Doctoral Dissertation

Conversion of Noisy or Long Sentences Into Readable Sentences

ITSUMI TAKAHASHI

MARCH 2020

Department of Information Processing
Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Itsumi Takahashi

March 2020

Thesis Committee:

Professor Yuji Matsumoto (Supervisor)
Professor Satoshi Nakamura (Co-supervisor)
Associate Professor Masashi Shimbo (Co-supervisor)
Assistant Professor Hiroyuki Shindo (Co-supervisor)

ACKNOWLEDGMENT

主指導教官の松本裕治教授に感謝いたします．会社の業務と博士研究の両立が難しい時期もありましたが，先生には終始暖かくご指導していただきました．また，お忙しい中研究に関する様々なご助言をいただきました．深く感謝いたします．

お忙しい中，副指導教官になっていただきました中村哲教授，新保仁准教授，進藤裕之助教に感謝いたします．先生方からは本論文に関し様々なご助言をいただきました．ありがとうございました．また，秘書の北川祐子さんには遠隔地で研究や論文審査を進めるにあたり様々なご支援をいただきました．ありがとうございました．

学位取得の機会を与えていただいたNTTメディアインテリジェンス研究所の方々に感謝いたします．私が入社した当初からお世話になった松尾元グループリーダ，浅野主幹研究員，今村元主任研究員，齋藤主幹研究員に感謝いたします．浅野主幹研究員には，特に社内での開発業務で現在まで多大なご助力をいただき，研究した技術の製品化に関してあらゆる面で支えていただきました．現フューチャーアーキテクトの貞光氏には，入社時のテーマ研究を指導していただき大変お世話になりました．当時のテーマ研究からの一連の研究が今回の博士論文の骨子となっており，貞光氏の指導によって博士論文執筆に至ることができました．また，現在東北大学におられる鈴木准教授にはテーマ研究のアドバイザをしていただき，その後もNTTコミュニケーション科学研究所に短期滞在させていただいたり論文執筆の指導をいただいたりするなど，お忙しい中多くのご指導を賜りました．感謝いたします．富田グループリーダ，西田特別研究員，大塚研究員，西田研究員に感謝いたします．西田特別研究員には論文執筆や研究テーマの議論など日々熱心な研究指導をいただくとともに，研究に集中できる環境を整えていただきました．現マイクロソフトデベロップメントの泉氏，現ソニーの宮崎氏にはNTTの先輩としてさまざまなことを教えていただくとともに，プライベートな相談などもさせてい

ただきました．ありがとうございました．

私の家族・親族に感謝します．私が研究職を志しこれまで自身の興味に向き合い続けることができたのは，家族の理解と支援によるところが大きいと考えます．

最後に，日々の生活を支えてくれている夫に感謝いたします．夫は不規則になりがちな私の生活を寛大な心で受け入れ，支えてくれました．一足先に社会人博士を取得し最大限のサポートをしてくれた夫に，最大の感謝を捧げます．

CONVERSION OF NOISY OR LONG SENTENCES INTO READABLE SENTENCES

崩れたテキストや長いテキストの読みやすいテキストへの変換

Abstract

Itsumi Takahashi

In this thesis, we propose methods of converting noisy or long sentences into readable sentences. Noisy texts such as social media and long documents are hard to read for humans and machines. Converting these texts to readable texts is an important task to analyze sentences accurately and to facilitate human communication. There are three main tasks we tackled in this thesis.

The first task is a joint estimation of word-level normalization and morphological analysis. In this study, we proposed a novel method of analyzing non-standard tokens. We achieved higher accuracy and recall for word segmentation, POS tagging, and normalization than those of conventional systems. Moreover, we proposed a novel method for automatically extracting pairs of a variant word and its standard form from the unannotated text. We incorporated the acquired variant-normalization pairs into Japanese morphological analysis, and they improved the recall of normalization.

The second task is a sentence-level normalization. We proposed simple but effective methods of data augmentation for encoder-decoder based neural normalization models. The combination of proposed methods outperformed baselines.

The third task is length-controllable summarization. We propose a new prototype-guided length-controllable abstractive summarization model to generate comprehensive texts from a long document. Our model first extracts a prototype text and creates a summary by jointly encoding and copying words from both the prototype text and source text. Experi-

ments with the CNN/Daily Mail dataset and the NEWSROOM dataset show that our model outperformed previous models in standard and length-controlled settings.

The above studies are a proposal of methods to convert noisy sentences and long sentences into readable sentences at a word level, sentence level, and document level.

Keywords: morphological analysis, normalization, summarization, encoder-decoder model

CONVERSION OF NOISY OR LONG SENTENCES INTO READABLE SENTENCES

崩れたテキストや長いテキストの読みやすいテキストへの変換

Abstract

Itsumi Takahashi

本研究では，文章の単語的な表記揺れを統制するための変換技術や，文章全体をシンプルで短い文に変換する技術など，崩れた文章や長い文章をわかりやすい文に変換する技術を提案する．本研究の成果は大きく3つある．

成果1は，Twitterのような辞書に存在しない崩れた語が多く含まれる口語体の文章について，単語レベルの正規化と形態素解析を同時に行う技術である．従来の形態素解析技術では，辞書にない単語が入力されると解析が失敗し，後段の処理に意味の異なる形態素が渡されてしまうことがあった．本研究では，辞書に存在しない崩れた口語を辞書に存在する正規の語に紐付ける正規化と形態素解析を同時に行うことで，形態素解析の精度向上と正規化による読みやすい文への変換の両方を達成した．

成果2はニューラルネットワークを用いて，方言を標準語に文レベルで変換する技術である．ニューラルネットを用いた文変換は，変換元のテキストと変換先のテキストペアの学習データが大量に存在する場合には高い精度を達成できることが知られている．ただし，方言から標準語への変換タスクにおいては学習データが少なく，大量のデータを人手で作成することも高コストであり現実的ではない．本研究では，少量の人手データから文字レベルの変換パタンを抽出し，大量のラベルなしデータに変換パタンを適用することによって疑似的なペアデータを大量に生成する手法を提案した．提案手法により，従来型の統計的機械翻訳を用いた手法に比べ複数方言ドメインでの精度向上を達成した．

成果3は，長い文書を短く読みやすい要約文へと変換する技術である．現在最も精度

が良い要約システムにおいては，要約元となるテキストに対して一つの要約テキストしか出力することができない．提案手法は，単語レベルの抽出型要約と生成型要約を効果的に組み合わせることによって，生成型要約の精度向上と長さの制御性の双方を達成した．

上記の研究は，単語レベル，文レベル，文書レベルのそれぞれにおいて崩れた文や長い文章を読みやすい文に変換する技術の提案である．

キーワード: 形態素解析, 正規化, 要約, エンコーダ・デコーダモデル

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Motivation

There are many texts that are difficult to read for humans, such as noisy texts like social media and long texts. For example, Social media texts are often written in a non-standard style and include many lexical variants such as insertions, phonetic substitutions, abbreviations that mimic spoken language. By converting such a variety of non-standard tokens into standard tokens, we can handle these texts more easily.

Moreover, it takes a long time to read long texts. We can save time by converting these texts into shorter and readable texts. Such conversions are essential to perform correct analysis for machines and facilitate human communications.

1.2 Task Settings

In this thesis, we tackled mainly three tasks: First two are related to converting noisy sentences to standard sentences and third one is related to converting long documents to short sentences.

Task 1 (Joint estimation of word-level normalization and morphological analysis)**.** *In this*

task, an input is a sentence, and outputs are word segmentations, POS-taggings, and standard forms of each word.

In Japanese, there is no space between words, and it is necessary to perform morphological analysis and normalization simultaneously.

Task 2 (Sentence-level normalization)**.** *In this task, an input is a sentence, and an output is a normalized sentence.*

In this task, we used a character-level encoder-decoder model and performed sentence-level normalization. Attention-based encoder-decoder models are widely used for generating natural languages. Unlike task 1, the encoder-decoder model integrates a language model and a character-level conversion model. This model can perform the sentence-level conversion in an end-to-end manner.

Task 3 (Converting long document to length-controllable summary texts)**.** *In this task, inputs are a long document and a desired number of words in summary, and an output is a summary text.*

Recent state-of-the-art abstractive summarization models based on encoder-decoder models generate only one summary per source text. However, controllable summarization, especially of the length, is an important aspect for practical applications. By generating length-controllable summaries, it is possible to generate readable summaries to meet user requirements.

1.3 Thesis outline

The rest of this thesis is organized as follows: As preliminaries, we will provide a brief introduction of Japanese morphological analysis and normalization tasks and background. We also introduce a sequence to sequence models. These techniques are leveraged in the

following chapters. In Chapter 3 and Chapter 4, we present a supervised and unsupervised model of normalization and Japanese morphological analysis for task 1. In Chapter 5, we present a sentence-level normalization model using character-based encoder-decoder model for task 2. In Chapter 6, we present a novel length-controllable summarization model for task 3. We summarize the thesis and set forth further directions in Chapter 7.

Chapter 2

Preliminaries

2.1 Japanese Morphological Analysis and Normalization

Many approaches to joint word segmentation and POS tagging including Japanese Morphological analysis can be interpreted as re-ranking while using a word lattice (Kaji and Kitsuregawa, 2013). There are two points to consider in the analysis procedure: how to generate the word lattice and how to formulate the cost of each path. In Japanese morphological analysis, the dictionary-based approach has been widely used to generate the word lattice (Kudo, Yamamoto, and Matsumoto, 2004, Kurohashi et al., 1994). In a traditional approach, an optimal path is sought by using the sum of the two types of costs for the path: the cost for a candidate word that reflects the word's occurrence probability, and the cost for a pair of adjacent POS that reflects the probability of an adjacent occurrence of the pair (Kudo, Yamamoto, and Matsumoto, 2004, Kurohashi et al., 1994). A greater cost means less probability. The Viterbi algorithm is usually used for finding the optimal path.

Several studies have been conducted on Japanese morphological analysis in the normalized form. The approach proposed by Sasano, Kurohashi, and Okumura (2013) aims to develop heuristics to flexibly search by using a simple, manually created derivational rule. Their system generates normalized character sequence based on the derivational rule, and

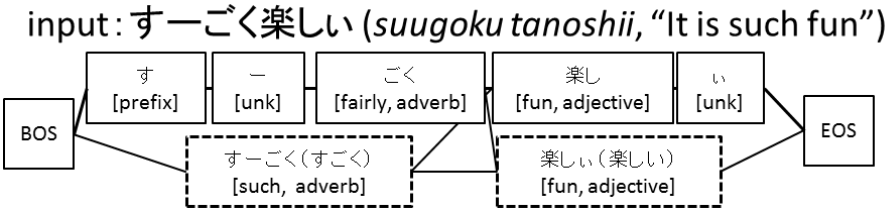

Figure 2.1 Example of Japanese morphological analysis and normalization

adding new nodes that are generated from normalized character sequence when generating the word lattice using dictionary lookup. Figure 4.1 presents an example of this approach. If the non-standard written sentence すーごく楽しい (*suugoku tanoshii*, "It is such fun") is input, the traditional dictionary-based system generates Nodes that are described using solid lines, as shown in Fig. 4.1. Since "すーごく" (*suugoku*, "such") and "楽しい" (*tanoshii*, "fun") are OOVs, the traditional system cannot generate the correct word segments or POS tags. However, their system generates additional nodes for the OOVs, shown as broken line rectangles in Fig. 4.1. In this case, derivational rules that substitute "ー" with "null" and "い" (*i*) with "い" (*i*) are used and the system can generate the standard forms "すごく" (*sugoku*, "such") and "楽しい" (*tanoshii*, "fun") and their POS tags. If we can generate sufficiently appropriate rules, these approaches seem to be effective. However, there are many types of derivational patterns in SNS text and it is difficult to cover all of them by hand. Moreover, it becomes a serious problem how to set the path cost for appropriately re-ranking the word lattice when the number of candidates increases. In Chapter 3 and Chapter 4, we tackled to the problem that automatically extend derivational patterns and incorporated them into the word lattice.

