, researchers have pursued the goal of enabling machines to comprehend and communicate using human language. In 1950, Alan Turing's seminal article "Computing Machinery and Intelligence" introduced what became known as the Turing test, establishing a framework for evaluating machine intelligence. This breakthrough led to decades of advancement in natural language processing.

The early era of natural language processing was marked by systems like ELIZA (Weizenbaum, 1966) and SHRDLU (Winograd, 1972), which employed human-written rule sets to transform linguistic input into symbolic representations. These systems generated responses through iterative conversations. However, contemporary research has demonstrated the inherent limitations of rule-based approaches in achieving genuine language understanding and generalization.

Statistical approaches evolved over the past three decades as a data-driven alternative (Brown et al., 1990; Manning and Schütze, 1999). These methods fundamentally shifted the field by examining common patterns in language use rather than relying on predefined rules. An evolution of these methods led to neural models, which combined with self-supervised learning techniques (Mikolov et al., 2013) accelerated progress by enabling effective scaling with vast amounts of text data. This evolution produced large-scale models capable of multiple tasks, both with and without task-specific training (Devlin et al., 2019; Howard and Ruder, 2018; Radford et al., 2019).

Despite these advances in model architectures and training approaches, fundamental challenges in processing human language persist. One of the most critical yet often overlooked challenges lies in the initial transformation of raw text into a format that machines can process efficiently. This

challenge becomes particularly acute as models scale to handle multiple languages simultaneously, each with its own unique characteristics and requirements.

A crucial step in all natural language processing systems is a processing step known as tokenization, which transforms text streams into algorithmically processable chunks. While current methods preserve input information more effectively than their predecessors, they face significant challenges when processing character-diverse languages. This challenge particularly affects Chinese, Japanese, and Korean (CJK) languages, which collectively serve more than 1.5 billion speakers yet remain underrepresented in modern natural language processing systems.

The implications of suboptimal CJK processing extend beyond academic interest to practical constraints. Recent research by Petrov et al. (2023) demonstrates that processing CJK text in state-of-the-art models like GPT-4 incurs 2-2.5 times higher computational costs compared to English text. This cost differential stems from fundamental inefficiencies in tokenization, where training corpus biases lead to inflated sequence lengths and high out-of-vocabulary rates for CJK languages.

These challenges exist partly due to an oversimplified approach to processing in multilingual models. Current systems often treat all languages uniformly, disregarding the unique characteristics of different writing systems. This one-size-fits-all approach particularly disadvantages languages with complex scripts or distinctive linguistic properties. While the field moved away from language-specific tokenization methods in favor of universal approaches that showed superior performance, these generic methods have revealed new inefficiencies when processing certain languages. This suggests an opportunity to selectively reintroduce language-specific processing to complement, rather than replace, modern tokenization approaches.

This dissertation examines these challenges through the specific lens of Korean language processing. In particular, I demonstrate how current tokenization approaches create an inherent trade-off between computational efficiency and information preservation, leading to reduced generalization and task performance.

1.2. Contributions

This research advances the field of natural language processing through several key contributions:

- Our first contribution focuses on tokenization schemes for character-diverse languages such as Chinese, Japanese, and Korean. We thoroughly examine current state-of-the-art approaches and propose specific enhancements to address their limitations for Korean.
- In the domain of transfer learning, we introduce a novel post-training method that significantly

reduces information loss in task-tuned models. This approach specifically targets out-of-vocabulary scenarios, resulting in improved model performance.

- This method's effectiveness is further validated through extensive testing against artificial models with intentionally degraded performance, establishing clear boundaries for information loss mitigation.
- For from-scratch model training, we develop an innovative approach that reduces the vocabulary budget rigidity and cost associated with processing Korean text in both monolingual and multilingual models.
- We conduct a comprehensive analysis of both proposed methods through the lens of recent developments in tokenization quality evaluation. This analysis provides valuable insights into the effectiveness and limitations of each approach.
- Finally, we present a thorough examination of tokenization quality metrics, both existing and newly defined. Our research demonstrates how these metrics interact and the necessary trade-offs between them, providing a framework for future tokenization research and development.

1.3. Outline

Following this introductory chapter, the dissertation is structured as follows:

Chapter 2: Background

This chapter establishes the foundational knowledge necessary for understanding tokenization and word segmentation, with particular emphasis on concepts central to this dissertation. We trace the evolution of tokenization methods from traditional approaches to contemporary techniques, examining how Chinese, Japanese, and Korean language tokenization has developed alongside these advances. Special attention is given to the unique challenges these languages present in natural language processing.

Additionally, this thesis focuses on two key qualities of tokenization and their interrelationship: efficiency and robustness.

Chapter 3: Related work

This chapter examines research that directly informs our proposed methods. We provide detailed analysis of relevant studies, highlighting methodological differences, limitations, and objectives.

The discussion encompasses language model-based transfer learning, tokenization strategies in prominent architectures, and the role of tokenization in both pre-training and post-training environments.

Chapter 4: The Effects and Mitigation of Out-of-vocabulary in Universal Language Models

The first major component of our research investigates how suboptimal tokenization creates adverse effects through information loss. We focus on pre-trained multilingual BERT as a case study, proposing a novel mitigation strategy for minimizing information loss in post-training scenarios. To demonstrate our method's efficacy, we test it under artificially degraded conditions, establishing its effectiveness even in worst-case scenarios.

Chapter 5: Jamo Pair Encoding: Subcharacter Tokenization of Korean

Our second major contribution addresses the specific challenges of tokenizing Korean text using current subword tokenization methods. We examine these challenges through the lens of vocabulary size and information loss, introducing a novel approach that leverages the Korean script's linguistic properties. This method significantly reduces both information loss and vocabulary costs, with performance validated against baseline models.

Chapter 6: Discussion

This chapter provides a deeper analysis of our two proposed methods for improving Korean tokenization and segmentation. While previous chapters demonstrate effectiveness through empirical evidence and task-specific metrics, here we incorporate recent developments in the field to explain the observed phenomena more comprehensively. This analysis offers new insights into our methods' success.

Chapter 7: Conclusion

The final chapter synthesizes our contributions to Korean tokenization and their downstream impact. We address remaining challenges, ethical considerations, study limitations, and promising directions for future research in this rapidly evolving field.

1.4. Publications

The work primarily discussed in this dissertation are the following peer-reviewed articles, in order of publication:

- **Moon, S.** and Okazaki N. (2020). Jamo pair encoding: Subcharacter representation-based extreme Korean vocabulary compression for efficient subword tokenization. In *Proceedings of the Twelfth Language Resources and Evaluation Conference (LREC)*.
- **Moon, S.** and Okazaki N. (2020). PatchBERT: Just-in-time, out-of-vocabulary patching. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- **Moon, S.**, and Okazaki N. (2021). Effects and mitigation of out-of-vocabulary in universal language models. In *Journal of Information Processing, Volume 29*.

While not directly discussed in the main content, the following peer-reviewed articles are discussed in supporting chapters:

- **Moon, S.**, Cho, W., Han, H., Okazaki, N., and Kim, N. (2022). OpenKorPOS: Democratizing Korean Tokenization with Voting-Based Open Corpus Annotation. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference (LREC)*.
- Cognetta, M., **Moon, S.**, Wolf-Sonkin, L., and Okazaki, N. (2023). Parameter-Efficient Korean Character-Level Language Modeling. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.
- Cognetta, M., Zouhar, V., **Moon, S.**, and Okazaki, N. (2024). Two Counterexamples to Tokenization and the Noiseless Channel. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*.

2

Background

This chapter presents the fundamental concepts essential for comprehending the subsequent material, and is primarily intended for readers without prior expertise in Natural Language Processing (NLP). Readers who possess substantial background knowledge in this domain may proceed directly to Section 2.9, as the foundational concepts covered here will likely be familiar.

2.1. Traditional Methods

Natural language processing (NLP) tasks begin by transforming input text into a feature representation that algorithms can process. This transformation, at its most fundamental level, represents a form of lossy tokenization where complex linguistic structures are reduced to simpler computational elements (Winograd, 1971). The resulting representation captures relevant characteristics from the text and is passed to a prediction function that produces the desired output.

Consider these example sentences and their corresponding sentiment classifications:

- **Input:** "I am sad because my dog is sick." (*Negative*) | "The weather makes me happy" (*Positive*)
- **Output:** *Negative* | *Positive*

In this example, the words *sad* and *happy* serve as strong indicators of sentiment, demonstrating how complex emotional expressions are reduced to individual *lexical items*. Traditional symbolic approaches leveraged such linguistic patterns to define sets of *rules* (Stone et al., 1966). From a feature representation perspective, these approaches focused primarily on identifying specific trigger words in the text.

The classification function can be formally expressed as a piecewise function, where X represents the input text and Y the predicted sentiment:

Slot	Word	Slot	Word
1	i	8	sick
2	am	9	the
3	sad	10	weather
4	because	11	makes
5	my	12	me
6	dog	13	happy
7	is	...	...

Table 2.1: A minimal bag-of-words (BOW) vocabulary.

$$Y = f(X) = \begin{cases} \text{positive} & \text{if “happy”} \in X \\ \text{negative} & \text{if “sad”} \in X \\ \text{unknown} & \text{otherwise} \end{cases}$$

This approach, however, proves fragile in practice. A single entry in a thesaurus for either “happy” or “sad” yields dozens of synonyms, each requiring explicit inclusion in the ruleset. Moreover, such systems struggle with negation and contextual inference. For instance, “I am no longer sad” would be incorrectly classified as *negative*, while “My dog is sick” would be classified as *unknown* despite its clear negative sentiment. These challenges extend beyond simple word matching, as negative sentiment in natural language can be expressed through various grammatical constructions, unlike the more structured nature of programming languages.

To address these limitations, evolutions of these systems adopted syntactic parsing approaches (Marcus, 1980). These parsers enabled more sophisticated rule definition by operating on grammatical structure rather than surface forms (Winograd, 1983). While successful for English, this approach faced scalability challenges: each language requires its own parser implementation and language-specific ruleset due to fundamental differences in grammatical structure (Gazdar, 1985).

These foundational complexities demonstrate why even seemingly straightforward tasks like sentiment analysis prove challenging with rule-based approaches. The inherent complexity of natural language, with its contextual nuances and semantic variations, ultimately led to the decline of purely symbolic methods in the 1980s. In their place emerged statistical approaches that could learn patterns directly from data, marking a fundamental shift in NLP methodology (Manning and Schütze, 1999).

Word	Count	Word	Count
i	1	sick	1
am	1	the	$2 \rightarrow 1$
sad	0	weather	1
because	0	makes	1
my	1	me	0
dog	1	happy	1
is	0	...	...

Table 2.2: Bag-of-words (BOW) encoding of "The weather makes the dogs sick, but I am happy". In a multiple-Bernoulli representation, the value for "*the*" is 1.

2.2. Statistical Methods

Natural language processing transitioned from symbolic to statistical approaches to address the limitations of hand-crafted rules. This shift fundamentally changed how we formulate classification functions: instead of deterministic outputs, statistical approaches produce probability distributions over possible outcomes. For prediction tasks, we denote the conditional probability of an event y given input x as $P(y|x)$. In our sentiment analysis example, this enables us to express *probable positive* cases as $P(y|\text{“happy”} \in x)$ and *probable negative* cases as $P(y|\text{“sad”} \in x)$.

The first challenge in statistical approaches is converting text into a mathematical representation suitable for computation. The bag-of-words (BOW) model (Harris, 1954) provides a straightforward solution by treating a document as an unordered collection of words. This approach discards word order and grammatical structure, focusing solely on word occurrences. While this simplification loses some linguistic information, it enables efficient statistical analysis. The underlying intuition behind this in statistical modeling is that some words, regardless of them not being obviously a salient feature, may carry signal of value to the model in the learning process.

To implement this model, we first construct a vocabulary of all unique words appearing in our document collection, then represent each document [1] as a vector indicating the frequency or presence of each word. Table 2.1 demonstrates this concept, showing a vocabulary constructed from our sentiment analysis examples.

Using the vocabulary defined in Table 2.1, we can encode any text as a vector where each position corresponds to a word. Table 2.2 illustrates this encoding for the sentence "The weather makes the dogs sick, but I am happy", where each word from our vocabulary is assigned a count based on its occurrence in the sentence.

The encoding shown in Table 2.2 transforms our text classification problem into a vector space

[1] In text processing, the term *document* is defined as one input to the algorithm; this can be a sentence, paragraph, or a full document - but in literature it is commonly referred to as a *document*.

problem, where each document becomes a point in a high-dimensional space. While this frequency-based representation is intuitive, we can enhance our statistical modeling by viewing the same information through a multiple-Bernoulli event space. In this alternative formulation, each word position becomes a binary random variable:

$$X_i = \begin{cases} 1 & \text{if word } w_i \text{ appears in the document} \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

The multiple-Bernoulli representation, first applied to text classification by McCallum and Nigam (1998), offers several important advantages. First, it aligns naturally with the way humans process text—the presence or absence of keywords often matters more than their frequency. For instance, in sentiment analysis, seeing the word "excellent" once is typically sufficient to indicate positive sentiment; its repetition may not provide significant additional information. Second, this binary representation simplifies probability estimation by reducing the space of possible outcomes, making the model more robust when training data is limited. Third, it provides a more principled way to handle rare words, where frequency counts might be unreliable due to data sparsity.

To formalize the classification problem in this framework, let Ω be our sample space, containing all possible text documents. The classification task involves partitioning this space into disjoint events corresponding to our sentiment classes: $\Omega = \Omega_{\text{positive}} \cup \Omega_{\text{negative}}$. In this framework, each document d represents an event in Ω, and our goal is to determine the probability of each class given the document's features.

The mathematical foundation for classification in this framework is Bayes' Rule, which relates the conditional probability $P(Y|X)$ of event Y given event X to their individual probabilities:

$$\begin{aligned} P(Y|X) &= \frac{P(X|Y)P(Y)}{P(X)} \\ &= \frac{P(X|Y)P(Y)}{\sum_{y \in Y} P(X|Y=y)P(Y=y)} \end{aligned} \tag{2.2}$$

To implement a Naive Bayes classifier (McCallum and Nigam, 1998) for sentiment analysis, we calculate the probability of each class given the input text. Using our representation choice, we can express this as:

$$\begin{aligned} P(c|d) &= \frac{P(d|c)P(c)}{P(d)} \\ &\propto P(d|c)P(c) \end{aligned} \tag{2.3}$$

where c represents the class and d represents the document. Since $P(d)$ is constant across classes, we can work with proportionality. The crucial simplifying assumption of Naive Bayes, known as the "naive independence assumption," is that all features are conditionally independent given the class. Under the frequency-based representation, this allows us to decompose $P(d|c)$ into:

$$P(d|c) = \prod_{i=1}^{n} P(w_i|c)^{f_i} \tag{2.4}$$

where w_i represents the i-th word, n is the vocabulary size, and f_i is the frequency of word w_i. Alternatively, under the multiple-Bernoulli representation (McCallum and Nigam, 1998), we have:

$$\begin{aligned} P(c|d) &\propto P(c) \prod_{i=1}^{n} P(X_i = x_i|c) \\ &= P(c) \prod_{i:x_i=1} P(X_i = 1|c) \prod_{i:x_i=0} P(X_i = 0|c) \end{aligned} \tag{2.5}$$

In practice, we estimate these probabilities from training data:

$$\begin{aligned} P(c) &= \frac{\text{count of documents in class } c}{\text{total number of documents}} \\ P(w_i|c) &= \frac{\text{count of word } w_i \text{ in class } c \text{ documents}}{\text{total words in class } c \text{ documents}} \end{aligned} \tag{2.6}$$

To prevent zero probabilities for unseen words, we apply Laplace smoothing (Chen and Goodman, 1996) with parameter α (typically 1):

$$P(X_i = 1|c) = \frac{\text{count}(w_i, c) + \alpha}{\text{count}(c) + 2\alpha} \tag{2.7}$$

The final classification decision rule becomes:

$$c^* = \underset{c}{\text{argmax}} \left[\log P(c) + \sum_{i:x_i=1} \log P(X_i = 1|c) + \sum_{i:x_i=0} \log P(X_i = 0|c) \right] \tag{2.8}$$

This formulation extends naturally to multi-class scenarios where $c \in \{c_1, ..., c_k\}$, with the predicted class maximizing the posterior probability:

$$c^* = \underset{c \in \{c_1,...,c_k\}}{\text{argmax}} \; P(c|d) \tag{2.9}$$

The multiple-Bernoulli model captures the significance of word presence rather than frequency, aligning well with applications where certain words strongly indicate document class regardless of repetition. While mathematically elegant, the model maintains the naive independence assumption, unable to capture dependencies between words (such as “not good” versus “very good”). Nevertheless, its simplicity and computational efficiency, combined with surprisingly good empirical performance, make it a valuable baseline for text classification tasks.

This simple classification example demonstrates the fundamental requirement in statistical NLP: transforming arbitrary text sequences into discrete, mathematically tractable representations. While we have explored the most straightforward approach through BOW and multiple-Bernoulli representations, the challenges of effective text transformation extend far beyond these basic methods. The next section examines these challenges in depth, exploring advanced tokenization approaches that better serve the requirements of modern statistical models.

2.3. Tokenization

In the previous section, we implicitly performed tokenization as part of our BOW implementation for sentiment analysis. To formalize this process, let $\mathcal{T}$ be the set of all possible text tokens and $\mathcal{V}$ be our target vocabulary. Tokenization and preprocessing can then be viewed as a series of functions $f : \mathcal{T} \to \mathcal{V}$ that map raw text to vocabulary elements.

While not explicitly detailed earlier, our process involved at least three distinct steps, as evidenced by Table 2.2:

1. Strip punctuation using regular expressions
2. Convert text to lowercase
3. Split text into tokens using whitespace as the delimiter

Strictly speaking, only the third step constitutes tokenization proper; the first two are preprocessing steps that enhance token robustness. Let f_{case} represent the case normalization function. Without this preprocessing, tokens like *“sick”* would retain trailing punctuation (*“sick,”*), and if the case normalization function f_{case} is a no-op, we would observe:

Slot	Word	Slot	Word
1	~~i~~	8	sick
2	~~am~~	9	~~the~~
3	sad	10	weather
4	*becaus*	11	*make*
5	~~my~~	12	~~me~~
6	dog	13	*happi*
7	~~is~~	...	...

Table 2.3: Our previous example vocabulary with stopwords removed and stemmed with the Porter stemmer algorithm.

$$f_{\text{case}}(\text{“The”}) \neq f_{\text{case}}(\text{“the”}) \tag{2.10}$$

causing token mismatches. These sequential preprocessing and tokenization steps form a composition of functions collectively known as the *tokenizer* or *tokenization pipeline*.

Case normalization through lowercasing, while computationally inexpensive, exemplifies the trade-offs inherent in text preprocessing. While it resolves case-based token mismatches by ensuring the following:

$$f_{\text{case}}(\text{“Apple”}) = f_{\text{case}}(\text{“apple”}) \tag{2.11}$$

it eliminates the distinction between proper nouns and common nouns—for instance, Apple (the company) becomes indistinguishable from apple (the fruit).

More sophisticated tokenization pipelines may incorporate additional preprocessing functions (assuming English text, but many languages follow a similar pipeline):

- Suffix stripping f_{stem} (Porter, 1980), where: $f_{\text{stem}}(\text{“likes”}) = \text{“like”}$
- Lemmatization f_{lemma} (Fellbaum, 1998), where: $f_{\text{lemma}}(\text{“corpora”}) = \text{“corpus”}$
- Stopword removal f_{stop} (Rijsbergen, 1979; Fox, 1992)
- Diacritic normalization f_{diac}, where: $f_{\text{diac}}(\text{“naïve”}) = \text{“naive”}$
- Number normalization f_{num}, where: $f_{\text{num}}(\text{“123”}) = \text{“<number>”}$

Table 2.3 demonstrates the effects of stopword removal and stemming. These transformations reduce vocabulary size significantly. Let $|\mathcal{V}|$ represent vocabulary size. Without preprocessing:

$$|\mathcal{V}_{\text{raw}}| = |\mathcal{V}_{\text{base}}| \times |\mathcal{V}_{\text{inflections}}| \times |\mathcal{V}_{\text{cases}}| \tag{2.12}$$

With preprocessing:

$$|\mathcal{V}_{\text{processed}}| \approx |\mathcal{V}_{\text{base}}| \tag{2.13}$$

where $|\mathcal{V}_{\text{processed}}| \ll |\mathcal{V}_{\text{raw}}|$ is guaranteed. This reduction enhances generalizability and learnability (Bellman, 1959) while reducing computational costs. However, these transformations can be detrimental for tasks requiring grammatical nuance, as such information is lost in the process.

These preprocessing transformations are inherently lossy. For any preprocessing function f, we generally have:

$$f^{-1}(f(t)) \neq t \tag{2.14}$$

where $t \in \mathcal{T}$ is an original token. While storing auxiliary information for reconstruction is possible, it adds complexity to the system and was historically rare (Manning and Schütze, 1999). Nevertheless, these transforms significantly enhance vocabulary robustness and token quality, making them standard practice in classical NLP approaches. This trend continued until the development of neural methods that could effectively learn from less processed input.

The adoption of these preprocessing steps was driven by both practical and theoretical considerations. Different morphological forms of words often show imbalanced frequency distributions, complicating the learning process. Moreover, maintaining a vocabulary covering every possible word form becomes computationally intractable. Even the base vocabulary of English presents challenges—current estimates suggest approximately one million words, though linguistic consensus on this figure varies by $\pm 25\%$.[2] Axiomatically, with a representation like our bag-of-words model, the resulting output would be a one million-dimension vector - which is likely a challenging proposition from a learnability perspective.

Despite extensive preprocessing, creating a truly representative vocabulary remains challenging. Our previous example's omission of the common word *"but"* illustrates this point, though more typical challenges include handling rare words, complex terms, and spelling variations or errors. The next section addresses these vocabulary coverage challenges in detail.

2.4. Out-of-Vocabulary Handling

While previous sections explored methods to transform text into discrete tokens and maximize vocabulary robustness, achieving perfect coverage remains impossible even for a single language

[2] According to Merriam-Webster's current estimate, with the noted caveat that linguists debate this figure.

Slot	Word	Slot	Word
1	sad	5	weather
2	*becaus*	6	*happi*
3	dog	7	*<unknown>*
4	*make*	...	...

Table 2.4: A bag-of-words (BOW) encoding using the trimmed vocabulary,with OOV handling (*<unknown>*).

like English. Formally, let $\mathcal{V}$ be our vocabulary and T be the set of all possible tokens in a language (Jurafsky and Martin, 2024). There will always exist some token $t \in T$ such that $t \notin \mathcal{V}$. This fundamental limitation necessitates a strategy for handling out-of-vocabulary (OOV) tokens.

Two primary approaches exist for OOV handling (Knight and Graehl, 1997). Let $f_{\text{OOV}} : T \to \mathcal{V}$ be our OOV handling function. The first approach, token dropping, can be expressed as:

$$f_{\text{drop}}(t) = \begin{cases} t & \text{if } t \in \mathcal{V} \\ \emptyset & \text{otherwise} \end{cases} \tag{2.15}$$

The second approach, special token replacement, can be formalized as:

$$f_{\text{replace}}(t) = \begin{cases} t & \text{if } t \in \mathcal{V} \\ \text{<unknown>} & \text{otherwise} \end{cases} \tag{2.16}$$

While these approaches might appear equivalent for unigram models, their implications differ significantly when considering contextual relationships.

The importance of context in OOV handling becomes clear through an example:

- **Original:** This cotton **candy** is sweet.
- **Drop Strategy:** f_{drop}(“This cotton candy is sweet”) = “This cotton is sweet”
- **Replace Strategy:** f_{replace}(“This cotton candy is sweet”) = “This cotton <unknown> is sweet”

In this example, when “candy” $\notin \mathcal{V}$, the two strategies produce notably different results. The drop strategy creates a potentially misleading sequence, while the replace strategy maintains structural integrity while explicitly marking the unknown term’s position.

From a learning perspective, let $P(w_i|w_{i-1})$ be the probability of word w_i given its previous word (Manning and Schütze, 1999). With the drop strategy, the model might incorrectly learn P(“sweet”|“cotton”) is high. The replacement strategy instead models P(“sweet”|“<unknown>”) and P(“<unknown>”|“cotton”), preventing such spurious correlations.

However, OOV handling has limitations. When the proportion of tokens $t \notin \mathcal{V}$ in an input

sequence exceeds some threshold θ, the resulting representation becomes too sparse for reliable inference. Modern approaches have largely mitigated these challenges through subword tokenization (Sennrich et al., 2016; Schuster and Nakajima, 2012; Wu et al., 2016).

Let $\mathcal{V}^*$ be the subword vocabulary, and $f_{\text{subword}} : T \to \mathcal{V}^*$ be a subword tokenization function that maps a token to a sequence of vocabulary elements. For example:

$$f_{\text{subword}}(\text{“makes”}) = [\text{“make”}, \text{“\_s”}] \tag{2.17}$$

While detailed discussion of these algorithms follows in the next chapter, their core insight lies in preserving morphological information through this decomposition. Contemporary methods apply this principle at a finer granularity, but the fundamental approach of information preservation remains constant.

Subword tokenization methods have largely addressed OOV challenges for alphabetic writing systems, but they build on an assumption that text can be effectively decomposed into smaller units drawn from a limited character set. This assumption holds well for languages like English, where words are formed from combinations of a few dozen letters. However, text processing systems face a fundamentally different scale of complexity when handling Chinese, Japanese, and Korean (CJK) languages, where the base character set itself numbers in the thousands. These languages not only challenge our approaches to OOV handling but also require us to reconsider basic assumptions about character-level text processing.

2.5. Complexities of CJK Languages

While subword tokenization methods have largely addressed OOV challenges for alphabetic writing systems, they build on an assumption that text can be effectively decomposed into smaller units drawn from a limited character set. This assumption holds well for languages like English, where words are formed from combinations of a few dozen letters. However, text processing systems face a fundamentally different scale of complexity when handling Chinese, Japanese, and Korean (CJK) languages, where the base character set itself numbers in the thousands, as demonstrated contrasting English, French, and Korean in Figure 2.1. These languages not only challenge our approaches to OOV handling but also require us to reconsider basic assumptions about character-level text processing.

The fundamental challenge with CJK languages stems from their vast character sets. While alphabetic languages combine a small set of letters to form words, CJK languages require thousands

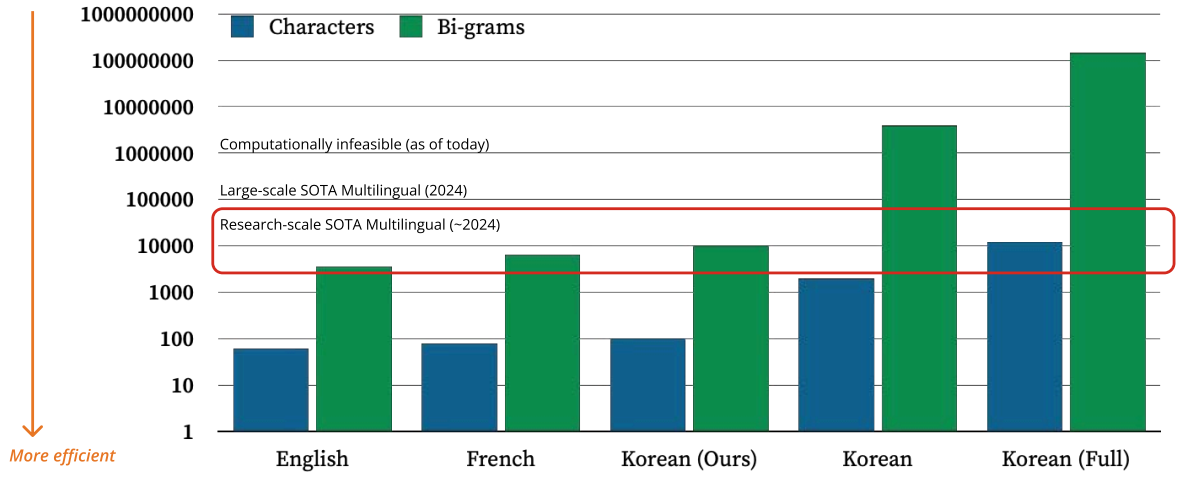


Figure 2.1: Character-level and character bi-gram vocabulary size bounds for different languages.

of distinct characters for basic literacy. This characteristic creates significant implications for text processing, particularly in vocabulary management and tokenization.

2.5.1. Chinese

Chinese presents perhaps the most direct example of this complexity through its use of ideographs (hanzi[3]). The complete set of Chinese characters spans nearly 100,000 Unicode code points, though most modern texts use a much smaller subset. This vast character space creates challenges for both computational representation and processing.

The evolution of Chinese text processing reflects these challenges. The initial GB 2312 standard supported only 6,763 ideographs, while its revision GB/T 2312-1980 expanded to 7,445 characters. For traditional Chinese, the Big5-1984 standard supported 13,053 characters. These limited character sets represented practical compromises between coverage and computational feasibility.

Unlike most languages which can be tokenized using whitespace as a delimiter, Chinese text lacks explicit word boundaries and uses minimal punctuation. This necessitates algorithmic approaches for initial tokenization, often combining language-specific heuristics with lexical knowledge to identify word boundaries. Early approaches focused on character sequence patterns and dictionary-based heuristics (Li and Yuan, 1998), which evolved into statistical methods (Xue, 2003; Shi, 2005) and later neural networks that can learn boundary patterns directly from data (Peng et al., 2004; Ma et al., 2018).

[3]Hanzi, kanji, and hanja are region-specific terms referring to the roughly the same set of ideographs. In a computational context, hanzi may refer to either traditional or simplified, while kanji refers to Japanese simplified (different than that of Chinese simplified), and hanja refers to traditional. However, Unicode codepoint-wise they are mutually interoperable (Unicode Consortium, 2024).

2.5.2. Japanese

Japanese text processing evolved through similar constraints. The initial JIS X 0201 standard supported only ASCII and 64 katakana characters. Later standards expanded to include 6,355 kanji characters, bringing the total to 6,879 characters (including 524 non-kanji characters). This mixed writing system, combining kanji with two phonetic scripts (hiragana and katakana), presents unique challenges for tokenization and vocabulary management.

Japanese also does not use spaces, but employs more extensive punctuation than Chinese, providing additional lexical boundary indicators. However, its agglutinative nature and complex grammatical structure, including multiple levels of formality and an intricate system of verb conjugation, create additional challenges. The morphological complexity is managed primarily through the hiragana writing system, which helps contain the character-level complexity despite the rich inflectional system (Kudo et al., 2004). The development of robust segmentation approaches for Japanese has progressed from rule-based systems through statistical approaches, to modern neural methods (Tolmachev et al., 2018).

2.5.3. Korean

Korean presents a distinct case where the challenge lies not in the writing system's inherent complexity, but in its computational representation. While Korean uses a phonetic alphabet (Hangul) with a relatively small set of basic characters (jamo), its writing system arranges these characters into two-dimensional morpho-syllabic blocks. For example, the syllable '한' (han) combines three jamo: 'ㅎ' (h), 'ㅏ' (a), and 'ㄴ' (n).

Historical computing constraints led to the adoption of syllable-based encoding schemes. Rather than representing the individual jamo components, systems stored pre-composed syllable blocks. The KSC-5601-1987 standard supported 2,350 such syllable blocks, while Unicode 1.1 expanded this to 11,172 precomposed syllables—a number that remains constant through Unicode 13.0.

Korean's uniqueness extends beyond its writing system to its spacing conventions. Unlike Chinese and Japanese, Korean uses spaces, but these spaces demarcate eojeol (어절) - units that can contain multiple morphemes. The rules governing morpheme agglutination into eojeol are relatively flexible, making simple whitespace tokenization insufficient for many applications. Korean verbs exhibit extensive inflectional morphology, with combinations of tense, aspect, honorifics, and grammatical modality forming a complex system of verbal expressions.

2.5.4. Implications for Text Processing

These large character sets create significant challenges for n-gram-based approaches. Given a character set $\hat{\Sigma}$, the potential number of character-level bigrams grows quadratically as $|\hat{\Sigma}|^2$. For CJK or other languages with ideographic script, this results in an exponentially larger vocabulary space compared to alphabetic languages.

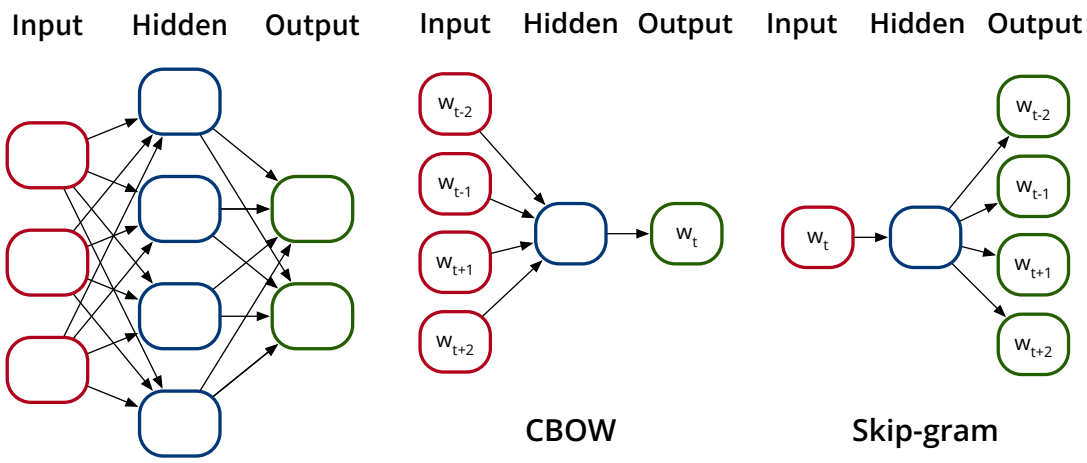
In practice, character frequency distributions in CJK languages follow Zipfian patterns (Zipf, 1949), making it possible to achieve reasonable coverage with a subset of characters. This frequency distribution often aligns with the character sets chosen for legacy encoding standards, as these standards were typically designed to support the most common characters in contemporary usage.