2.2 Neural Sequence to Sequence Models

Neural sequence to sequence models such as an encoder-decoder model is widely used for many language generation tasks such as machine translation and summarization. We use an encoder-decoder model in Chapter 5 and Chapter 6. There are several variations on the

encoder-decoder model. We describe these variations in the following subsections.

2.2.1 Attention based Encoder-Decoder model

The attention-based neural encoder-decoder model was proposed by Bahdanau, Cho, and Bengio (2015). It is an extension model of a simple encoder-decoder model (Cho et al., 2014). Instead of converting the input context vector into a fixed length vector, the attention-based encoder-decoder model compute different context vectors depending on the decoder state. By introducing attention mechanism, long sentences can be converted with high accuracy.

Let $\boldsymbol{s} = (s_1, s_2, \ldots, s_n)$ be an input sentence. Similarly, let $\boldsymbol{t} = (t_1, t_2, \ldots, t_m)$ be an output sentence. The notations n and m are the total lengths of the characters in $\boldsymbol{s}$ and $\boldsymbol{t}$, respectively.

Then, the generation probability of $\boldsymbol{t}$ given input sentence $\boldsymbol{s}$ can be written as

$$p(\boldsymbol{t}|\boldsymbol{s}, \theta) = \prod_{j=1}^{m} p(t_j|t_{<j}, \boldsymbol{s}), \tag{2.1}$$

where θ represents a set of all model parameters in the encoder-decoder model, which are determined by the parameter-estimation process of a standard softmax cross-entropy loss minimization using training data. Therefore, given θ and $\boldsymbol{s}$, output sequence is generated by finding $\boldsymbol{t}$ with maximum probability:

$$\hat{\boldsymbol{t}} = \text{argmax}_{\boldsymbol{t}}\{p(\boldsymbol{t}|\boldsymbol{s}, \theta)\}, \tag{2.2}$$

where $\hat{\boldsymbol{t}}$ represents the solution.

More specifically, $p(t_j|t_{<j}, \boldsymbol{s})$ can be written as

$$p(t_j|t_{<j}, \boldsymbol{s}) = p(t_j|t_{<j}, \boldsymbol{u}_j, \boldsymbol{c}_j), \tag{2.3}$$

Where $\boldsymbol{u}_j$ is the state of a decoder at time step j and $\boldsymbol{c}_j$ is the context vector at time step j. Context vector $\boldsymbol{c}_j$ is calculated as follows:

$$e_{ji} = S(\boldsymbol{h}_i, \boldsymbol{s}_{j-1}) \tag{2.4}$$

$$\alpha_{ji} = \frac{\exp(e_{ji})}{\sum_{k=1}^{n} \exp(e_{jk})} \tag{2.5}$$

$$\boldsymbol{c}_j = \sum_{i=1}^{n} \alpha_{ji} \boldsymbol{h}_i \tag{2.6}$$

In Chapter 5, we applied this model to character sequences.

2.2.2 Pointer-Generator based Encoder-Decoder Model

See, P. J. Liu, and Manning (2017) proposed a pointer-generator model that directly copies words from the source text via pointing (Vinyals, Fortunato, and Jaitly, 2015). Use of a pointer network improves accuracy and handling of OOV words, and many studies especially for summarization tasks have used this model as a basis. We also used this model as a basis in Chapter 6.

2.2.3 Introducing Controllability to An Encoder-Decoder Model

There are another lines of extension of the encoder-decoder model. Some researchers have investigated how to introduce controllability to the encoder-decoder model. Kikuchi et al. (2016) were the first to propose using length embedding for length-controlled abstractive summarization. Fan, Grangier, and Auli (2018) also introduced special embeddings at the beginning of the decoder module for control the summary length, summary style and summary focus. Takase and Okazaki (2019) introduced positional encoding that represents the remaining length at each decoder step of Transformer-based encoder-decoder model. These previous models use simple special embeddings for controlling the length or content in the decoding module. We introduced a new idea that using an extractive module instead of

special embeddings for controlling the summary length.

Chapter 3

Morphological Analysis for Japanese Noisy Text Based on Character-level and Word-level Normalization

3.1 Introduction

Social media texts attract a lot of attention in the fields of information extraction and text mining. Although texts of this type contain a lot of information, such as one's reputation or emotions, they often contain non-standard tokens (lexical variants) that are considered out-of-Vocabulary (OOV) terms. We define an OOV as a word that does not exist in the dictionary. Texts in micro-blogging services such as Twitter are particularly apt to contain words written in a non-standard style, e.g., by lengthening them ("goooood" for "good") or abbreviating them ("thinkin' " for "thinking"). This is also seen in the Japanese language, which has standard word forms and variants of them that are often used in social media texts. To take one word as an example, the standard form is おいしい (*oishii*, "It is delicious") and its variants include おいしいいいいい(*oishiiiii*), おいしぃ(*oishii*), and おいしー(*oishii*), where the underlined characters are the differences from the standard form. Such non-

standard tokens often degrade the accuracy of existing language processing systems, which are trained using a clean corpus.

Almost all text normalization tasks for languages other than Japanese (e.g., English), aim to replace the non-standard tokens that are explicitly segmented using the context-appropriate standard words (Han, Cook, and Baldwin, 2012; Han and Baldwin, 2011; Hassan and Menezes, 2013; Chen Li and Yang Liu, 2012; F. Liu, Weng, and X. Jiang, 2012; F. Liu, Weng, B. Wang, et al., 2011; Pennell and Yang Liu, 2011; Cook and Stevenson, 2009; Aw et al., 2006). On the other hand, the problem is more complicated in Japanese morphological analysis because Japanese words are not segmented by explicit delimiters. In traditional Japanese morphological analysis, word segmentation and part-of-speech (POS) tagging are simultaneously estimated. Therefore, we have to simultaneously analyze normalization, word segmentation, and POS tagging to estimate the normalized form using the context information. For example, the input パンケーキぉぃしーぃ(*pan-keiki oishiiii*, "This pancake tastes good") written in the standard form is パンケーキおいしい (*pan-keiki oishii*). The result obtained with the conventional Japanese morphological analyzer MeCab (T. Kudo, 2005) for this input is **パンケーキ(pancake, noun)**/**ぉぃし(unk)**/**ー(unk)**/**ぃ(unk)**/, where slashes indicate the word segmentations and "unk" means an unknown word. As this result shows, Japanese morphological analyzers often fail to correctly estimate the word segmentation if there are unknown words, so the pipeline method (e.g., first estimating the word segmentations and then estimating the normalization forms) is unsuitable.

Moreover, Japanese has several writing scripts, the main ones being Kanji, Hiragana, and Katakana. Each word has its own formal written script (e.g., 教科書 (*kyoukasyo*, "textbook") as formally written in Kanji), but in noisy text, there are many words that are intentionally written in a different script (e.g., きょうかしょ (*kyoukasyo*, "textbook") is the Hiragana form of 教科書). These tokens written in different script also degrade the performance of existing systems because dictionaries basically include only the standard script. Unlike the character-level variation we described above, this type of variation occurs on a word–level one.

Therefore, there are both character-level and word-level non-standard tokens in Japanese informal written text. Several normalization approaches have been applied to Japanese text. Sasano, Kurohashi, and Okumura (2013) and Oka et al. (2011) introduced simple character level derivational rules for Japanese morphological analysis that are used to normalize specific patterns of non-standard tokens, such as for word lengthening and lower-case substitution. Although these approaches handle Japanese noisy text fairly effectively, they can handle only limited kinds of non-standard tokens.

We propose a novel method of normalization in this study that can handle both character- and word-level lexical variations in one model. Since it automatically extracts character-level transformation patterns in character-level normalization, it can handle many types of character-level transformations. It uses two steps (character- and word-level) to generate normalization candidates, and then formulates a cost function of the word sequences as a discriminative model. The contributions this research makes can be summarized by citing three points. First, the proposed system can analyze a wider variety of non-standard token patterns than the conventional system by using our two-step normalization candidate generation algorithms. Second, it can largely improve the accuracy of Japanese morphological analysis for non-standard written text by simultaneously performing the normalization and morphological analyses. Third, it can automatically extract character alignments and in so doing reduces the cost of manually creating many types of transformation patterns. The rest of this paper is organized as follows. Section 2 describes the background to our research, including Japanese traditional morphological analysis, related work, and data collection methods. Section 3 introduces the proposed approach, which includes lattice generation and formulation, as a discriminative model. Section 4 discusses experiments we performed and our analyses of the experimental results. Section 5 concludes the paper with a brief summary and a mention of future work.

3.2 Background

3.2.1 Related Work

Several studies have been conducted on Japanese morphological analysis in the normalized form. The approach proposed by Sasano, Kurohashi, and Okumura (2013) aims to develop heuristics to flexibly search by using a simple, manually created derivational rule. Our approach is also based on the dictionary-based approach, however, our approach is significantly dissimilar from their approach in two ways. First, we automatically generate derivational patterns (we call them transformation tables) based on the character-level alignment between non-standard tokens and their standard forms. Compared to generating the rules by hand, our approach can generate broad coverage rules. Second, we use discriminative methods to formulate a cost function. W. Jiang, Mi, and Q. Liu (2008) and Kaji and Kitsuregawa (2013) introduce several features to appropriately re-rank the added nodes. This enables our system to perform well even when the number of candidates increases.

On the other hand, several studies have applied a statistical approach. For example, Sasaki et al. (2013) proposed a character-level sequential labeling method for normalization. However, it handles only one-to-one character transformations and does not take the word-level context into account. The proposed method can handle many-to-many character transformations and takes word-level context into account, so the scope for handling non-standard tokens is different. Many studies have been done on text normalization for English; for example Han and Baldwin (2011) classifies whether or not OOVs are non-standard tokens and estimates standard forms on the basis of contextual, string, and phonetic similarities. In these studies it was assumed that clear word segmentations existed. However, since Japanese is an unsegmented language the normalization problem needs to be treated as a joint normalization, word segmentation, and POS tagging problem.

In the field of speech recognition, similar techniques have been developed, such as conversion from phoneme to word using WFST (Mohri, Pereira, and Riley, 2002) and many-to-many

type	non-standard form	standard form
(1) Insertion	ありがとーーう (*arigatoou*)	ありがとう (*arigatou*, "Thank you")
(2) Deletion	さむ_ (*samu*)	さむい (*samui*, "cold")
(3) Substitution with phonetic variation	かわええ (*kawaee*)	かわいい (*kawaii*, "cute")
(4) Substitution with lowercases and uppercases	ぁりがとぅ (*arigatou*)	ありがとう (*arigatou*, "Thank you")
(5) Hiragana substitution	あいでぃー (*aidei*)	ID (*aidei*, "identification card")
(6) Katakana substitution	アリガトウ (*arigatou*)	ありがとう (*arigatou*, "Thank you")
(7) Any combination of (1) to (6)	かぅんたー (*kaunta*)	カウンター (*kaunta*, "counter")
	あっつぃ (*attsui*)	あつい (*atsui*, "hot")

Table 3.1 Types of non-standard tokens and examples of annotated data

character alignment (Bisani and Ney, 2008; Sittichai, Grzegorz, and Tarek, 2007). Although the method of lattice extension is based on the idea of these existing researches, our study introduces these method for morphological analysis and normalization task for the first time. In addition, in the training of decoder, we introduced several specific features for this task. In the Japanese morphological analysis, many words can be analyzed by existing dictionary words. Therefore, it is important to normalize the non-standard words without degrading the overall accuracy. We customized the features that suppresses excessive normalization and the overall accuracy degradation was prevented.

3.2.2 Data Collection and Analysis of Non-standard Tokens

In previous studies (Hassan and Menezes, 2013; Ling et al., 2013; F. Liu, Weng, B. Wang, et al., 2011), the researchers proposed unsupervised ways to extract non-standard tokens and their standard forms. For Japanese text, however, it is very difficult to extract word pairs in an unsupervised way because there is no clear word segmentation. To address this problem we first extracted non-standard tokens from Twitter text and blog text and manually annotated their standard (dictionary) forms. In total, we annotated 4808 tweets and 8023 blog text sentences. Table 1 lists the types of non-standard tokens that we targeted in this study and examples of the annotated data. Types (1), (2), (3) and (4) are similar to English transform patterns. Types (5) and (6) are distinctive patterns in Japanese. As previously

mentioned Japanese has several kinds of scripts, the main ones being Kanji, Hiragana, and Katakana. These scripts can be used to write the same word in several ways. For example, the dictionary entry 先生 (*sensei*, "teacher") can also be written in Hiragana form せんせい (*sensei*) or Katakana form センセイ (*sensei*). Most words are normally written in the standard form, but in informal written text (e.g., Twitter text), these same words are often written in a non-standard form. In examining Twitter data for such non-standard tokens, we found that 55.0% of them were types (1) to (3) in Table 1, 4.5% were type (4), 20.1% were types (5) to (6), 2.7% were type (7), and the rest did not fall under any of these types since they were the result of dialects, typos, and other factors. In other words, a large majority of the non-standard tokens fell under types (1) to (7). We excluded those that did not as targets in this study because our proposed method cannot easily handle them. Types (1) to (4) occur at character-level and so can be learned from character-level alignment, but types (5) to (6) occur at word-level and it is inefficient to learn them on a character–level basis. Accordingly, we considered generating candidates and features on two levels: character-level and word-level.

3.3 Proposed Method

3.3.1 Overview of Proposed System

We showed the structure of the proposed system in Fig. 3.1. Our approach adds possible normalization candidates to a word lattice and finds the best sequence using a Viterbi decoder based on a discriminative model. We introduced several features that can be used to appropriately evaluate the confidence of the added nodes as normalization candidates. We generate normalization candidates as indicated in Fig. 3.2. We describe the details in the following section.

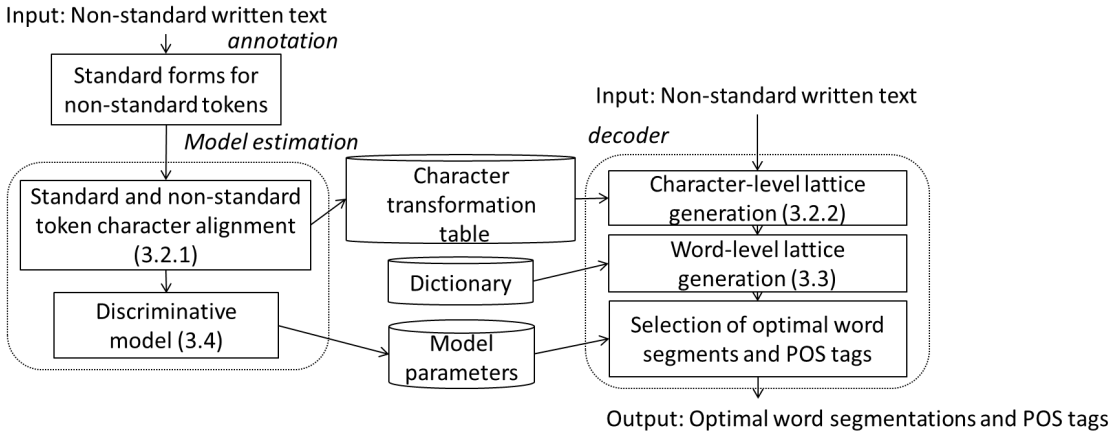

Figure 3.1 Structure of proposed system

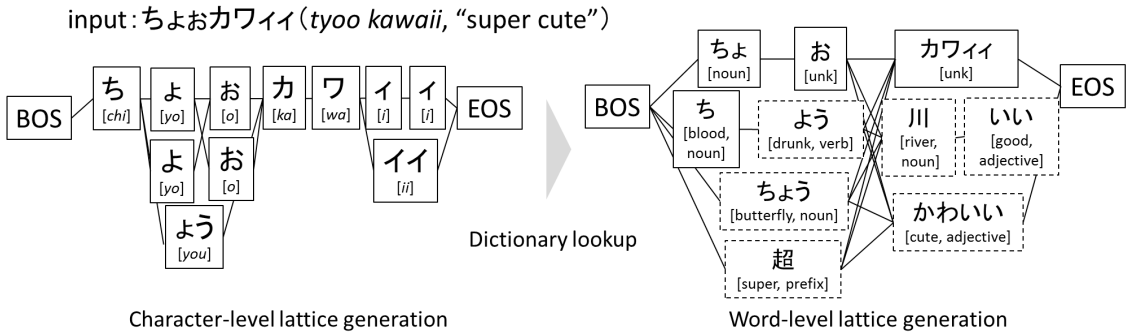

Figure 3.2 Example of candidate generation

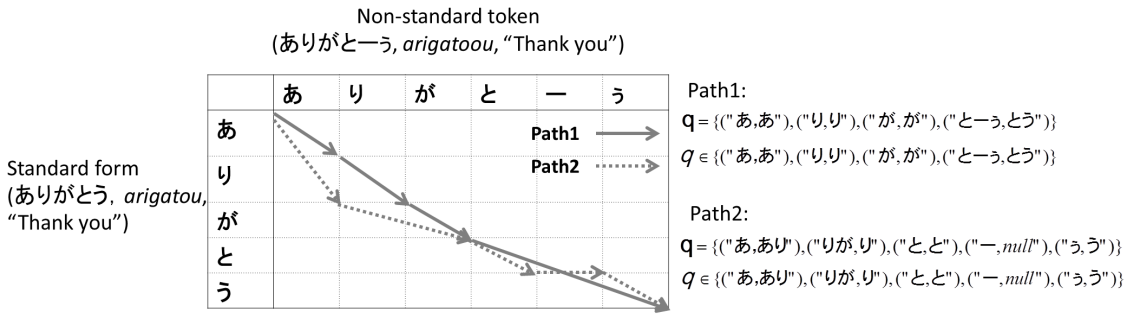

Figure 3.3 Example of character alignment

3.3.2 Character-level Lattice

Character Alignment between Non-standard Tokens and Their Normalized Forms

We have to create a character-level transformation table to generate the character-level lattice. We used the joint multigram model proposed by Sittichai, Grzegorz, and Tarek (2007) to create the transformation table because this model can handle many-to-many character alignments between two character sequences. In observing non-standard tokens and their standard forms, we find there are not only one-to-one character transformations but also many-to-many character transformations. Furthermore, unlike in translation, there is no character reordering so the problems that arise are similar to those in transliteration. Accordingly, we adopted a joint multigram model that is widely used for transliteration problems. The optimal alignment can be formulated as $\hat{\mathbf{q}} = _{\mathbf{q} \in K_d} \prod_{q \in \mathbf{q}} p(q)$, where d is a pair of non-standard tokens and its standard form (e.g., d is ありがとーぅ (*arigatoou*), ありがとう (*arigatou*). Here, q is a partial character alignment in d (e.g., q is “とーぅ, とう”), $\mathbf{q}$ is the character alignment q set in d (e.g., $\mathbf{q}$ of path 1 in Fig. 3.3 is {(“あ, あ”), (“り, り”), (“が, が”), (“とーぅ, とう”)}. K_d is the possible character alignment sequence candidates generated from d. We generate n-best optimal path for K_d in this study.

$$
\begin{aligned}
p(q) &= \gamma_q / \sum_{q \in Q} \gamma_q \qquad (3.1) \\
\gamma_q &= \sum_{d \in D} \sum_{\mathbf{q} \in K_d} p(\mathbf{q}) n_q(\mathbf{q}) = \sum_{d \in D} \sum_{\mathbf{q} \in K_d} \frac{\prod_{q \in \mathbf{q}} \bar{p}(q)}{\sum_{\mathbf{q} \in K_d} \prod_{q \in \mathbf{q}} \bar{p}(q)} n_q(\mathbf{q}),
\end{aligned}
$$

and where D is the number of the d pair, Q is the set of q, and $n_q(\mathbf{q})$ is the count of q that occurred in $\mathbf{q}$. In our system, we allow for standard form deletions (i.e., mapping of a non-standard character to a null standard character) but not non-standard token deletions. Since we use this alignment as the transformation table when generating a character-level lattice, the lattice size becomes unnecessarily large if we allow for non-standard form deletions. In the calculation step of the EM algorithm, we calculate the expectation (partial counts) γ_q of each alignment in the E-step, calculate the joint probability $p(q)$ that maximizes the likelihood function in the M-step as described before, and repeat these steps until convergence occurs. $\bar{p}(q)$ indicates the result of $p(q)$ calculated in the previous step over the iteration. When generating the character-level lattice, we used alignments that were expected to exceed a predefined threshold. We used γ_q ($q = (c_t, c_v)$) and $r(c_t, c_v)$ as thereshold, where c_t and c_v are the partial character sequence of non-standard token and it's standard form respectively. $r(c_t, c_v)$ is calculated by $r(c_t, c_v) = \gamma_q / n_{c_v}.$, where n_{c_v} is the number of occurrences of c_v in the training data. We set the threshold $\gamma_{q\_thres} = 0.5$, and $r(c_t, c_v)_{thres} = 0.0001$ in this study. We also used $r(c_t, c_v)$ as a feature of cost function in subsection. 3.4.2. When calculating initial value, we set $p(c_t, c_v)$ high if the character c_t and c_v are the same character and the length of each character is 1. We also give the limitation that a Kanji character does not change to a different character and is aligned with same character in the calculation step of the character alignment.