Beyond the sheer size of their character sets, each language presents distinct processing challenges. Chinese requires algorithmic word boundary detection due to its lack of explicit delimiters and minimal punctuation. Japanese combines multiple writing systems with complex grammatical structures, requiring sophisticated morphological analysis to handle its agglutinative properties. Korean, while using spaces, must contend with flexible word formation rules and extensive morphological transformations that occur at the sub-character level.

These characteristics compound to create particularly challenging out-of-vocabulary scenarios. While alphabetic languages typically encounter OOV issues at the word level, CJK languages face a multi-level OOV challenge. At the character level, rare or specialized characters may be entirely absent from the system's character set. At the word level, the combinatorial nature of character composition means that even when all individual characters are known, their combinations may create novel tokens never seen in training. For agglutinative cases like Korean and Japanese, morphological transformations can generate valid but previously unseen character combinations, further expanding the potential OOV space.

The traditional approaches of dropping or replacing unknown tokens become particularly problematic in this context. Dropping characters risks corrupting the fundamental meaning of expressions, while replacement strategies must contend with uncertainty at multiple linguistic levels. For instance, an unknown character in a Chinese word could change not just the word's meaning but also its grammatical role and word boundaries. Similarly, an unseen Korean syllable block might represent a crucial grammatical transformation that affects the entire sentence's interpretation.

Traditional approaches to these challenges often employed morphological analysis pipelines (Tseng and Chen, 2002; Kurohashi, 1994; Sang-ho, 1995) passing normalized forms to downstream



(a) An artificial neural network with input, hidden, and output layers. (Source)

(b) The continuous bag-of-words (CBOW) and skip-gram training schemes proposed by the word2vec architecture.

Figure 2.2: Examples of two shallow artificial neural network architectures.

components. However, these methods shared the limitations of other lossy transformations: they were only as robust as their normalization rules and often struggled with neologisms or novel usage patterns. Modern approaches, particularly those based on subword tokenization, have begun to address these limitations, though the fundamental challenge of managing exponentially larger vocabulary spaces remains a distinctive characteristic of CJK language processing.

2.6. Embeddings and Neural Networks

Our exploration of statistical methods showed how text could be transformed into mathematical objects through vocabulary-based representations, while our examination of tokenization and out-of-vocabulary handling revealed the challenges of maintaining effective discrete vocabularies. Neural approaches retain the fundamental concept of a vocabulary-based representation but transform how we use it: instead of treating vocabulary indices as the features themselves, each index maps to a vector of learned parameters. These learned representations, known as embeddings, form the foundation of more complex neural architectures for text processing - the most basic form is learnt through samples by a shallow neural network and the backpropagation algorithm (Mikolov et al., 2011).

An embedding is a mapping from a source domain into a vector space that preserves specific structural properties. More formally, it is an injective function that maps elements from the source domain to vectors of real numbers, where the mapping preserves certain mathematical structures we care about. In machine learning (ML) applications, these embeddings are typically learned from data, with the preserved structures determined by the learning objective and architecture of the

model.

While embeddings can be learned through various methods, neural networks have proven particularly effective at discovering useful embedding spaces. To understand how neural networks learn embeddings, we first need to examine their fundamental structure and learning mechanism.

A neural network, or more precisely, an artificial neural network, is a ML method inspired by biological neural networks. These models consist of artificial neurons (called nodes or units) connected by edges, forming a graph structure. Each neuron computes a weighted sum of its inputs, which is then transformed by an activation function to produce its output. More formally, each neuron computes:

$$o = \sigma(\sum_{i=1}^{n} w_i x_i + b) \tag{2.18}$$

where σ is the activation function, w_i are the weights, x_i are the inputs, and b is a bias term.

A set of these artificial neurons are defined as a layer, which in the simplest implementation would be three (input, hidden, and output). There can be multiple intermediate hidden layers, and the more layers the network has the deeper (in the sense of *deep learning*) it is deemed. The values of the hidden layers in this neural network, or so called *weights*, $\boldsymbol{\theta}$ - will be the product of learning from samples, or commonly referred to as - training.

Re-using our previous rule-based sentiment classification example, where we implemented the classification function $f(X)$ with a set of rules:

$$Y = f(X) = \begin{cases} \text{positive} & \text{if “happy”} \in X \\ \text{negative} & \text{if “sad”} \in X \\ \text{unknown} & \text{otherwise} \end{cases} \tag{2.19}$$

Instead imagine this classifier is re-implemented such that given input X, produces a sentiment Y, given weights $\boldsymbol{\theta}$, formulated as $Y = f(X; \boldsymbol{\theta})$. Whether any given $y_n \in Y$ is *happy* or *sad* is determined by the values learnt through samples, and encoded as knowledge in $\boldsymbol{\theta}$.

For each training iteration, the network computes a loss value measuring its prediction errors, calculates gradients through backpropagation, and updates the weights to minimize this loss. This process continues until either the loss stabilizes or a maximum number of iterations is reached. While the internal mechanics are more complex, this high-level understanding suffices for our discussion. For a comprehensive treatment, see Goodfellow et al. (2016).

This learning mechanism can be applied to learn embeddings for words using a simple shallow

neural architecture and no labeled samples. The concept of learning neural word embeddings was first introduced by Bengio et al. (2003), but achieved widespread adoption through word2vec (Mikolov et al., 2013), which proposed two novel training objectives. The theoretical foundation for these approaches rests on the *Distributional Hypothesis* about language and word meaning, which states that words which occur in the same context tend to have similar meanings (Harris, 1954).

An illustration of this can be seen in Figure 2.2b, where we have the continuous bag-of-words (CBOW) architecture, where the input is the surrounding words of a given word, and the output layer is expected to predict the word given the context. An alternative architecture - Skip-gram is also proposed, where it is the other way around; in this case the output layer must predict the surrounding words given the word.

Both CBOW and Skip-gram architectures optimize related but distinct objective functions. The Skip-gram model maximizes:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t) \tag{2.20}$$

where T is the length of the text sequence, c is the size of the context window, and the inner sum iterates over the context positions while excluding the target word position ($j \neq 0$). For each target word w_t, the model learns to predict its context words w_{t+j}. In contrast, CBOW maximizes:

$$\frac{1}{T}\sum_{t=1}^{T}\log p(w_t|\mathbf{h}_\mathrm{t}) \tag{2.21}$$

where $\mathbf{h}_\mathrm{t}$ represents the average of the context word vectors:

$$\mathbf{h}_\mathrm{t} = \frac{1}{2c}\sum_{-c\leq j\leq c, j\neq 0}\mathbf{x}_{\mathrm{t+j}} \tag{2.22}$$

Here, the model learns to predict the target word w_t from the average of its context word representations. While both approaches learn useful word embeddings, their different objectives lead to slightly different properties in the learned representations, with Skip-gram often performing better on rare words due to treating each context position independently.

Importantly, while the training scheme in a conventional machine learning sense is *supervised learning*, there are no labeled samples; the training is done through written text, which in the modern internet era can be sourced easily at scale, unlike annotated data which can be costly to create or source, especially in large quantities. This type of training regime is often called *self-supervised learning*.

It is worth noting that the input here is not a sparse BOW (where the majority of the scalars in the vector are zero) representation, it is a dense vector (where the majority of scalars in the vector are non-zero) for each word. This vector is a row in a embedding matrix that represents the entire vocabulary. The main product used for different downstream tasks after training is this embedding matrix.

An embedding matrix trained through word2vec establishes a one-to-one mapping between tokens in the vocabulary and their corresponding embedding vectors. While more sophisticated architectures have superseded word2vec, its key innovations—self-supervised learning on large text corpora and learned dense embeddings—remain fundamental to modern approaches. The vectors learn to capture distributional properties, with semantically or syntactically similar words located closer together in the embedding space. They even exhibit approximate algebraic relationships, famously demonstrated by examples such as:

$$\text{vec}(\text{“queen”}) \approx \text{vec}(\text{“king”}) - \text{vec}(\text{“man”}) + \text{vec}(\text{“woman”}) \tag{2.23}$$

In the next section, we examine in detail how these mechanisms come to effect in recent methods.

2.7. Transfer Learning

Traditional supervised machine learning assumes abundant labeled data for each task and domain. However, this assumption often fails in practice, leading to significant challenges in deploying machine learning systems at scale. Transfer learning is a practice that addresses these limitations by enabling models to leverage knowledge from related tasks and domains.

Following the survey of Ruder et al. (2019), we can disambiguate transfer learning starting from a classical supervised machine learning perspective. If we train a model for some task and domain, there is an implicit assumption that there is labeled data for the given task and domain. The model is trained on this dataset and is expected to perform reasonably well on unseen data of the same task and domain. In our sentiment analysis example, this would be done by a (ideally balanced) set of labeled samples. A model trained with this data is expected to be able to perform sentiment analysis on unseen examples, as long as the unseen samples are roughly of the same input domain (e.g. language, as the most obvious example).

However, this regime depends on a reasonably large set of samples which may not always be readily available; additionally, the produced models are only capable of one task-domain pair - and

as a product of the abovementioned limitations, scale poorly as there is a necessity to serve more tasks and domains.

Transfer learning addresses these limitations by leveraging data from a related task or domain, referred to as the *source task* and *source domain*. Knowledge acquired through training for the source task and domain is applied to a *target task* and *target domain*.

As Pan and Yang (2010) explains, a binary classification task such as our sentiment analysis example can be used to explain this content. A domain $\mathcal{D}$ consists of a feature space $\mathcal{X}$, and a marginal probability distribution $P(X)$ over the feature space, where $X = x_1, ..., x_n \in X$. Here, $\mathcal{X}$ is the space of all possible input features, and $x_i \in X$ is a single sample from the training set X.

Given a domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$, each task $\mathcal{T}$ consists of an output space $\mathcal{Y}$, a prior distribution $P(Y)$, and a learned conditional probability distribution $P(Y|X)$ from training data pairs $(x_i \in X, y_i \in \mathcal{Y})$. In our sentiment analysis example, this means $\mathcal{Y} = \{\text{positive, negative}\}$ and ideally $Y = \{\text{positive, negative}\}$.

Now, given a source domain-task pair $\mathcal{D}_s, \mathcal{T}_s$ and target domain-task pair $\mathcal{D}_t, \mathcal{T}_t$, where the domain and task inequality constraint is met $\mathcal{D}_s \neq \mathcal{D}_t$ and $\mathcal{T}_s \neq \mathcal{T}_t$, the objective of transfer learning is to learn the target conditional probability distribution $P_t(Y_t|X_t)$ in target domain $\mathcal{D}_t$.

Following the taxonomy defined in Pan and Yang., 2010, transfer learning can be either *inductive* or *transductive*, and within those two families there are leaf classifications. In *inductive transfer learning*, depending on the learning scheme across the source and target domains $\mathcal{T}_s, \mathcal{T}_t$, it can be either *sequential learning* if the ability to solve $\mathcal{T}_t$ is learned *after* $\mathcal{T}_s$ or *multi-task learning* if $\mathcal{T}_s, \mathcal{T}_t$ is learned in conjunction. Generally, supervised fine-tuning (SFT) schemes fall under *sequential learning*.

Under the *transductive* branch, if $\mathcal{T}_s \approx \mathcal{T}_t$ and $\mathcal{D}_s \neq \mathcal{D}_t$ holds, it falls under *domain adaptation*, which is what a continual pre-training (CPT) scheme optimizes for.

More formally, in domain adaptation, we can express the relationship between source and target domains mathematically. Let $D(\cdot, \cdot)$ be a distance metric between probability distributions (e.g. KL-divergence). The domain adaptation scenario is characterized by having significantly different marginal distributions between source and target domains:

$$D(P_s(X), P_t(X)) > \epsilon \tag{2.24}$$

where ϵ is some threshold that determines meaningful distributional difference. However, the key assumption enabling successful domain adaptation is that the conditional distributions remain

similar, given small threshold δ:

$$D(P_s(Y|X), P_t(Y|X)) < \delta \tag{2.25}$$

While there are other classifications within the taxonomy proposed by Pan and Yang (2010), *sequential learning* and *domain adaptation* are the only two variants addressed within the scope of the related methods discussed in later chapters and the methods proposed in this thesis.

The field's terminology has evolved over time. What was previously known as *sequential learning* is now commonly referred to as *supervised fine-tuning* (SFT), while *domain adaptation* has become known as *continual pre-training* (CPT). This thesis will use these contemporary terms in subsequent sections.

The success of transfer learning in computer vision inspired similar approaches in other domains. Pre-trained models, now commonly called foundation models, have become fundamental building blocks across machine learning. These models are trained on large-scale datasets and can be adapted to numerous downstream tasks with minimal task-specific training data.

Specifically, *sequential learning* has shown to be incredibly effective, and is now almost universally used in the domain of computer vision. This was done by initially training a large convolutional neural network (LeCun et al., 1998; Krizhevsky et al., 2012; Simonyan and Zisserman, 2014) against a large-scale classification dataset (Deng et al., 2009), and using the pre-trained model for classification for other tasks (Donahue et al., 2013; Girshick, 2015). This also applied to *domain adaptation*, in particular for domains lacking in large training sets. As of the time of writing, it is uncommon to find newly proposed methods which do not employ some form of transfer learning; to a point where not doing so could be considered a risk (Mahajan et al., 2018). These pre-trained models are now commonly referred to as *foundation models*.

The next section covers basics of a language model in NLP, and how these language models have evolved to become foundation models.

2.8. Language Models and Foundation Models in NLP

Earlier, we surveyed a basic neural word embedding architecture - word2vec (Mikolov et al., 2013) and discussed the high-level intuition behind transfer learning. In this section, we will discuss how these two schemes intertwine when transfer learning in NLP becomes the eventual goal.

A language model, in the simplest possible explanation, is a model that is trained to predict a word given some context. A common use of a language model is the autocomplete feature provided

in many software packages - for example - software keyboards provided in everyday mobile devices. More formally, it serves the purpose of assigning a probability to sentences in a language - e.g. "what is the probability of seeing the sentence *Good morning*?", but also assigning a probability for the likelihood of a word to appear given context - e.g. "what is the probability of seeing the word *you*, given the past context is *how are*?

While language models are commonly constrained to predicting words from past context, this is not a strict requirement. Models trained on other prediction tasks also qualify as language models. For example, word2vec's CBOW architecture predicts missing words from surrounding context (e.g., *I bought a ____ at Starbucks*), while its Skip-gram architecture predicts words at varying distances from the input word. However, the most common form is to predict the next token, given past words - given an arbitrary sequence of words $w_{1:n}$, estimating the probability $P(w_{1:n})$ is done as follows:

$$\begin{aligned} P(w_{1:n}) &= P(w_1)P(w_2|w_1)P(w_3|w_{1:2})...P(w_n|w_{1:n-1}) \\ &= P(w_1)\prod_{i=2}^{n} P(w_i|w_{1:i-1}) \end{aligned} \tag{2.26}$$

As discussed earlier, one powerful property of training a language model is that they are capable of being trained without labeled data; only needing written text, which can be sourced in large quantities from the internet. This means that as long as the model is capable of learning more, all one needs to do is to source more text for training. Recently proposed scaling laws (Kaplan et al., 2020; Wei et al., 2022; Hoffmann et al., 2022) suggests that models are capable of storing more knowledgeas long as the model parameter size is scaled proportional to the quantity of training data.

However, before these laws were discovered, seminal work such as BERT (Devlin et al., 2019) and ULMFiT(Howard and Ruder, 2018) empirically demonstrated that language models trained on large volumes of text are able to serve as foundation models for NLP. BERT introduced masked language modeling and next sentence prediction objectives, enabling the model to learn bi-directional context - a novel advancement over previous unidirectional approaches. Both methods employ sequential training schemes with optional domain adaptation, though they differ in their underlying neural architectures and training objectives. BERT, which is used in one of our proposed methods, is discussed in further detail in the next chapter.

Despite these advances in transfer learning for NLP, a fundamental technical challenge persists in

vocabulary management. The vocabulary of these models must typically remain fixed throughout the transfer learning process for two key reasons. First, modifying a model's vocabulary requires retraining the embedding layers, which is computationally expensive for large-scale models. Second, vocabulary modifications can trigger catastrophic forgetting, where the model loses previously acquired knowledge during the adaptation process. These constraints mean that core challenges, such as handling out-of-vocabulary tokens and adapting to domain-specific terminology, remain significant obstacles even as models are transferred to new domains. The next section discusses the trade-offs between computational efficiency and linguistic precision in vocabulary management, examining how different approaches balance these competing demands.

2.9. Robustness and Efficiency, and its Trade-offs

The preceding sections have traced the evolution of NLP methods from traditional to statistical approaches, establishing tokenization as a fundamental component of NLP systems. We examined the challenges of vocabulary coverage in tokenization, with particular attention to the complexities of processing CJK text. We then explored transfer learning and its implementation through language models, identifying tokenizer vocabulary as a persistent bottleneck in these methods. Building on these foundations, this section examines two critical concepts—efficiency and robustness—that underpin the novel methods proposed in subsequent chapters. Understanding these concepts is essential for appreciating both the motivation behind our approaches and their practical impact on NLP systems.

2.9.1. Robustness

Robustness is a metric that measures a tokenizer's ability to reliably encode and decode information. The underlying assumption is that a given tokenizer is potentially, but not guaranteed to be, a lossy encoder, where a round trip encoding-decoding process may introduce information loss. This can be formally expressed using domain vocabulary and character sets.

Let $\hat{\mathcal{V}}$ represent all possible tokens in the domain, and $\mathcal{V}$ represent the tokens observed during training, where $\mathcal{V} \subset \hat{\mathcal{V}}$. Similarly, let $\hat{\Sigma}$ represent all possible characters across all tokens in the domain, and Σ represent the characters observed across $\mathcal{V}$ during training, where $\Sigma \subset \hat{\Sigma}$.

Given a vocabulary budget constraint k and training set X, we define a trained vocabulary $\mathcal{V}' \subset \mathcal{V}$ where $|\mathcal{V}'| \leq k$. While $\mathcal{V}' \subset \mathcal{V}$ holds by definition, ideally we want $\mathcal{V}' = \mathcal{V}$ for optimal coverage. The token-level information loss δ for a training sample $x \in X$ can be expressed as:

$$\delta(x, \mathcal{V}') = \frac{1}{|x|} \sum_{t \in \text{tok}(x, \mathcal{V}')} I[t \notin \mathcal{V}'] \tag{2.27}$$

where $\text{tok}(x, \mathcal{V}')$ is a function that returns a sequence of tokens given input x and vocabulary $\mathcal{V}'$, and I is the indicator function that returns 1 if the token is not in vocabulary $\mathcal{V}'$. This formulation directly measures the proportion of out-of-vocabulary tokens for a given sample.

The expected information loss Δ across the entire training set is then:

$$\Delta = \frac{1}{|X|} \sum_{x \in X} \delta(x, \mathcal{V}') \tag{2.28}$$

An intuitive example of information loss occurs with out-of-vocabulary tokens, where any token receiving either drop or replacement treatment will be lost after the round-trip encoding-decoding process. Information loss is inevitable when $\mathcal{V}' \neq \hat{\mathcal{V}}$ and, particularly for CJK languages, when $\Sigma \neq \hat{\Sigma}$.

The relationship between robustness and model transferability can be formalized in the context of domain adaptation. Given source and target domains $\mathcal{D}_s$ and $\mathcal{D}_t$, the ability to adapt to $\mathcal{D}_t$ is inversely related to tokenization information loss. Higher information loss impairs the model's capacity to learn meaningful correlations during both continual pre-training (CPT) and supervised fine-tuning (SFT). This relationship can be expressed through the KL-divergence between domains for CPT:

$$\min_{V \subset \mathcal{V}} \text{KL}(\mathcal{D}_t || \mathcal{D}_s) \text{ subject to } \Delta \leq \epsilon \tag{2.29}$$

where ϵ is an information loss threshold that must be maintained for effective transfer.

Robustness also impacts model generalization. Consider a training subset $\mathcal{D}'_s$ of the source domain $\mathcal{D}_s$, where $\mathcal{D}'_s \subset \mathcal{D}_s$. Low robustness, characterized by high information loss Δ, leads to poor generalization within $\mathcal{D}_s$ and diminished transfer capability to $\mathcal{D}_t$. Furthermore, in text generation applications, high information loss particularly compromises output quality, as the model's ability to produce coherent text depends critically on the fidelity of its token representations.

2.9.2. Efficiency

Related to robustness, efficiency is a metric that quantifies the downstream computation cost of a tokenizer. A tokenizer with a smaller vocabulary size typically requires fewer computational resources than one with a larger vocabulary if both are assumed to completely cover the corpus.

This relationship is produced by two key mechanisms: a larger vocabulary requires more embedding weight parameters to fully capture in training, and producing output probability distributions over a larger vocabulary is directly associated with a greater computational cost (Yang et al., 2017; Kanai et al., 2018).

We can formalize tokenizer efficiency η as a normalized measure based on vocabulary size, given a trained vocabulary $\mathcal{V}'$, assuming a zero information loss scenario:

$$\eta(\mathcal{V}') = 1 - \frac{\log(|\mathcal{V}'|)}{\log(k)} \tag{2.30}$$

where k is the maximum vocabulary budget constraint. This formulation ensures that efficiency is bounded between 0 and 1, where:

$$\eta(\mathcal{V}') = \begin{cases} 1 & \text{when } |\mathcal{V}'| = 1 \text{ (maximum efficiency)} \\ 0 & \text{when } |\mathcal{V}'| = k \text{ (minimum efficiency)} \end{cases} \tag{2.31}$$

The logarithmic scaling reflects the sub-linear relationship between vocabulary size and computational costs, while normalization allows meaningful comparison across different vocabulary budget constraints. However, maximizing η alone is insufficient, as it must be balanced against the information loss Δ we discuss in the context of robustness:

$$\max_{\mathcal{V}' \subset \mathcal{V}} \eta(\mathcal{V}') \text{ subject to } \Delta \leq \epsilon \tag{2.32}$$

where ϵ is the maximum acceptable Δ threshold, optimally $\epsilon = 0$.

From an information theory perspective (Shannon), efficiency can be quantified as the minimum number of bits needed to encode a sequence while maintaining fidelity. The theoretical lower bound is a single bit (Zongker, 2006), achieved when the vocabulary contains just one token. However, such extreme compression renders the model practically useless due to loss of information to reproduce the original signal (Zipf, 1949), due to high information loss.

The efficiency of a tokenizer can be measured by its compression capabilities (Gallé, 2019; Ali et al., 2024; Goldman et al., 2024). In attention-based architectures, where computational complexity increases quadratically with sequence length, the relationship between vocabulary size and sequence length creates an optimization challenge.

While larger vocabularies can achieve more efficient tokenization and shorter sequences, this advantage is constrained by the increased computational demands these vocabularies place on

other model components. Conversely, minimizing vocabulary size leads to shorter tokens and consequently longer sequences, which increases computational (and monetary) costs (Ahia et al., 2023). An optimal solution requires striking a careful balance between vocabulary size and sequence length to minimize overall computational requirements while maintaining model performance.

Given these efficiency considerations, this work examines both vocabulary size and sequence length, while prioritizing vocabulary efficiency within acceptable information loss bounds. Specifically, our objective is to develop tokenization methods that maintain or minimize sequence length while improving tokenizer efficiency through optimal vocabulary utilization. This approach ensures that improvements in computational efficiency do not come at the cost of model performance or robustness.

2.10. Trade-offs

As discussed earlier with the single-word vocabulary example, efficiency can be increased by sacrificing robustness, and vice-versa. The mutually competing goals of training a high-quality tokenizer can be summarized as a convex optimization problem to jointly minimize the following:

1. Information loss (OOV) (Moon and Okazaki, 2020b)
2. Unnecessary rare tokens in-vocabulary (Land and Bartolo, 2024)
3. Vocabulary size (Sennrich et al., 2016)
4. Sequence length (Ahia et al., 2023)

This has to be a balancing act, as we cannot simply optimize against one objective function, as that will inevitably result in a overall worse tokenizer.

For example, increasing vocabulary size infinitely can reduce both sequence length and information loss. While this effectively reduces computational cost from a sequence length perspective and eliminates OOV issues, it becomes computationally infeasible beyond a certain size, and setting an upper bound for the vocabulary budget becomes increasingly challenging. Additionally, the long tail results in many tokens being undertrained in this scenario, leading to poor model quality (Land and Bartolo, 2024). As a result, an anecdotal upper bound chosen by the largest models as of this writing tends to be approximately 150,000.

2.11. Summary

With all the background discussed so far, we can make a short summary of the points to better justify the motivation of our work.

- In NLP, traditional rule-based methods had limits on generalizability and were fragile to domain shift, thus were superseded by statistical methods that allowed learning from samples.
- Statistical methods are able to correlate nonobvious relations from input signal to the expected output label, and hence benefit by maximizing information preservation.
- Tokenization methods have evolved to serve the information preservation needs of statistical models.
- Character diversity and complex grammar make robust tokenization a challenging task in CJK languages.
- Models trained using large amounts of training data for an unrelated task are effective at improving downstream task performance; also known as transfer learning.
- Transfer learning still operates with a fixed vocabulary, so any downstream models can be hindered by a suboptimal tokenizer.
- The vocabulary generalization problem is further amplified in CJK languages.
- Therefore, in a transfer learning setup, the vocabulary is an important component for the generalizability of the foundation model.
- When a vocabulary is defined, careful consideration of the trade-offs are needed to balance information loss, computability, and sequence length (computation cost).

While current tokenization techniques have advanced significantly, they face notable limitations when applied to Korean due to its unique linguistic properties. To better understand the state of the art and identify opportunities for innovation, the next chapter critically reviews related work, focusing on tokenization strategies for character-diverse languages and recent advancements in NLP.

3

Related Work

3.1. The Bounds of Relation

It is rather challenging and even potentially contentious to firmly define the boundary of related work, especially with recent developments. Much of the literature surrounding modern-day foundation models such as those commonly referred to as large-language models (LLM), is built upon the same foundations that this work touches on.

3.2. Foundation Models

3.2.1. Sequence Modeling with Neural Networks

In this section, we briefly discuss how neural networks are used to process and generate sequences which can be used for many purposes, including NLP. As of the time of writing, almost every input modality, where obvious cases such as textual, aural, and time-series have existed, but newer schemes touching less obvious modalities such as visual (Dosovitskiy et al., 2020) and even conventional tabular data (Huang et al., 2020) has a corresponding architecture that utilizes sequence modeling, largely attributable to the popularity and capabilities of the universally applicable Transformer architecture.

Recurrent Models and Long-Short Term Memory

A recurrent neural network (RNN) (Rosenblatt, 1961) processes sequential data by maintaining an internal state that evolves over time. The key insight behind RNNs is that they capture sequential dependencies by allowing information to persist across multiple processing steps. Unlike FFNSs that process each input independently, RNNs maintain a “memory” of previous inputs through their hidden state.

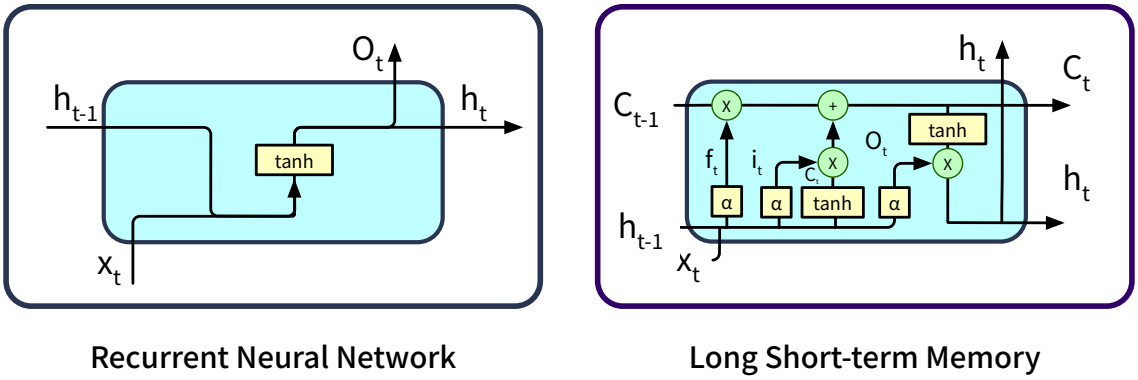


Figure 3.1: RNN compared with LSTM.

This hidden state mechanism can be thought of as the network's working memory, updated with each new input while retaining information from previous inputs:

$$h_t = \phi(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{3.1}$$

Here, the hidden state h_t at time t combines the current input x_t with the previous hidden state h_{t-1}. The weights W_{xh} and W_{hh} determine how much importance to give to the current input versus previous information, while ϕ is a nonlinear activation function that allows the network to learn complex patterns.

Training these networks requires an extension of the standard backpropagation algorithm called backpropagation through time (BPTT) (Werbos, 1990). BPTT unfolds the recurrent network across time steps and computes gradients through this unfolded network. However, this temporal dependency creates a significant challenge: as gradients flow backward through time, they tend to either vanish or explode (Bengio et al., 1994). This can be seen in the gradient computation:

$$\frac{\partial L}{\partial h_t} = \prod_{i=t}^{T} \frac{\partial h_{i+1}}{\partial h_i} \tag{3.2}$$

The repeated multiplication of terms in this equation means that gradients either become vanishingly small or explosively large over many time steps, making it difficult for RNNs to learn long-range dependencies.

To address these limitations, researchers developed two main architectures: long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and gated recurrent units (GRU) (Cho et al., 2014). LSTMs, which became particularly prominent in NLP applications, introduce a sophisticated gating mechanism that controls information flow. These gates act like regulators, determining which information to keep, update, or discard:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad \text{(forget gate)} \tag{3.3}$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad \text{(input gate)} \tag{3.4}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad \text{(output gate)} \tag{3.5}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_c[h_{t-1}, x_t] + b_c) \quad \text{(cell state)} \tag{3.6}$$

$$h_t = o_t \odot \tanh(c_t) \quad \text{(hidden state)} \tag{3.7}$$

The forget gate f_t controls what information to discard from the cell state, the input gate i_t determines what new information to store, and the output gate o_t controls what parts of the cell state should be output. This gating mechanism allows LSTMs to maintain relevant information over many time steps while avoiding the vanishing gradient problem.

These recurrent architectures are a natural choice as a foundation for language models, as they share similarities to how humans write text. (Peters et al., 2018) demonstrated that bi-directional LSTMs (BiLSTMs) can produce contextualized representations that significantly outperformed static embeddings on downstream tasks. (Howard and Ruder, 2018) further advanced the field with ULMFiT, an LSTM-based model pre-trained on Wikitext-103 (Merity et al., 2016). Their approach introduced a two-phase training procedure combining continual pre-training (CPT) and supervised fine-tuning (SFT), achieving substantial improvements over previous methods.

Transformers: Attention is All You Need

The Transformer (Vaswani et al., 2017) architecture represents a paradigm shift in sequence modeling, departing from recurrent architectures in favor of a purely attention-based approach. Originally proposed for machine translation, the architecture consists of stacked encoder and decoder layers, processing source and target sequences respectively through self-attention mechanisms.

Building upon the encoder-decoder framework (Sutskever et al., 2014), the Transformer maps an input sequence $X = (x_1, \ldots, x_n)$ to an intermediate representation $Z = (z_1, \ldots, z_n)$ via the encoder, which the decoder then transforms into an output sequence $Y = (y_1, \ldots, y_m)$. Unlike RNNs, the Transformer processes all sequence elements simultaneously through self-attention mechanisms (Bahdanau et al., 2014).

A fundamental building block of the architecture is the scaled dot-product attention mechanism:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.8}$$

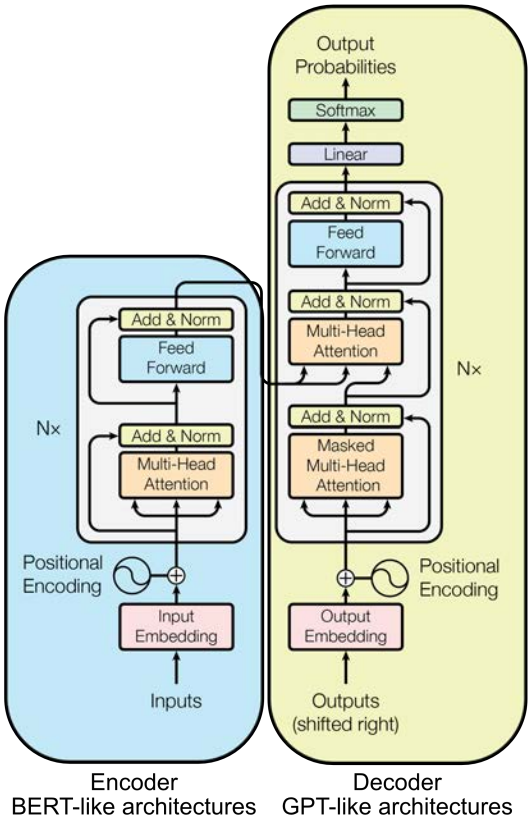


Figure 3.2: The Transformer architecture, separated by the encoder and decoder.

where Q, K, and V represent queries, keys, and values respectively, and d_k is the dimensionality of the key vectors. This mechanism is extended to multi-head attention:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \tag{3.9}$$

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{3.10}$$

To preserve sequence order information in this parallel processing framework, the Transformer employs sinusoidal positional encodings:

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{\text{model}}}) \tag{3.11}$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{\text{model}}}) \tag{3.12}$$

Each encoder and decoder layer contains a position-wise feed-forward network (FFN) following the attention mechanisms:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{3.13}$$

The complete architecture includes residual connections and layer normalization after each sub-layer:

$$\text{LayerNorm}(x + \text{Sublayer}(x)) \tag{3.14}$$

The next two sections demonstrate two seminal foundation models built using methods proposed in the original Transformers work.

3.2.2. GPT: Generative Pre-trained Transformer

The Generative Pre-trained Transformer (GPT) (Radford and Narasimhan, 2018) marked a significant advancement in language modeling by demonstrating the effectiveness of large-scale pre-training using the Transformer architecture. Unlike the original Transformer model designed for translation, GPT adapts the architecture for generative language modeling, using only the decoder component (3.2) with unidirectional attention.

The architecture comprises a 117M parameter model trained on a diverse text corpus using self-supervised learning. GPT employs byte-pair encoding (BPE) (Sennrich et al., 2016) for tokenization, allowing it to handle arbitrary text while maintaining a manageable vocabulary size. For downstream tasks, the model attaches task-specific layers and undergoes supervised fine-tuning (SFT).

GPT's pre-training objective follows the conventional autoregressive language modeling framework, maximizing the likelihood of predicting the next token given the previous tokens:

$$L(\theta) = \sum_i \log P(x_i | x_{<i}; \theta) \tag{3.15}$$

where $L(\theta)$ represents the log likelihood of the entire sequence under the model parameters θ. For each position i in the sequence, the model computes the conditional probability of token x_i given all previous tokens $x_{<i}$. This formulation captures the autoregressive nature of the model, where each prediction depends only on the tokens that came before it.

The model achieves this through a stack of Transformer decoder blocks that process input tokens sequentially:

$$h_0 = WE + PE \tag{3.16}$$

$$h_l = \text{transformer\_block}(h_{l-1}) \quad \forall l \in [1, n] \tag{3.17}$$

$$P(x) = \text{softmax}(h_n W_e^T) \tag{3.18}$$

where WE represents token embeddings, PE denotes positional encodings, and each transformer block consists of:

$$\text{transformer\_block}(h) = \text{LN}(h + \text{FFN}(\text{LN}(h + \text{MHA}(h)))) \tag{3.19}$$

$$\text{MHA}(h) = \text{MultiHead}(h, h, h; \text{mask}) \tag{3.20}$$

This architecture enables GPT to generate remarkably coherent text within its context window, while also learning representations useful for downstream tasks. The success of this approach led to increasingly larger models, culminating in systems like GPT-2 (Radford et al., 2019), GPT-3 (Brown et al., 2020), and PaLM (Chowdhery et al., 2022), which power popular conversational AI systems.

While our work does not directly experiment with GPT-based architectures [1], we discuss this decoder-based autoregressive architecture as a reference to contrast to that of an encoder-based architecture in the next section.

3.2.3. BERT: Bi-directional Encoder Representations from Transformers

BERT (Devlin et al., 2019) emerged contemporaneously with GPT, introducing a different approach to pre-training that achieved state-of-the-art performance across numerous downstream tasks. While both models utilize the Transformer architecture and employ self-supervised pre-training followed by task-specific fine-tuning, BERT's approach differs fundamentally in both architecture and training objectives. BERT was released in two sizes for English: a base model (110M parameters) and a large model (340M parameters). Additionally, BERT includes a multilingual variant, which is only available in the base size of 110M parameters, which is used in our proposed method in the upcoming chapter. [2]

Unlike GPT's unidirectional decoder architecture, BERT uses the Transformer's encoder (3.2)

[1] GPT-1 and GPT-2 were trained exclusively on English data, making them unsuitable for our work with Korean text.
[2] There are also cased and uncased variants, which we do not discuss.

blocks, allowing it to process input sequences bi-directionally. This architectural choice enables BERT to consider both left and right context when building representations. The pre-training phase employs two complementary objectives: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP).

In MLM, BERT randomly masks input tokens and learns to predict these masked tokens using bi-directional context. This can be formalized as:

$$L_{MLM}(\theta) = \mathbb{E}_{x\sim D}\left[\sum_{i\in\mathcal{M}} \log P(x_i|x_{\mathcal{M}};\theta)\right] \tag{3.21}$$

where $\mathcal{M}$ represents the set of masked token positions, $x_{\mathcal{M}}$ denotes the input with masked tokens, and D is the training corpus.

The NSP objective teaches the model to understand relationships between sentences by predicting whether two sentences appear consecutively in the original text:

$$L_{NSP}(\theta) = \mathbb{E}_{(s_1,s_2)\sim D}\left[\log P(y|s_1, s_2;\theta)\right] \tag{3.22}$$

where y indicates whether sentence s_2 follows s_1 in the original text.

The combined training objective becomes:

$$L(\theta) = L_{MLM}(\theta) + L_{NSP}(\theta) \tag{3.23}$$

This fundamental difference in training objectives reflects the distinct goals of these architectures. While GPT's autoregressive objective naturally suits text generation tasks, BERT's bi-directional approach excels at understanding tasks that benefit from full contextual awareness. GPT learns to generate coherent text sequences, developing useful representations as a byproduct. In contrast, BERT directly optimizes for rich contextual representations, though it sacrifices the ability to generate text autoregressively.

From our perspective of examining tokenization and transfer learning, both architectures serve as sophisticated sequence processors, transforming input tokens into contextual representations. Their architectural differences, while significant for their respective applications, become less relevant when viewing them as sequence processors from a tokenizer perspective, and conversely, from a model's perspective the tokenizer is simply a discrete encoder. In the context of discussing specific architectures, the tokenizer and model are considered as a pair, but in practice they are components which operate independently from each other. In the next section, we discuss the

tokenizers these foundation models use.

3.3. Tokenization

As introduced in 2.3, tokenization is an initial processing step for most NLP systems that breaks an arbitrary sequence of text into processable tokens. This transformation is necessary for algorithms that operate in discrete time steps, such as those used in language models. In this section, we survey the tokenization algorithms that serve as the foundation for our methods.

3.3.1. Segmentation and Subword Tokenization

Segmentation and *tokenization* are strongly related, but not strictly identical concepts. The definition by Universal Dependencies v2 (Nivre et al., 2020) differentiate based on whether it is intra-word (segmentation) or inter-word (tokenization).

While this definition holds well for many languages, languages that are classified as *scriptio continua*, such as CJK languages have a much more fuzzier boundary; based on the processing scheme, it is more likely they fall under segmentation, especially Japanese and Chinese which do not use spacing. Korean on the other hand does use spaces, but after tokenizing using whitespace as a delimiter, the tokens are not yet at word-level, as we discussed in Section 2.5). Traditional methods which break the document into smaller linguistic units, for example, through a morphological analyzer, are performing segmentation.

Smaller-than-word methods such as WordPiece (Schuster and Nakajima, 2012; Wu et al., 2016), BPE (Sennrich et al., 2016) and SentencePiece (Kudo and Richardson, 2018), which include breaking contiguous character sequences such as words apart, are in the strictest sense performing segmentation. These methods are commonly referred to as *subword tokenizers*. While there are semantic nuances that differentiate the two, in practice it is often used interchangeably. In our work, we will not be treating them separately, and will be using a single term - *tokenization* for simplicity.

These subword tokenization methods have become prevalent in modern NLP systems for several key reasons. They effectively balance vocabulary size with semantic coverage by breaking rare words into meaningful subunits, helping models handle out-of-vocabulary words, and morphologically rich languages. For example, the word “unimaginable” might be broken into “un”, “imagin”, and “able”, allowing the model to understand its meaning through its components. This approach has proven particularly valuable for multilingual models, where vocabulary size constraints would otherwise limit coverage across languages.

```
Input: set of strings D, target vocab size k
procedure BPE(D, k)
    V ← all unique characters in D
    while |V| < k do                          ▷ Merge tokens
        t_L, t_R ← Most frequent bigram in D
        t_new ← t_L + t_R                     ▷ Make new token
        V ← V + [t_new]
        Replace each occurrence of t_L, t_R in
            D with t_new
    end while
    return V
end procedure
```

```
Input: set of strings D, target vocab size k
procedure UnigramLM(D, k)
    V ← all substrings occurring more than once in D
    while |V| > k do                          ▷ Prune tokens
        Fit unigram LM θ to D
        for t ∈ V do                          ▷ Estimate token 'loss'
            L_t ← p_θ(D) − p_θ'(D)
            where θ' is the LM without token t
        end for
        Remove min(|V| − k, ⌊α|V|⌋) of the
        tokens t with highest L_t from V,
        where α ∈ [0, 1] is a hyperparameter
    end while
    Fit final unigram LM θ to D
    return V, θ
end procedure
```

Figure 3.3: BPE compared to Unigram LM.

Some recent work, such as that of Schmidt et al. (2024), refers to the step of whitespace tokenization as pre-tokenization; in particular this differentiation is rooted to the residual processing of whitespaces into the tokenizer vocabulary (Gow-Smith et al., 2022; Jacobs and Pinter, 2022). This is mentioned for completeness on the subject matter, and is not investigated within the scope of this thesis.

In the upcoming sections, the set of all possible unicode characters (within scope of the target language of the model) is denoted as $\hat{\Sigma}$, all characters seen in corpus as Σ, and characters in the final vocabulary Σ'. Similarly, following the same convention, word-level vocabulary is indicated as $\hat{\mathcal{V}}$, $\mathcal{V}$, and $\mathcal{V}'$. Subword-level vocabulary is differentiated from $\mathcal{V}$ with a $*$, e.g. $\mathcal{V}^*$ and $\mathcal{V}'^*$. σ is every possible byte (0x00-0xFF).

3.3.2. WordPiece

WordPiece (Schuster and Nakajima, 2012; Wu et al., 2016) is the earliest data-driven method for subword tokenization [3]. The method was originally developed to address vocabulary size limitations in Japanese and Korean voice search systems.

The algorithm begins with a character-level unigram language model using all Unicode characters $\hat{\Sigma}$ from the target language as its initial vocabulary, such that $\hat{\Sigma} = \Sigma$. It then follows an iterative process: the model samples bigrams and evaluates them based on likelihood maximization against the training data's distribution. When the algorithm identifies a candidate that maximizes likelihood, it adds the merged subword unit to the vocabulary. This process continues until reaching a predetermined vocabulary size limit, $k = |\mathcal{V}^*|$.

[3] (Mikolov et al., 2011) proposed a subword language model earlier, but it relied on pre-defined heuristics for subword boundaries, requiring manual tuning for each language.

To illustrate the tokenization process, consider this example:

- **Original**: Jet makers feud over seat width with big orders at stake
- **WordPiece**: _J et _makers _fe ud _over _seat _width _with _big _orders _at _stake

By design, the trained subword vocabulary $\mathcal{V}'^*$ includes all characters from the initial character set, ensuring $\Sigma \subset \mathcal{V}'^*$, barring any undocumented vocabulary pruning steps.

This tokenization method gained widespread adoption through its use in BERT (Devlin et al., 2019) and serves as the foundation for our first method presented in later chapters. However, several aspects of WordPiece remain understudied in the literature, particularly its end-to-end training process and its handling of whitespace, punctuation, and other preprocessing heuristics.

3.3.3. BPE: Byte Pair Encoding

Sennrich et al. (2016) adapted Byte Pair Encoding, a general-purpose compression algorithm originally proposed by Gage (1994), to address the out-of-vocabulary (OOV) problem in neural machine translation. BPE differs from WordPiece in two fundamental aspects: its deterministic frequentist approach and its vocabulary initialization strategy.

Unlike WordPiece's probabilistic language model approach, BPE follows a deterministic process based on frequency counts. Additionally, while WordPiece initializes its vocabulary with the complete Unicode range of target languages, BPE constructs its initial character vocabulary Σ solely from characters observed in the training corpus.

The BPE algorithm operates iteratively:

1. Initialize the vocabulary with characters from the training corpus
2. Count frequencies of adjacent character pairs (bigrams) in the corpus
3. Merge the most frequent bigram pair into a new token
4. Add the new merged token to the vocabulary
5. Repeat steps 2-4 until reaching the target vocabulary size $k = |\mathcal{V}^*|$

For example, given the word *low* appearing frequently in a corpus, BPE might perform these merges:

- Initial: l o w e r
- After first merge: lo w e r
- After second merge: low e r

- Final: lower

Despite its name referencing bytes, standard BPE implementations typically operate on Unicode characters rather than bytes. However, byte-level variants have emerged for specific applications. Wang et al. (2020) proposed a true byte-level BPE, while recent large language models employ either a hybrid approach combining subword tokens with byte-level fallback ($\mathcal{V}' + \sigma$) or pure byte-level operation with byte-based subwords ($\Sigma = \sigma$). The byte-level approach offers universal coverage across all possible inputs, eliminating the out-of-vocabulary problem entirely at the cost of potentially longer token sequences. As of the time of writing, BPE is the most widely-used subword tokenization method, and is used by GPT and many of its variants.

3.3.4. SentencePiece

SentencePiece (Kudo and Richardson, 2018; Kudo, 2018) represents a significant evolution in subword tokenization, offering both BPE and a novel unigram language model-based approach. While its unigram method shares conceptual similarities with WordPiece, SentencePiece differs in its vocabulary construction strategy.

Unlike WordPiece's bottom-up merging or BPE's frequency-based approach, SentencePiece's unigram method employs a top-down pruning strategy. It begins with a large seed vocabulary $\hat{\mathcal{V}}^*$ derived from the most frequent substrings in the corpus, then iteratively removes tokens to reach the target vocabulary size $k = |\mathcal{V}^*|$ through Expectation Maximization (EM).

The SentencePiece algorithm proceeds through EM iterations as follows:

1. Initialize a large vocabulary from frequent substrings
2. Expectation step:
   (a) Compute the most likely segmentation for current vocabulary
   (b) Calculate maximum-likelihood using Viterbi algorithm
3. Maximization step:
   (a) Update subword occurrence probabilities
   (b) For each token x_i, compute its removal impact:

$$loss_i = \log P(D) - \log P(D \setminus x_i) \tag{3.24}$$

where D denotes the training corpus as a sequence of characters, $P(D)$ represents the

probability of generating the entire corpus under the current model, and $D \setminus x_i$ indicates the corpus modeled without token x_i in the vocabulary

(c) Remove tokens with minimal impact on corpus likelihood

4. Repeat steps 2-3 until reaching target vocabulary size
5. Generate final model with pruned vocabulary

SentencePiece has two operational modes - unigram and BPE, where the latter closely follows standard BPE implementations.

3.4. Related Methods

In this section, we examine methods that are closely related to our proposed approaches, either as direct competitors addressing the same problem space or as parallel developments that arrived at similar solutions while targeting different challenges. This examination provides context for our work and highlights the distinctions of our approach.

As discussed in Section 2.3, byte-level BPE (BBPE) (Wang et al., 2020) provides a foundational approach to handling unicode coverage by operating at the byte level rather than the character level. While byte-level processsing effectively addresses universal text coverage (Xue et al., 2022), it introduces additional complexity in token sequence length, particularly for non-ASCII text.

3.4.1. Vocabulary-free Methods

A significant line of research has explored methods that eliminate the need for explicit vocabulary construction, offering alternatives to traditional tokenization approaches. These methods aim to address scalability challenges and reduce preprocessing complexity while maintaining model performance.

Weinberger et al. (2009) introduced feature hashing for large-scale machine learning (ML) models, demonstrating that random projections through hashing could effectively reduce dimensionality while preserving important feature relationships. This technique, also known as the “hashing trick,” maps high-dimensional feature spaces to lower-dimensional spaces without requiring explicit vocabulary maintenance, though at the cost of potential hash collisions.

Building on this foundation, Bojanowski et al. (2017) and Joulin et al. (2017) applied similar principles to text classification with their fastText approach. They employed n-gram feature hashing to capture partial word information without maintaining a fixed vocabulary, achieving remarkable ef-

ficiency in both training and inference while maintaining competitive accuracy on text classification tasks.

Svenstrup et al. (2017) further refined the hashing approach with hash embeddings, introducing a novel technique that combines multiple hash functions with trainable parameters. This method reduces collision problems inherent in previous hashing approaches while maintaining the benefits of vocabulary-free operation. Their work demonstrated that hash embeddings could match or exceed the performance of traditional word embeddings while using significantly less memory.

Most recently, Clark et al. (2022) proposed CANINE, a transformer-based architecture that operates directly on character sequences without any tokenization. CANINE employs local self-attention and downsampling to process raw Unicode characters efficiently, eliminating tokenization entirely from the NLP pipeline. This approach demonstrates competitive performance on multilingual tasks while avoiding the complexity and potential biases introduced by subword tokenization.

These vocabulary-free methods represent a growing trend toward simpler, more flexible text processing pipelines. While early approaches focused on hashing techniques for efficiency, recent methods leverage advanced neural architectures to eliminate tokenization entirely or learn it dynamically during training. This evolution suggests a potential future where explicit vocabulary construction may become unnecessary for many NLP tasks.

3.4.2. Vocabulary Adaptation

Vocabulary adaptation addresses the challenge of extending pretrained models to new languages or domains without complete retraining. This approach offers a middle ground between using fixed vocabularies and training entirely new models.

Wang et al. (2019) introduced a systematic approach for expanding the vocabulary of pretrained multilingual models. Their method maintains the original model's knowledge while incorporating new tokens specific to target languages. The process involves three key steps: first, identifying new vocabulary items through frequency analysis in the target language; second, initializing new token embeddings using a mix of existing embeddings; and third, fine-tuning the expanded vocabulary while freezing the original model parameters. Their experiments demonstrated significant improvements in cross-lingual transfer, particularly for languages with distinct scripts from the pretraining languages.

This work builds upon earlier efforts in vocabulary expansion such as (Chronopoulou et al., 2019), who explored vocabulary adaptation for cross-domain transfer, and Artetxe et al. (2020),

who investigated cross-lingual vocabulary alignment.

Vocabulary adaptation methods offer particular advantages for low-resource languages and specialized domains, where the original model's vocabulary may provide inadequate coverage. However, these approaches must carefully balance the benefits of expanded vocabulary coverage against the potential for catastrophic forgetting of the original model's learned patterns.

3.4.3. OOV Mitigations

The handling of out-of-vocabulary (OOV) tokens represents a fundamental challenge in natural language processing, particularly for machine translation systems where coverage of rare words is crucial for meaning preservation.

Luong et al. (2015) proposed a novel approach to handling OOV words in neural machine translation through a hybrid system that combines both word and character-level representations. Their key innovation was the introduction of a "positional all" model that aligns and recovers OOV tokens through character-level back-translation. The system maintains a record of OOV token positions during the translation process, enabling accurate placement of recovered words in the final output. This approach demonstrated significant improvements in translation quality, particularly for technical and domain-specific content where OOV words are more prevalent.

Li et al. (2016) introduced a novel substitution-translation-restoration method for neural machine translation (NMT). Their approach addresses rare words by substituting them with similar in-vocabulary words, using a similarity model learned from monolingual data. This substitution occurs both during training and testing. After translation, a post-processing step restores the original rare words' translations. The method reduces ambiguity caused by unknown tokens while preserving sentence structure, demonstrating significant improvements in BLEU scores.

Kolachina et al. (2017) proposed a method for improving dependency parsing by replacing OOV words with semantically or morphologically similar in-vocabulary words. The approach uses distributional similarity, computed from a large background corpus, and also considers common suffixes to identify suitable replacements. This method was tested using both count-based and dense neural vector-based semantic models. The study found that count-based methods, specifically a distributional thesaurus, performed better than neural vector-based methods for OOV replacement in dependency parsing.

Zhao et al. (2018) advanced this line of work by focusing on "troublesome words" - including not just OOV tokens but also low-frequency and ambiguous words. Their approach leverages contextual

information and external memory to improve translation quality for these challenging cases. By maintaining an external memory of word translations and incorporating surrounding context, their system demonstrated improved handling of rare and difficult words in neural machine translation.

Finally, Moon and Okazaki (2020b) introduces a method for mitigating OOV issues in multilingual BERT models during SFT (and optionally, CPT) for downstream tasks. This approach identifies OOV subwords in the task corpus and maps them to in-vocabulary subwords using various strategies. The key innovation is ensuring a one-to-one mapping between OOV tokens and their in-vocabulary surrogates. During fine-tuning, the embeddings for both the new subword and its surrogate are shared and updated jointly.

Recent approaches have shifted toward more integrated solutions where OOV handling is incorporated directly into the model architecture rather than treated as a post-processing step. This evolution reflects the growing recognition that OOV mitigation benefits from access to the model's full representational capacity and contextual understanding.

3.4.4. Normalizing Methods

An alternative approach to handling multilingual text processing involves lossy transformation methods that convert text into a simplified or standardized form. These approaches trade perfect reconstruction ability for increased processing efficiency and reduced complexity.

Li et al. (2018) demonstrated that romanization of Chinese text can improve cross-lingual transfer in multilingual models while reducing vocabulary size requirements. Their work showed that pinyin-based representations, while losing some semantic information, enabled better parameter sharing across languages.

Sekizawa et al. (2017) propose a paraphrase-based preprocessing method for neural machine translation (NMT) that addresses out-of-vocabulary (OOV) words by replacing them with frequent synonyms from the target training corpus. Their approach combines paraphrase dictionary scores with language model scores to maintain both semantic accuracy and fluency. The iterative paraphrasing process progressively reduces vocabulary complexity. Similarly, Štajner and Popovic (2016) investigate text simplification as a preprocessing step for statistical machine translation (SMT), focusing on under-resourced languages, which yields vocabulary reduction as a beneficial side effect.

Kim and Shin (2013) propose romanization as a normalizing technique for colloquial Korean input. Their method addresses the challenges of processing irregular SMS language through translit-

eration of Korean into Roman letters, enabling more straightforward application of linguistic rules for unspaced text, phonetic variations, and lexical reductions. The approach employs Yale Romanization to establish a one-to-one correspondence between Korean and English letters, facilitating the application of linguistic rules for vowel contraction, reduction, and word-ending addition to generate a comprehensive lexicon of word variations.

Ding et al. (2018) explore a method to simplify abugida writing systems, focusing on four southern Brahmic scripts: Thai, Burmese, Khmer, and Lao. Rather than aligning multiple language alphabets, their approach aims to reduce the complexity of individual abugidas by omitting most vowel diacritics and merging consonant letters with similar phonetic values. Their main objective is to address sample sparsity, but as a byproduct they achieve better robustness in their trained vocabulary.

These lossy transformation methods offer practical benefits for specific applications, particularly in scenarios where perfect reconstruction of the original text is not critical. However, the trade-off between reduced complexity and information loss must be carefully considered based on the target application's requirements.

3.4.5. Sub-character Architectures

Sub-character approaches have emerged as a particularly effective method for processing East Asian languages, where individual characters can be decomposed into meaningful components. These architectures leverage the compositional nature of Chinese, Japanese, and Korean writing systems to improve model performance and reduce vocabulary requirements.

In the context of Chinese sub-character methods, there are two major families of approaches; increasing the representational power or robustness of character embeddings through subcharacter archictures (Shi et al., 2015; Sun et al., 2014), or utilizing subcharacter information to further increase the representational power of a model (Yin et al., 2016; He et al., 2018).

For Japanese text processing, character-level approaches pose challenges due to the language comprising of a combination of ideographs and phonetic script. The latter is less likely to have meaningful information at character-level. Given that constraint, methods mainly focused on using sub-character information as auxiliary information.

As an example, Nguyen et al. (2017) developed a neural language model that operates on multiple granularity levels, similar to that of Yin et al. (2016), incorporating both character and sub-character information. Their model processes kanji characters by decomposing them into

radicals and components, demonstrating that this sub-character information improves model performance, particularly for rare characters and words.

Karpinska et al. (2018) conducted a comprehensive analysis of when sub-character information benefits Japanese language processing. Their work established that while sub-character decomposition can improve performance for rare kanji and technical vocabulary, it may not provide significant benefits for common characters. This research provided important guidelines for when to apply sub-character approaches in Japanese NLP systems.

For Korean,Stratos (2017) introduced a pioneering sub-character architecture for Korean language processing that decomposes Hangul characters into their constituent Jamo components. By treating these sub-character units as the basic elements of processing, the model achieved improved performance on various Korean language tasks while significantly reducing the vocabulary size. This approach effectively leverages Korean's unique alphabetic syllabary structure, where characters are composed of initial consonants, vowels, and final consonants.

Most recently, Kim et al. (2024) advanced the field with KOMBO, a sophisticated model for Korean text processing that incorporates explicit rules for combining sub-characters. Their approach not only decomposes characters but also learns the structural relationships between components, leading to improved performance on various Korean language tasks while maintaining computational efficiency.

These sub-character architectures represent a significant advancement in East Asian language processing, offering a middle ground between character-level and byte-level approaches. They leverage linguistic structure to reduce model complexity while preserving important semantic and phonetic information encoded in character components.

Despite advancements in tokenization methods, the problem of out-of-vocabulary (OOV) tokens remains a significant challenge, particularly for character-diverse languages like Korean. The upcoming chapters investigate the effects of OOV on foundation models and proposes novel strategies to mitigate these issues.

4

The Effects and Mitigation of Out-of-vocabulary in Universal Language Models

Building on the foundational concepts of OOV in NLP transfer learning we discussed in Chapter 1 and 2, this chapter presents a systematic empirical investigation of a novel OOV mitigation strategy.

In this chapter, we extend our previous research (Moon and Okazaki, 2020b) through controlled experiments that quantify OOV impact and evaluate multiple mitigation approaches. Finally, our investigation compares these approaches against both unmitigated baselines and vocabulary expansion techniques, providing insights into performance recovery.

This chapter is a summary of the following peer-reviewed work:

- **Moon, S.** and Okazaki N. (2020). PatchBERT: Just-in-time, out-of-vocabulary patching. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- **Moon, S.**, and Okazaki N. (2021). Effects and mitigation of out-of-vocabulary in universal language models. In *Journal of Information Processing, Volume 29*.

4.1. Motivation

Our high-level motivation for this work can be summarized as follows:

- Unstratified, large-scale web corpus training is inevitably imbalanced, and at train time this will bias the trained tokenizer.
- Due to scalability constraints, tokenizer training is typically done on a small sample of the entire pre-train dataset, further amplifying the bias.

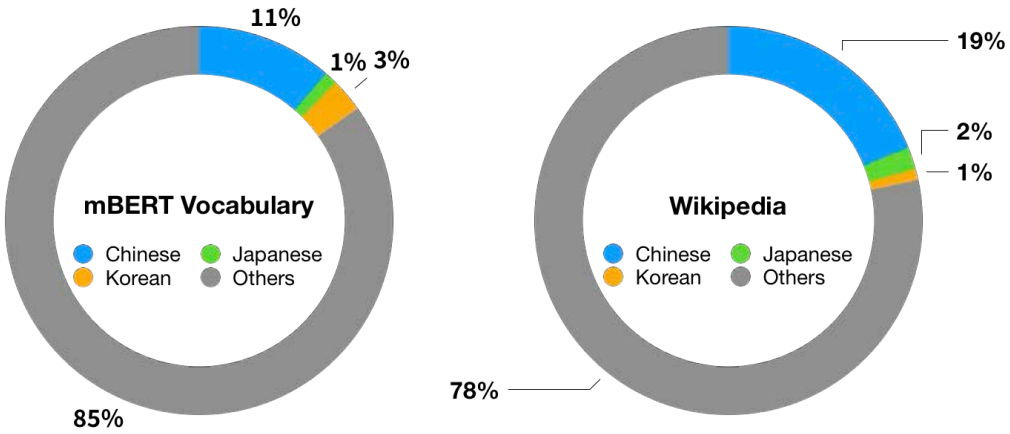


Figure 4.1: Multilingual BERT vocabulary distribution for CJK languages.

- CJK (Chinese, Japanese, Korean) languages represent a minority in the broader context of language usage on the internet.
- We expect these issues will result in suboptimal handling and inevitably lead to OOV (out-of-vocabulary) tokens for CJK languages in a multilingual setup.
- We hypothesize that there are possible mitigations to these problems in a post-train setup, during continued pre-training (CPT), and supervised fine-tuning (SFT).

Given the known complexities in processing text for CJK languages, this work investigates the adequate handling of CJK languages in a well-known multilingual model. Our work aims to understand the language-specific downstream impact of a tokenizer trained in a fully data-centric large-model training regime, which we expect to have inherited bias from imbalances in the pre-train corpus.

For an initial hypothesis test, we compared the ratio of articles between Wikipedia's statistics as of the time of writing, and compared the distribution of articles written in CJK languages to that of multilingual BERT's vocabulary. The language detection for multilingual BERT's (mBERT) subword tokens was performed with the lingua-py library in accuracy mode with the prefix notation for suffix subwords (##) stripped. The results of the analysis are shown in Figure 4.1, where we observe that the allocation of CJK vocabulary in the mBERT tokenizer has a summed ratio of 15%, while Wikipedia shows an allocation of 22%.

An interesting observation is that the relative micro-distribution shows a skew, where Chinese, Japanese, and Korean respectively represent 86.36%, 9.09%, and 4.54% approximate by training

data, contrasted with a relative distribution of 73.33%, 6.6%, and 20% allocated in the vocabulary. While the Chinese and Japanese distribution cannot be considered reliable due to the overlap in usage of ideographs, the relative vocabulary budget allocated for Korean is unexpected.

While both the macro and micro-distributions do not exactly match the distribution of Wikipedia articles, the trends observed suggest that the train-time imbalance has likely impacted the subword distribution; therefore, we justify this work based on this initial hypothesis test. We expect the following outcomes from this work:

- Confirming that pre-train imbalance and resulting bias lead to observable OOV issues in downstream task setups.
- Demonstrating that a computationally inexpensive mitigation for OOV is feasible.
- Showing that reduction or removal of OOV results in better downstream task performance.

4.2. Terminology Evolution

At the time this work was proposed, the terminology of transfer learning was still going through evolution, lacking universally agreed terms to describe identical concepts. While this chapter attempts to remain faithful to the original publication, it adopts converged terminology updates for better readability.

Foundation Model

A *foundation model* is a pre-trained model that was typically trained for a different, generally large-scale task, which is then adapted for orthogonal tasks following a *sequential learning* regime. This approach represents a key application of transfer learning, where knowledge from one domain transfers to benefit another. An example of this in computer vision is the VGG network introduced in 2014 (Simonyan and Zisserman, 2014).

In the context of this chapter, the foundation model is BERT (Bidirectional Encoder Representations from Transformers), which serves as the basis for the subsequent adaptation methods discussed.

Continued Pre-training (CPT)

In the original work of Devlin et al. (2019), the concept of domain adaptation using a self-supervised training objective, such as language modeling (or in the case of BERT, masked language modeling) is described as *unsupervised fine-tuning*. Similarly, Howard and Ruder (2018) refers to this as *LM*

fine-tuning. While domain adaptation appears as a related concept, it is worth noting that this is the *outcome* and not the *method*, therefore, it is not strictly a term that can be used interchangeably.

Recently, the field has converged on the term *continued pre-training* (CPT) to describe the process of achieving domain adaptation through the same task used to pre-train the foundation model. This process, also known as domain-adaptive pre-training in some literature, typically involves self-supervised language modeling.

Supervised Fine-tuning (SFT)

Fine-tuning refers to the process of taking a *foundation model* and performing additional training to address a task different from its original purpose, through a *sequential learning* regime. While historically referred to simply as *fine-tuning*, recent terminology has converged on *supervised fine-tuning* (SFT). This shift helps distinguish it from *unsupervised fine-tuning* and aligns with the field's convention of using three-letter acronyms to designate distinct methodological approaches.

4.3. Preliminaries

4.3.1. Foundation Model Tokenizer Training in Practice

Given n languages, character sets $\hat{\Sigma}_l$ for each language, and a desired character level n-gram length for the subwords m, the upper bound of the vocabulary size can be approximated as follows:

$$|\mathcal{V}| \approx \sum^{n} 2|\hat{\Sigma}_{(l,n)}|^m \tag{4.1}$$

Since this upper bound is too large to be reasonably trainable, models typically sample portions of the entire corpora or relax constraints on character-level coverage for these languages to prevent the vocabulary from growing to an unmanageable scale. As of today, this remains an unavoidable trade-off when training multilingual models.

This limitation introduces an aritificial bottleneck for downstream tasks, as any omitted character causes information loss. The effect amplifies when a large character set and scriptio continua [1] exist in the same context, which is the case for all CJK languages.

Table 4.1 illustrates typical patterns of OOV in CJK languages: Chinese exhibits dropped punctuation and rare characters, Japanese shows loss in emojis, and Korean demonstrates loss of entire text segments. While these patterns are representative, they do not encompass all possible OOV

[1] Languages written without spaces. Chinese and Japanese qualify as scriptio continua, while Korean represents a special case where spacing rules are liberal and text can appear without spaces in colloquial writing.

scenarios but rather provide insight into the forms of OOV occurring across different language-task setups.

Some researchers have proposed mitigating these issues through character decomposition (Stratos, 2017; Moon and Okazaki, 2020a), which significantly reduces the vocabulary budget while retaining all information. However, these approaches have seen limited adoption in practice.

In a monolingual setup, researchers can utilize pre-trained models specific to the target task language. However, a multilingual setup introduces additional complexity when using an ensemble of monolingual models, as it requires language detection to determine the appropriate model and tokenization scheme for each input. The most straightforward solution involves pre-training a monolingual model with a shared tokenization scheme across all required languages.