Generation of Character-level Lattice Based on Transformation Table

First, repetitions of more than one letter of "ー", "〜", "-", and "っ" are reduced back to one letter (e.g., ありがとーーーう (*arigatooooou*, "Thank you") is reduced to ありがとーう (*arigatoou*)) for the input text. In addition, repetitions of more than three letters other than "ー", "〜", "-", and "っ" are reduced back to three letters (e.g., うれしいいいいい (*uresiiiiiii*, "I'm happy") is reduced back to うれしいいい (*uresiiii*)). These preprocessing rules are inspired by Han and Baldwin (2011) and determined by taking the Japanese characteristics into consideration. We also used these rules when we estimated the alignments of the non-standard tokens and their standard forms. Next, we generate the character-level normalization candidates if they match the key transformation table in the input text. For example, if the transformation table contains (q, log$p(q)$)= ("ょぉ (*yoo*), よう (*you*)", -8.39), ("ぉ (*o*), お (*o*)", -7.56), and the input text includes the character sequence "ちょぉ" (*tyoo*), we generate a new sequence "ちょう" (*tyou*) and "ちょお" (*tyoo*). In other words, we add new nodes "よう" (*you*) and "お" (*o*) in the position of "ょぉ" (*yoo*) and "ぉ" (*o*), respectively (see Fig. 3.2).

3.3.3 Generation of Word-level Lattice

We generate the word lattice based on the generated character-level lattice using dictionary lookup. We exploit dictionary lookup by using the possible character sequence of the character-level lattice while the traditional approach exploits it by using only the input character sequence. For example, we exploit dictionary lookup for character sequences such as "ちょぉカワィィ" (*tyoo kawaii*) and "ちょうカワィィ" (*tyou kawaii*) and "ちよぉカワィィ" (*chiyou kawaii*) and "ちょぉカワイイ" (*tyoo kawaii*) (see Fig. 3.2)

Furthermore, we use the phonetic information of the dictionary to generate the normalization candidates for Hiragana and Katakana substitution. For example, assume "超" (*tyou*, "super") and "かわいい" (*kawaii*, "cute") are the dictionary words. Then, if the input text

contains the character sequences “ちょう” (*tyo*) (which is written in Hiragana) and “カワイイ” (*kawaii*) (which is written in Katakana), we add “超” (*tyo*, “super”) and “かわいい” (*kawaii*, “cute”) to the word lattice as the normalization candidates since the two character sequences are pronounced identically. By using this two-step algorithm, we can handle any combinational derivational patterns, such as Katakana substitutions or substitutions of lowercases like “カワィィ” (*kawaii*) → “カワイイ” (*kawaii*)→ “かわいい” (*kawaii*, “cute”) (see Fig. 3.2). Note that we filtered candidates on the basis of a predefined threshold to prevent the generation of unnecessary candidates. The threshold was defined on the basis of the character sequence cost of normalization, which is described in subsection 3.4.2. Furthermore, we limited the number of character transformations to two per word.

3.3.4 Decoder

Objective Function

The decoder selects the optimal sequence $\hat{y}$ from $L(s)$ when given the candidate set $L(s)$ for sentence s. This is formulated as $\hat{y} =_{y \in L(s)} \mathbf{w} \cdot \mathbf{f}(y)$ (W. Jiang, Mi, and Q. Liu, 2008; Kaji and Kitsuregawa, 2013), where $\hat{y}$ is the optimal path, $L(s)$ is the lattice created for sentence s, and $\mathbf{w} \cdot \mathbf{f}(y)$ is the dot product between weight vector $\mathbf{w}$ and feature vector $\mathbf{f}(y)$. The optimal path is selected according to the $\mathbf{w} \cdot \mathbf{f}(y)$ value.

Features

The proposed lattice generation algorithm generates a lattice larger than that generated in traditional dictionary-based lattice generation. Therefore, we need to introduce an appropriate normalization cost into the objective function. We listed the features we used in Table 2. Let w_i be the ith word candidate and p_i be the POS tag of w_i. p_{i-1} and w_{i-1} are adjacent POS tag and word respectively. We also used the word unigram cost $f_{w_i p_i}$, the cost for a pair of adjacent POS f_{p_{i-1}, p_i} that are quoted from MeCab (T. Kudo, 2005), and five additional

Name	Feature
Word unigram cost	$f_{w_i p_i}$
POS bi-gram cost	f_{p_{i-1},p_i}
Word-POS bi-gram cost	$-\log p_{w_{i-1}p_{i-1},w_i p_i}$
Character sequence cost	$\log(p'_s/p'_{t_i})$
	where, $p'_x = p_x^{1/length(x)},\ p_x = \prod_{j=1}^{n} p(c_j \mid c_{j-5}^{j-1}),\ x \in \{s, t_i\}$
Character transformation cost	$\phi_{trans_i} \cdot (-\log r(c_t, c_v))$
Hiragana substitution cost	$\phi_{h_i} \cdot f_{w_i p_i}$
Katakana substitution cost	$\phi_{k_i} \cdot f_{w_i p_i}$

Table 3.2 Feature list of the decoder. ϕ_{trans_i} is 1 if w_i is generated by character transformation, otherwise 0. ϕ_{h_i} is 1 if w_i is generated by Hiragana substitution, otherwise 0. ϕ_{k_i} is 1 if w_i is generated by Katakana substitution, otherwise 0.

types of costs. These are the word-pos bi-gram cost $-\log p_{w_{i-1}p_{i-1},w_i p_i}$ of a blog corpus; the character transformation cost $\phi_{trans_i} \cdot (-\log r(c_t, c_v))$, which is calculated in Section3.3.2, for nodes generated by character transformation; the Hiragana substitution cost $\phi_{h_i} \cdot f_{w_i p_i}$ for nodes generated by Hiragana substitution; the Katakana substitution cost $\phi_{k_i} \cdot f_{w_i p_i}$ for nodes generated by Katakana substitution; and the character sequence cost $\log(p'_s/p'_{t_i})$ for all the normalized nodes. The character sequence cost reflects the character sequence probability of the normalization candidates. Here, s and t_i are input string and transformed string respectively. (e.g., In Fig. 3, for the normalized node "かわいい" (cute, adjective), s is "ちょぉカワィィ" and t_i is "ちょぉかわいい"). Then p_s and p_{t_i} are calculated by using the character 5-gram of a blog corpus, which is formulated by $p_s = p(c_1 \cdots c_n) = \prod_{j=1}^{n} p(c_j \mid c_{j-5}^{j-1})$, where c_j is the j th character of character sequence s. p'_{t_i} and p'_s are normalized by using the length of each string s and t_i as $p'_{t_i} = p_{t_i}^{1/length(t_i)}$. We set the threshold $(p'_s/p'_{t_i})_{thres} = 1.5$ for generating a Hiragana or Katakana normalization candidate in this study. Since all those features can be factorized, the optimal path is searched for by using the Viterbi algorithm.

Training

We formulated the objective function for tuning weights $\mathbf{w}$ by using Eq. 3.2. The weights $\mathbf{w}$ are trained by using the minimum error rate training (MERT) (Machery, Och, and Thayer, 2008). We defined the error function as the differences between the reference word segmentations and the POS tags of the reference sequence y_{ref} and the system output ${}_{y \in L(s)}\mathbf{w} \cdot \mathbf{f}(y)$.

$$\hat{\mathbf{w}} =_{\mathbf{w} \in \mathbf{W}} \sum_{i=1}^{N} \text{error}(y_{ref}, {}_{y \in L(s)}\, \mathbf{w} \cdot \mathbf{f}(y)) \tag{3.2}$$

3.4 Experiments

3.4.1 Dataset and Estimated Transformation Table

We conducted experiments to confirm the effectiveness of the proposed method, in which we annotated corpora of a Japanese blog and Twitter. The Twitter corpus was split into three parts: the training, development, and test sets. The test data comprised 300 tweets, development data comprised 500 sentences and the training data comprised 4208 tweets. We randomly selected the test data which contained at least one non-standard token. The test data comprised 4635 words, 403 words of them are non-standard token and are orthographically transformed into normalized form and POS tags. The blog corpus comprised 8023 sentences and all of them were used as training data. Training data was used for extracting character transformation table and development data was used for estimating parameters of discriminative model. We used the IPA dictionary provided by MeCab to generate the word-level lattice and extracted the dictionary-based features. We itemized the estimated character transformation patterns in Table 3. There were 5228 transformation patterns that were learned from the training data and we used 3268 of them, which meets the predefined condition. The learned patterns cover most of the previously proposed rules. In addition, our method can learn more of the variational patterns that are difficult to create manually.

non-standard character c_t	standard character c_v	logp(q)	non-standard character c_t	standard character c_v	logp(q)
ー	null	-4.233	っす(*ssu*)	です(*desu*)	-5.999
まぁ(*maa*)	まあ(*maa*)	-5.059	どー(*doo*)	どう(*dou*)	-6.210
しょ(*syo*)	しょう(*syou*)	-5.211	ねー(*nee*)	ない(*nai*)	-6.232
だろ(*daro*)	だろう(*darou*)	-5.570	りゃ(*rya*)	れは(*reha*)	-6.492
っ(*ttsu*)	null	-5.648	てん(*ten*)	てる(*teru*)	-6.633
んと(*nto*)	んとう(*ntou*)	-5.769	ゆう(*yuu*)	いう(*iu*)	-6.660
わ(*wa*)	は(*wa*)	-5.924	なん(*nan*)	なの(*nano*)	-6.706

Table 3.3 Example of character-level transformation table

3.4.2 Baseline and Evaluation Metrics

We compared the five methods listed in Table 4 in our experiments. Traditional means that which generates no normalization candidates and only uses the word cost and the cost for a pair of adjacent POS, so we can consider it as a traditional Japanese morphological analysis. We compared three baselines, Baseline1, Baseline2 and Baseline3. Baseline1 is the conventional rule-based method (considering insertion of long sound symbols and lowercases, and substitution with long sound symbols and lowercases), which was proposed by Sasano, Kurohashi, and Okumura (2013). In Baseline2, 3, and Proposed, we basically use the proposed discriminative model and features, but there are several differences. Baseline2 only generates character-level normalization candidates. Baseline3 uses our two-step normalization candidate generation algorithms, but the character transformation cost of all the normalization candidates that are generated by character normalization is the same. Proposed generates the character-level and Hiragana and Katakana normalization candidates and use all features we proposed.

We evaluated each method on the basis of precision and recall and the F-value for the overall system accuracy. Since Japanese morphological analysis simultaneously estimates the word segmentation and POS tagging, we have to check whether or not our system is negatively affected by anything other than the non-standard tokens. We also evaluated the

	word segmentation			word segmentation and POS tag			
method	precision	recall	F-value	precision	recall	F-value	recall*
Traditional	0.716	0.826	0.767	0.683	0.788	0.732	-
Rule based (BL1**)	0.753	0.833	0.791	0.717	0.794	0.754	0.092
Proposed	**0.856**	**0.883**	**0.869**	**0.822**	**0.849**	**0.835**	**0.667**
- without Hiragana and Katakana normalization (BL2)	0.834	0.875	0.854	0.798	0.838	0.818	0.509
- character transformation cost is fixed (BL3)	0.838	0.865	0.851	0.807	0.834	0.821	0.533

* considering only normalized words, ** BL:baseline

Table 3.4 Results of precision and recall of test data

recall with considering only normalized words. That value directly reflects the performance of our normalization method. We registered emoticons that occurred in the test data in the dictionary so that they would not negatively affect the systems' performance.

3.4.3 Results and Discussion

The results are classified in Table 4. As the table shows, the proposed methods performed statistically significantly better than the baselines and the traditional method in both precision and recall ($p < 0.01$), where the precision was greatly improved. This indicates that our method can not only correctly analyze the non-standard tokens, but can also reduce the number of wrong words generated. Baseline1 also improved the accuracy and recall compared to the traditional method, but the effect was limited. When we compare Proposed with Baseline2, we find the F-value is improved when we take the Hiragana and Katakana substitution into consideration. Baseline3 also improved the F-value but its performance is inferior to proposed method.This proves that even if we can generate sufficient normalization candidates, the results worsen if the weight parameter of each normalization candidate is not appropriately tuned. The column of "recall*" in Table 4 specifies the improvement rates of the non-standard tokens. The proposed methods improve about seven times when using Baseline1 while preventing degradation. These results prove that we have to generate appropriate and sufficient normalization candidates and appropriately tune the cost of each candidate to improve both the precision and recall.

input	traditional	proposed	gold standard
(1)あぢー(*adii*)	あ(*a*)/ぢ(*di*)/ー	**あつい**(*atsui*)	あつい(*atsui*, "hot")
(2)すげー(*sugee*)	すげ(suge)/ー	**すごい**(*sugoi*)	すごい(*sugoi*, "great")
(3)ごっめーん(*gommeen*)	ご(*go*)/っ/め(*me*)/ー/ん(*n*)/	**ごめん**(*gomen*)	ごめん(*gomen*, "I'm sorry")
(4)ひつよう(*hitsuyou*)	ひつ(*hitsu*)/よう(*you*)	**必要**(*hitsuyou*)	必要(*hitsuyou*, "necessary")
(5)だいちゅき(*daichuki*)	だ(*da*)/いち(*ichi*)/ゅ(*yu*)/き(*ki*)/	**大好き**(*daisuki*)	大好き(*daisuki*, "like very much")
(6)おせえええ(*oseee*)	おせ(*ose*)/ええ(*ee*)/え(*e*)	おせ(*ose*)	おそい(*osoi*, "slow")
(7)かんわいい(*kanwaii*)	かん(*kan*)/わ(*wa*)/いい(*ii*)	官話(*kanwa*)/いい(*ii*)	かわいい(*kawaii*, "cute")
(8) いない(*inai*)	**い(*i*)/ない(*nai*)**	以内(*inai*)	い/ない(*i*/*nai*, "absent")

Table 3.5 System output examples

We show examples of the system output in Table 5. In the table, slashes indicate the position of the estimated word segmentations and the words that were correctly analyzed are written in bold font. Examples (1) to (5) are examples improved by using the proposed method. Examples (6) to (7) are examples that were not improved and example (8) is an example that was degraded. Examples (1) to (3) include phonetic variations and example (4) is a Hiragana substitution. Example (5) is a combinational transformation pattern of a phonetic variation and Hiragana substitution. We can see our system can analyze such variational non-standard tokens for all these examples. Two types of errors were identified. The first occurred as the result of a lack of a character transformation pattern and the second was search errors. Example (6) shows an example of a case in which our system couldn't generate correct normalization candidate because there was not corresponding character transformation pattern, even though there was a similar phonetic transformation pattern. To ensure there will be no lack of transformation patterns, we should either increase the parallel corpus size to enable the learning of more patterns or derive new transformation patterns from the learned patterns. Example (7) shows an example of a case in which a normalized candidate was generated but a search failed to locate it. Example (8) shows an example of a case in which the result was degraded. Our system can control the degradation well, but there are several degradation caused by normalization. We will need to develop a more complicated model or introduce other features into the current model to reduce the number of search errors.

3.5 Conclusion and Future Work

We introduced a text normalization approach into joint Japanese morphological analysis and showed that our two-step lattice generation algorithm and formulation using discriminative methods outperforms the previous method. In future work, we plan to extend this approach by introducing an unsupervised or semi-supervised parallel corpus extraction for learning character alignments to generate more patterns at a reduced cost. We also plan to improve our model's structure and features and implement it with a decoding method to reduce the number of search errors. In addition, we should consider adding other types of unknown words (such as named entities) to the morphological analysis system to improve its overall performance.

Chapter 4

Automatically Extracting Variant-Normalization Pairs for Japanese Text Normalization

4.1 Introduction

Social media texts contain many non-standard tokens (lexical variants), e.g., by lengthening (“goooood" for “good") or abbreviating them (“tmrw" for “tomorrow"). Current language processing systems often fail to analyze such non-standard tokens, so normalizing them into standard tokens as a preprocess is promising for analyzing such noisy texts robustly (Cook and Stevenson, 2009; Han, Cook, and Baldwin, 2012; Chen Li and Yang Liu, 2012; Chen Li and Yang Liu, 2014). The normalization task mainly consists of two components. One is detecting variant words and generating normalization candidates. The other is constructing a word lattice from possible normalization candidates and decoding to select the best normalized word sequences. Early work on normalization focused on supervised approaches using labeled text, e.g., an approach based on a statistical machine translation (Aw et al., 2006; Pennell and Yang Liu, n.d.). However, social network service (SNS) text has a dy-

namic nature, and large SNS text is costly to annotate. Recent work has been focused on unsupervised approaches. For example, Han, Cook, and Baldwin (2012) proposed generating variant-normalization pairs automatically on the basis of distributional similarity and string similarity. Hassan and Menezes (2013) developed the approach by using a graph-based approach. Yang and Eisenstein (2013) introduced a highly accurate unsupervised normalization model. As just described, unsupervised methods have been developed for English normalization tasks.

Japanese SNS text also contains variant words, and several normalization methods have been proposed (Sasano, Kurohashi, and Okumura, 2013; Kaji and Kitsuregawa, 2014; Saito et al., 2014). The basic framework of Japanese normalization is quite similar to that of English normalization. However, the problem is more complicated in Japanese normalization because Japanese words are not segmented using explicit delimiters, so we have to estimate word segmentation simultaneously in the decoding step. Variant words are also more difficult to extract automatically in Japanese than in explicitly segmented languages such as English. Unlike English normalization, the approaches for generating normalization candidates in Japanese are based on manually created rules or supervised training using annotated text. Japanese normalization contains problems to which the English unsupervised approach is simply applied. Although the English unsupervised approach assumes that there are explicit word segmentations, conventional analyzers often fail to segment non-standard words in Japanese. Therefore, to extract variants in an unsupervised fashion, we have to introduce an idea to generate correct word segmentation of variant words.

Our idea for this problem is to use short sentences and phrases in SNS text. SNS text, like tweets from Twitter, contains many short sentences and phrases consisting of a single word or several words. For Example, “おっはよーん！ (*ohayon*, Good Morning)" is a variant form of “おはよう！ (*ohayou*)," and “ちょーさみー (*cho samii*, It is very cold)" is a variant form of “超 (*cho*, very)/寒い (*samui*, cold)." Since these short sentences often contain variant words, they can be used as efficient cues for extracting a variant word. Our idea is to not

extract a variant-normalization pair in one step. Instead, we present a two-step normalization approach. In the first step, we extract coarse candidates for variant-normalization pairs from unlabeled text, and in the second step, we incorporate the extracted pairs into Japanese morphological analysis and normalization. The appropriate normalization candidates are selected in the second step. We use training data for morphological analysis in the second step but do not use annotated data in the first step. Therefore, we can efficiently extract many types of variant-normalization pairs that appear in real text.

The contributions of this study are summarized as follows.

- We developed a new method for extracting pairs of variant and normal forms from tweets, which have no explicit delimiters, by focusing on short phrases and sentences in Twitter.

- We incorporated the variant-normalization pairs extracted by our method from tweets into a Japanese morphological analysis method and statistically significantly improved the accuracy for variant words without degrading the overall accuracy for Japanese morphological analysis.

4.2 Background

4.2.1 Japanese Morphological Analysis

As we mentioned above, we have to consider Japanese normalization tasks with Japanese morphological analysis. In this section, we describe the basic idea of Japanese morphological analysis. Japanese Morphological analysis can be interpreted as ranking while using a word lattice and scores of each path (Kaji and Kitsuregawa, 2013). There are two points to consider in the analysis procedure: how to generate the word lattice and how to formulate the score of each path. In Japanese morphological analysis, the dictionary-based approach has been widely used to generate word lattices (Kudo, Yamamoto, and Matsumoto, 2004;

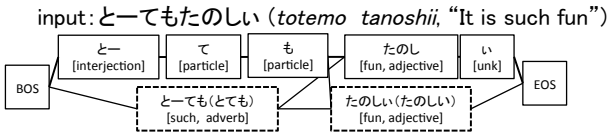

Figure 4.1 Example of Japanese morphological analysis and normalization

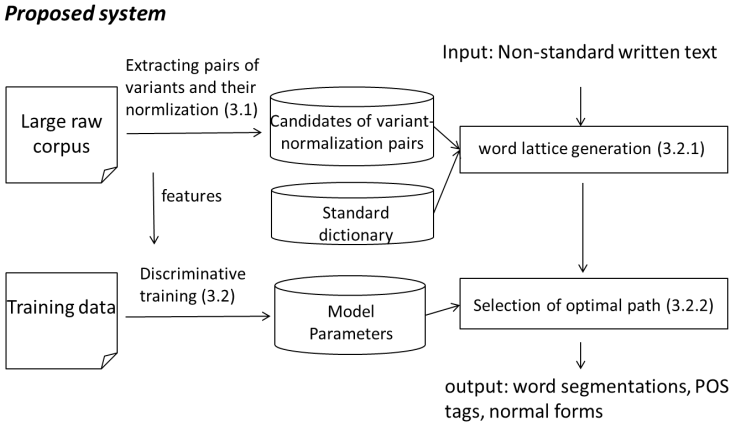

Figure 4.2 Overview of proposed system

Kaji and Kitsuregawa, 2013). To calculate the score of each path, two main scores are widely used: the score for a candidate word and the score for a pair of adjacent parts-of-speech (POSs). We can consider other various scores by using discriminative model (Kudo, Yamamoto, and Matsumoto, 2004; Kaji and Kitsuregawa, 2013).