However, this approach faces significant challenges: acquiring a large corpus is demanding, and training a large multilingual model remains financially infeasible for many researchers. Given the high upfront cost and complexity of implementing a multilingual system, transfer learning using an open, multilingual model presents an economically attractive alternative. Unfortunately, due to corpus imbalance during pre-training, less-investigated languages—especially those with diverse character sets like CJK languages—frequently encounter OOV issues. Our work aims to improve the performance of these languages without substantially increasing computational costs when using open-source pre-trained models.

4.3.2. BERT Tokenizer

The multilingual BERT model **bert-base-multilingual-cased** (Devlin et al., 2019) we used performs two-phase tokenization, first with whitespace (word-like) followed by WordPiece tokenization (subwords). An example output of the tokenizer is explained in Figure **??**. The prefix forms of the subwords are expressed in their original form, while suffix forms are expressed by appending a ## prefix.

If either form of the subword is missing in a token, the tokenization fails, and the token surface is treated as OOV. Using the example in Figure **??**, the suffix form of **î** is not in the vocabulary, hence the entire surface of the token **plaît** becomes OOV. This is due to the greedy merging nature of the WordPiece algorithm and is not universal to all subword-based methods.

Due to the dependency on initial whitespace tokenization, BERT's tokenization is not expected to work well with scriptio continua languages, especially if it has a diverse alphabet. To workaround this limitation, BERT's tokenizer implements special handing which artificially injects whitespace

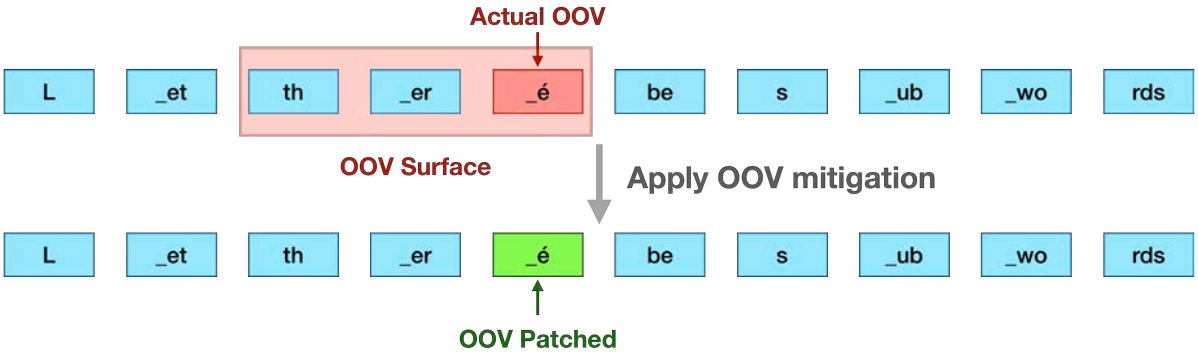


Figure 4.2: OOV surface contrasted to subword OOV, illustrated with mitigation.

before and after CJK ideographs. This mechanism is not enabled for Korean. .

4.3.3. OOV Mitigation

Our experiments utilized four CJK datasets for evaluation. The experimental process consisted of learning OOV words, optionally performing continued pre-training (CPT), and conducting supervised fine-tuning (SFT). We evaluated the effects at specific milestones throughout this process as part of an ablation study.

For each dataset, we defined the OOV rate as the ratio of sentences containing at least one OOV token. We selected sentiment analysis datasets specifically for this study, as they provided an effective test case for domain shift due to their extensive use of colloquialisms. The pre-trained model used in our experiments (bert-base-multilingual- cased, 110M parameters) was trained on Wikipedia (2500M words for English, with larger volumes for other languages) and BooksCorpus (800M words)[2]. The transition from these well- formed corpora to user-generated content increased the likelihood of encountering OOV tokens. Among the CJK languages, Korean showed the highest expected OOV rates because the BERT tokenizer lacks specialized handling for Korean text, making it particularly vulnerable to the WordPiece tokenizer's greedy merging behavior.

4.4. Method

In this section, we propose a method to mitigate OOV without training a new model. This approach is based on our hypothesis that OOV has adversarial effects on task performance, which we verify through experiments in Section 4.6. Our method modifies the BERT tokenizer, and we evaluate its effectiveness both with and without continued pre-training (CPT).

[2]This training dataset composition should be considered approximate, as the exact training dataset for multilingual BERT has not been publicly released

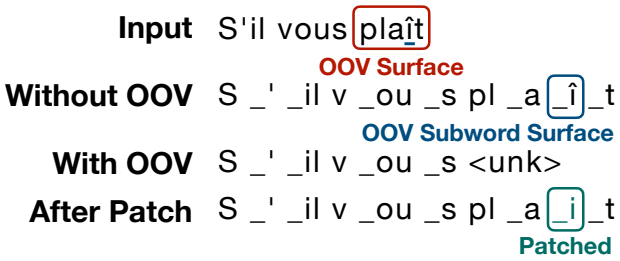


Figure 4.3: The hierarchy of OOV and the high level process explained with a simplified example.

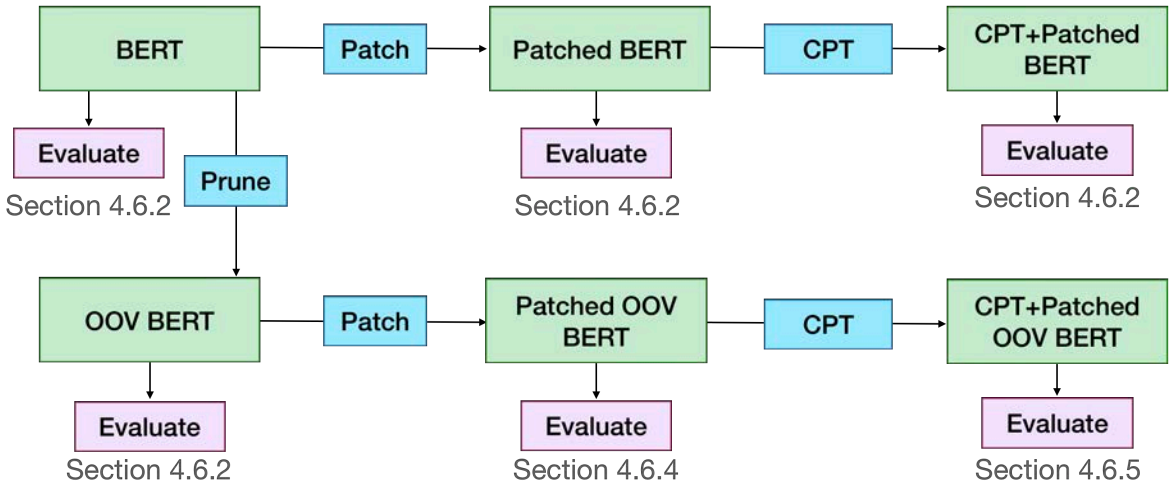


Figure 4.4: Different experiment variants, illustrated as a pipeline.

The proposed modification enables the BERT tokenizer to support a secondary vocabulary that maps new words to existing words. This modified tokenizer replaces the original tokenizer in the BERT model. The approach comprises three key steps.

OOV Surface Identification

We begin with a comprehensive corpus analysis to identify all OOV surfaces by tokenizing the task corpus. In the context of BERT, an OOV surface refers to an entire space-tokenized token. For each OOV occurrence, we record both the complete OOV surface and its surrounding context.

For each identified OOV surface, we conduct a brute-force search to identify the maximally specific OOV subword surface. An OOV subword surface represents a specific subword missing from an OOV surface. During this step, we generate frequency tables for both OOV and in-vocabulary subwords to inform our mitigation strategy. Our experiments revealed that most OOV subword surface cases stemmed from a single missing character in the vocabulary—a direct consequence of incomplete character coverage in the pre-training corpora.

OOV Patching

We then leverage this information to construct a mitigation strategy for the OOV subwords. When possible, we utilize the computed frequency of OOV tokens to prioritize frequent cases over rare ones. We evaluate several mitigation algorithms, each detailed in subsequent method sections. Following OOV mitigation, we optionally perform CPT and compare against the baseline.

Continued Pre-training

CPT serves two essential purposes: it enables the model to learn the newly introduced vocabulary from our mitigations, and it helps develop representations better suited to the task domain. This step becomes particularly crucial when a surrogate is assigned to a subword from a different language, such as when handling previously unseen subwords. Additionally, since CPT does not require an annotated corpus, we can enhance model robustness by incorporating additional corpora.

While the core concept of substitution for OOV mitigation has been previously explored (Kolachina et al., 2017), existing approaches typically rely on auxiliary knowledge from lexical dictionaries or semantic similarity derived from word embeddings trained on auxiliary corpora and models. These traditional methods prove challenging to apply to subword models, as subwords often lack sufficient semantic meaning to learn meaningful representations and do not appear in standard dictionaries. Our approach's key contribution lies in its applicability to subword models and its practical implementation, requiring only a downstream task corpus and a pre-trained model.

4.4.1. Surrogated Tokens

Surrogates, simply put, map a subword missing from the vocabulary to a subword that is already in the vocabulary of a pre-trained model. There are intuitive ways to find substitute words in a word-level setup, the most obvious being choosing a semantically similar word from a thesaurus. In a subword context, this is not as straightforward, as a subword generally has no meaning. In our work, we discuss different surrogate selection processes. The surrogate selection process assigns multiple subwords to the same embedding, which is a trade-off that limits the utility of the proposed method for generation tasks. As surrogates are only assigned once, to perform generation tasks when a subword is polysemic, one would need to use an auxiliary binary classifier to determine which subword the prediction actually is. This is not required for tasks that do not require generation, such as classification.

The embeddings between the newly added subword and the surrogate are shared and updated

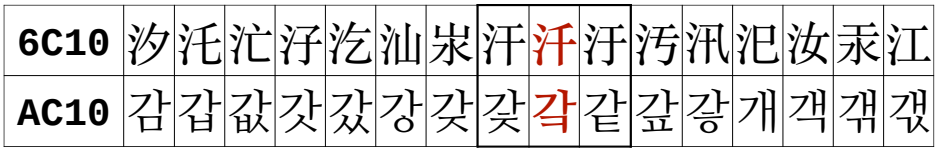


Figure 4.5: Intuition of character distance method, showing radical sharing and phonetic similarities.

together in the fine-tuning process. The OOV subword frequency table we constructed in the second step of the process above is used to break ties and minimize conflicts. For example, token A and B, both of which are OOV subwords, can end up with the same proposals $\{X, Y\}$ in preference order. In this case, given A has a higher frequency, it gets precedence over B, so the surrogate map becomes $A \to X$ and $B \to Y$. Our goal is to refine the proposals to be in a state where one surrogate is assigned to only one OOV token.

Character Distance

This method selects the surrogate with the shortest Unicode codepoint distance from the OOV subword, limited to subword tokens within the vocabulary of the same length. In this process, we perform an exhaustive search, formulated as the following.

$$\underset{w \in W'}{\operatorname{argmin}} |\operatorname{ord}(v) - \operatorname{ord}(w)|_1 \tag{4.2}$$

In the formula above, v is the OOV subword, and W' is a subset of the vocabulary W which satisfies UTF-8 character level length equality $|v| = |w|$ for $w \in W$. ord is the Unicode ordinal conversion function.

The intuition of this method builds on the characteristics of the CJK Unicode blocks, which allow us to cheaply approximate text or semantic similarity through the scalar values of the Unicode codepoints as seen in Figure 4.5. The properties which we intend to exploit are different, depending on the target language. In CJK ideographs, adjacent characters tend to share a radical, hence has a bias towards semantic similarity.

On the other hand, in Korean, phonetically similar characters are adjacent. This approximates edit distance, as a Korean character in Unicode is a combination of multiple sub-characters. This phonetic similarity differs from edit distance, as it tends to disallow edits on the first two components of the character. In the event of a distance tie, we used the candidate with a lower codepoint.

Frequent subword tokens get preferential treatment and hence get surrogates with closer distance to an infrequent token. Once a token has been assigned, it is not re-used as a surrogate.

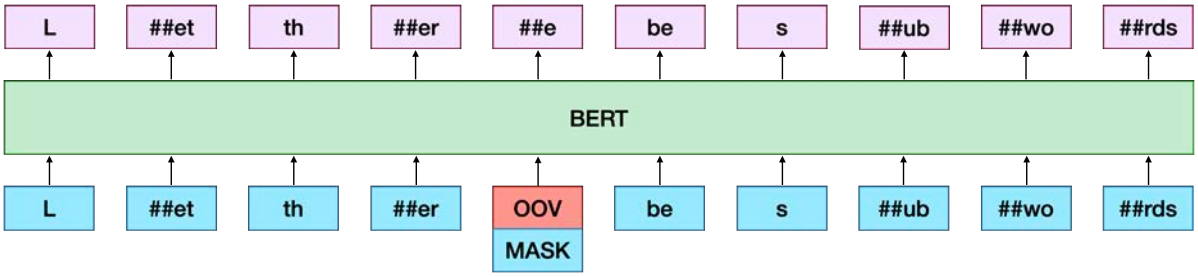


Figure 4.6: Masked language-model OOV surrogate candidate selection.

Unseen Subwords

We select tokens from the in-vocabulary token frequency table, which were never seen in the current task as surrogates. As downstream tasks for evaluation do not require the entire vocabulary, we select random tokens with a frequency of 0 as surrogates. In our experiments, this was implemented by overwriting the existing unseen subword to the target subword. This allows guaranteed reconstruction of the original text, making it usable for generation tasks, but at the cost of the embeddings being assigned to ones that the model has not seen in the context.

This method is analogous to increasing the model parameters (via vocabulary size), then pruning back to the original size, but as an in-place operation. Any word previously assigned was held out to prevent re-assignment. As the vocabulary will have a large number of tokens never seen in most downstream tasks, we do not use any frequency preference here.

This method can also be implemented with an inverse objective, to select the token with the maximum character distance from a list of random unseen subwords, to minimize the chances of surrogates being assigned in the same Unicode page. We expect the outcome of this approach to be similar to that of using unseen subwords, therefore was not included in our experiments.

Masked Language Model

The masked language model-based method uses BERT's masked language head to generate surrogate proposals, as illustrated in Figure 4.6. Each subword OOV surface is replaced with the mask token and passed to the masked LM head with the whole context. The subword token with the highest probability is selected for each context, stored in a frequency table, to select the most common prediction later. This results in deterministic surrogate mappings.

We use the same frequency preference as character distance, which allows frequent OOV subwords to have precedence when selecting surrogates. As with other methods, once a surrogate is assigned, it is held out. Therefore, less frequent words are assigned to the next most locally frequent surrogate. After the entire process, OOV subwords that were not assigned a surrogate are

assigned to the candidate with the lowest frequency. This method has the highest computation cost, as it requires inference on the model.

4.4.2. Additional Tokens

Here, we add new tokens to the vocabulary and increase the model size, motivated by prior work (Wang et al., 2019). As this increases the network parameters, these are used as a secondary baseline to be compared with surrogates.

Random Initialization

After adding the missing subword to the vocabulary, then the corresponding embedding is randomly initialized. This is analogous to how a model is commonly initialized, and also how new tokens are added to an existing vocabulary.

Transfer Initialization

Transfer initialization is done by following the first step of the masked language model task to generate a list of surrogates. We then initialize by copying the embedding vector of the topmost probable candidate of the OOV subword into the newly added OOV subword's slot in the embedding matrix. These two tokens share the same initial embeddings but are expected to diverge through fine-tuning.

4.4.3. Post-mitigation

After our OOV mitigation method has been applied to the model using the downstream corpus, the following steps are also necessary to complete the task-oriented tuning procedure.

Continued Pre-training: Masked Language Modeling

The original BERT paper mentions CPT as related work, but it is not employed as part of the proposed methodology; instead the authors defer entirely to SFT to tune the model appropriately. We hypothesize that because the representations for the language which the downstream tasks are already well-trained, this might have been sufficient; especially given that the evaluation was done against a monolingual English model.

We suspect that multilingual learning is much more difficult; and as a result the expectation is that a minority language in a multilingual model would not have strong representational power, along with the bias towards well-formed book-like text for training resulting in a domain shift for downstream tasks.

Language	Example
NSMC (Korean)	재밌습니다.재밌습니다. → [UNK] . [UNK] .
Twitter (Japanese)	…１５回は押した💢💢💢💢💢… → … １５ 回 は 押 [UNK] …
INEWS (Chinese)	双叶湖密山骑友 ——哒哒香花海之旅 ！ → 双叶湖密山骑友 [UNK] 香花海之旅！

Table 4.1: Examples of OOV in the task datasets, demonstrating different patterns.

Dataset	OOV Tokens	Total Tokens	Token Rate	OOV Sentences	Total Sentences	Sentence Rate
NSMC	81603	5185891	1.5%	60151	200000	30.1%
KorQuAD	14159	5134799	2.8%	8569	144000	5.9%
Twitter	10310	985345	1.0%	5518	22000	25.1%
INEWS	2570	158212	1.6%	1278	6355	20.1%

Table 4.2: OOV rates on the task datasets with multilingual BERT.

For these reasons, we drift from the standard BERT fine-tuning regime and introduce a CPT step with the intent for the model to adapt to a new domain, but also to learn representations for the subwords which have been surrogated for OOV mitigation.

CPT is done through the same task as BERT's pre-train scheme, with the exception of not employing the next-sentence prediction part. The rationale for this is that many of the downstream tasks only have single-sentence as input, which make it inappropriate for next-sentence prediction as there is likely no logical continuity between two distinct samples. All layers are trained during CPT.

Supervised Fine-tuning: Fine-tuning with Task Head

Supervised fine-tuning is performed following the same scheme as the reference BERT evaluation protocol, utilizing a task-specific head. All of the layers trained during fine-tuning.

4.5. Tasks

We used four tasks for downstream task performance evaluation. Each dataset's OOV rate has been computed in Table 4.2, with the OOV rates rounded to the first decimal digit. (0.0% is any value under 0.05%.)

4.5.1. Classification: Naver Sentiment Movie Corpus

The Naver Sentiment Movie Corpus[3] (NSMC) (Cho et al., 2020) is a Korean sentiment analysis task, containing 200,000 user comments and a corresponding binary label which indicates positive or negative sentiment. The OOV rate on the pre-trained BERT model was 30.1% due to a large number of typos and the domain gap.

4.5.2. Classification: Japanese Twitter Sentiment Analysis

As a second validation target language, we used a subset[4] of a Japanese Twitter dataset (Suzuki, 2019) [5], which is a sentiment analysis task with five possible labels. The subset contains 20K Tweets and 2K Tweets, respectively, for training and test. We observed that a large portion of the OOV was from emojis during analysis, resulting in an OOV rate of 25.1% on the pre-trained BERT model.

4.5.3. Classification: Chinese News Sentiment Analysis

The INEWS dataset is part of the ChineseGLUE[6] dataset. The input is a short sentence from a news article, and the label is one of three labels denoting the tone of the sentence. This is also a sentiment analysis task, with a split of 5K train and 1K validation, and an OOV rate of 20.1% on the pre-trained BERT model.

4.5.4. Question Answering: KorQuAD 1.0

KorQuAD 1.0[7] is a Korean version of the SQuAD (Rajpurkar et al., 2016) reading comprehension task. The task involves answering a question given a passage of text, and consists of 10K passages with 66K questions. The passages are from Wikipedia, which is commonly used as a part of large-scale training corpora. The result of this is a low OOV rate of 5.9% on the pre-trained BERT model. For this task, CPT was omitted to prevent the model from memorizing answers. We added this additional task to validate our method against a low-OOV task.

[3] https://github.com/e9t/nsmc
[4] https://github.com/cynthia/japanese-twitter
[5] http://www.db.info.gifu-u.ac.jp/data/Data_5d832973308d57446583ed9f
[6] https://github.com/chineseGLUE
[7] https://korquad.github.io/

Task	Optimizer	Adam ϵ	LR	GradAccum	Weight Decay	Length	Batch Size	Epochs
CPT	Adam	1e-8	5e-5	1	0.0	512	6	3
OOV Correlation (NSMC)	Adam	1e-8	2e-5	1	0.0	160	160	3
Mitigation (GLUE)	Adam	1e-8	2e-5	1	0.0	512	10	3
Question Answering (KorQuAD)	Adam	1e-8	3e-5	1	0.0	512	12	3
OOV Recovery (NSMC)	Adam	1e-8	2e-5	1	0.0	160	160	3

Table 4.3: Hyperparameters used to train each of the downstream task models.

4.6. Experiments

To validate the effectiveness of our method proposed in the previous section, we perform multiple experiments against multiple CJK datasets in the upcoming sections. To thoroughly evaluate the effects of our proposed scheme, we validate against both real and synthetic setups, using the different mitigation schemes explained in Section 4.4.1 and 4.4.2.

The high-level flow of all experiments we do here work is explained in Figure 4.4. We compare the effects of different methods using a pre-trained multilingual BERT (bert-base-multilingual-cased). Each method was tested with fine-tuning, including CPT, or by fine-tuning only against the task.

Task-level fine-tuning was included in every experiment to ensure fairness and is done by attaching a task head and training the downstream task model. This allows the model to learn how to accomplish the task while adapting itself to produce better representations for the task. For our experiments, we limited CPT to the task corpus to make the experiments reproducible with only the task datasets.

All experiments that involved training the model were trained for three epochs. The full list of hyperparameters used for the experiments is listed in Table 4.3, in this paper's appendix. Every experiment was run five times each, with the random seed fixed to an integer value of the run number in the range of $[1..5]$.

The runs are then compared to the baseline scores to observe the statistical significance of the different scores for each method. For the significance test, we performed a dependent t-test for paired samples (Dror et al., 2018). We used a p-value of $p < 0.05$ to determine statistical significance and a fixed seed (42) for any random algorithm to make the results deterministic, which guarantees reproducibility. The evaluation was done with the reference implementation [8].

4.6.1. Hyperparameters

We ran our experiments as close as possible to the baseline parameters used by the publicly available benchmark scripts for each task type. This means most of the hyperparameters for all of the evaluation was done as close to the default values as possible. For the OOV correlation and recovery tasks, we optimized the sequence length and batch size parameter specifically to the task to maximize VRAM usage for faster experimentation. The exact hyperparameters are disclosed in 4.3 [9]

The masking probability for CPT was set to 0.15. We did not employ whole word masking in our experiments, as the languages under investigation require an auxiliary pipeline (e.g., a morphological analyzer) to identify word boundaries.

4.6.2. Results on Task Datasets

The evaluation was done through the SST-2 GLUE task metrics (Wang et al., 2018) for the sentiment analysis tasks, and EM/F1 evaluation from the SQuAD metrics for KorQuAD, as the two tasks are compatible. Each model used the same dataset and training parameters as the baseline, only with different OOV mitigation methods. The results of these experiments are in Table 4.4.

Additionally, while Chinese and Japanese are both scriptio continua languages, BERT's tokenizer treats CJK ideograph text differently and breaks at every character by artificially injecting whitespaces. This makes the affected surface from OOV significantly smaller, resulting in less information loss. We expect to see more considerable gains in Korean for these reasons, as the per-character break is not enabled.

Naver Sentiment Movie Corpus

Due to the larger OOV surface and frequency, we expected to observe only a modest increase compared to the baseline. As shown in Table 4.4, we observed that OOV mitigation improves accuracy regardless of the method used, and these improvements are statistically significant. The OOV tokens in our analysis came from colloquialisms in user-generated content, which differs from the well-formed text (e.g., books) used for pre-training. This suggests that even without robust, representative embeddings, mitigation is still preferable to information loss during tokenization. We hypothesize that performance improves through domain adaptation via CPT because the initial

[8] https://github.com/cynthia/testSignificanceNLP by Dror et al. (2018), patched to support Python 3 https://github.com/-cynthia/testSignificanceNLP.

[9] The parameters used will use 23.5GBs out of 24GB of available VRAM when training using IEEE754 half-precision floating point tensors.

Model	Value	NSMC (ko)		Twitter (ja)		INEWS (zh)		KorQuAD (ko)	
		Acc@+CPT	Acc	Acc@+CPT	Acc	Acc@+CPT	Acc	EM	F1
BERT (Baseline)	Mean	0.8824	0.8785	0.7284	0.7192	0.8138	0.8074	0.7037	0.9005
	Std	0.0017	0.0006	0.0041	0.0058	0.0064	0.0047	0.0016	0.0013
Add (Transfer)	Mean	0.8916	0.8844	**0.7319**	0.7223	0.8116	0.8082	0.7097	0.9030
	Std	0.0007	0.0006	0.0040	0.0060	0.0022	0.0082	0.0023	0.0011
	p-value	0.0002	0.0000	0.1091	0.1623	0.2599	0.4437	0.0091	0.0041
Add (Random)	Mean	**0.8928**	0.8848	0.7310	0.7211	**0.8186**	**0.8106**	**0.7098**	0.9029
	Std	0.0006	0.0004	0.0046	0.0041	0.0049	0.0065	0.0034	0.0018
	p-value	0.0000	0.0000	0.2263	0.1639	0.1280	0.0601	0.0128	0.0248
Char. Distance	Mean	0.8926	**0.8855**	0.7304	**0.7238**	0.8122	0.8092	0.7094	**0.9031**
	Std	0.0009	0.0013	0.0037	0.0024	0.0097	0.0070	0.0026	0.0019
	p-value	0.0001	0.0005	0.1499	0.0108	0.3152	0.1567	0.0115	0.0358
Unseen Subwords	Mean	0.8922	0.8846	0.7304	0.7225	0.8142	0.8102	0.7037	0.9013
	Std	0.0002	0.0013	0.0039	0.0038	0.0079	0.0065	0.0017	0.2112
	p-value	0.0000	0.0000	0.1554	0.0649	0.4403	0.1441	0.5000	0.2934
Masked LM	Mean	0.8915	0.8842	0.7307	0.7225	0.8100	0.8090	0.7089	0.9027
	Std	0.0009	0.0006	0.0043	0.0058	0.0063	0.0047	0.1647	0.1283
	p-value	0.0004	0.0002	0.1103	0.1219	0.1451	0.2801	0.0177	0.0614

Table 4.4: Experiment results, with statistically significant p-values underlined (< 0.05). +CPT indicates usage of CPT, Acc is *accuracy* and Std is *standard deviation*.

embeddings are not representative of the subwords in context.

As this dataset showed the most significant performance gains, we qualitatively analyze different cases in Figure 4.7. *Positive* represents labels cases where the OOV patch worked as intended; *Positive with bad patches* shows cases where the vocabulary patch appeared inadequate yet resulted in improved performance; and *Negative* illustrates cases where the vocabulary patch appeared adequate but resulted in degraded performance. The task-wide effects are summarized in Table 4.5, which shows that more cases improved with our method.

Japanese Twitter Sentiment Analysis

This corpus showed a high OOV rate due to the frequent occurrence of emoji in the text, and improper normalization of Unicode punctuation. We observe similar patterns with the results from NSMC. Generally, we see only minor improvements, except for character distance - which was statistically significant. We observed that character distance assigned surrogates to Korean characters.[10]

Chinese News Sentiment Analysis

While we observed a high OOV rate in this dataset, the improvement was negligible. Analyzing the surrogates, we observed that most of the OOV tokens were punctuation or uncommon ideographs,

[10] This would have been appropriate to demonstrate with examples, but due to the Twitter license agreement, reproducing the original text in this paper was not possible.

Type	Model	Text
Positive	Input	어릴때 **재밌게** 봤던 영화~
	Translation	A movie I **enjoyed** when I was young
	Baseline	어릴때 **[UNK]** 봤던 영화 ~
	Patched	어릴때 **재밋게** 봤던 영화 ~
	Input	**정맒재밋다**
	Translation	**Really interesting**
	Baseline	**[UNK]**
	Patched	**정말재밋다**
Positive (w/Bad Patches)	Input	**줫나 재미없다ㅅ1ㅂ** 영화보다가 **존건생ㅇ 처음림이랗그유치할수가**;; 평점 **ㅁㄷ고** 보다가 **개났ㄲ임**
	Translation	**Really boring. First time I fell asleep** during a movie. **Really lame. Tricked** by review scores.
	Baseline	**[UNK] [UNK]** 영화보다가 **[UNK] [UNK]** ; ; 평점 **[UNK]** 보다가 **[UNK]**
	Patched	**있나 재미없다ㅈ1ㅆ** 영화보다가 **존건생ヴ 처음림이랙그유치할수가** ; ; 평점 **ㅋン고** 보다가 **개났モ임**
Negative	Input	유갓서비드가 **훨잼씀**
	Translation	"You got served" **is much more interesting**
	Baseline	유갓서비드가 **[UNK]**
	Patched	유갓서비드가 **훨쟁씀**
	Input	실질적으로 일반인들이 탈수있는차를 소개해달라니까 뭔 폴쉐911이야 **ㅋㅋㅋㅋㅋㅋ뭐하자는거임 ㅋㅋ**
	Translation	They introduce a Porsche 911 as a car for an average person? **(Laugh) What are they thinking?**
	Baseline	실질적으로 일반인들이 탈수있는차를 소개해달라니까 뭔 폴쉐911이야 **[UNK] [UNK]**
	Patched	실질적으로 일반인들이 탈수있는차를 소개해달라니까 뭔 폴쉐911이야 · · · · · · **뭐하자는거임** · ·

Figure 4.7: Qualitative inspection of positive, positive (with bad patches) and negative vocabulary patch cases.

Dataset	Regressed	Improved	Delta
NSMC	392	528	136
KorQuAD	64	79	15
Twitter	21	32	11
INEWS	11	11	0

Table 4.5: Improvements and regressions with OOV samples, best Character Distance models compared to best baseline models.

which we expected to, and confirmed to have little effect in the downstream task performance. In Table 4.5, not only does the improved cases cancel out, looking at the OOV cases we considered the difference to be training noise. We hypothesize that small size of the dataset is likely to have contributed to the negative results.

KorQuAD 1.0

We did not expect significant improvements due to the low OOV rate, and the results reflect this. While we still saw minor improvements across the board, the difference is incremental at best, although some methods produced p-values which were considered statistically significant. The small delta can most likely be attributed to the relatively low OOV rate and omission of CPT.

Given that our experiments' results demonstrate that mitigating OOV improves task performance, in the next section, we explore if our method can recover performance in high-OOV models, which we have synthetically created through initial OOV and performance correlation experiments.

4.6.3. Effects of OOV on Task Performance

In the previous section, we demonstrated the effects of our method on different tasks and languages. These experiments were conducted based on the hypothesis that OOV has an adversarial impact on task performance. In this section, we artificially induce OOV on a pre-trained model through vocabulary pruning and correlate the OOV rate to task performance. With these synthetic OOV models, we use one of the tasks to investigate how OOV affects task performance in a BERT model. Following this, we apply our scheme to these synthetic models to verify if our proposed method is effective at recovering the performance of a broken model.

In this section, we investigate the correlation between OOV and task performance by evaluating task performance using the baseline BERT (bert-base-multilingual-cased) model, then compare the results of that to models with varying OOV rates.

Additionally, using NSMC the task, we artificially induce OOV in a model and perform task evaluation using this model. In our experiments, we use three different strategies for artificially inducing OOV.

We use the three methods to eliminate the most frequent words, the least frequent words, and random sampling. We compare different methods to ensure fairness, as the different methods exhibit different scenarios of how an OOV can be introduced in a downstream task. NSMC was chosen because it was the largest dataset we had for our experiments, and we assumed that the larger the task corpus is, the more likely it will have a diverse vocabulary, hence being more susceptible to OOV.

For the frequency computation, we used two datasets. The first dataset we used is the kosentences[11] corpus. This corpus is a Korean corpus cleansed of Wiki markup from multiple publicly available Wiki dumps. As we only use the Wikipedia part of kosentences, we will refer to the corpus as KoWiki in this paper. We considered this to be a good approximation of what the backbone model (bert-based-multilingual-cased) was initially trained with. This is because almost every large-scale pre-train corpus contains Wikipedia in some form. For this case, the frequency table was initialized with every Korean subword in the model, and the frequencies against the KoWiki corpus were updated on the frequency table. Subwords in the model's vocabulary, but not in the KoWiki corpus, were kept at a 0. The second dataset used was the actual task corpus, as using the task corpus is the most effective way to introduce OOV artificially.

This experiment intends to correlate the relation between OOV rate and task performance to

[11] https://github.com/cynthia/kosentences

confirm our initial hypothesis. It is worth noting that as we do not train a model from scratch, this is an approximation and not an accurate representation of what a pre-trained model's vocabulary would have due to the properties of subword tokenization depending on the character level n-gram distribution. This trade-off was made for computational efficiency reasons, as pre-training, a new model requires a significant amount of computing power, and for our experiments, we will need to train 42 models, which was computationally infeasible.

In our experiments, we prune subwords from the frequency table in different ratios - for our experiments, we chose 0.1%, 1%, 5%, 10%, 20%, and 50% as the target ratios. 20% and 50% are used to test extreme scenarios, to a point where it is likely that the model predictions can be considered equivalent to random choice. We use three different strategies for pruning the vocabulary, which we discuss in the subsections below. The ratio here is the ratio of words of the frequency table's vocabulary we prune from the vocabulary and should not be mistaken with the OOV ratio discussed in the datasets section.

Common Words

Removing the most frequent words is not a common scenario in any form, especially when it comes to a pre-trained model setup. Ranking the vocabulary in order of frequency, we prune the vocabulary from the top ranking (most frequent) word based on the ratio to be pruned. For example, in a 1000 word vocabulary with a 5% prune rate, the end result will be a model that is missing 50 of the most frequent words. This was chosen to demonstrate the extreme cases of unusable models, for instance, if a language that was expected to be supported was accidentally omitted from the training data. However, it is worth noting that this is a common form of removing stopwords in traditional NLP and information retrieval setups. This exercise intends to find out roughly which part of the vocabulary begins to contribute to the task performance.