4.2.2 Related Work

Several studies have been conducted on Japanese morphological analysis and normalization. The approach proposed by Sasano, Kurohashi, and Okumura (2013) developed heuristics to flexibly search by using a simple, manually created derivational rule. Their system generates a normalized character sequence based on derivational rules and adds new nodes when generating the word lattice using dictionary lookup. Figure 4.1 presents an example of this approach. If the non-standard written sentence “とーてもたのしぃ (*totemo tanoshii*, It is such fun)" is input, the traditional dictionary-based system generates nodes that are described using solid lines, as shown in Figure 4.1. Since “とーても (*totemo*, such)" and “たの

しぃ (*tanoshii*, fun)" are Out Of Vocabulary (OOVs), the traditional system cannot generate the correct word segments or POS tags. However, their system generates additional nodes for the OOVs, shown as broken line rectangles in Figure 4.1. In this case, derivational rules are used that substitute "ー" with "null" and "ぃ (i)" with "い (i)", and the system can generate the standard forms "とても (*totemo*, such)" and "たのしい (*tanoshii*, fun)" and their POS tags. If we can generate sufficiently appropriate rules, these approaches seem to be effective. However, there are many types of derivational patterns in SNS text, and they are difficult to all cover manually. Moreover, how to set the path score for appropriately ranking the word lattice when the number of candidates increases becomes a serious problem.

Saito et al. (2014) proposed supervised extraction of derivational patterns (we call them transformation patterns), incorporated these patterns into a word lattice, and formulated morphological analysis and normalization using a discriminate model. Although this approach can generate broad-coverage normalization candidates, it needs a large amount of annotation data of variant words and their normalization. Kaji and Kitsuregawa (2014) also proposed morphological analysis and normalization based on a discriminative model and created variant words on the basis of hand-made rules. As far as we know, automatic extraction of variant-normalization pairs has not been researched. If we can extract variant-normalization pairs automatically, we can decrease the annotation cost and possibly increase accuracy by combining our method with other conventional methods.

Several studies have applied a character-based approach. For example, Sasaki et al. (2013) proposed a character-level sequential labeling method for normalization. However, it handles only one-to-one character transformations and does not take the word-level context into account. The proposed method can handle many-to-many character transformations and takes word-level context into account, so it has a wider scope for handling non-standard tokens.

Many studies have been done on text normalization for English; for example, Han and Baldwin (2011) classifies whether or not OOVs are non-standard tokens and estimates stan-

dard forms on the basis of contextual, string, and phonetic similarities. Han, Cook, and Baldwin (2012) and Hassan and Menezes (2013) developed the method of extracting variant-normalization pairs automatically for English. Yang and Eisenstein (2013) introduced a highly accurate unsupervised normalization model using log-linear model. In these studies, clear word segmentations were assumed to exist. However, since Japanese is unsegmented, the normalization problem needs to be treated as a joint normalization, word segmentation, and POS tagging problem.

Thus, we propose automatically extracting normalization candidates from unlabeled data and present a method for incorporating these candidates into Japanese morphological analysis and normalization. Our method can extract new variant patterns from real text.

4.3 Proposed Method

Our method consists of two parts. The first involves extracting normalization candidates and their normal forms from unlabeled data. The second involves a morphological analysis and normalization using extracted variants. Basically, we use a previously proposed dictionary based approach (Sasano, Kurohashi, and Okumura, 2013; Saito et al., 2014; Kaji and Kitsuregawa, 2014), but the method for generating normalization candidates and some features used in a discriminative model are new. The proposed system is illustrated in Figure 4.2.

In the first part, we generate a coarsely segmented corpus and calculate the pairwise similarity of two arbitrary nodes that appear in the segmented corpus. In a previous study (Han and Baldwin, 2011), the nodes were assumed to be single words. On the other hand, our method assigns a node to not only single words but also short phrases (See. 4.3.1). To calculate similarity between two nodes, we use semantic similarity and phonetic similarity. After calculating similarity, we filter the pairs that do not exceed a similarity threshold. We use word embeddings as a semantic similarity measure. We describe this more precisely in 4.3.1.

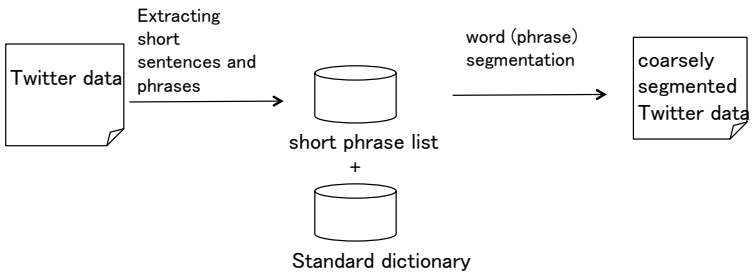

Figure 4.3 Flow of generating coarsely segmented corpus

In the second part, the problem is how to incorporate extracted variants into Japanese morphological analysis. We use a discriminative model and Viterbi decoding for estimating word segmentation, POS tagging, and normalization. To prevent degradation induced by incorporating extracted variants, we introduce many types of features. We describe this more precisely in 4.3.2.

4.3.1 Extracting Candidates of Variant-Normalization Pairs from Twitter Texts

Preparation of Coarsely Segmented Corpus Using Short SNS Sentences and Phrases

We have to prepare a segmented corpus for generating normalization candidates and calculating similarity. The flow of generating coarsely segmented corpus is shown in Figure 4.3. As we mentioned above, we cannot determine the explicit word segmentation of unlabeled data, especially for variant words. However, we can assume that there are some explicit segmentations in text: the left and right ends of sentences and symbols such as punctuation, brackets, pictographs, emoticons, linefeed characters, commas, and spaces. SNS text contains many short sentences and phrases consisting of a word or several words. Our idea is to use the units of short sentences and short phrases delimited in symbols in SNS text as cues for extracting variant words.

More specifically, we first segment large Twitter text with several predefined symbols,

extract character sequences consisting of ten or fewer characters, and insert them into a standard dictionary. Then we segment the large twitter corpus using an expanded dictionary and conventional analyzer. An example of a segmented sentence using an expanded dictionary is "おっはよーう(*ohhayou*)/！(Good morning!)", whereas the result using a standard dictionary is "おっ(*o*)/は(*ha*)/よー(*yo*)/う(*u*)/！" Since this segmented text contains several segmentation errors and noise, we extract reliable candidates using a similarity threshold described in the next subsection.

Similarity Measures

To calculate the similarity between two nodes w_i and w_j appearing in a segmented corpus (4.3.1), we mainly use two similarity measures: semantic and phonetic.

Semantic Similarity We calculate semantic similarity between w_i and w_j by inducing dense real-valued low-dimensional embeddings from large unlabeled text (Mikolov, Yih, and Zweig, 2013). We use the tool word2vec [1] to calculate embeddings of each node. Semantic similarity is defined as

$$semsim(w_i, w_j) = cos(vec(w_i), vec(w_j)) \tag{4.1}$$

This semantic similarity is used as a feature of a normalization and morphological analysis model (4.3.2). We set the embedding size is 200.

Phonetic Similarity We first convert a surface string into pronunciations (Japanese Kanji to Hiragana) and calculate the edit distance. We use two types of edit distance: standard and modified. For calculating modified edit distance, we set the substitution cost of two strings 0.5 when two strings have the same vowels, two strings have the same consonant

[1] http://code.google.com/p/word2vec/

or two strings are vowels. We also set insertion and deletion cost of vowels 0.5, while the standard cost of substitution, insertion and deletion is 1.

We use standard edit distance and modified edit distance as a threshold of candidate filtering (4.3.1) and modified edit distance as an element of a feature of a normalization and morphological analysis model (4.3.2). For a feature and threshold, we set the phonetic similarity $psim(w_i, w_j)$ as follows:

$$psim(w_i, w_j) = \prod_{m_i, m_j} p(m_i, m_j) \tag{4.2}$$

$$p(m_i, m_j) = 1 - MED(m_i, m_j)/OC(m_i, m_j) \tag{4.3}$$

Where $OC(m_i, m_j)$ indicates the total number of operations for calculating edit distance of m_i and m_j and $MED(m_i, m_j)$ indicates the modified edit distance. m_i and m_j indicate the morphemes in w_i and w_j, respectively.

To calculate morphological-level features, we analyzed w_j using conventional morphological analyzer and make morphological-level alignment using character-level alignment of w_i and w_j and morphological information of w_j. Here, w_j and w_i are regarded as a normal form and a variant form, respectively. Here is an example of w_i = "たんじょーび (birthday)" and w_j = "たんじょうび (birthday)". In this case, morphological-level alignment is (たんじょー/たんじょう, び/び) since character-level alignment is (た/た, ん/ん, じ/じ, ょ/ょ, ー/う, び/び) and word segmentation of w_j is (たんじょう, び).

Candidate Filtering using Similarity Measures

We calculate the pairwise similarity between two nodes appearing in a segmented corpus (4.3.1) and their filter using a similarity measure we defined (4.3.1). The set of nodes N consists of all tokens w that appear in the segmented corpus. Since there is a huge number of node pairs and most are irrelevant, we have to filter the pairs that have low similarities. Here,

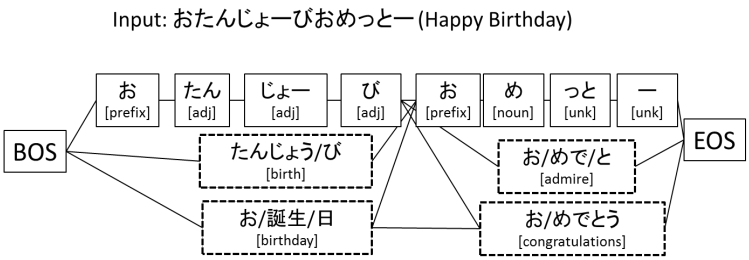

Figure 4.4 Example of proposed lattice

we set the threshold of *semsim* to 0.4, and the threshold of standard phonetic edit distance is 2. Moreover, we filter the pairs in which the consonants of the beginning phonetic symbols of each morphemes are different or morphological-level phonetic similarity $p(m_i, m_j)$ is lower than 0.6. We also filtered the candidates when the number of consonants and all characters in a variant form (except for "っ", "ん", and lower case letters) are larger than that of a normal form at morphological level. If a variant word is already exists in a standard dictionary, we filtered the candidate. Note that this filtering is not intended to exactly identify whether the pair has the relationship of variant and normalization. Since we use only two similarities, simple phonetic information and the coarsely segmented corpus in this phase, we only extract candidates of variants and their normal forms coarsely in the first step. Then, we exactly identify the word segmentation and normalization simultaneously in the second step (4.3.2).

4.3.2 Normalization and Morphological Analysis

Incorporating Normalization Candidates in Japanese Morphological Analysis

We generate the word lattice using extracted candidate for variant-normalization pairs and dictionary lookup (See Figure 4.4). The broken line rectangles in Figure4.4 are nodes added by the proposed method. We exploit dictionary lookup by using the possible character sequence of the extracted normalized character sequences when variant character sequences match the input character sequences. For example, we exploit dictionary lookup for input

Name	Feature
Word unigram score	$f_{m_i p_i}$
POS bi-gram score	f_{p_{i-1},p_i}
Character sequence score	$1/\log(p'_{t_i}/p'_s)$ if $p'_{t_i} > p'_s$ otherwise 0. where, $p'_x = p_x^{1/length(x)}$, $p_x = \prod_{j=1}^{k} p(c_j \mid c_{j-5}^{j-1})$, $x \in (s, t_i)$
Character similarity score	$\log(psim)$
Semantic similarity score	$\log(semsim)$
Character frequency score	$\log(freq_w + 1)$
Character and morph transformation score	$\log(freq_{ct} + 1)$, $\log(freq_{mt} + 1)$, ϕ_{ct}, ϕ_{mt}

Table 4.1 Feature list

character sequences such as “おたんじょーびおめっとー (happy birthday)" and add the possible normalized word sequences such as “お/誕生/日（birthday）" and “お/めでとう (congratulations)" which are from extracted variant-normalization candidates. The proposed method is intended to generate normalized word sequences. In the first step, appropriateness of word segmentation is not taken into account, but in this phase, we can exactly evaluate whether the acquired pair is appropriate for normalization or not by considering the morphological connectivity.