Rare Words

This method was chosen to simulate a scenario where the corpus was sampled, or character coverage was reduced due to computational constraints. As least frequent subwords in a corpus will be omitted from the vocabulary, we consider this a rough approximation of what would happen when trade-offs are made due to the computational limitations. The process is the same as common words, but in this case, pruning is in order of least frequent words.

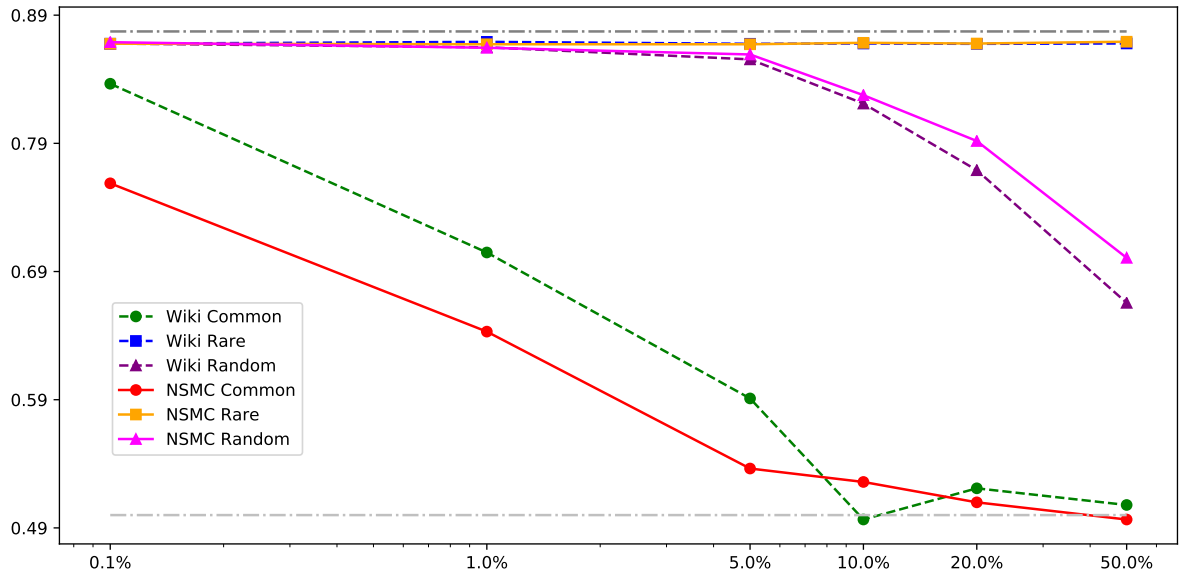


Figure 4.8: Performance degradation trends of artificial OOV with different pruning strategies.

Frequency Table	Method	0%	0.1%	1%	5%	10%	20%	50%
KoWiki	Common	0.87730	0.75882	0.64312	0.53632	0.52584	0.50994	0.49654
KoWiki	Rare	0.87730	0.86772	0.86730	0.86730	0.86842	0.86786	0.86928
KoWiki	Random	0.87730	0.86892	0.86456	0.85934	0.82758	0.79182	0.70068
NSMC	Common	0.87730	0.83650	0.70486	0.59100	0.49656	0.52088	0.50788
NSMC	Rare	0.87730	0.86772	0.86918	0.86766	0.86790	0.86762	0.86794
NSMC	Random	0.87730	0.86784	0.86494	0.85550	0.82104	0.76904	0.66556

Table 4.6: Performance degradation data of artificial OOV with different pruning strategies.

Random Words

In random words, we randomly eliminate subwords from the vocabulary. The subwords list in the frequency table is used to select target subwords to remove from the vocabulary. Based on the target subword list, we randomly choose a word for removal and evaluate the performance.

This is also another approximation of the consequences of computational feasibility trade-offs, as with least frequent words. As the distribution of subword frequency is expected to follow Zipf's law, even with random removal, we assume that the probability of an infrequent subword being chosen for deletion is inversely proportional to the frequency of the given subword.

4.6.4. Task Performance and OOV

The results of these experiments are summarized in Figure 4.8, accompanied by the full results in Table 4.6. An important point to note here is that as this is a balanced, binary classification task, it is unlikely that a model's accuracy can go significantly below 0.5. As it converges towards 0.5, we can consider the model's output to be equivalent to an equidistributed binary random number

generator, hence a random model.

Based on the experiment results, the first straightforward observation we made is that removing rare words does not affect task performance at all, regardless of how many are removed. Analyzing the removed words, we observed that the removed words were mostly words from a different language, which we suspect will not have substantial contributions to task performance. On the other hand, the other two methods used for pruning affect accuracy, especially as the ratio increases.

We observed that pruning common words had immediate effects, especially using the KoWiki corpus - as the effects are apparent even at 0.1%. This is because the vocabulary of the frequency table of KoWiki is larger than that of NSMC, 0.1% pruned more subwords than the NSMC frequency table, which had a smaller vocabulary. Removal of common words can have devastating effects, as, without a matching suffix form of a subword, the tokenizer's greedy will fail. In both cases, we can see that starting from around 5%; the model converges towards a random model's performance. In these worst-case scenarios, we observed that the model's input had more OOV than actual subwords. In many cases, input to the model exclusively consisted of OOV tokens.

Random pruning, on the other hand, tended to have a slower effect on task performance. This is expected, as the probability of pruning a common subword is lower than the probability of pruning a less common word. We can still observe noticeable performance decreases on both KoWiki and NSMC starting from 5%. Unlike common, the model did not end up in a state comparable to random choice.

Even when scaled up to 50%, pruning rare subwords did not have significant effects on the performance. This was somewhat unexpected, as we initially hypothesized it to affect the task performance with that many words removed. The reason turned out to be that even at 50%, most of the rare words only appeared once, and only a small portion (less than 5%) of these rare words were Korean - which explains the minimal effect on performance.

Based on these results, we can conclude that removing any subword that is not extremely rare does have adversarial effects on task performance. As OOV in a pre-trained model can have devastating effects on the model's final applicability as a universal language model, early mitigation is crucial. Early mitigation can be done by preparing enough diverse training data, so that the resulting model is robust in many settings and by thoroughly validating for OOV before training the model.

However, when OOV happens due to the lack of early mitigation in a model, late mitigation is

Frequency Table	Sampler	Mitigation	0%	0.1%	1%	5%	10%	20%	50%
KoWiki	Common	None	0.87730	0.75882	0.64312	0.53632	0.52584	0.50994	0.49654
		Patched (CD)	0.88390	0.87132	0.85702	0.85014	0.84756	0.85146	0.85120
		Patched (CD) + CPT	0.88850	0.88446	0.86514	0.86256	0.86918	0.86662	0.86562
KoWiki	Rare	None	0.87730	0.86772	0.86730	0.86730	0.86842	0.86786	0.86928
		Patched	0.88390	0.88094	0.88072	0.88072	0.88060	0.88124	0.88020
		Patched (CD) + CPT	0.88850	0.88588	0.88556	0.88572	0.88628	0.88530	0.88646
KoWiki	Random	None	0.87730	0.86892	0.86456	0.85934	0.82758	0.79182	0.70068
		Patched	0.88390	0.88092	0.88078	0.87752	0.87452	0.87012	0.85542
		Patched (CD) + CPT	0.88850	0.88582	0.88490	0.88522	0.88380	0.88334	0.87710
NSMC	Common	None	0.87730	0.83650	0.70486	0.59100	0.49656	0.52088	0.50788
		Patched	0.86830	0.87942	0.87530	0.86340	0.85716	0.84872	0.85000
		Patched (CD) + CPT	0.88850	0.88432	0.87788	0.86590	0.86146	0.86122	0.86040
NSMC	Rare	None	0.87730	0.86772	0.86918	0.86766	0.86790	0.86762	0.86794
		Patched	0.86830	0.88176	0.88208	0.88264	0.88242	0.88248	0.88266
		Patched (CD) + CPT	0.88850	0.88610	0.88622	0.88648	0.88446	0.88588	0.88490
NSMC	Random	None	0.87730	0.86784	0.86494	0.85550	0.82104	0.76904	0.66556
		Patched	0.86830	0.88288	0.88112	0.88264	0.87702	0.87506	0.86194
		Patched (CD) + CPT	0.88850	0.88488	0.88634	0.88196	0.87882	0.87388	0.86554

Table 4.7: Recovery experiment results. Patched is with mitigation, Patched+CPT is with mitigation and CPT.

the next best option to maximize the model's capabilities. In the next section, we propose a late mitigation for OOV in a non-synthetic scenario, using an openly available pre-trained model that is missing subwords needed for a task.

Using our proposed methods, we investigate the extent of performance recovery possible and identify the threshold for diminishing returns. We artificially introduce OOV tokens based on one of the methods from our OOV-task performance correlation analysis, then apply our vocabulary patching scheme to evaluate its effectiveness in salvaging model performance—essentially simulating the behavior of a poorly trained model. For this analysis, we utilized the same models described in Section 4.6.2.

We use the models from this experiment with the same protocol we proposed to mitigate OOV for the recovery experiments. The only difference here is that OOV mitigation is done against a synthetic high-OOV model instead of the baseline model. Among the multiple methods proposed, we use character distance (CD), as it was shown to be effective while being computationally efficient, which allowed us to experiment with many different configurations.

4.6.5. Recovery with Proposed Method

The results of this experiment are visualized in Figure 4.9, and the full results are in Table 4.7. The trends we observed in the original mitigation NSMC experiments repeat here. A model that has been both through CPT and OOV-patching consistently outperformed a model without CPT. In this particular setup, we hypothesize that CPT contributions in models with higher OOV rates can be attributed to the fact that many subwords have now been mapped to semantically different words,

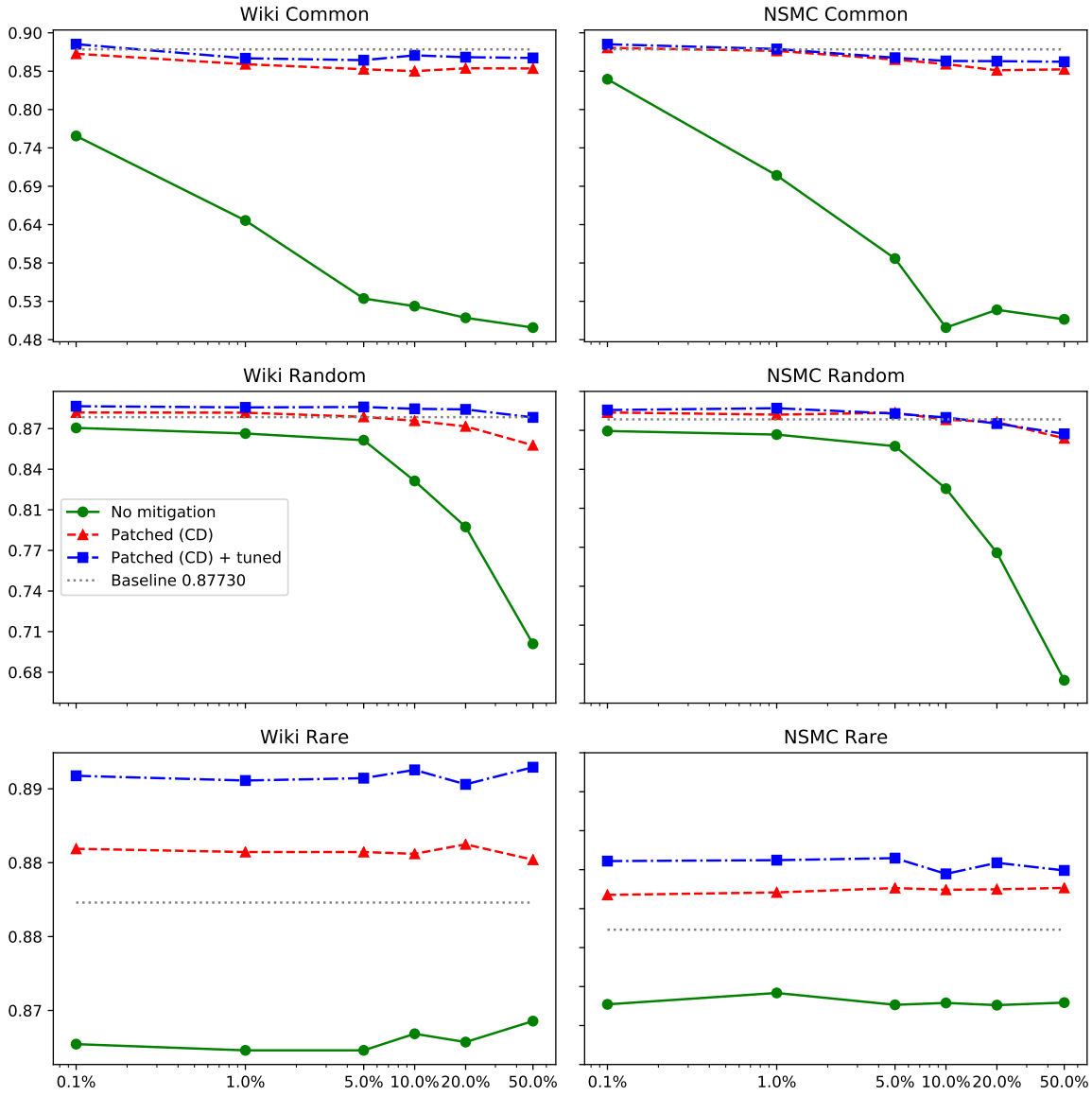


Figure 4.9: Performance recovery under different OOV rates. (Note: Plot range for the y-axis differs between frequencies.)

so the model has to learn the structure of the text nearly from scratch. However, our results suggest that the proposed method can even be effective at improving performance in high-OOV conditions, such as a model that was pre-trained on corpora extremely disparate from the target task's domain.

While the model does not fully manage to recover to the best-case performance fully, we also observed that extreme case models such as those with comparable performance to a random model could also recover quite well. However, in these extreme case models, we observed that due to the high amount of surrogates needed, the model started borrowing subwords from other languages, from the CJK ideograph block as a surrogate for Korean subwords.

We do not have any theoretical justification of why extreme cases like this can also recover

near-baseline performance. We hypothesize that without CPT, the model's representations lack semantic or contextual information; hence it acts as a random embeddings model. This model can still be used to classify data with a classification head. With CPT, the model re-learns the structure from the input, only with different embeddings, and due to the surrogate assignment being exclusive in our method, the model can adapt to inherent structure from the task corpus.

A hypothesis is that this becomes equivalent to a classification through a dense BOW representation at the last layers [12]. With fine-tuning, the model re-learns a weak representation from the input, only with different embeddings, and due to the surrogate assignment being exclusive in our method, the model can adapt to inherent structure from the task corpus.

4.7. Applicability to Other Models

4.7.1. Multilingual Models

While our experiments are limited to BERT, the method can be applied to any model. Generally, our proposed method is most effective when applied to greedy merging tokenizers such as WordPiece, which is used by both BERT and ELECTRA (Clark et al., 2020). This is due to the fact that greedy merging results in whole chunks of text being lost during tokenization, as observed in 4.7.

However, the method is applicable to most subword tokenization methods, such as Byte-pair Encoding (BPE), used by the multilingual model XLM (Conneau and Lample, 2019), and SentencePiece (Kudo and Richardson, 2018), used by another multilingual model, XLM-R (Conneau et al., 2020) also can benefit from this. The effects will be less significant since both tokenizers are not greedy. The expected effect is a diversification of the UNK token by re-assigning it to different subwords instead of all OOV tokens being mapped to a single embedding. This is expected to make it easier to train. In an actual byte-level tokenizer, such as the one used in GPT-2 (Radford et al., 2019) or more recent models (Brown et al., 2020; Chowdhery et al., 2022), our method is not expected to have significant gains as there will always be a byte-level fallback.

4.7.2. Comparison with Monolingual Models

Following the discussion on our method's applicability to different multilingual models, we also investigated whether or not OOV is also a phenomenon in language-specific models. As our method depends on the occurrence of OOV in the first place, if there is a low OOV rate, the contributions of OOV mitigation are also expected to be minor. To investigate this, we used three separate

[12] This hypothesis is re-visited in Section 6.2.1.

Dataset	Model	OOV Tokens	Total Tokens	Token Rate	OOV Sentences	Total Sentences	Sentence Rate
NSMC	bert-base-multi	81603	5185891	1.5%	60151	200000	30.1%
NSMC	KR-BERT	360	4773732	0.0%	336	200000	0.1%
KorQuAD	bert-base-multi	14159	5134799	2.8%	8569	144K	5.9%
KorQuAD	KR-BERT	5978	4396060	1.4%	2393	144K	1.7%
Twitter	bert-base-multi	10310	985345	1.0%	5518	22000	25.1%
Twitter	cl-tohoku-base-v2	26566	951286	2.8%	10165	22000	46.2%
INEWS	bert-base-multi	2570	158212	1.6%	1278	6355	20.1%
INEWS	bert-base-chinese	2338	158065	1.5%	1119	6355	17.6%

Table 4.8: OOV rates on the datasets, comparing multilingual BERT with monolingual.

monolingual models for each language.

For Chinese, we used the official BERT Chinese model (bert-base-chinese) released as part of the pre-trained models in (Devlin et al., 2019), with the BERT tokenizer, and for Japanese we used [13]. Finally, for Korean, we used KR-BERT (Lee et al., 2020) with Normalization Form Compatibility Decomposition (NFKD)[14] pre-processing. The subcharacter decomposition is similar to the work proposed in (Moon and Okazaki, 2020a) and makes this method much more robust against OOV. Each of the monolingual models was used to tokenize the respective language dataset compared with the multilingual model used in this work. The results are disclosed in Table 4.8, with the rates rounded to the first decimal digit (0.0% is any value under 0.05%.).

We observed that Korean, which is the most effective language to our scheme, we can see that the amount of OOV tokens in this model is extremely low. Due to this, it is unlikely to have adversarial effects on performance. While the OOV token ratio was still above 1% for KorQuAD, most of this turned out to be caused by subwords in a foreign language (e.g. CJK Ideographs), which is unlikely to have severe effects as it is assumed that the reader does not necessarily have to comprehend this from the passage to produce an answer[15] for the task.

Japanese, on the other hand, showed an increase in OOV. This is likely because the pre-training corpus was Wikipedia, which is well-formed text lacking colloquial writing, and Emojis, common in data sourced from social networks. In Chinese, there was very little difference as with the multilingual model, so the effects of applying our method are likely to be the same as a multilingual model.

4.8. Conclusion

This work investigated the correlation between OOV occurrence and task performance in transfer learning contexts. Through experiments with varying OOV rates, we confirmed our hypothesis

[13] https://github.com/cl-tohoku/bert-japanese
[14] This model does not work if this normalization is omitted.
[15] We confirmed that none of the answers expected an answer in a different language from the dataset.

that OOV significantly impacts model performance in transfer setups—in some cases degrading performance to that of random selection.

After demonstrating the effects of OOV-triggered information loss on task performance under transfer learning conditions, we proposed several OOV mitigation methods for downstream task fine-tuning. Our evaluation compared three approaches: no mitigation (baseline), mitigation through network modification, and vocabulary surrogates (requiring no network modification). The results demonstrated that vocabulary surrogates can enhance performance without additional computational cost at the model level, particularly when combined with continued pre-training (CPT). Our experiments also confirmed that task performance degradation correlates directly with OOV rates—tasks with lower OOV rates showed better resilience than those with higher rates.

To validate the broader applicability of our approach, we tested our proposed mitigation methods on high-OOV models initially used to test our hypothesis. The results demonstrated that our mitigation strategies could effectively restore model capabilities, highlighting the crucial role of tokenization in determining a pre-trained model's performance in transfer learning scenarios.

While the proposed OOV mitigation strategies significantly reduce information loss, they do not fully address the unique tokenization challenges posed by Korean's morpho-syllabic structure. To overcome these challenges, the next chapter introduces Jamo Pair Encoding, a novel tokenization method inspired by the linguistic properties of the Korean script.

5

Jamo Pair Encoding: Subcharacter-tokenization of Korean

Motivated by the problematic patterns in pre-trained Korean subword vocabularies in the previous chapter, this chapter is dedicated to explore the possibility of eliminating OOV altogether in Korean tokenization. Our investigation is inspired by the alphabetic nature of Korean script (Hangul), which is not adequately utilized due to the Unicode representation of the script.

This work has been proposed and peer-reviewed in the following venues:

- **Moon, S.** and Okazaki N. (2020). Jamo pair encoding: Subcharacter representation-based extreme Korean vocabulary compression for efficient subword tokenization. In *Proceedings of the Twelfth Language Resources and Evaluation Conference (LREC)*.

5.1. Motivation

Although Korean is typically grouped with Chinese and Japanese under the Unicode CJK classification, it represents a distinct case within this family. While these languages share substantial vocabulary roots, Korean employs a fundamentally different character system. Unlike Chinese and Japanese, which can effectively share character-level vocabulary, Korean requires a different approach.

The key distinction lies in the Korean writing system, which is phonetic in nature, contrasting with the ideographic systems of Chinese and Japanese [1]. This phonetic structure, while systematic, creates challenges for algorithmic transformation between shared lexical roots and their written representations, as illustrated in Figure 5.1.

[1] Japanese employs two phonetic alphabets, hiragana and katakana, alongside kanji (ideographs) in its writing system.

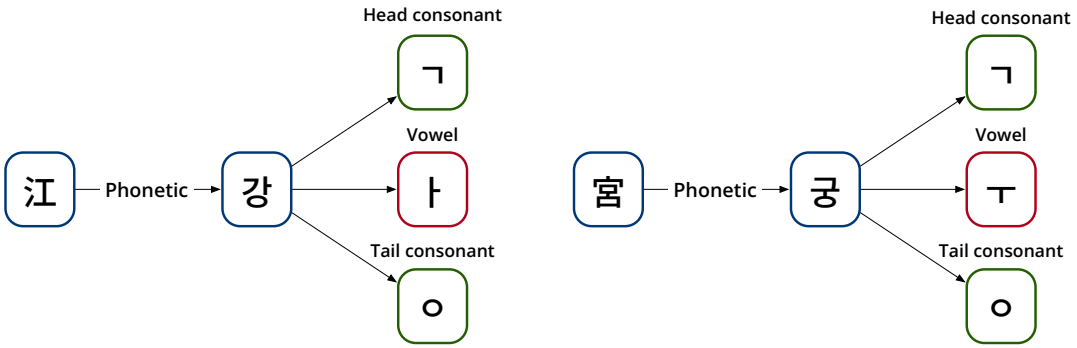


Figure 5.1: Two samples of transition from ideograph to syllable, syllable to phonemes (jamo).

Korean	Ideograph Candidates
강	傋僵冈江刚剛薑匹咣唌唬囥堈埐壃夆奋姜嫝岗岡岂崗螂嵣康弱弿強强彊忼慷扛捆摃摡杠棡椌樣橿殭江浰溤漮焖牥犅獍犟犴猜猛玒琋珧畔畺疆疆矼砊碉礓穅箊粇糠糡絳綱繈纃纊绛繵罡羌羌羟豺耩腔膙舡茳萎薔薑蛹袶襁講謽讲豇鰉鋼錒鏮鏹钢鐪降韁顜鰉剄苅娎康峒嵣慮掤擋枞柳歔洸犟瓶瓻旷痊碉磕廉緽蘠蝀蠔舩軑璗颶魧 (=140)
궁	匑匑穹宮宮弓杗焢焢穷穹窮竆芎蘑躬躬倗涅莮誇躬鉚 (=23)

Table 5.1: Example of 1:n relationship between syllable and ideographs.

The transcription process from ideographs to syllables inherently loses information, as illustrated in the first phonetic transform step of Figure 5.1. Reversing this process requires sophisticated contextual understanding to select the appropriate ideograph, as the phonetic information alone maps to multiple possible characters (Table 5.1). This challenge parallels the next-token disambiguation problem in language modeling tasks. The inherent complexity of these phonetic-to-ideographic mappings makes it particularly difficult to create a unified character system that effectively bridges Korean, Chinese, and Japanese writing systems.

The Korean writing system utilizes *Hangul*, a compact phonetic script comprising consonants and vowels, called *Jamo*, as its foundation. Following the notation established by Stratos (2017), we define $\mathcal{J}$ as the Jamo alphabet capable of representing all possible Korean characters $\hat{\Sigma}$ through composition. As shown in Table 5.2, $|\mathcal{J}| = 51$, with each component serving a specific role in character formation.

As discussed in Section 2.5.3, the Unicode standard disregards this alphabetic foundation, instead representing Korean syllables as composite phonetic units across 11,172 codepoints. This approach significantly complicates the process of learning optimal subword boundaries in both merge-driven methods, such as BPE (Sennrich et al., 2016), and prune-driven methods, including SentencePiece (Kudo and Richardson, 2018; Kudo, 2018).

The *computational* Hangul alphabet can be formally defined as $\mathcal{J} = \mathcal{J}_h \cup \mathcal{J}_v \cup \mathcal{J}_t$, where $\mathcal{J}_h$ repre-

Level	Computational Jamo (Subcharacters)
$\mathcal{J}_h$, Head consonant	ᄀ ᄁ ᄂ ᄃ ᄄ ᄅ ᄆ ᄇ ᄈ ᄉ ᄊ ᄋ ᄌ ᄍ ᄎ ᄏ ᄐ ᄑ ᄒ
$\mathcal{J}_v$, Vowel	ᅡ ᅢ ᅣ ᅤ ᅥ ᅦ ᅧ ᅨ ᅩ ᅪ ᅫ ᅬ ᅭ ᅮ ᅯ ᅰ ᅱ ᅲ ᅳ ᅴ ᅵ
$\mathcal{J}_t$, Tail consonant	<nil> ᆨ ᆩ ᆪ ᆫ ᆬ ᆭ ᆮ ᆯ ᆰ ᆱ ᆲ ᆳ ᆴ ᆵ ᆶ ᆷ ᆸ ᆹ ᆺ ᆻ ᆼ ᆽ ᆾ ᆿ ᇀ ᇁ ᇂ
Level	**Written Jamo (Subcharacters)**
Head consonant	ᄀ ᄁ ᄂ ᄃ ᄄ ᄅ ᄆ ᄇ ᄈ ᄉ ᄊ ᄋ ᄌ ᄍ ᄎ ᄏ ᄐ ᄑ ᄒ
Base vowel	ᅡ ᅣ ᅥ ᅧ ᅩ ᅭ ᅮ ᅲ ᅳ ᅵ
Compound vowel	ᅢ ᅤ ᅦ ᅨ ᅪ ᅫ ᅬ ᅯ ᅰ ᅱ ᅴ
Tail consonant	<nil> ᆨ ᆩ ᆫ ᆮ ᆯ ᆷ ᆸ ᆺ ᆻ ᆼ ᆽ ᆾ ᆿ ᇀ ᇁ ᇂ
Compound tail consonant	ᆪ ᆬ ᆭ ᆰ ᆱ ᆲ ᆳ ᆴ ᆵ ᆶ ᆹ

Table 5.2: Computational and written (or typed) Jamo alphabet contrasted, along with their roles for composition.

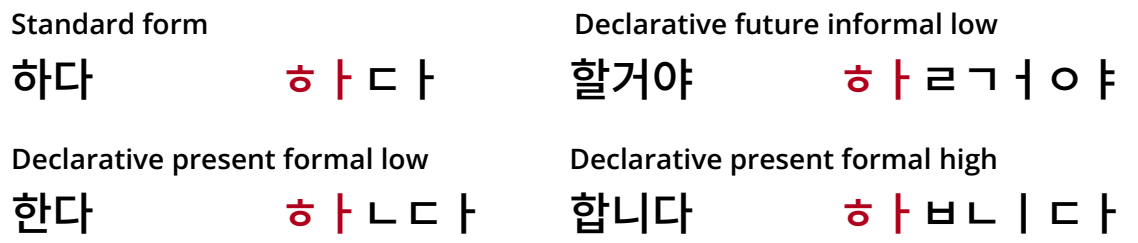


Figure 5.2: Common Jamo surface morpheme under different verb conjugations.

sents Choseong (head consonant), $\mathcal{J}_v$ represents Jungseong (vowel), and $\mathcal{J}_t$ represents Jongseong (tail consonant). This structure is exemplified in Figure 5.1, where ᄀ$\in \mathcal{J}_h$, ᅡ$\in \mathcal{J}_v$, and ᄋ$\in \mathcal{J}_t$. It is worth noting that $\mathcal{J}_t$ may be omitted, in which case it is represented as <nil> $\in \mathcal{J}_t$.

The distinction between *computational* and *written* Hangul is not immediately apparent without knowledge of Korean writing conventions. For computational processing of Jamo, one must use specific offsets that point to compound Jamo variants, whereas in written or typed Korean, base variants are used for character composition. While this adds complexity from a *computational* perspective, working directly with text stored in computer memory allows for significant algorithmic simplification.

Previous work has explored these characteristics in various contexts: as part of an end-to-end architecture (Stratos, 2017), for learning subword embeddings (Park et al., 2018), and in classification tasks (Cho et al., 2019). However, a critical limitation of existing methods is their lack of guaranteed round-trip consistency. While these approaches discuss sub-character (Jamo) based methods, their evaluation was limited to non-generative tasks, and reconstruction capabilities were not addressed. Our work attempts to resolve this limitation, drawing parallels to how subword tokenization methods introduced guarantees of lossless encoding and decoding, in contrast to conventional lossy methods such as lemmatization, stemming, and other normalization approaches.

5.2. Method

Our method addresses the Unicode-related challenges described in the previous section by exposing both the alphabetic composition characteristics and the agglutinative nature of Korean to subword tokenizers. A key feature of our approach is the introduction of a strict round-trip requirement, ensuring lossless reconstruction of transformed text while maintaining compatibility with multilingual training scenarios.

The implementation leverages unused code points in the Hangul Unicode block as processing hints. These special characters, while invisible in the final output, are processed as standard characters within the same Unicode block by tokenizers. This design choice is essential for maintaining compatibility with tokenizer implementations that restrict merging operations to character pairs from the same Unicode block.

The method comprises two distinct modules: a pre-processor for Jamo decomposition and a post-processor for reconstruction into human-readable form. While the pre-processor must be applied to all input text, the post-processor is only necessary for generative tasks where human readability is required.

We present two algorithmic variants: an **aligned** method that maintains a fixed three-character grid structure, and an **automaton** method that implements a more flexible, state-machine-based approach. Both variants incorporate special handling for orphaned Jamo (e.g., ㅋㅋ, traditionally used to represent laughter in digital communication) by prefixing them with post-processor hints to ensure proper reconstruction.

The distinction and trade-offs between these approaches can be illustrated with a romanized example. Consider the first example in Figure 5.2, **하다/ha-da**), which contains two syllabic blocks: **하/ha** and **다/da**. This differs from the second example, **한다/han-da**, in that it lacks tail consonants. The process of decomposing this example into Jamo and reconstructing it back to syllables proceeds as follows:

- **Decomposition**: 하/ha 다/da → ᄒ/h ᅡ/a ᄃ/d ᅡ/a
- **Reconstruction**: ᄒ/h ᅡ/a ᄃ/d ᅡ/a → 하/ha 다/da

This reconstruction process reveals a significant challenge. When using a left-to-right parser, after processing **ᄒ/h ᅡ/a**, the parser will place the next Jamo **ᄃ/d** into the tail consonant position. This creates a single unit **ᄒ/h ᅡ/a ᄃ/d**, leaving the subsequent vowel **ᅡ/a** isolated. As a result, the output becomes “**핟ᅡ**” (**ᄒ/h ᅡ/a ᄃ/d - ᅡ/a**) instead of “**하다**” (**ᄒ/h ᅡ/a - ᄃ/d ᅡ/a**). Our work

explores two distinct approaches to prevent this misalignment.

These two approaches offer distinct advantages and trade-offs. The **aligned** method facilitates reconstruction through a straightforward post-processing step and provides robust reconstruction guarantees. In contrast, the **automaton** method requires a more sophisticated state-machine-based post-processor, operating similarly to Input Method Processors (IMEs), but offers greater flexibility in character composition.

5.2.1. General Decomposition

The decomposition process leverages Unicode-specific properties of Korean text, following principles similar to Unicode's Normalization Form Compatibility Decomposition (NFKD), but with modifications intended to enhance reconstruction reliability when processing non-deterministic model outputs, such as those from partially converged models. The decomposition operates through arithmetic operations on the Unicode codepoints. For a given character c with integer Unicode codepoint c_i, the decomposition process requires constants $k_1 = 44032$ (Hangul syllable base), $k_2 = 588$ (jamo per block), and $k_3 = 28$ (jongseong per block). The decomposition is formally defined as:

$$\begin{aligned} c_i' &= c_i - k_1 \\ i_h &= \frac{c_i'}{k_2} \\ i_v &= \frac{c_i' - (k_2 \cdot i_h)}{k_3} \\ i_t &= (c_i' - (k_2 \cdot i_h)) - k_3 \cdot i_v \end{aligned} \tag{5.1}$$

These constants represent specific offset values within the Unicode Korean block. The computed indices $i_h \in [0, 18]$, $i_v \in [0, 20]$, and $i_t \in [0, 27]$ correspond to entries in Table 5.2 for $\mathcal{J}_h$, $\mathcal{J}_v$, and $\mathcal{J}_t$ respectively.

For orphaned Jamo characters occurring independently rather than as part of a composite character, we implement a prefix-based identification mechanism using the character **U+115F** (Hangul Choseong Filler, <o>). During reconstruction, this character serves as a look-ahead hint, signaling independent character processing rather than composite character construction. This approach is optimized for the relative rarity of orphaned Jamo in standard text.

The decomposition procedure is applied selectively within a corpus C: for each character

Input	강	강가	ㅋㅋ
Output	ㄱㅏㅇ	ㄱㅏㅇㄱㅏ<f>	<o>ㅋ<o>ㅋ

Table 5.3: An example of the decomposition in aligned mode, with orphan hints (<o>) and fillers (<f>).

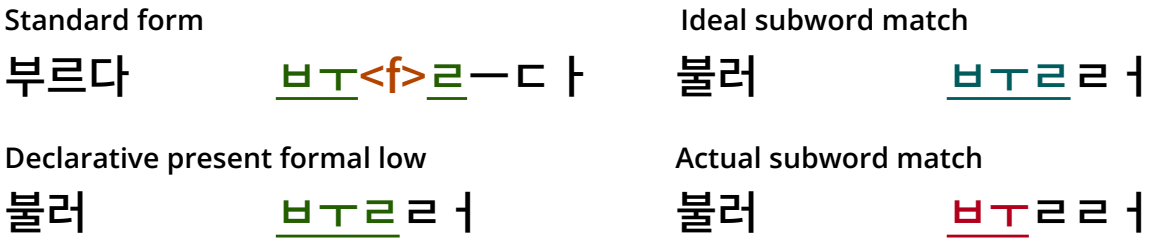


Figure 5.3: Filler (<f>) interfering a morpheme surface merge.

$c \in \hat{\Sigma}$, if c has a codepoint within the composite Korean character range (`0xAC00-0xD7A3`), the decomposition is performed; otherwise, c remains unmodified. This selective application ensures compatibility with mixed-script text while maintaining the integrity of non-Korean characters.

5.2.2. Aligned Processing

The *aligned* processing approach employs post-decomposition padding to ensure that both decomposition and reconstruction operations proceed in consistent triplets of Jamo characters. This alignment strategy significantly simplifies the reconstruction process, particularly in error state handling, by maintaining predictable character boundaries.

Aligned Decomposition

In the aligned method's pre-processor, we systematically replace instances of `<nil>` $\in \mathcal{J}_{t_{i_v}}$ with the special character **U+11FF** (Hangul Jongseong Ssangnieun) from the Unicode Jamo block. This specific character was selected because input is impossible through normal means from an Input Method Editor (IME), making it ideal as a processing marker. Furthermore, this replacement ensures compatibility with tokenization algorithms that restrict merging operations to character pairs within the same Unicode block.

The introduction of this marker enables the post-processor to implement a consistent three-character read-ahead strategy when encountering a Choseong (head consonant).

Aligned Reconstruction

The reconstruction process then operates as an inverse transform of the decomposition procedure to recover the original codepoint c_i. For input text C, the post-processor examines each character $c \in$

Input	강	강가	ㅋㅋ
Output	ㄱㅏㅇ	ㄱㅏㅇㄱㅏ	<o>ㅋ<o>ㅋ

Table 5.4: Decomposition in automaton mode, with orphan hints (<o>) but without fillers (<f>).

C. Upon encountering a character in the Choseong set $\mathcal{J}_h$, it reads the subsequent two characters, which must correspond to elements of $\mathcal{J}_v$ and $\mathcal{J}_t$ respectively.

Given the triplet of characters c_h, c_v, and c_t, we determine their respective indices i_h, i_v, and i_t such that:

$$\begin{aligned} c_h &= \mathcal{J}_{h_{i_h}} \\ c_v &= \mathcal{J}_{v_{i_v}} \\ c_t &= \mathcal{J}_{t_{i_t}} \end{aligned} \tag{5.2}$$

When c_t corresponds to the filler character **U+11FF**, we set $i_t = 0$. The inverse transform then reconstructs the original character's codepoint through the following formula:

$$c_i = k_1 + (i_h \cdot k_2 + i_v \cdot k_3 + i_t) \tag{5.3}$$

While this aligned approach offers implementation simplicity, it introduces certain constraints on the exposure of Korean's agglutinative properties. The filler character acts as a merge constraint during vocabulary training, forcing alignment to complete character boundaries in cases where the filler appears. This characteristic can result in suboptimal handling of shared morphemes, as demonstrated in Figure 5.3.

5.2.3. Automaton (Unaligned) Processing

The *automaton* method, also referred to as the *unaligned* variant, eliminates the need for padding characters and fixed triplet alignment by implementing a state machine-based reconstruction approach. This design enables flexible reconstruction of the original character sequence without the constraints of a fixed window size. The method draws inspiration from Korean Input Method Editor (IME) implementations, though it employs a simplified state machine architecture optimized for text reconstruction.

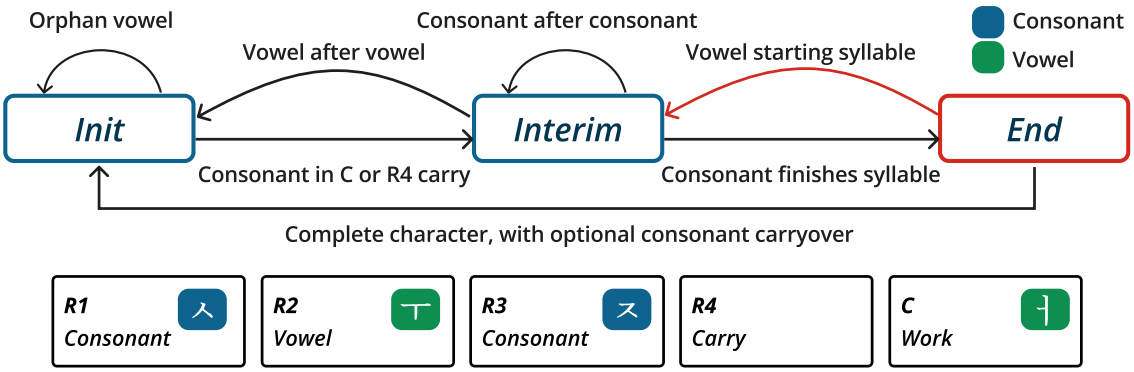


Figure 5.4: State machine explaining the automaton method, demonstrating consonant carryover state transition.

Automaton Decomposition

The automaton method's decomposition process follows the same fundamental principles outlined in Equation 5.1. However, it introduces a key optimization: the filler character (<f>) becomes optional since the post-processing state machine can autonomously transition to a completed character state without explicit padding signals. Table 5.4 illustrates the practical implications of this distinction between the aligned and automaton approaches.

Automaton Reconstruction

The unaligned nature of the Jamo output necessitates a stateful parser for reconstruction. Our approach implements a variant of the state machine commonly used in Korean IMEs. Our state machine implementation for reconstruction adopts a simplified variant of Korean IME architectures, with two key optimizations:

- **Unidirectional Processing**: Eliminates state rollback mechanisms as model generation is forward-only
- **Direct Jamo Handling**: Bypasses compound validation since input consists of pre-validated computational Jamo

Standard Korean IME implementations necessarily support bidirectional text manipulation, as users require deletion capabilities during text input. This deletion functionality typically operates at the Jamo level, requiring the state machine to maintain and restore previous states. While character-level deletion is supported in both initial and end states, the complexity of state restoration is unnecessary for our unidirectional generation task. By eliminating this bidirectional requirement, we achieve significant simplification of the state machine architecture.

The removal of Jamo compounding operations provides additional algorithmic simplification.

Algorithm 1 Pseudocode for the automaton reconstruction algorithm.

```
Reset: R1, R2, R3, R4 ← {0, 0, 0, 0}; Initialize: S ← INIT;
for c ∈ C such that c ∈ J do
    if S = INIT then
        if c ∈ Jv then
            if R4 ≠ 0 then
                Compose(R1, R2, R3)
                R1, R2, R3, R4 ← {R4, c 0, 0}
            end if
        else
            if R4 ≠ 0 then
                I ← Combine(R3, R4)
                if I ≠ R3 then
                    Compose(R1, R2, I)
                else
                    Compose(R1, R2, R3)
                    Compose(R4, 0, 0)
                end if
            else
                Compose(R1, R2, R3)
            end if
            Reset
            R1 ← c; S ← INTERIM;
        end if
    else if S = INTERIM then
        if c ∈ Jv then
            if R2 ≠ 0 then
                I ← Combine(R2, c)
                if I ≠ R2 then
                    Compose(R1, I, 0)
                else
                    Compose(R1, R2, 0)
                    Compose(0, c, 0)
                end if
                Reset; Initialize;
            else
                R2 ← c
            end if
        else
            if R2 ≠ 0 then
                R3 ← c; S ← END
            else
                Compose(R1, 0, 0)
                Reset; R1 ← c; S ← END
            end if
        end if
    else if S = END then
        if c ∈ Jv then Compose(R1, R2, 0)
            if R3 ≠ 0 then
                R1, R2, R3, R4 ← {R3, c, 0, 0}
                S ← INTERIM
            else
                Reset; Initialize;
            end if
        else
            I ← Combine(R3, c)
            if I ≠ R3 then
                Initialize; R3 ← I;
            else
                Compose(R1, R2, R3)
                Reset; R1 ← c; S ← INTERIM;
            end if
        end if
    end if
end for
```

Algorithm 2 Pseudocode for a simplified automaton reconstruction algorithm.

Reset: $R_1, R_2, R_3, R_4 \leftarrow \{0, 0, 0, 0\}$; **Initialize:** $S \leftarrow$ `INIT`;
for $c \in C$ **such that** $c \in \mathcal{J}$ **do**
    if $S =$ `INIT` **then**
        if $c \in \mathcal{J}_v$ **then**
            if $R_4 \neq 0$ **then**
                Compose(R_1, R_2, R_3)
                $R_1, R_2, R_3, R_4 \leftarrow \{R_4, c\, 0, 0\}$
            end if
        else
            if $R_4 \neq 0$ **then**
                Compose(R_1, R_2, R_3)
                Compose$(R_4, 0, 0)$
            else
                Compose(R_1, R_2, R_3)
            end if
            Reset
            $R_1 \leftarrow c$; $S \leftarrow$ `INTERIM`;
        end if
    else if $S =$ `INTERIM` **then**
        if $c \in \mathcal{J}_v$ **then**
            if $R_2 \neq 0$ **then**
                Compose$(R_1, R_2, 0)$
                Compose$(0, c, 0)$
                Reset; Initialize;
            else
                $R_2 \leftarrow c$
            end if
        else
            if $R_2 \neq 0$ **then**
                $R_3 \leftarrow c$; $S \leftarrow$ `END`
            else
                Compose$(R_1, 0, 0)$
                Reset; $R_1 \leftarrow c$; $S \leftarrow$ `END`
            end if
        end if
    else if $S =$ `END` **then**
        if $c \in \mathcal{J}_v$ **then Compose**$(R_1, R_2, 0)$
            if $R_3 \neq 0$ **then**
                $R_1, R_2, R_3, R_4 \leftarrow \{R_3, c, 0, 0\}$
                $S \leftarrow$ `INTERIM`
            else
                Reset; Initialize;
            end if
        else
            Compose(R_1, R_2, R_3)
            Reset; $R_1 \leftarrow c$; $S \leftarrow$ `INTERIM`
        end if
    end if
end for

In typical IME implementations, compounding requires validation of intermediate state values against compound characters in $\mathcal{J}_v$ and $\mathcal{J}_t$ when consecutive vowels or consonants are encountered. However, this validation becomes unnecessary in our approach because the decomposition process ensures that the input sequence consists of *computational* Jamo, where all components are already in their appropriate compound form.

The state machine architecture employs four purpose-specific registers (R_1, R_2, R_3, R_4), a scratch register (I) and has three distinct states, as illustrated in Figure 5.4: initial, interim, and end. The initial state activates when the current character c satisfies $c \in \mathcal{J}_h$. The interim state represents intermediate conditions where registers are partially populated but insufficient for character emission. The end state occurs upon register completion, triggering character emission.

The register system accommodates values according to the following constraints: $R_1 \in \mathcal{J}_h$, $R_2 \in \mathcal{J}_v$, $R_3 \in \mathcal{J}_t$, and $R_4 \in \mathcal{J}_h$. A complete pseudocode implementation is detailed in Algorithm 1, along with a simplification in Algorithm 2, which omits handling of compound characters.

Character composition is triggered under two primary conditions: complete population of registers R1 through R3, or population of R1 and R2 combined with a termination condition. A termination condition can arise when R2 is populated and a vowel is encountered, as our method utilizes pre-combined vowels. In such cases, the vowel position saturation necessitates the emission of the current register values as a character, followed by a register reset. This scenario is formally specified in Algorithm 1 where $S = \text{INTERIM}, c \in \mathcal{J}_v, R2 \neq 0$.

The carryover operation represents a special case utilizing R4, occurring when R1 through R3 are populated and an additional consonant is encountered. In this scenario, the incoming consonant is assigned to R4, and the state machine returns to its initial state.

The **Compose** function serves as an interface to the c_i reconstruction algorithm previously defined in Section 5.2.2. When **Compose** receives the arguments c_h, c_v, and c_t, it determines indices i_h, i_v, and i_t such that $c_h = \mathcal{J}_{h_{i_h}}$, $c_v = \mathcal{J}_{v_{i_v}}$, and $c_t = \mathcal{J}_{t_{i_t}}$. An exception exists where c_h is processed as an orphaned character if $c_v = c_t = 0$.

The optional **Combine()** function performs consonant combination operations, mapping separate consonants and vowels (e.g., ㄹ + ㄱ) in $\mathcal{J}_t$ and $\mathcal{J}_v$ to their compound form (e.g., ㄺ). This mapping is implemented through a predefined lookup table of valid compound consonant characters (e.g., ㄹ + ㄱ $\rightarrow$ ㄺ and ㅡ + ㅣ $\rightarrow$ ㅢ).

Model	Supported languages	Vocab Size	Korean Vocab Size	Average
mBERT	104	119547	1568	5.39 / 1.35
XLM	100	200000	2581	6.61 / 1.37
KoBERT	1	8002	7378	2.70 / 1.67
BERTKr	1	32006	11607	3.04 / 2.58

Table 5.5: Comparison of Korean capabilities across different models.

5.3. Experiments

Our experimental evaluation is situated within the context of unsupervised, subword-tokenized large corpus pre-training in multilingual settings—the context that initially revealed the challenges our work addresses. We assess our method against several established baseline models, with BERT (Devlin et al., 2019) serving as a primary reference point through its multilingual pre-trained model (bert-base-multilingual-cased). Additional comparisons include XLM's (Conneau and Lample, 2019) 100-language model (xlm-mlm-100-1280) and two specialized Korean BERT implementations: KoBERT[2] and BERT Korean (BERTKr)[3]. For these baseline models, we limit our evaluation specifically on their tokenization components.

To evaluate our methods, we conducted experiments using the SentencePiece tokenizer (Kudo and Richardson, 2018; Kudo, 2018), training on a diverse corpus comprising 500,000 randomly sampled sentences from the kosentences Wikipedia dataset and 50,000 sentences from the Naver Sentiment Movie Dataset's test set. The latter was specifically included to represent user-generated content. Using this combined corpus, we trained SentencePiece models across multiple vocabulary sizes: 350, 1K, 2.5K, 5K, and 10K tokens. During training, we excluded sentences exceeding the maximum designated length. The complete set of tokenizer training hyperparameters is documented in the appendix.

5.3.1. Datasets

Our experimental evaluation employs multiple datasets to assess the effectiveness of our method across different domains. Given that our primary focus is on tokenization, we utilize these corpora specifically to validate our method's robustness against domain shift. This targeted approach acknowledges the varying performance characteristics of underlying models and the unavailability of their pre-training corpora. Consequently, our evaluation emphasizes tokenization-level metrics.

[2]https://github.com/SKTBrain/KoBERT
[3]https://github.com/yeontaek/BERT-Korean-Model

For vocabulary training, we utilize varying-sized subsets from the Wikipedia portion of the kosentences corpus (Moon et al., 2022). Consistent with standard transfer learning protocols, we learn the vocabulary during the pre-training phase and apply it unchanged to downstream tasks. Accordingly, we evaluate tokenization performance across train-test splits.

Naver Sentiment Movie Corpus

The Naver Sentiment Movie Corpus (Cho et al., 2020)[4] comprises 200,000 user-generated movie reviews with associated sentiment labels. Our evaluation utilizes only the review text from the training split, as the test set data was incorporated during vocabulary pre-training.

KorQuAD 1.0

KorQuAD[5] represents a Korean adaptation of the SQuAD 1.0 (Rajpurkar et al., 2016) reading comprehension task. The dataset encompasses 10,645 passages and 66,181 questions, requiring systems to generate answers based on provided text passages. Our evaluation incorporates all passages, questions, and answers from the dataset.

Text Mining Dataset

The Text Mining Dataset[6] consolidates Korean text from two distinct sources: Naver news and Naver movie reviews. The news corpus contains 1,661,786 sentences from article content and 2,655,847 user comments, while the movie review corpus comprises 3,280,685 review entries.

Namuwiki

The kosentences dataset integrates content from both Wikipedia and Namuwiki sources. Namuwiki content exhibits characteristics of casual writing and contains substantial amounts of paraphrased non-Korean text. We deliberately restricted our pre-training to the Wikipedia portion to leverage Namuwiki as an evaluation dataset.

Given the extensive size of the Namuwiki corpus (exceeding 27 million sentences), we conducted our evaluation on a random sample of 1 million sentences to ensure computational feasibility while maintaining statistical significance.

5.3.2. Hyperparameters

The SentencePiece model training utilized specific hyperparameter configurations to ensure optimal performance. The normalization rule was set to identity to prevent automatic Unicode

[4] Available at: https://github.com/e9t/nsmc
[5] Available at: https://korquad.github.io/
[6] Available at: https://github.com/lovit/textmining_dataset

normalization from recombining decomposed characters. A maximum sentence length threshold of 8,000 tokens was implemented, and the unigram model type was selected for tokenization.

All of the SentencePiece models used for the experiments share the following parameters:

- **model_type**: unigram
- **max_sentence_length**: 8000
- **normalization_rule_name**: identity

Character coverage rates were calibrated according to vocabulary size through a manual search:

- $|V| = 350$: coverage rate of 0.994
- $|V| = 1,000$: coverage rate of 0.995
- $|V| = 2,500$: coverage rate of 0.997
- $|V| = 5,000$: coverage rate of 0.9985
- $|V| = 10,000$: coverage rate of 1.0

These coverage rates were empirically determined to optimize the trade-off between vocabulary completeness and model efficiency across different vocabulary sizes. The progressive increase in coverage rates corresponds to the expanding capacity for character representation at larger vocabulary sizes.

5.3.3. Results

Out-of-Vocabulary Robustness

The primary contribution of this work lies in its significant improvement of OOV handling capabilities. Our method reduces the minimum required character- level budget for representing all standard cases (excluding the three extension blocks) from 22,852 character-level subwords to a theoretical minimum of 102 character-level subwords[7].

The experimental results presented in Figure 5.5, Figure 5.6, Table 5.6, and Table 5.7 demonstrate substantial improvements in OOV mitigation compared to baseline approaches, even with significantly reduced vocabulary sizes allocated for Korean subwords. Due to the absence of well-established metrics for evaluating OOV robustness, we developed the following intrinsic metrics as an evaluation criteria:

[7] The combined Unicode Hangul Syllables and Jamo contain 11,426 characters, necessitating approximately double this number in the vocabulary for most subword tokenizers.

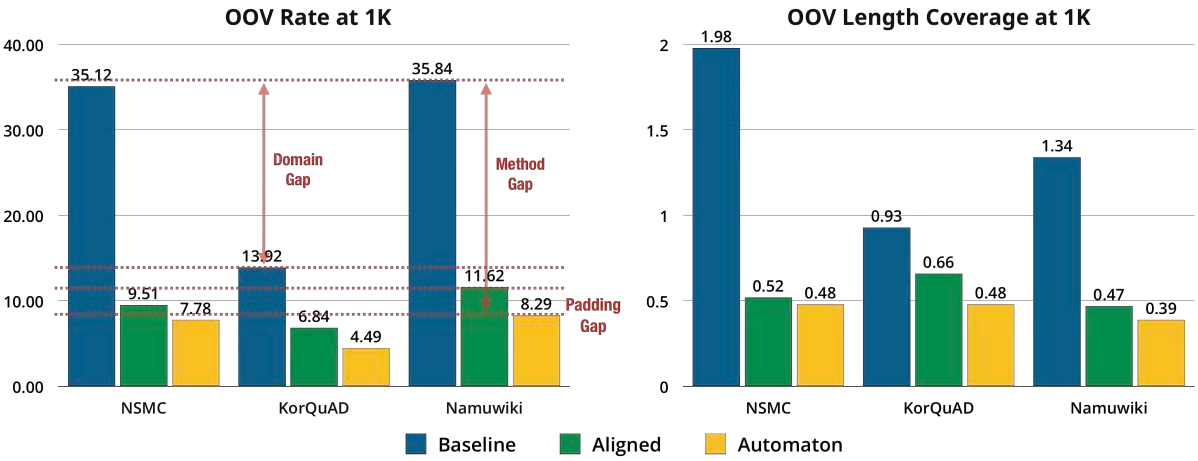


Figure 5.5: OOV Rate (OR) and OOV Length Coverage (LC) compared at 1K vocabulary size.

- **OOV Count (OC)**: Total number of OOV tokens encountered
- **OOV Rate (OR)**: Percentage of sentences containing OOV tokens, percentile scale
- **Length Coverage (LC)**: Ratio of total OOV token surface length to original surface length
- **Mean Length (ML)**: Average length of OOV token surfaces

With a vocabulary size of 10K, where Korean utilizes less than 25% of the total budget, our method achieves consistently negligible OOV rates below 0.2% across multiple domains. This performance contrasts sharply with the domain-dependent behavior observed in baseline models.

Notably, these improvements are achieved despite training on a smaller dataset compared to the baselines. The effectiveness stems from our decomposition approach, which reduces the character-level vocabulary requirements. This reduction enables improved handling of rare combinations in $\mathcal{V}_v$ and $\mathcal{V}_t$, which traditionally require extensive training corpora when working with composite characters.

Elasticity of Vocabulary Budget

Our method's reduction of the minimum character-level vocabulary requirement to 102 subwords has significant implications for vocabulary allocation. This reduction enables the allocation of approximately 22,750 additional slots for meaningful subwords, rather than reserving them for complete characters that may rarely appear in the corpus. This optimization is particularly valuable for pure Korean corpora.

To validate the enhanced vocabulary budget elasticity, we conducted extensive experiments across multiple vocabulary sizes, as detailed in Figure 5.8 and Table 5.6. The results demonstrate that even with limited vocabulary sizes (e.g., 1K tokens), our method achieves OOV handling capa-

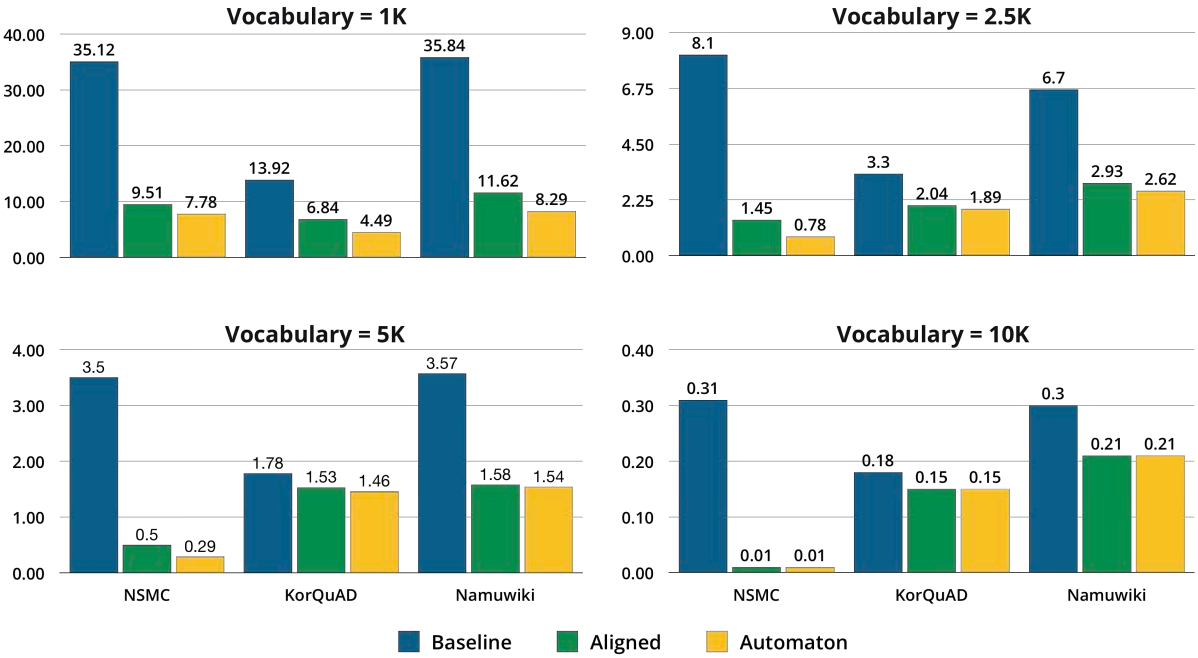


Figure 5.6: OOV rates (OR) across different vocabulary sizes.

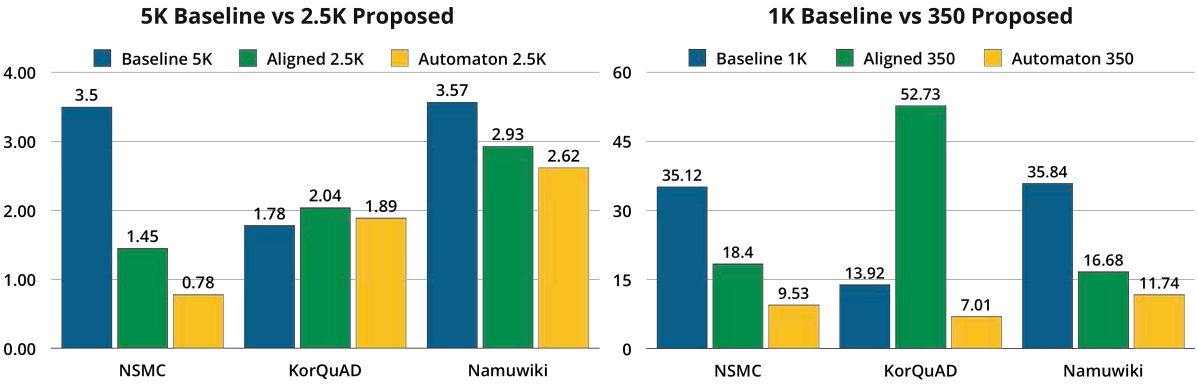


Figure 5.7: OOV rates (OR) on larger vocabulary baseline contrasted with proposed using a smaller vocabulary.

bilities comparable or superior to tokenizers utilizing substantially larger vocabularies. Through empirical analysis with a coverage rate threshold of 0.994, we established theoretical lower boundaries of 109 subwords for the aligned variant and 116 subwords for the automaton (unaligned) variant. While these alphabet-sized vocabularies serve primarily as theoretical minimums and may not be practical for real-world applications, they demonstrate the method's efficiency in vocabulary utilization.

Subword Surface Length

The length of subword surfaces significantly impacts model performance through two primary mechanisms: sequence length reduction and computational efficiency. Shorter sequences mitigate the risk of truncation or omission due to architectural constraints, while also reducing com-

Model	Size	NSMC				TM News-Body				TM News-Comments			
		OC	OR	LC	ML	OC	OR	LC	ML	OC	OR	LC	ML
mBERT	120K	81603	30.08	6.10	5.80	151175	8.26	0.50	4.17	1351933	35.83	5.60	6.58
XLM	200K	259266	62.78	4.53	2.06	2627434	61.86	2.62	2.92	4233920	67.39	3.92	2.45
KoBERT	8K	38480	14.98	0.96	1.84	30724	1.77	0.03	1.23	574058	17.00	0.81	2.00
BertKR	32K	63729	24.93	5.68	6.52	119017	6.46	0.34	3.58	1159335	31.36	4.55	6.11
SP	1K	101738	35.12	1.98	1.61	759562	30.70	0.78	1.76	2414089	51.92	2.45	1.99
	2.5K	18412	8.10	0.35	1.23	79231	4.60	0.08	1.23	408751	13.07	0.41	1.32
	5K	10096	4.50	0.19	1.19	30950	1.89	0.03	1.16	263799	8.77	0.26	1.25
	10K	662	0.31	0.01	1.15	659	0.04	0.00	1.08	24005	0.86	0.02	1.13
Aligned	350	48282	18.40	1.03	1.91	164518	9.89	0.00	1.00	359789	10.98	0.10	1.79
	1K	23912	9.51	0.52	2.20	60189	3.56	0.00	1.00	221439	7.16	0.08	1.70
	2.5K	3868	1.45	0.10	2.19	1708	0.11	0.00	1.00	4546	0.17	0.00	1.21
	5K	1376	0.50	0.04	2.22	422	0.03	0.00	1.00	1524	0.06	0.00	1.14
	10K	23	0.01	0.00	1.55	1	0.00	0.00	1.00	18	0.00	0.00	1.00
Automaton	350	23955	9.53	0.53	2.20	60189	3.56	0.00	1.02	221438	7.16	0.08	1.68
	1K	19016	7.78	0.48	2.22	30468	1.84	0.00	1.02	153298	5.23	0.04	1.43
	2.5K	2163	0.78	0.05	2.19	1708	0.11	0.00	1.00	4546	0.17	0.00	1.21
	5K	832	0.29	0.02	2.50	422	0.03	0.00	1.00	1524	0.06	0.00	1.14
	10K	23	0.01	0.00	1.55	1	0.00	0.00	1.00	18	0.00	0.00	1.00

Model	Size	TM Movies				KorQuAD				Namuwiki			
		OC	OR	LC	ML	OC	OR	LC	ML	OC	OR	LC	ML
mBERT	120K	1664275	37.37	9.18	6.51	14775	4.34	0.47	4.75	123873	10.17	0.82	3.70
XLM	200K	4770386	66.32	5.49	2.19	142926	25.05	2.22	3.98	1220239	62.69	2.63	1.95
KoBERT	8K	728933	17.38	1.23	1.87	21278	4.02	0.49	5.46	67869	4.94	0.26	2.58
BertKR	32K	1157216	27.26	5.79	5.63	15414	6.48	0.92	6.28	202436	15.58	1.44	4.24
SP	1K	1587614	34.29	2.02	1.56	46931	13.92	0.93	2.99	532261	35.84	1.34	1.73
	2.5K	519287	13.48	0.65	1.28	9793	3.30	0.19	2.53	94949	6.70	0.26	1.76
	5K	114981	3.20	0.14	1.16	5052	1.78	0.09	2.34	51700	3.57	0.13	1.66
	10K	15960	0.46	0.02	1.12	337	0.18	0.01	1.98	4651	0.30	0.01	1.74
Aligned	350	120149	3.21	0.04	1.47	108167	52.73	1.84	1.60	261738	16.68	0.63	2.39
	1K	91907	2.52	0.03	1.49	30295	6.84	0.66	5.30	175069	11.62	0.47	2.59
	2.5K	91907	2.52	0.03	1.49	9974	2.04	0.33	7.39	45762	2.93	0.25	4.04
	5K	540	0.02	0.00	1.00	7432	1.53	0.17	5.18	34634	1.58	0.17	4.92
	10K	0	0.00	0.00	0.00	291	0.15	0.01	2.17	3660	0.21	0.01	1.99
Automaton	350	91907	2.52	0.03	1.47	32194	7.01	0.69	5.34	177032	11.74	0.48	2.57
	1K	60536	1.70	0.02	1.40	17094	4.49	0.48	5.52	122446	8.29	0.39	2.79
	2.5K	60536	1.70	0.02	1.40	9491	1.89	0.28	6.85	40823	2.62	0.23	4.19
	5K	540	0.02	0.00	1.00	6745	1.46	0.15	4.67	36463	1.54	0.14	4.42
	10K	0	0.00	0.00	0.00	291	0.15	0.01	2.17	3660	0.21	0.01	1.99

Table 5.6: Complete experiment results comparing different tokenizers and datasets with our method.

putational requirements in sequence models. Our experimental results, presented in Table 5.7, demonstrate meaningful improvements in subword surface length characteristics.

The introduction of the aligned (Al) variant yields modest improvements in surface length compared to the SentencePiece baseline. However, the automaton (Au) variant achieves substantially better results, approximately doubling the surface length relative to other approaches with equivalent vocabulary budgets. This enhancement in surface length efficiency relative to allocated vocabulary size represents a significant advancement over baseline approaches, offering improved sequence compression while maintaining vocabulary efficiency.

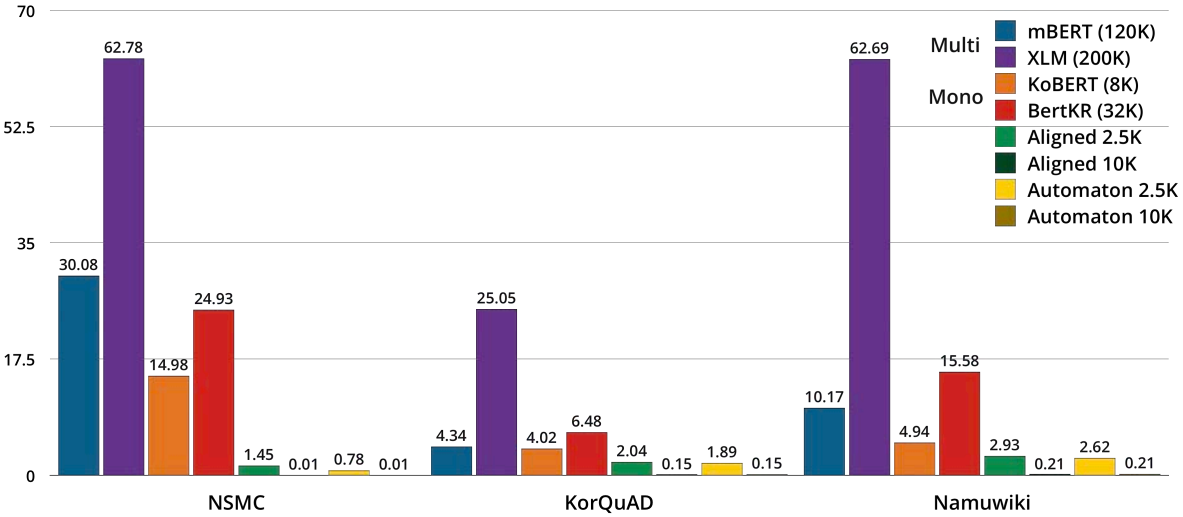


Figure 5.8: OOV rates (OR) comparing proposed method with other multilingual and monolingual models.