Objective Function

We used a discriminative model for incorporating many features. The decoder selects the optimal sequence $\hat{y}$ from $L(s)$ when given the candidate set $L(s)$ for sentence s. This is formulated as follows (W. Jiang, Mi, and Q. Liu, 2008; Kaji and Kitsuregawa, 2013):

$$\hat{y} = _{y \in L(s)} \mathbf{w} \cdot \mathbf{f}(y) \tag{4.4}$$

Where $\hat{y}$ is the optimal path, $L(s)$ is the lattice created for s, $\mathbf{w} \cdot \mathbf{f}(y)$ is the dot product between weight vector $\mathbf{w}$ and feature vector $\mathbf{f}(y)$. The optimal path is selected in accordance with the $\mathbf{w} \cdot \mathbf{f}(y)$ value. For estimating parameters, we used the averaged perceptron, which is widely used (Collins, 2002).

Features

The proposed lattice generation method generates a lattice larger than that generated in traditional dictionary-based lattice generation. Therefore, we need to introduce appropriate normalization scores into the objective function to prevent degradation. Table 4.1 lists the features we used. Let m_i be the ith word candidate and p_i be the POS tag of m_i. p_{i-1} and m_{i-1} are adjacent POS tag and word, respectively. We used the word unigram score $f_{m_i p_i}$, the cost for a pair of adjacent POSs f_{p_{i-1},p_i} that are estimated by MeCab [2], and additional scores.

The character sequence score reflects the character sequence probability of the normalization candidates (Saito et al., 2014). Here, s and t_i are input string and transformed string, respectively (e.g., in Figure 4.4, for the normalized node "お誕生日 (birthday)", s is "おたんじょーびおめっとー" and t_i is "お誕生日おめっとー". Then p_s and p_{t_i} are calculated by using the character 5-gram of a news corpus. c_j is the jth character of character sequence. We also used character sequence score as a candidate filter. We filtered the candidates that did not satisfy the pre-defined condition that $p'_s \leq p'_{t_i}$.

The character similarity score is calculated using $psim$ (see 4.3.1). The semantic similarity score is calculated using $semsim$ (see 4.3.1). The Character frequency score is a frequency of surface character sequences of variant nodes appeared in news data. Since variant words rarely appear in the news data, we use this feature to identify variant words and standard words. The character and morph transformation score is related to transformation patterns. $\log(freq_{ct}+1)$ and $\log(freq_{mt}+1)$ are the frequency of transformation patterns ct (character-level) and mt (morphological-level) that are extracted from variant-normalization candidates, respectively. ϕ_{ct} and ϕ_{mt} are 1 if a node contains transformation patterns ct and mt, otherwise 0, respectively. The scale of features were adjusted.

Since all those features can be factorized, the optimal path is searched by using the

[2]http://taku910.github.io/mecab/ (in Japanese)

variant forms	norm forms	translation	*semsim*
うれしーい (*ureshiii*)	うれしい (*ureshii*)	happy	0.655
うれしー (*ureshii*)	うれしい (*ureshii*)	happy	0.684
うれしぃ (*ureshii*)	うれしい (*ureshii*)	happy	0.575
うれすぃ (*uresui*)	うれしい (*ureshii*)	happy	0.568
うれちい (*uretii*)	うれしい (*ureshii*)	happy	0.649
うれしす (*ureshishu*)	うれしい (*ureshii*)	happy	0.715
かわええ (*kawaee*)	かわいい (*kawaii*)	cute	0.744
かんわいい (*kanwaii*)	かわいい (*kawaii*)	cute	0.683
きゃわいい (*kyawaii*)	かわいい (*kawaii*)	cute	0.770
お/ねげー/し/ます (*o/negee/shi/masu*)	お/ねがい/し/ます (*o/negai/shi/masu*)	please	0.657
お/ながい/し/ます (*o/nagai/si/masu*)	お/ねがい/し/ます (*o/negai/shi/masu*)	please	0.590
お/ねがい/し/まふ (*o/negai/shi/mahu*)	お/ねがい/し/ます (*o/negai/shi/masu*)	please	0.678
ちかれ/た (*tikare/ta*)	疲れ/た (*tukare/ta*)	I'm tired	0.742
お/めでとー (*o/medeto*)	お/めでとう (*o/medetou*)	congratulations	0.911
お/めでとぉ (*o/medeto*)	お/めでとう (*o/medetou*)	congratulations	0.796
お/めっとー (*o/metto*)	お/めでとう (*o/medetou*)	congratulations	0.753
お/めてとう (*o/meteto*)	お/めでとう (*o/medetou*)	congratulations	0.665

Table 4.2 Example of extracted pair of variants-normalization candidates

Viterbi algorithm.

Candidate Expansion

Although our method can extract many variants, we expand the variants to achieve higher recall. We use a simple rule for adding simple variation in the decoding step. For example, first, repetitions of more than one character of "ー", "～" and "っ" are reduced to one character and repetitions of more than three characters of Japanese Hiragana and Katakana are reduced to three characters and one character. Moreover, we use the patterns of deletions of "ー", "～", "っ" and lowercase characters (Saito et al., 2014).

4.4 Experiments

4.4.1 Data and Settings

We prepared the Balanced Corpus of Contemporary Written Japanese (BCCWJ) (Maekawa et al., 2014), which is a commonly used dataset in Japan and Twitter data. For unsupervised

variant extraction, we used about 70 million unlabeled Twitter corpora. We used 2,000 sentences of BCCWJ text for training the decoder and Twitter data for the test. Twitter data contain manually annotated word segmentations, POS tags, and normal forms for variant words and consist of 7,213 sentences and 995 variant words. We used Unidic (unidic-mecab) [3] as a standard dictionary.

4.4.2 Baselines and Evaluation Metrics

We compared the three methods listed in Table 4.3 in our experiments. Conventional means a method that generates no normalization candidates and only uses the word cost and the cost for a pair of adjacent POSs, so we can consider it as a conventional Japanese morphological analysis. Rule-based means the conventional rule-based method proposed by Sasano, Kurohashi, and Okumura (2013). The rule-based method considers insertion of long sound symbols and lowercase characters and substitution with long sound symbols and lowercase characters. We basically use the transformation rule of Sasano, Kurohashi, and Okumura (2013) and use three features for the model of Rule-based method: the word score, score for a pair of adjacent POSs, and character transformation score. The character transformation score is constant for all transformation rules. Proposed is our method. We used extracted variant-normalization pairs and all features described in 4.3.2.

We evaluated each method on the basis of precision, recall, and the F-value for the overall system accuracy and the recall for normalization. Since Japanese morphological analysis simultaneously estimates the word segmentation and POS tagging, we have to assess whether or not adding the normalization candidates negatively affects a system.

	word seg			word seg and POS tag			normalization
method	prec	rec	F	prec	rec	F	rec
Conventional	0.865	0.949	0.905	0.837	0.919	0.876	-
Rule-based	0.879	0.952	0.914	0.848	0.918	0.881	0.294
Proposed	0.882	0.951	**0.915**	0.851	0.918	**0.883**	**0.340**

Table 4.3 Test data (Twitter) precision, recall, and F-value results

proposed	conventional	gold	*translation*
(1) おも (思っ) /た *omo* (omott) /*ta*	おもた *omota*	おも (思っ) /た *omo* (omott) /*ta*	I thought
(2) ばよりん (バイオリン) *bayorin* (*baiorin*)	ば/より/ん *ba*/*yo*/*rin*	ばよりん (バイオリン) *bayorin* (*baiorin*)	violin
(3) かんわいい (かわいい) *kanwaii* (*kawaii*)	かん/わ/いい *kan*/*wa* /*ii*	かんわいい (かわいい) *kanwaii* (*kawaii*)	cute
(4) さっむ (寒い) *samu* (*samui*)	さっ/む *sa*/*mu*	さっむ (寒い) *samu* (*samui*)	It is cold
(5) わろえる (笑える) *waroeru* (*waraeru*)	わろ/える *waro*/*eru*	わろえる (笑える) *waroeru* (*waraeru*)	It is funny
(6) おー/こく *oo*/*koku*	おー/こく *oo*/*koku*	おーこく (王国) *ookoku* (*oukoku*)	Kingdom
(7) ついった (ツイート) *tuitta* (*tuitta*)	つ/いっ/た *tu*/*i*/*ta*	ついった (ツイッター) *tuitta* (*tuitta*)	Twitter

"/" indicates the estimated word segmentation. Words in parentheses "()" are estimated normal forms. Underlined words are variant words.

Table 4.4 Example of morphological analysis and normalization outputs

4.4.3 Results

Results of Extracted Variant-Normalization Candidates

Table 4.2 lists examples of the extracted variant-normalization candidates. Our method automatically extracted well-known transformation patterns such as substitution of lowercase characters, insertion of lowercase characters, and insertion of "ー" and "っ" such as "うれしーい (*ureshii*)".

Our method also extracted more variant phonetic transformation patterns such as substitution of "shi" with "pi," "ji," or "hi" and these combinations such as "うれちい (*uretii*)".

[3]http://osdn.jp/projects/unidic/ (in Japanese)

We also list examples of extracted multi-word variant-normalization pairs. The phrase “お/ねげー/し/ます (*o/nege/shi/masu*)" is a variant pattern of the original phrase “お/ねがい/します (*o/negai/shi/masu*)". Our method extracted these multi-word mappings.

Moreover, our method can extract typing errors such as “お/めでとう (*o/medetou*)” with “お/めてとう (*o/meteto*)" and slang such as “うれしす (*ureshisu*)” with “うれしい (*ureshii).*” Such relatively less frequent patterns were often excluded from normalization targets. Our method also extracts many paraphrases: semantically and phonetically similar pairs that are not variant-normalization pairs. This often degrades the results of morphological analysis. We use a discriminative model to prevent such paraphrase pairs appearing in the decoding step.

Morphological Analysis and Normalization Results

Tables 3 and 4 list the results for the Twitter text. The F-value of the proposed method is significantly higher than those of the conventional method and rule-based method. Our method was able to extract broad-coverage variant words, and these candidates also improve the recall of normalization without degrading the overall accuracy of morphological analysis.

Table 4.4 show examples of the system output. In the table, slashes indicate the positions of the estimated word segmentations, and the correctly analyzed words are written in bold. Examples (1) to (5) are examples improved by using the proposed method. Examples (6) and (7) are examples that were not improved.