5.4. Limitations and Future Work

A significant limitation of both proposed methods lies in their dependency on well-formed model output for successful reconstruction. This challenge parallels the Unicode pair reconstruction issues observed in byte-level approaches, such as those employed by GPT-2 (Radford et al., 2019) and other byte-level tokenization methods (Gillick et al., 2016). In these cases, reconstruction failure can occur when the output sequence does not conform to valid Unicode character specifications, potentially compromising the integrity of the entire output sequence.

While potential mitigation strategies exist, such as incorporating checksum encodings within subword tokens to facilitate error detection and selective output filtering in generative tasks, these approaches remain unexplored in the current work. Future research could investigate these error-handling mechanisms to enhance the robustness of reconstruction processes, particularly in scenarios involving partially trained models or noisy input data.

5.5. Conclusion

This work addresses a significant but frequently overlooked challenge in multilingual pre-training: the computational inefficiencies and complexities associated with incorporating CJK languages, particularly Korean, into large- scale language models. Our research makes several key contributions to this domain.

First, we present a novel, tokenizer-agnostic methodology that enhances subword tokenization for Korean text while maintaining compatibility with existing multilingual frameworks. The

Model	Count	Min	Max	Mean	Std
mBERT	1568	1	5	1.32	0.62
XLM	2581	1	8	1.55	0.72
KoBERT	7378	1	9	2.12	0.93
BERTKr	11607	1	14	2.19	1.00
SP@1K	861	1	3	1.02	0.13
SP@2.5K	1902	1	5	1.31	0.57
SP@5K	2903	1	5	1.39	0.64
SP@10K	2748	1	3	1.02	0.15
Aligned@350	150	0.4	1.2	0.96	0.91
Aligned@1K	460	0.4	2	1.18	0.99
Aligned@2.5K	1001	0.4	3.6	1.46	1.42
Aligned@5K	1463	0.4	**4.8**	**1.59**	**1.67**
Aligned@10K	1049	0.4	3.6	1.44	1.50
Automaton@350	272	0.4	2.8	0.92	0.95
Automaton@1K	845	0.4	5.2	1.32	1.57
Automaton@2.5K	2125	0.4	6	1.71	1.93
Automaton@5K	3514	0.4	**6.4**	**1.90**	**2.05**
Automaton@10K	2466	0.4	6	1.72	1.88

Table 5.7: Count of Korean subwords, minimum, maximum, average, and variance of the token length. Jamo methods scaled down by 2.5 for a fair comparison.

method's efficiency is particularly noteworthy: requiring only a single pre-processing pass for unseen text and introducing no adverse effects in polyglot corpora. This approach not only facilitates more effective subword learning but also provides complete mitigation of out-of-vocabulary issues across all downstream Korean language tasks while substantially reducing the required character-level vocabulary budget.

Second, we introduce a fully reversible transformation process with an optimized state machine implementation for text reconstruction. By removing bidirectional processing requirements and simplifying Jamo compound validation, we achieve significant computational efficiency while maintaining reconstruction accuracy. These optimizations represent a departure from traditional Korean IME implementations, specifically tailored for the unidirectional nature of model generation tasks. This streamlined approach extends the method's applicability to generative downstream tasks, including machine translation, text summarization, and language modeling, while ensuring the preservation of linguistic information throughout the processing pipeline.

Our experimental results demonstrate that this methodology enhances both the flexibility and performance of current state-of-the-art approaches while effectively addressing common out-of-vocabulary challenges in multilingual model training. These improvements offer a cost-effective and practical pathway for incorporating Korean language processing into future multilingual research initiatives.

Our results demonstrate that the proposed methods significantly reduce information loss and enhance tokenization efficiency for Korean text processing. The following chapter examines these findings in the context of recent methodological advances in natural language processing. We analyze how our approach compares to newly proposed techniques and evaluate its continued relevance given evolving trends in multilingual language model development. This analysis provides valuable insights into the broader implications of our work for large-scale language processing applications.

6

Discussion

The field of Natural Language Processing has undergone transformative advancements since the initial publication of the methods proposed in this thesis. The emergence of large language models, particularly those built on the Transformer architecture, have demonstrated unprecedented capabilities in sequence modeling across diverse domains. These developments have driven significant progress in model architectures, training methodologies, and evaluation frameworks.

While previous chapters maintained fidelity to our original publications, this approach necessarily limited our discussion of subsequent methodological innovations. Given the rapid evolution of the field and renewed academic interest in tokenization approaches, an examination of recent developments is essential for contextualizing our work's continued relevance.

Recent research has introduced novel frameworks for the intrinsic evaluation of tokenization methods, providing valuable new perspectives for analyzing existing approaches. In light of these developments and the accelerating pace of architectural innovation in the field, a systematic re-examination of our proposed methods is warranted. This chapter presents a comprehensive analysis of our work through contemporary evaluation frameworks while assessing its ongoing applicability to current NLP systems.

6.1. Evaluation of Tokenization Quality

Historically, and also in our work, when a new method or enhancement for tokenization was proposed, they have usually been accompanied with results from extrinsic evaluations through a downstream task (Kudo and Richardson, 2018; Hiraoka et al., 2019; Provilkov et al., 2020; Hiraoka et al., 2020; Vilar and Federico, 2021; Hiraoka et al., 2021; Takase et al., 2022; Moon et al., 2022; Saleva and Lignos, 2023). The impact and influence of the tokenizer with downstream tasks were often used to justify the novelty and significance of the work.

While it is not the only extrinsic evaluation, often machine translation was used as the gold standard. As translation requires both requires tokenization and de-tokenization, across multiple language pairs, it is deemed to be a good proxy to estimate the proposed tokenization method's robustness. Additionally, due to many past investigations, shared tasks such as those published regularly from Conference on Machine Translation (WMT) or Workshop on Asian Translation (WAT) have well-established baselines for well-known methods.

When the work in this thesis was initially proposed, this was still the status quo; and until recent developments; this trend continued. Only in recent years tokenization as a stand-alone algorithm to be evaluated independently has come to light; earlier developments beginning with alignment with linguistically meaningful tokens (Klein and Tsarfaty, 2020; Hofmann et al., 2021, 2022; Gow-Smith et al., 2022) and token length and frequency (Bostrom and Durrett, 2020; Yehezkel and Pinter, 2023) along with initial findings of correlation to downstream performance (Gallé, 2019; Gutierrez-Vasques et al., 2023).

Later developments began to further correlate the tokenized sequence length and frequency distribution, otherwise tokenizer compression rates, and correlating these metrics with downstream task performance (Zouhar et al., 2023; Goldman et al., 2024; Ali et al., 2024) and also investigations where this trend did not hold (Cognetta et al., 2024; Schmidt et al., 2024). In this chapter, we revisit our work and evaluate the effects of our work through the lens of these recent developments.

6.1.1. Corpus Token Count, Fertility, Compression, and Information Theory

The relationship between sequence length and vocabulary size represents a fundamental trade-off in tokenization. While shorter sequences (achieved through higher compression rates at the cost of larger vocabularies $|\mathcal{V}|$) reduce computational costs in both attention-based and RNN architectures, excessive optimization for compression can lead to the inclusion of rare tokens in the vocabulary. These infrequently occurring tokens often result in undertrained representations, exemplified by phenomena such as the "SolidGoldMagikarp" effect (Land and Bartolo, 2024).

Corpus Token Count (CTC) represents the total number of tokens produced when tokenizing a corpus. Recent work (Gallé, 2019; Ali et al., 2024; Goldman et al., 2024) suggests that reduced sequence lengths correlate positively with downstream task performance.

Subword fertility (f_{sub}), introduced by (Ács, 2019) and further developed by (Rust et al., 2021), measures the tokenization expansion ratio:

$$f_{\text{sub}} = \frac{|T|}{|W|} \tag{6.1}$$

where $|T|$ is the number of subword tokens and $|W|$ is the number of original words. This provides a word-level interpretation of corpus token count (CTC), distinct from the statistical machine translation concept of fertility (ϕ) introduced by (Brown et al., 1993), which measures the number of target language words generated per source word. A subword fertility value of $f_{\text{sub}} = 1$ indicates complete word coverage in the tokenizer's vocabulary, meaning each word maps to exactly one token. Related metrics include the proportion of suffix subwords (p_{suffix})—tokens designed to merge with other subwords—and our proposed per-subword surface length (l_{surf}) measure, described in Section 5.3.3, which exhibits inverse correlations with both f_{sub} and p_{suffix}.

The application of these metrics to CJK languages presents unique challenges due to their reliance on word boundaries typically identified through spacing. The absence of consistent spacing in CJK texts, combined with the character-level segmentation commonly employed (e.g., BERT's handling of CJK ideographs), artificially deflates the proportion of suffix subwords. Consequently, our application of these metrics in subsequent sections should be considered approximate.

Information-theoretic metrics, particularly Rényi efficiency, have been proposed as tokenization quality metrics which estimate model quality. This approach, introduced by Zouhar et al. (2023), challenges the assumption of a Zipfian distribution in tokenization. Instead, it suggests that optimizing for a more uniform token distribution may be beneficial, as it can reduce the occurrence of undertrained, rare tokens.

Rényi efficiency is derived from Rényi entropy, a generalization of Shannon entropy. Shannon entropy measures the average amount of information needed to represent an event from a probability distribution, where X is a discrete random variable and $p(x)$ is the probability of outcome x:

$$H(X) = -\sum_{x \in X} p(x) \log p(x) \tag{6.2}$$

Rényi entropy introduces a parameter α (where $\alpha \geq 0$) that can emphasize different parts of the probability distribution. As α approaches 1, Rényi entropy converges to Shannon entropy:

$$H_\alpha(X) = \frac{1}{1-\alpha} \log \left(\sum_{x \in X} p(x)^\alpha \right) \tag{6.3}$$

Rényi efficiency is a normalized version of Rényi entropy, where $H_\alpha(X)$ is the Rényi entropy and

$|V|$ is the vocabulary size:

$$Eff_{\alpha}(X) = \frac{H_{\alpha}(X)}{\log |V|} \tag{6.4}$$

By accounting for vocabulary size, it becomes suitable for comparing tokenizers with different vocabularies. Experiments with small-scale machine translation models demonstrate that Rényi efficiency correlates with downstream performance, indicating its potential as a intrinsic measure of tokenizer quality. It's important to note that the term 'efficiency' here refers to Rényi efficiency, a tokenization metric, not computational efficiency. We will therefore refer to this metric as entropy in the upcoming sections for clarity.

Lastly, recent work by Schmidt et al. (2024) presents evidence that challenges prevailing assumptions about tokenization efficiency. Their study introduces a dynamic programming approach that achieves provably minimal sequence lengths given a fixed vocabulary ($\min_{S \in \mathcal{S}} |S|$ for vocabulary $\mathcal{V}$). Through experiments with large language models (350M-2.4B parameters), they demonstrate that corpus token count (CTC) exhibits only weak negative correlation ($\rho < -0.2$) with downstream performance. These findings suggest that the relationship between sequence length and model performance extends beyond simple compression metrics, with factors such as pre-tokenization method and vocabulary construction playing crucial roles in determining tokenizer effectiveness.

6.1.2. Parity

The concept of *parity* in multilingual tokenization, introduced by Petrov et al. (2023), provides a quantitative framework for assessing tokenization fairness across languages. This metric complements existing measures like subword fertility and extends our understanding of multilingual tokenization efficiency.

For a sentence s_A in language A and its translation s_B in language B, a tokenizer t achieves parity when the ratio of their respective tokenized lengths approaches unity:

$$\frac{|t(s_A)|}{|t(s_B)|} \approx 1 \tag{6.5}$$

where $|t(s_A)|$ represents the length of the tokenized sequence for sentence s_A. This ratio, termed the *premium* of language A relative to language B, serves as a key metric for evaluating tokenization equity.

While related to tokenization fertility metrics introduced by Ács (2019), which measure the

expansion ratio within a single language as $f_{\text{sub}} = \frac{|T|}{|W|}$, parity provides a cross-lingual perspective on tokenization efficiency. Both metrics capture aspects of tokenization quality: fertility quantifies how efficiently a tokenizer represents text within a language, while parity measures relative efficiency between languages for equivalent content.

Using parallel corpora such as FLORES-200 (Guzmán et al., 2019; Goyal et al., 2022; Costa-jussà et al., 2022), researchers can systematically compute and compare these premiums across different tokenization approaches. This analysis has revealed significant disparities in tokenization efficiency. For instance, languages like Shan exhibit premiums up to 15 times higher than English in popular models like GPT-4, indicating severe tokenization inequity. Such disparities particularly affect low-resource languages in multilingual models, with implications for both computational efficiency and accessibility of language technologies.

6.2. Retrospective: The Effects and Mitigation of OOV

6.2.1. Why does it work?

The effectiveness of our method stems primarily from reducing information loss during tokenization. Our experiments with artificially degraded models reveal that performance deteriorates at a near-exponential rate as Out-Of-Vocabulary (OOV) occurrences increase. This pattern emerges because WordPiece merge failures in Korean text affect larger surface areas compared to Japanese or Chinese text processing. For example, when a single subword OOV appears within a word-level token (bounded by whitespace or punctuation), it compromises the entire surface representation. This substantial information loss impairs model performance because the attention mechanism cannot identify relevant contextual relationships when critical tokens are missing.

The design of BERT's tokenization scheme presents a notable challenge for Korean language processing, as it lacks the single-character break exceptions that exist for other CJK languages. However, implementing such exceptions for Korean would likely prove ineffective, since individual Korean characters carry significantly less semantic information than individual Chinese or Japanese ideographs. This assessment aligns with current research trends, which show limited adoption of character-level Korean processing methods. The situation resembles the inefficiency of arbitrarily segmenting English text into three or four-character chunks without considering underlying bigram frequency distributions.

Our recovery experiments yielded unexpected insights into model behavior under corruption. While we initially hypothesized that extensive model corruption would reduce BERT to an overpa-

Table 6.1: OOV Recovery Performance Comparison

Method	Accuracy	Δ vs. Baseline
FastText Baseline	0.8550	—
CPT (worst case)	0.8604	+0.0054

Table 6.2: Translation performance using byte-level BPE across language pairs.

Language Pair	Sentence Pairs	BLEU	chrF	TER
en-zh	2.23M	0.5	1.9	127.3
en-ja	3.47M	1.6	6.5	467.2
ja-ko	2.99M	19.0	35.8	114.4

rameterized non-linear classifier, our results suggest a more nuanced interpretation. The FastText baselines reported by Park (2024) achieved 85.50% accuracy, notably lower than our worst-case synthetic OOV recovered results using CPT at 86.04(see Table 6.1, full results in Table 4.7). This performance gap indicates that the recovery process, combined with preserved model knowledge, produces better outcomes than training a non-linear classifier from scratch. The role of contextualization through attention mechanisms may provide additional performance benefits, though this remains an open research question requiring further investigation.

6.2.2. Validity as of Today

The landscape of tokenization has evolved significantly since our work's initial proposal. When first introduced, OOV issues were widespread, with our experiments revealing suboptimal tokenizer training even in monolingual models (Table 4.8). The greedy merging behavior of WordPiece in BERT particularly amplified our method's effectiveness. However, WordPiece itself has largely been superseded by SentencePiece, with the last significant mention from its original proposers dating to 2018 (Song et al., 2021).

Contemporary large-scale language models have largely resolved the OOV challenge through the adoption of byte-level BPE (BBPE) implementations atop BPE or SentencePiece, ensuring consistent token coverage through byte-level fallback mechanisms. This development effectively eliminates the original motivation for our OOV mitigation approach. XGLM (Lin et al., 2022) represents the last major model exhibiting traditional OOV issues, with subsequent large-scale models implementing complete vocabulary coverage.

Our method may retain utility for smaller-scale models that cannot implement byte-level fallback. While our approach's success primarily addressed WordPiece's suboptimal merging behavior, it could benefit BPE implementations with constrained vocabulary sizes. Our experiments with

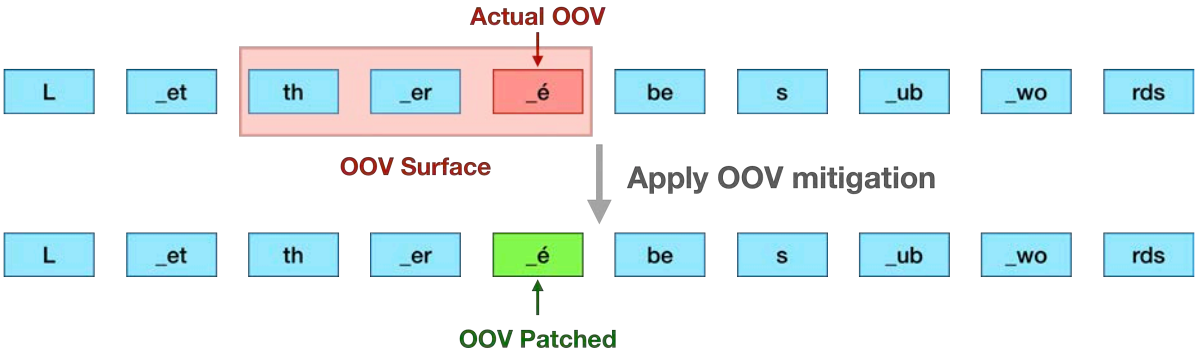


Figure 6.1: Comparison between OOV surface forms and subword OOV, with mitigation strategies illustrated.

a vanilla transformer across three language pair translation tasks (Barrault et al. (2020), AI Hub) demonstrate BBPE's limitations in smaller models (Table 6.2). These models often struggle to generate valid Unicode sequences, creating a potential application for our approach. However, implementing our method versus retraining models with expanded vocabularies requires careful cost-benefit analysis.

Despite these limitations, our method may find new applications in contemporary contexts, particularly in continuous pre-training (CPT) scenarios. Drawing on findings from Zouhar et al. (2023), we can leverage corpus entropy calculations to estimate domain adaptation potential for individual subwords. Land and Bartolo (2024)'s research suggests that rare tokens in training corpora adds a risk of undertrained tokens, indicating potential benefits from tokenizer optimization in these contexts. This application warrants further empirical validation but presents a promising direction for continued relevance.

6.2.3. Revisiting with Current Frameworks

Recent tokenizer quality metrics require complete vocabulary coverage (no OOV tokens) to produce comparable results. The presence of OOV tokens, particularly in the word-surface obstructing form encountered in WordPiece, introduces bias that significantly distorts these metrics.

To analyze this effect, we revisit our OOV example and fertility formulation from earlier, illustrated in Figure 6.1. The subword fertility f_{sub} is defined as the ratio between the corpus token count (CTC) of subword tokenization $|T|$ and the CTC of word tokenization $|W|$:

$$f_{\text{sub}} = \frac{|T|}{|W|} \tag{6.6}$$

This formulation requires refinement to account for vocabulary coverage. We must restrict T to

tokens t that exist in the vocabulary ($t \in \mathcal{V}$) and exclude words where the tokenized form produces OOV tokens (<unk>) to ensure fair comparison with WordPiece. For any randomly sampled word w from the corpus where the tokenized length exceeds the input length ($|t_w| > |w|$), this inequality persists when the tokenized OOV surface is $w =$ <unk>, as $|t_w| > 1$. Consequently, an OOV-patched tokenization T' will always satisfy $|T'| > |T|$, leading to:

$$\frac{|T'|}{|W|} > \frac{|T|}{|W|} \tag{6.7}$$

CTC: This metric becomes unsuitable for comparison in this context because $|T'| \geq |T|$ is invariant, with equality ($|T'| = |T|$) representing the optimal case.

Fertility: Without OOV filtering, the relationship $\frac{|T'|}{|W|} > \frac{|T|}{|W|}$ holds due to $|T'| = |T|$. When adjusted for OOV, we find $T' = T$ as the tokenizations become equivalent for non-OOV surfaces.

Entropy: From an information theoretic perspective, entropy changes depend fundamentally on modifications to the underlying vocabulary distribution. While adding subwords might appear to increase entropy, vocabulary patching through polysemic mappings maintains a constant $|\mathcal{V}|$. The process merely redistributes probability mass from <unk> to surrogate tokens.

Consequently, entropy improvements occur in proportion to the original <unk> frequency, though this improvement reverses if the surrogate token frequency becomes higher than the original <unk> frequency.

Parity: Including OOV in calculations axiomatically produces negative trends due to sequence length dependencies. When comparing a baseline T_s, comparison T_t, and patched T'_t with OOV discounted the parity is identical:

$$\frac{|T'_t|}{|T_s|} = \frac{|T_t|}{|T_s|} \tag{6.8}$$

This approximation holds because $|T'_t| = |T_t|$ is guaranteed when OOV tokens are excluded. Conversely, including OOV yields $|T'_t| \geq |T_t|$, with $|T'_t| = |T_t|$ representing the best possible outcome.

6.3. Retrospective: Jamo Pair Encoding

6.3.1. Why does it work?

The effectiveness of this method challenges conventional assumptions about Korean text representation. Rather than questioning why Jamo pair encoding succeeds, we should consider why syllabic block representation remains the prevalent standard despite its limitations. Beyond typesetting re-

quirements, the Unicode syllabic block representation introduces unnecessary complexity without providing significant computational advantages.

Our earlier analysis highlighted two critical challenges with syllabic block representation: the processing complexity of an extensive character set and its impact on vocabulary size. From an information theory perspective, this representation effectively implements an oversized, inefficient encoding at the character level. These inefficiencies compound significantly when combined with subword tokenization methods. The suboptimal nature of the base representation inevitably propagates to downstream processing methods built upon it.

Jamo pair encoding succeeds by exposing the fundamental phonetic units directly to tokenization algorithms. This approach reduces the problem complexity to a level comparable with English text processing, providing a more efficient foundation for Korean language processing. The method's effectiveness stems not from any special property of Korean phonetics, but from its alignment with basic principles of efficient text representation and processing.

6.3.2. Validity as of Today

Our analysis indicates that the Jamo pair encoding method remains not only valid but increasingly relevant in today's landscape of language model development. The emergence of large-scale multilingual models has intensified the challenges of vocabulary budget allocation, making efficient character encoding schemes more crucial than ever.

The method's utility extends across model scales, though with important considerations at each level. Recent research has established strong correlations between tokenizer performance and downstream task effectiveness (Zouhar et al., 2023; Goldman et al., 2024; Ali et al., 2024). These findings reinforce the continued relevance of our approach for smaller models. However, we must note that very small models present an inherent risk: output instability could compromise accurate character reconstruction from Jamo units.

For large-scale models, maximizing the method's effectiveness requires careful integration with other low-level tokenization strategies (see Section 6.5.3). Alternatively, the development of complementary subcharacter methods for related languages offers another promising direction for continued relevance (see Section 6.5.3).

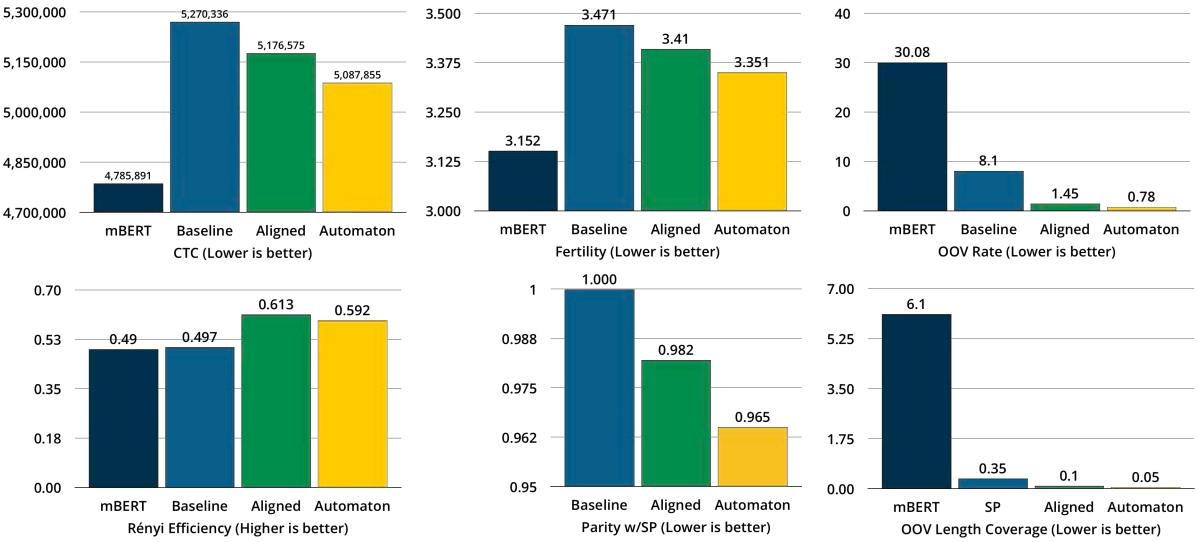


Figure 6.2: Comparison of multilingual BERT and our 2.5K model.

6.3.3. Revisiting with Current Frameworks

Our second method permits evaluation using recently established tokenization quality metrics, though the experiments retain some OOV considerations. This evaluation framework enables direct comparison with current approaches.

The sequence length metrics demonstrate an inverse correlation with allocated vocabulary budget. This relationship allows meaningful comparison between a baseline SentencePiece tokenizer and our proposed method.

To establish broader context, we conducted comparisons with baseline tokenizers from pretrained models containing similar Korean vocabulary sizes. Multilingual BERT, with its 1,568 Korean subwords, provides a comparison point for our 2.5K tokenizers, while XLM's 2,581 Korean subwords align with our 10K models. These comparisons are necessarily approximate, as SentencePiece does not provide direct control over language-specific vocabulary allocation. For this evaluation, we utilized the complete NSMC dataset, as the absence of model training eliminated the need for train/test partitioning.

CTC: The comparison between multilingual BERT and 2.5K models reveals that SentencePiece exhibits lower compression efficiency, with a 6.37-10.13increase compared to BERT. Between SentencePiece 2.5K models, our method achieves modest improvements, demonstrating up to 3.5% lower CTC compared to the baseline SentencePiece implementation.

The XLM to 10K comparison yields more varied results. CTC differences range from an 11.37% decrease in the best case to an 11.62% increase in the worst case. Within SentencePiece 10K

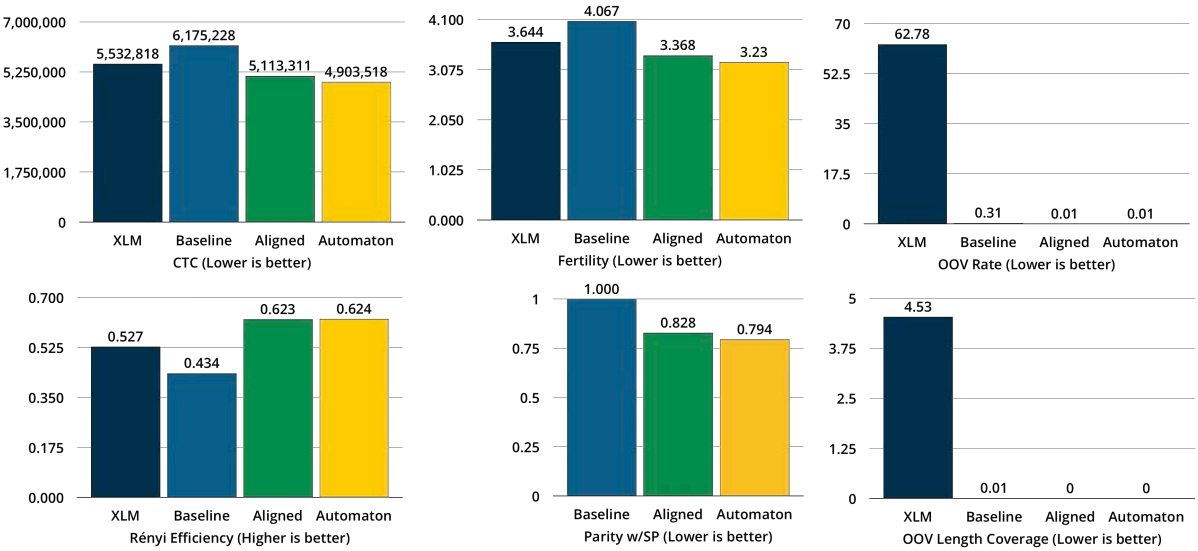


Figure 6.3: Comparison of XLM and our 10K model.

	Vocab	ko-Vocab	CTC	Fertility	Renyi	Parity w/SP
mBERT	119,547	1,568	4,785,891	3.152	0.490	x
SP@2.5K	2,500	1,903	5,270,336	3.471	0.497	x
Aligned@2.5K	2,500	2,160	5,176,575	3.410	0.613	0.982
Automaton@2.5K	2,500	2,131	5,087,855	3.351	0.592	0.965
XLM	200,000	2,581	5,532,818	3.644	0.527	x
SP@10K	10,000	2,789	6,175,228	4.067	0.434	x
Aligned@10K	10,000	2,517	5,113,311	3.368	0.623	0.828
Automaton@10K	1,0000	2,513	4,903,518	3.230	0.624	0.794

Table 6.3: Raw data comparing our method (2.5K/10.K) with recently proposed metrics on NSMC.

models, our method achieves substantial improvements, with the best case showing 20.6% lower CTC compared to baseline. Notably, across both scenarios, baseline SentencePiece consistently demonstrates higher CTC values than the respective pre-trained comparison models.

Fertility: Analysis of 2.5K models indicates increased fertility in SentencePiece models compared to baseline, suggesting reduced tokenization efficiency. In contrast, 10K models implementing our method demonstrate fertility improvements, while baseline SentencePiece shows decreased performance.

Entropy: The 2.5K comparison reveals lower performance in baseline SentencePiece compared to BERT. However, our methods demonstrate increased Rényi efficiency, indicating improved tokenization quality and suggesting potential downstream performance benefits.

Parity: Direct comparison between baseline SentencePiece and our proposed methods shows favorable parity metrics for our approach in both aligned and automaton implementations.

Our analysis indicates that when trained with identical corpora, parameters, and vocabulary

Figure 6.4: Example of tokenization-induced model misbehavior in ChatGPT.

budgets, our method enhances SentencePiece's tokenization properties. However, we observe varying patterns in tokenization quality between 2.5K and 10K implementations, likely influenced by vocabulary budget and character coverage hyperparameters.

6.4. The Significance of Tokenization

The landscape of tokenization has evolved substantially with the emergence of large language models. Contemporary architectures with billions of parameters have introduced sophisticated approaches to handle rare tokens, including byte-level tokenization schemes (Sennrich et al., 2016; Wang et al., 2020). While these advances have addressed certain fundamental challenges in tokenization, particularly around robustness, they have also revealed new complexities in the relationship between tokenization strategies and model performance.

A striking illustration of these complexities emerged through the discovery of the *SolidGold-Magikarp phenomenon* (Figure 6.4) in large language models (Land and Bartolo, 2024). As demonstrated in Figure 6.4, this specific string triggered unexpected model behaviors, revealing how distribution shifts between tokenizer training and model pretraining can lead to undertrained tokens and subsequent model hallucinations. This case exemplifies the broader challenges in tokenization that persist despite recent advances in model architecture and scale.

A quote from Karpathy (2024) highlights the impact of tokenization on model capabilities:

- Why can't LLM spell words? Tokenization.
- Why can't LLM do super simple string processing tasks like reversing a string? Tokenization.
- Why is LLM worse at non-English languages (e.g. Japanese)? Tokenization.

- Why is LLM bad at simple arithmetic? Tokenization.
- Why did GPT-2 have more than necessary trouble coding in Python? Tokenization.
- Why did my LLM abruptly halt when it sees the string "<|endoftext|>"? Tokenization.
- What is this weird warning I get about a "trailing whitespace"? Tokenization.
- Why the LLM break if I ask it about "SolidGoldMagikarp"? Tokenization.
- Why should I prefer to use YAML over JSON with LLMs? Tokenization.
- Why is LLM not actually end-to-end language modeling? Tokenization.
- What is the real root of suffering? Tokenization.

These effects manifest across multiple domains: character-level operations such as string manipulation and spelling, multilingual processing (particularly for languages with different character systems), numerical operations, and code generation. The limitations extend beyond mere technical constraints, fundamentally affecting the model's ability to perform end-to-end language modeling.

The significance of tokenization extends beyond these individual challenges. As language models increasingly adopt multilingual capabilities as a standard feature, the allocation of vocabulary budget across different languages and domains has become a critical consideration in model development. This shift demands more sophisticated approaches to tokenizer training, moving beyond treating it as an auxiliary component to recognizing it as a fundamental element of model architecture.

We propose three key directions for future research in tokenization:

1. Development of scalable training methodologies that better capture global distributional characteristics of training data, particularly addressing the challenges of sampling bias and domain shift.
2. Investigation of adaptive tokenization strategies that can dynamically optimize to reflect underlying probability distributions in the training data.
3. Integration of tokenizer optimization with model pretraining to reduce the impact of distribution shifts and improve handling of rare tokens.

These challenges suggest that while the fundamental problems of tokenization robustness may be well understood, significant opportunities remain for improving how tokenization supports and enhances model capabilities. The path forward requires treating tokenization not as a solved

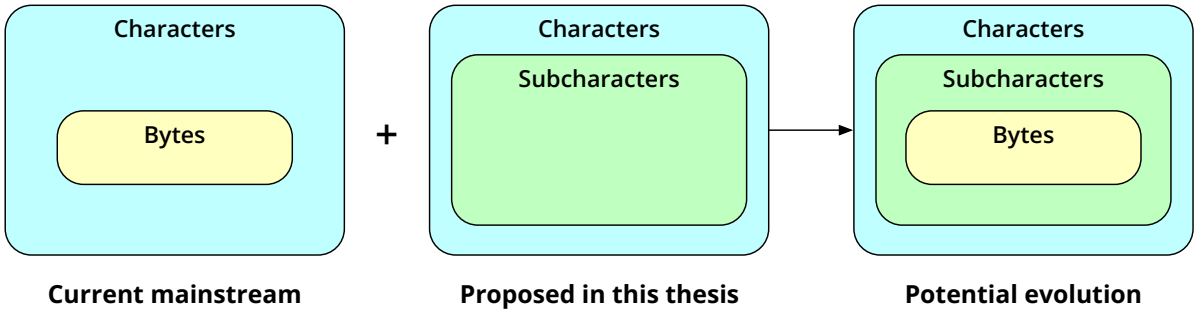


Figure 6.5: Proposed evolution of character-level processing.

preprocessing step, but as an integral component of language model development that warrants continued research attention.

6.5. Future Work

6.5.1. Opportunities for Enhanced Evaluation

When the methods discussed in this thesis were initially proposed, Korean NLP benchmarking was still developing. Many widely-used evaluation tasks had not undergone rigorous peer review by the academic community, yet served as de facto standards for performance assessment.

NSMC, which features prominently in our experimental work, exemplifies these limitations. While we selected it for its challenging tokenization requirements due to prevalent colloquialisms, it remains fundamentally a binary classification task. Recent research has established its performance ceiling at 93.0% accuracy, achieved by HyperCLOVA (39B) using p-tuning, a SFT technique (Liu et al., 2022). Our analysis revealed eight samples with empty text fields[1], raising concerns about additional unidentified, unsolvable instances within the dataset.

The field has since evolved significantly with the introduction of robust evaluation frameworks. The KLUE benchmark (Park et al., 2021) and systematic task curation efforts (Cho et al., 2020) now provide more reliable assessment methodologies. While our current work has not yet been evaluated against these enhanced benchmarks, such validation represents a critical area for future research. A comprehensive assessment using these more rigorous benchmarks would provide valuable insights into the generalizability and effectiveness of our proposed methods.

6.5.2. Subword Architectures for Other Languages

The subcharacter-oriented subword approach we explored for Korean in Chapter 5 holds significant potential for other writing systems with rich character sets and well-defined subcharacter linguistic units in Unicode. The Chinese, Japanese, and Korean (CJK) ideographic block in Unicode presents a particularly promising avenue for extending this research.

Our investigation of Korean demonstrated the value of leveraging subcharacter structures, as detailed in Section 2.5.3 and Section 5.1. Recent research has begun to explore similar approaches in related writing systems. Si et al. (2023) developed a subcharacter architecture for Chinese and validated it using a monolingual BERT model, establishing a foundation for future work in this direction. Wu et al. (2024) have further expanded this concept by examining subcharacter representations from a multimodal perspective.

The Chinese writing system presents unique opportunities and challenges for subcharacter architectures. Its multiple systems of phonetic transcription (e.g., Zhuyin and Pinyin) and various non-phonetic input methods (e.g., Cangjie, Dayi, and Wubi, inter alia) provide multiple potential approaches for implementing subcharacter processing. However, this diversity also expands the search space for optimal methodologies, requiring careful consideration of trade-offs between different approaches.

Beyond CJK ideographs, the Unicode emoji block represents another area where character-level processing faces coverage challenges. Both CJK ideographs and emojis see extensive real-world usage, yet achieving efficient processing while maintaining comprehensive coverage remains an open problem. Future research should address these challenges, particularly given the growing importance of both character sets in modern digital communication.

6.5.3. Integration with Byte-Level BPE

Briefly discussed in Section , Figure 6.5 illustrates a hypothetical future framework for integrating subcharacter-based methods as an intermediate layer between byte-level and Unicode character-level processing for languages with diverse character sets. This architecture addresses a fundamental limitation of pure byte-level approaches: their lack of linguistic context. While byte embeddings offer language-agnostic processing, they require the model's attention mechanism to learn basic character formation rules and language identification that could be more efficiently encoded at the subcharacter level.

[1] Empty text fields were identified at line numbers 46473, 60737, 77667, 84100, 127019, 172377, 173528, and 197281, with conflicting labels.

Current byte-level approaches force models to dedicate computational resources to tasks such as UTF-8 validation and language identification. By incorporating subcharacter-level processing, models can leverage inherent linguistic structure, potentially reducing the computational burden on the attention mechanism while preserving the flexibility of byte-level tokenization.

6.6. Disclosure and Broader Implications

6.6.1. Usage of AI Assistants

This thesis manuscript was revised with assistance from several commercial language models for stylistic refinement, grammatical correction, and fact-checking. The following systems were employed individually, or as an ensemble (in alphabetical order):

- Claude 3.5 Sonnet (Anthropic PBC)
- Github Copilot (GitHub, OpenAI)
- Grammarly (Grammarly Inc)
- NotebookLM (Google LLC)

It is important to note that these systems were used solely for editorial and verification purposes. All original research, analysis, and primary content were developed independently by the author and collaborators.

6.6.2. Impact Assessment: Ethical and Societal

The methodologies presented in this thesis have been internally evaluated for their ethical and societal implications. Our algorithmic approaches maintain neutrality by operating solely on syntactic structures without embedding or propagating demographic or cultural biases. The research methodology deliberately avoided the creation of new datasets or the involvement of human subjects and annotators, thereby minimizing potential ethical concerns regarding data privacy and annotation bias.

The societal impact of this research demonstrates tangible benefits, particularly in environmental sustainability. Our improvements in training efficiency and model reusability directly contribute to reducing computational resource requirements. Based on the amount of compute used for our experimental results, we expect reduction in downstream training compute requirements compared to baseline approaches, translating to proportional decreases in energy consumption for large-scale language model development.

A central feature of our methodology is its accessibility. All experiments were conducted using single-GPU configurations, demonstrating that meaningful contributions to the field are possible without access to extensive computational resources. This approach ensures that our methods remain practical for researchers and developers working with limited computational budgets, particularly in academic environments or smaller organizations.

These advancements in tokenization efficiency and model reusability contribute to the broader initiative within artificial intelligence research to develop more sustainable methodologies. While individual implementations may show incremental improvements, widespread adoption of these techniques could significantly reduce the environmental impact of natural language processing research and deployment.

6.6.3. Compute Costs

The research utilized the following computational environments:

- ABCI 2.0 (rt_G.small): Intel Xeon Gold 6148 (2.4 GHz, 6-cores), NVIDIA Tesla V100 (16GB VRAM), 60GB RAM
- PC1: AMD Ryzen 9 3900XT (3.8 GHz), NVIDIA RTX 3090 (24GB VRAM), 32GB DDR4-3600 RAM
- PC2: Intel Xeon E5-2680 v3 (2.5 GHz), NVIDIA Titan RTX (24GB VRAM), 64GB DDR4-2133 RAM

The Effects and Mitigation of OOV

For our initial methodology's CPT and SFT experiments, we utilized *ABCI 2.0* [2]. Training data and experimental code were accessed via GPFS 3.5 over EDR Infiniband with HCA connectivity. Individual experiments varied in duration, with wall-clock times ranging from 2 to 10 hours (mean: 5.5 hours) per experimental run.

The OOV correlation and recovery NSMC experiments were conducted on *PC1*, with training data and experimental code stored on a Phison E16 NVMe drive (1TB, PCIe 4.0). These experiments maintained consistent computational requirements, with each experimental run completing in approximately 40 minutes using the optimized hyperparameter configurations described in Section 3.4.

Jamo Pair Encoding

The second methodology's experiments were executed on *PC2*. While precise timing metrics were not captured prior to the system's decommissioning, reproduction runs on *PC1* completed within

[2] ABCI 2.0 has been decommissioned as of October 2024.

approximately 18 hours with no GPU acceleration or parallelization.

6.7. Summary

This chapter evaluated the continued relevance of the proposed methods through contemporary frameworks while assessing their applicability to current NLP systems. The analysis covered several areas, beginning with evaluation frameworks for tokenization quality, followed by detailed examination of our methods under the lens of recent developments in the field.

We now turn to the final chapter to synthesize our contributions and their implications for the field. The preceding analysis has demonstrated both the enduring value and limitations of our approaches in the context of modern NLP systems. As we move to our conclusions, we will articulate the contributions of this work.

7

Conclusion

7.1. Summary

This work contributes to the field of Natural Language Processing (NLP) by examining challenges in tokenization for character-diverse languages and the handling of out-of-vocabulary (OOV) tokens. My research investigates approaches for improving the efficiency and robustness of language models, with a particular focus on Korean text processing.

7.1.1. Contributions

This research addresses fundamental challenges in Korean language processing through novel approaches to tokenization and vocabulary management. My contributions demonstrate that current limitations represent solvable engineering problems rather than inherent constraints.

At its most fundamental level, this thesis revolutionizes Korean language processing by solving two critical challenges: eliminating OOV-related performance degradation through innovative surrogate mapping and reducing vocabulary requirements by 90% while maintaining complete coverage through sub-character decomposition.

First, I present a systematic investigation of out-of-vocabulary (OOV) effects in multilingual language models, demonstrating that pre-training corpus imbalances lead to suboptimal tokenizer vocabularies. Through empirical analysis, I found that CJK languages, particularly Korean, receive disproportionately small vocabulary allocations relative to their representation in training corpora, directly impacting downstream task performance through information loss.

Second, I introduce a novel OOV mitigation strategy that enables post-training recovery of model performance without requiring complete retraining. The method creates a secondary vocabulary mapping between OOV and in-vocabulary subwords through several proposed algorithms:

character distance, masked language model prediction, and unseen subword assignment. When combined with continued pre-training, this approach demonstrates significant performance improvements across multiple downstream tasks, achieving meaningful gains even in high-OOV scenarios.

Third, I develop Jamo Pair Encoding, a novel sub-character tokenization method that addresses fundamental limitations in Korean text processing. The method resolves challenges in Unicode-based processing through two distinct algorithmic variants: an aligned approach using a fixed three-character grid, and an automaton method employing a flexible state machine. This method guarantees round-trip consistency between original text and tokenized forms, enabling its use in text generation tasks while dramatically reducing the required character-level vocabulary. The approach maintains complete coverage of the Korean writing system without compromising reconstruction fidelity, making it suitable for both understanding and generation tasks.

Finally, I evaluate these methods through both traditional and newly proposed tokenization quality metrics including corpus token count, fertility, compression, and entropy. This analysis establishes a framework for understanding trade-offs between efficiency and robustness in tokenization optimization.

7.1.2. Conclusion

This research demonstrates that significant improvements in both the robustness and efficiency of Korean language processing are achievable through careful consideration of writing system characteristics and vocabulary allocation strategies. The findings reveal that current limitations in processing character-diverse languages are not fundamental constraints but rather addressable challenges through improved tokenization approaches.

Through the OOV mitigation strategy, I achieved substantial improvements in model robustness across multiple domains. The combination with continued pre-training led to significant reductions in OOV rates across various tasks and domains. These improvements in OOV handling directly translate to enhanced model performance, with statistically significant gains in downstream tasks.

The Jamo Pair Encoding method achieves remarkable reductions in vocabulary requirements while maintaining complete coverage of the Korean writing system. This dramatic improvement in vocabulary efficiency frees up substantial vocabulary capacity that can be allocated to meaningful subwords rather than complete characters. The automaton variant further enhances efficiency by improving surface length efficiency compared to baseline approaches while maintaining the same

vocabulary budget.

The effectiveness of both approaches—OOV mitigation and Jamo Pair Encoding—demonstrates that improvements in robustness and efficiency need not be mutually exclusive. The OOV mitigation strategy proves particularly effective for greedy merging tokenizers like WordPiece, while still offering benefits for non-greedy methods through UNK token diversification. Meanwhile, Jamo Pair Encoding's guaranteed round-trip consistency ensures that efficiency gains do not compromise information preservation.

These findings have significant implications for multilingual language model development. This work has shown that targeted interventions in tokenization strategies can substantially improve model performance without requiring extensive computational resources for retraining. The improvements in both OOV rates and vocabulary requirements establish a new benchmark for efficient processing of character-diverse languages.

Looking ahead, these contributions open several promising directions for future research. The success of the sub-character approach for Korean suggests potential applications to other writing systems, particularly those with composable characters. The vocabulary optimization strategies may inform the development of more efficient multilingual models, while the evaluation metrics provide a framework for assessing future improvements in tokenization methods. Most importantly, this thesis demonstrates that the seemingly competing goals of tokenization robustness and efficiency can be simultaneously improved through principled approaches to writing system analysis and vocabulary management.

8

References

Orevaoghene Ahia, Sachin Kumar, Hila Gonen, Jungo Kasai, David Mortensen, Noah Smith, and Yulia Tsvetkov. 2023. Do all languages cost the same? tokenization in the era of commercial language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9904–9923, Singapore. Association for Computational Linguistics.

Mehdi Ali, Michael Fromm, Klaudia Thellmann, Richard Rutmann, Max Lübbering, Johannes Leveling, Katrin Klug, Jan Ebert, Niclas Doll, Jasper Buschhoff, Charvi Jain, Alexander Weber, Lena Jurkschat, Hammam Abdelwahab, Chelsea John, Pedro Ortiz Suarez, Malte Ostendorff, Samuel Weinbach, Rafet Sifa, Stefan Kesselheim, and Nicolas Flores-Herr. 2024. Tokenizer choice for LLM training: Negligible or crucial? In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3907–3924, Mexico City, Mexico. Association for Computational Linguistics.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4623–4637, Online. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations*.

Loïc Barrault, Magdalena Biesialska, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Matthias Huck, Eric Joanis, Tom Kocmi, Philipp Koehn, Chi-kiu Lo, Nikola Ljubešić, Christof Monz, Makoto Morishita, Masaaki Nagata, Toshiaki Nakazawa, Santanu Pal, Matt Post, and Marcos Zampieri. 2020. Findings of the 2020 conference on machine translation (WMT20). In *Proceedings of the Fifth Conference on Machine Translation*, pages 1–55, Online. Association for Computational Linguistics.

Richard E. Bellman. 1959. *Dynamic Programming*. Rand Corporation Research Study. Princeton University Press.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5:157–166.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Kaj Bostrom and Greg Durrett. 2020. Byte pair encoding is suboptimal for language model pretraining. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4617–4624, Online. Association for Computational Linguistics.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *34th Annual Meeting of the Association for Computational Linguistics*, pages 310–318, Santa Cruz, California, USA. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Won Ik Cho, Seok Min Kim, and Nam Soo Kim. 2019. Investigating an effective character-level embedding in korean sentence classification. *arXiv preprint arXiv:1905.13656*.

Won Ik Cho, Sangwhan Moon, and Youngsook Song. 2020. Open Korean corpora: A practical report. In *Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS)*, pages 85–93, Online. Association for Computational Linguistics.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways.

Alexandra Chronopoulou, Christos Baziotis, and Alexandros Potamianos. 2019. An embarrassingly simple approach for transfer learning from pretrained language models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2089–2095, Minneapolis, Minnesota. Association for Computational Linguistics.

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

Marco Cognetta, Vilém Zouhar, Sangwhan Moon, and Naoaki Okazaki. 2024. Two counterexamples to tokenization and the noiseless channel. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 16897–16906, Torino, Italia. ELRA and ICCL.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual Language Model Pretraining. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. 2022. No Language Left Behind: Scaling Human-Centered Machine Translation. ArXiv:2207.04672 [cs].

Jia Deng, Wei Dong, Richard Socher, Li Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009*, pages 248–255.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Chenchen Ding, Masao Utiyama, and Eiichiro Sumita. 2018. Simplified abugidas. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 491–495, Melbourne, Australia. Association for Computational Linguistics.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2013. Decaf: A deep convolutional activation feature for generic visual recognition. *31st International Conference on Machine Learning, ICML 2014*, 2:988–996.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas

Unterthiner, Mostafa Dehghani, Matthias Minderer, G. Heigold, S. Gelly, Jakob Uszkoreit, and N. Houlsby. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations*.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker‘s guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia. Association for Computational Linguistics.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal*.

Matthias Gallé. 2019. Investigating the effectiveness of BPE: The power of shorter sequences. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1375–1381, Hong Kong, China. Association for Computational Linguistics.

Gerald. Gazdar. 1985. Generalized phrase structure grammar. page 276.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2016. Multilingual language processing from bytes. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1296–1306, San Diego, California. Association for Computational Linguistics.

Ross Girshick. 2015. Fast r-cnn.

Omer Goldman, Avi Caciularu, Matan Eyal, Kris Cao, Idan Szpektor, and Reut Tsarfaty. 2024. Unpacking tokenization: Evaluating text compression and its correlation with model performance. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2274–2286, Bangkok, Thailand. Association for Computational Linguistics.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. The MIT Press.

Edward Gow-Smith, Harish Tayyar Madabushi, Carolina Scarton, and Aline Villavicencio. 2022. Improving tokenisation by alternative treatment of spaces. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11430–11443, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Naman Goyal, Cynthia Gao, Vishrav Chaudhary, Peng-Jen Chen, Guillaume Wenzek, Da Ju, Sanjana Krishnan, Marc’Aurelio Ranzato, Francisco Guzmán, and Angela Fan. 2022. The Flores-101

evaluation benchmark for low-resource and multilingual machine translation. *Transactions of the Association for Computational Linguistics*, 10:522–538.

Ximena Gutierrez-Vasques, Christian Bentz, and Tanja Samardžić. 2023. Languages through the looking glass of BPE compression. *Computational Linguistics*, 49(4):943–1001.

Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary, and Marc'Aurelio Ranzato. 2019. The FLORES evaluation datasets for low-resource machine translation: Nepali–English and Sinhala–English. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6098–6111, Hong Kong, China. Association for Computational Linguistics.

Zellig S. Harris. 1954. Distributional structure. *WORD*, 10:146–162.

Han He, Lei Wu, Xiaokun Yang, Hua Yan, Zhimin Gao, Yi Feng, and George Townsend. 2018. Dual Long Short-Term Memory Networks for Sub-Character Representation Learning. In *Advances in Intelligent Systems and Computing*.

Tatsuya Hiraoka, Hiroyuki Shindo, and Yuji Matsumoto. 2019. Stochastic tokenization with a language model for neural text classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1620–1629, Florence, Italy. Association for Computational Linguistics.

Tatsuya Hiraoka, Sho Takase, Kei Uchiumi, Atsushi Keyaki, and Naoaki Okazaki. 2020. Optimizing word segmentation for downstream task. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1341–1351, Online. Association for Computational Linguistics.

Tatsuya Hiraoka, Sho Takase, Kei Uchiumi, Atsushi Keyaki, and Naoaki Okazaki. 2021. Joint optimization of tokenization and downstream model. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 244–255, Online. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9:1735–1780.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero,

Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. 2022. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, pages 30016–30030, Red Hook, NY, USA. Curran Associates Inc.

Valentin Hofmann, Janet Pierrehumbert, and Hinrich Schütze. 2021. Superbizarre is not superb: Derivational morphology improves BERT's interpretation of complex words. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3594–3608, Online. Association for Computational Linguistics.

Valentin Hofmann, Hinrich Schuetze, and Janet Pierrehumbert. 2022. An embarrassingly simple method to mitigate undesirable properties of pretrained language model tokenizers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 385–393, Dublin, Ireland. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. 2020. Tabtransformer: Tabular data modeling using contextual embeddings.

Cassandra L. Jacobs and Yuval Pinter. 2022. Lost in space marking.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain. Association for Computational Linguistics.

Daniel Jurafsky and James H. Martin. 2024. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*, 3rd edition edition. Online manuscript.

Sekitoshi Kanai, Y. Fujiwara, Yuki Yamanaka, and S. Adachi. 2018. Sigsoftmax: Reanalysis of the softmax bottleneck. *Neural Information Processing Systems*.

Jared Kaplan, Sam McCandlish, Tom Henighan OpenAI, Tom B Brown OpenAI, Benjamin Chess OpenAI, Rewon Child OpenAI, Scott Gray OpenAI, Alec Radford OpenAI, Jeffrey Wu OpenAI, and Dario Amodei OpenAI. 2020. Scaling laws for neural language models.

Andrej Karpathy. 2024. Tokenization, Colab Notebok.

Marzena Karpinska, Bofang Li, Anna Rogers, and Aleksandr Drozd. 2018. Subcharacter information in Japanese embeddings: When is it worth it? In *Proceedings of the Workshop on the Relevance of Linguistic Structure in Neural Architectures for NLP*, pages 28–37, Melbourne, Australia. Association for Computational Linguistics.

SungHo Kim, Juhyeong Park, Yeachan Kim, and SangKeun Lee. 2024. KOMBO: Korean character representations based on the combination rules of subcharacters. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5102–5119, Bangkok, Thailand. Association for Computational Linguistics.

Youngsam Kim and Hyopil Shin. 2013. Romanization-based approach to morphological analysis in Korean SMS text processing. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 145–152, Nagoya, Japan. Asian Federation of Natural Language Processing.

Stav Klein and Reut Tsarfaty. 2020. Getting the ##life out of living: How adequate are word-pieces for modelling complex morphology? In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 204–209, Online. Association for Computational Linguistics.

Prasanth Kolachina, Martin Riedl, and Chris Biemann. 2017. Replacing OOV words for dependency parsing with distributional semantics. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pages 11–19, Gothenburg, Sweden. Association for Computational Linguistics.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 230–237, Barcelona, Spain. Association for Computational Linguistics.

Sadao Kurohashi. 1994. Improvements of japanese morphological analyzer juman. *Proc. of The International Workshop on Sharable Natural Language Resources, 1994*, pages 22–38.

Sander Land and Max Bartolo. 2024. Fishing for magikarp: Automatically detecting under-trained tokens in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11631–11646, Miami, Florida, USA. Association for Computational Linguistics.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278–2323.

Sangah Lee, Hansol Jang, Yunmee Baik, Suzi Park, and Hyopil Shin. 2020. Kr-bert: A small-scale korean-specific language model.

Haizhou Li and Baosheng Yuan. 1998. Chinese word segmentation. In *Proceedings of the 12th Pacific Asia Conference on Language, Information and Computation*, pages 212–217, Singapore. Chinese and Oriental Languages Information Processing Society.

Wei Li, Xinyan Xiao, Yajuan Lyu, and Yuanzhuo Wang. 2018. Improving neural abstractive document summarization with explicit information selection modeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1787–1796, Brussels, Belgium. Association for Computational Linguistics.

Xiaoqing Li, Jiajun Zhang, and Chengqing Zong. 2016. Towards Zero Unknown Word in Neural Machine Translation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*, pages 2852–2858.

Xi Victoria Lin, Todor Mihaylov, Mikel Artetxe, Tianlu Wang, Shuohui Chen, Daniel Simig, Myle Ott, Naman Goyal, Shruti Bhosale, Jingfei Du, Ramakanth Pasunuru, Sam Shleifer, Punit Singh Koura,

Vishrav Chaudhary, Brian O'Horo, Jeff Wang, Luke Zettlemoyer, Zornitsa Kozareva, Mona Diab, Veselin Stoyanov, and Xian Li. 2022. Few-shot learning with multilingual generative language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9019–9052, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, Dublin, Ireland. Association for Computational Linguistics.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China. Association for Computational Linguistics.

Ji Ma, Kuzman Ganchev, and David Weiss. 2018. State-of-the-art Chinese word segmentation with Bi-LSTMs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4902–4908, Brussels, Belgium. Association for Computational Linguistics.

Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. 2018. Exploring the limits of weakly supervised pretraining. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11206 LNCS:185–201.

Chris Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*.

Mitchell P.. Marcus. 1980. A theory of syntactic recognition for natural language. page 335.

A. McCallum and K. Nigam. 1998. A comparison of event models for naive bayes text classification. *AAAI Conference on Artificial Intelligence*.

Stephen Merity, Caiming Xiong, James Bradbury, and R. Socher. 2016. Pointer sentinel mixture models. *International Conference on Learning Representations*.

Tomas Mikolov, I. Sutskever, Anoop Deoras, H. Le, Stefan Kombrink, and J. ⊠ernocký. 2011. Subword language modeling with neural networks.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26.

Sangwhan Moon, Won Ik Cho, Hye Joo Han, Naoaki Okazaki, and Nam Soo Kim. 2022. OpenKorPOS: Democratizing Korean tokenization with voting-based open corpus annotation. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 4975–4983, Marseille, France. European Language Resources Association.

Sangwhan Moon and Naoaki Okazaki. 2020a. Jamo pair encoding: Subcharacter representation-based extreme Korean vocabulary compression for efficient subword tokenization. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 3490–3497, Marseille, France. European Language Resources Association.

Sangwhan Moon and Naoaki Okazaki. 2020b. PatchBERT: Just-in-time, out-of-vocabulary patching. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7846–7852, Online. Association for Computational Linguistics.

Viet Nguyen, Julian Brooke, and Timothy Baldwin. 2017. Sub-character neural language modelling in Japanese. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pages 148–153, Copenhagen, Denmark. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France. European Language Resources Association.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22:1345–1359.

Jangwon Park. 2024. Evaluating NSMC with KoBERT. Original-date: 2019-12-21T16:40:09Z.

Sungjoon Park, Jeongmin Byun, Sion Baek, Yongseok Cho, and Alice Oh. 2018. Subword-level word vector representations for Korean. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2429–2438, Melbourne, Australia. Association for Computational Linguistics.

Sungjoon Park, Jihyung Moon, Sungdong Kim, Won Ik Cho, Ji Yoon Han, Jangwon Park, Chisung Song, Junseong Kim, Youngsook Song, Taehwan Oh, Joohong Lee, Juhyun Oh, Sungwon Lyu, Younghoon Jeong, Inkwon Lee, Sangwoo Seo, Dongjun Lee, Hyunwoo Kim, Myeonghwa Lee, Seongbo Jang, Seungwon Do, Sunkyoung Kim, Kyungtae Lim, Jongwon Lee, Kyumin Park, Jamin Shin, Seonghyun Kim, Lucy Park, Alice Oh, Jung-Woo Ha (NAVER AI Lab), and Kyunghyun Cho. 2021. KLUE: Korean Language Understanding Evaluation. *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 1.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 562–568, Geneva, Switzerland. COLING.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Aleksandar Petrov, Emanuele La Malfa, Philip H.S. Torr, and Adel Bibi. 2023. Language model tokenizers introduce unfairness between languages. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, pages 36963–36990, Red Hook, NY, USA. Curran Associates Inc.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. BPE-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892, Online. Association for Computational Linguistics.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. Technical report.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Frank Rosenblatt. 1961. Principles of neurodynamics. perceptrons and theory of brain mechanisms.

Sebastian Ruder, John G Breslin, and Parsa Ghaffari. 2019. *Neural Transfer Learning for Natural Language Processing*. Ph.D. thesis.

Phillip Rust, Jonas Pfeiffer, Ivan Vulic, Sebastian Ruder, and Iryna Gurevych. 2021. How good is your tokenizer? on the monolingual performance of multilingual language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3118–3135, Online. Association for Computational Linguistics.

Jonne Saleva and Constantine Lignos. 2023. What changes when you randomly choose BPE merge operations? not much. In *Proceedings of the Fourth Workshop on Insights from Negative Results in NLP*, pages 59–66, Dubrovnik, Croatia. Association for Computational Linguistics.

Lee Sang-ho. 1995. Kts : A korean part-of-speech tagging system with handling unknown words. *The Acoustical Society Of Korea Workshop*, pages 195–199.

Craig W Schmidt, Varshini Reddy, Haoran Zhang, Alec Alameddine, Omri Uzan, Yuval Pinter, and Chris Tanner. 2024. Tokenization is more than compression. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 678–702, Miami, Florida, USA. Association for Computational Linguistics.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pages 5149–5152.

Yuuki Sekizawa, Tomoyuki Kajiwara, and Mamoru Komachi. 2017. Improving Japanese-to-English neural machine translation by paraphrasing the target language. In *Proceedings of the 4th Workshop on Asian Translation (WAT2017)*, pages 64–69, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

C E Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:623–656.

Wu-Guang Shi. 2005. Chinese word segmentation based on direct maximum entropy model. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*.

Xinlei Shi, Junjie Zhai, Xudong Yang, Zehua Xie, and Chao Liu. 2015. Radical embedding: Delving deeper to Chinese radicals. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 594–598, Beijing, China. Association for Computational Linguistics.

Chenglei Si, Zhengyan Zhang, Yingfa Chen, Fanchao Qi, Xiaozhi Wang, Zhiyuan Liu, Yasheng Wang, Qun Liu, and Maosong Sun. 2023. Sub-character tokenization for Chinese pretrained language models. *Transactions of the Association for Computational Linguistics*, 11:469–487.

K. Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*.

Xinying Song, Alex Salcianu, Yang Song, Dave Dopson, and Denny Zhou. 2021. Fast WordPiece tokenization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2089–2103, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Sanja Štajner and Maja Popovic. 2016. Can text simplification help machine translation? In *Proceedings of the 19th Annual Conference of the European Association for Machine Translation*, pages 230–242.

Karl Stratos. 2017. A sub-character architecture for Korean language processing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 721–726, Copenhagen, Denmark. Association for Computational Linguistics.

Yaming Sun, Lei Lin, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2014. Radical-enhanced chinese character embedding. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*.

Yu Suzuki. 2019. Filtering method for twitter streaming data using human-in-the-loop machine learning. *Journal of Information Processing*, 27:404–410.

Dan Svenstrup, Jonas Meinertz Hansen, and Ole Winther. 2017. Hash embeddings for efficient word representations. *Advances in Neural Information Processing Systems*, 30.

Sho Takase, Tatsuya Hiraoka, and Naoaki Okazaki. 2022. Single model ensemble for subword regularized models in low-resource machine translation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2536–2541, Dublin, Ireland. Association for Computational Linguistics.

Arseny Tolmachev, Daisuke Kawahara, and Sadao Kurohashi. 2018. Juman++: A morphological analysis toolkit for scriptio continua. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 54–59, Brussels, Belgium. Association for Computational Linguistics.

Huihsin Tseng and Keh-Jiann Chen. 2002. Design of Chinese morphological analyzer. In *COLING-02: The First SIGHAN Workshop on Chinese Language Processing*.

Unicode Consortium. 2024. Uax #38: Unicode han database (unihan).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, ☒ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

David Vilar and Marcello Federico. 2021. A statistical extension of byte-pair encoding. In *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*, pages 263–275, Bangkok, Thailand (online). Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Changhan Wang, Kyunghyun Cho, and Jiatao Gu. 2020. Neural machine translation with byte-level subwords. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:9154–9160.

Hai Wang, Dian Yu, Kai Sun, Jianshu Chen, and Dong Yu. 2019. Improving pre-trained multilingual model with vocabulary expansion. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 316–327, Hong Kong, China. Association for Computational Linguistics.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature hashing for large scale multitask learning. *ACM International Conference Proceeding Series*, 382.

Joseph Weizenbaum. 1966. Eliza - a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9:36–45.

Paul J. Werbos. 1990. Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, 78:1550–1560.

T. Winograd. 1983. *Language as a Cognitive Process: Syntax*. Computer Science Series. Addison Wesley Publishing Company.

Terry Winograd. 1971. Procedures as a representation for data in a computer program for understanding natural language.

Terry Winograd. 1972. Understanding natural language. *Cognitive Psychology*, 3:1–191.

Xiaofeng Wu, Karl Stratos, and Wei Xu. 2024. The Impact of Visual Information in Chinese Characters: Evaluating Large Models' Ability to Recognize and Utilize Radicals. ArXiv:2410.09013 [cs].

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. In *International Journal of Computational Linguistics & Chinese Language Processing, Volume 8, Number 1, February 2003: Special Issue on Word Formation and Chinese Language Processing*, pages 29–48.

Zhilin Yang, Zihang Dai, R. Salakhutdinov, and William W. Cohen. 2017. Breaking the softmax bottleneck: A high-rank rnn language model. *International Conference on Learning Representations*.

Shaked Yehezkel and Yuval Pinter. 2023. Incorporating context into subword vocabularies. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 623–635, Dubrovnik, Croatia. Association for Computational Linguistics.

Rongchao Yin, Quan Wang, Peng Li, Rui Li, and Bin Wang. 2016. Multi-granularity Chinese word embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 981–986, Austin, Texas. Association for Computational Linguistics.

Yang Zhao, Jiajun Zhang, Zhongjun He, Chengqing Zong, and Hua Wu. 2018. Addressing troublesome words in neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 391–400, Brussels, Belgium. Association for Computational Linguistics.

G.K. Zipf. 1949. *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*. Addison-Wesley Press.

Doug Zongker. 2006. Chicken chicken chicken: Chicken chicken. *Annals of Improbable Research*, 12(5):16–21.

Vilém Zouhar, Clara Meister, Juan Gastaldi, Li Du, Mrinmaya Sachan, and Ryan Cotterell. 2023. Tokenization and the noiseless channel. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5184–5207, Toronto, Canada. Association for Computational Linguistics.

Judit Ács. 2019. Exploring BERT's Vocabulary.