Comparison with the Supervised Model In Chapter3, we proposed supervised model for normalization and morphological analysis. When we train the model with supervised learning, the recall of normalization was about 0.5. In this study, the recall of normalization was about 0.34. Therefore, our unsupervised learning can acquire about 60% of normalization patterns which can be obtained by supervised learning. Since variant-normalization patterns changes day by day and varies depending on the domain, it is considered that the coverage of

variant-normalization patterns can be improved by applying this unsupervised technique to the domain with no supervised data or the latest resources. Moreover, we used the acquired phrase level pattern as it is in this study. However, we can increase the recall by acquiring finer character-level patterns, as we did in Chapter3.

Error Analysis There were roughly two types of errors. The first occurred as a result of a lack of variant-normalization candidates, and the second was search errors. Example (6) shows an example of a case in which our method could not generate the correct normalized form because we could not extract the correct normalized form. Because we extract normalization candidates by phrase level, some patterns are difficult to extract as a word unit. To increase recall, we need to extract character-level and morph-level transformation patterns that occur frequently from phrase-level patterns and add them into morphological analysis and normalization. Example (7) shows an example of a case in which a normalized candidate was generated but a search failed. We will need to develop a more complicated model or introduce other features into the current model to reduce the number of search errors.

Besides the above errors, there are some errors in which correct normalization candidates were filtered. In this study, we filtered many candidates to eliminate noise. Some normalization candidates are filtered, and the correct normalization candidates cannot be generated in the word lattice. To increase recall further, we have to filter functions or calculate similarity scores more precisely. Also, some errors are associated with unknown words. Twitter data contain many unknown words such as names, and our system sometimes treats these names as other nouns. Non-standard and standard words needs to be more precisely discriminated between for higher accuracy.

4.5 Conclusion and Future Work

We introduced a new idea for extracting variant words from an unsegmented corpus and incorporated it into morphological analysis. The proposed method can effectively analyze noisy words without manual annotation. The limitation of this work is that this method is based on phonetic similarity. Although our method can extract many variant patterns, it cannot extract a pair of words that have quite low phonetic similarity. In addition, our method is based on a heuristic segmentation method for extracting normalization candidates. Though it works well in practice, we want to extend this idea for a more general framework.

In future work, we would like to increase the coverage of variant-normalization pairs. For this, we have to extract the character- and morph-level transformation patterns from the acquired phrase level variant-normalization pairs.

Chapter 5

Improving Neural Text Normalization with Data Augmentation at Character- and Morphological Levels

5.1 Introduction

Text normalization is an important fundamental technology in actual natural language processing (NLP) systems to appropriately handle texts such as those for social media. This is because social media texts contain non-standard texts, such as typos, dialects, chat abbreviations[1], and emoticons; thus, current NLP systems often fail to correctly analyze such texts (Huang, 2015; Sajjad, Darwish, and Belinkov, 2013; Han, Cook, and Baldwin, 2013). Normalization can help correctly analyze and understand these texts.

One of the most promising conventional approaches for tackling text normalizing tasks is using statistical machine translation (SMT) techniques (Junczys-Dowmunt and Grundkiewicz, 2016; Yuan and Briscoe, 2016), in particular, utilizing the Moses toolkit (Koehn et al., 2007). In recent years, encoder-decoder models with an attention mechanism (Bahdanau,

[1]short forms of words or phrases such as "4u" to represent "for you"

Cho, and Bengio, 2014) have made great progress regarding many NLP tasks, including machine translation (Luong, Pham, and Manning, 2015; Sennrich, Haddow, and Birch, 2016), text summarization (A. M. Rush, Chopra, and Weston, 2015) and text normalization (Xie et al., 2016; Yuan and Briscoe, 2016; Ikeda, Shindo, and Matsumoto, 2017). We can also simply apply an encoder-decoder model to text normalization tasks. However, it is well-known that encoder-decoder models often fail to perform better than conventional methods when the availability of training data is insufficient. Unfortunately, the amount of training data for text normalization tasks is generally relatively small to sufficiently train encoder-decoder models. Therefore, data utilization and augmentation are important to take full advantage of encoder-decoder models. Xie et al. (2016) and Ikeda, Shindo, and Matsumoto (2017) reported on improvements of data augmentation in error correction and variant normalization tasks, respectively.

Following these studies, we investigated data-augmentation methods for neural normalization. The main difference between the previous studies and this study is the method of generating augmentation data. Xie et al. (2016) and Ikeda, Shindo, and Matsumoto (2017) used simple morphological-level or character-level hand-crafted rules to generate augmented data. These predefined rules work well if we have sufficient prior knowledge about the target text-normalization task. However, it is difficult to cover all error patterns by simple rules and predefine the error patterns with certain text normalization tasks, such as dialect normalization whose error pattern varies from region to region. We propose two-level data-augmentation methods that do not use prior knowledge.

The contributions of this study are summarized as follows: (1) We propose two data-augmentation methods that generate synthetic data at character and morphological levels. (2) The experimental results with Japanese dialect text normalization demonstrate that our methods enable an encoder-decoder model, which performs poorly without data augmentation, to perform better than Moses, which is a strong baseline method when there is a small number of training examples.

5.2 Text Normalization using Encoder-Decoder Model

In this study, we focus on the dialect-normalization task as a text-normalization task. The input of this task is a dialect sentence, and the output is a “standard sentence” that corresponds to the given input dialect sentence. A “standard sentence" is written in normal form.

We model our dialect-normalization task by using a character-based attentional encoder-decoder model. More precisely, we use a single layer long short-term memory (LSTM) for both the encoder and decoder, where the encoder is bi-directional LSTM. Let $\boldsymbol{s} = (s_1, s_2, \ldots, s_n)$ be the character sequence of the (input) dialect sentence. Similarly, let $\boldsymbol{t} = (t_1, t_2, \ldots, t_m)$ be the character sequence of the (output) standard sentence. The notations n and m are the total lengths of the characters in $\boldsymbol{s}$ and $\boldsymbol{t}$, respectively. Then, the (normalization) probability of $\boldsymbol{t}$ given dialect sentence $\boldsymbol{s}$ can be written as

$$p(\boldsymbol{t}|\boldsymbol{s}, \theta) = \prod_{j=1}^{m} p(t_j|t_{<j}, \boldsymbol{s}), \tag{5.1}$$

where θ represents a set of all model parameters in the encoder-decoder model, which are determined by the parameter-estimation process of a standard softmax cross-entropy loss minimization using training data. Therefore, given θ and $\boldsymbol{s}$, our dialect normalization task is defined as finding $\boldsymbol{t}$ with maximum probability:

$$\hat{\boldsymbol{t}} = \text{argmax}_{\boldsymbol{t}}\{p(\boldsymbol{t}|\boldsymbol{s}, \theta)\}, \tag{5.2}$$

where $\hat{\boldsymbol{t}}$ represents the solution.

5.3 Proposed Methods

This section describes our proposed methods for generating augmented-data. The goal with augmented data generation is to generate a large amount of corresponding standard and dialect sentence pairs, which are then used as additional training data of encoder-decoder models. To generate augmented data, we construct a model that converts standard sentences to dialect sentences since we can easily get a lot of standard sentences. Our methods are designed based on different perspectives, namely, morphological- and character-levels.

5.3.1 Generating Augmented Data using Morphological-level Conversion

Suppose we have a small set of standard and dialect sentence pairs and a large standard sentences. First, we extract morphological conversion patterns from a (small) set of standard and dialect sentence pairs. Second, we generate the augmented data using extracted conversion patterns.

Extracting Morphological Conversion Patterns For this step, both standard and dialect sentences, which are basically identical to the training data, are parsed using a pre-trained morphological analyzer. Then, the longest difference subsequences are extracted using dynamic programming (DP) matching. We also calculate the conditional generative probabilities $p(m_s|m_t)$ for all extracted morphological conversion patterns, where m_s is a dialect morphological sequence and m_t is a standard morphological sequence. We set $p(m_s|m_t) = F_{m_s,m_t}/F_{m_t}$, where F_{m_s,m_t} is the joint frequency of (m_s, m_t) and F_{m_t} is the frequency of m_t in the extracted morphological conversion patterns of training data. Table 5.1 gives examples of extracted patterns from Japanese standard and dialect sentence pairs, which we discuss in the experimental section.

standard morphs (m_t)	dialect morphs (m_s)	$p(m_s\|m_t)$
し/ない(*shinai*)	せん (*sen*)	0.767
の/です/が (*nodesuga*)	ん/じゃ/けど (*njyakedo*)	0.553
ください (*kudasai*)	つか/あ/さい (*tukasai*)	0.517

Table 5.1 Examples of extracted morphological conversion patterns

Algorithm 1 Generating Augmented Data using Morphological Conversion Patterns

```
morphlist ← MorphAnalyse(standardsent)
newmorphlist ← []
for i = 0 ... len(morphlist) do
  sent ← CONCAT(mrphlist[i:])
  MatchedList ← CommonPrefixSearch(PatternDict, sent)
  m_si ← SAMPLE(MatchedList, P(m_s|m_t))
  newmorphlist ← APPEND(newmorphlist, m_si)
end for
synthesizedsent ← CONCAT(newmrphlist)
return synthesizedsent
```

Generating Augmented Data using Extracted Morphological Conversion Patterns After we obtain morphological conversion patterns, we generate a corresponding synthesized dialect sentence of each given standard sentence by using the extracted morphological conversion patterns. Algorithm 1 shows the detailed procedure of generating augmented data.

More precisely, we first analyze the standard sentences with the morphological analyzer. We then look up the extracted patterns for the segmented corpus from left to right and replace the characters according to probability $p(m_s|m_t)$. Table 5.2 shows an example of generated augmentation data. When we sample dialect pattern m_s from MatchedList, we use two types of $p(m_s|m_t)$. The first type is fixed probability. We set $p(m_s|m_t) = 1/\text{len(MatchedList)}$ for all matched patterns. The second type is generative probability, which is calculated from the training data (see the previous subsection). The comparison of these two types of probabilities is discussed in the experimental section.

input (standard sentence): インストールしなかった
morph sequence: インストール/し/なかっ/た/
matched conversion pattern:
(m_t, m_s) = (し/なかっ/た, せん/かった)
replaced morph sequence: インストール/せん/かった
output (augmented sentence): インストールせんかった

Table 5.2 Example of generated augmentation data using morphological conversion patterns

5.3.2 Generating Augmented Data using Character-level Conversion

For our character-level method, we take advantage of the phrase-based SMT toolkit Moses for generating augmented data. The idea is simple and straightforward; we train a 'standard-to-dialect' sentence SMT model at a character-level and apply it to a large non-annotated standard sentences. This model converts the sentence by using character phrase units. Thus, we call this method "character-level conversion".

5.3.3 Training Procedure

We use the following two-step training procedure. (1) We train model parameters by using both human-annotated and augmented data. (2) We then retrain the model parameters only with the human-annotated data, while the model parameters obtained in the first step are used as initial values of the model parameters in this (second) step. We refer to these first and second steps as "pre-training" and "fine-tuning", respectively. Obviously, the augmented data are less reliable than human-annotated data. Thus, we can expect to improve the performance of the normalization model by ignoring the less reliable augmented data in the last-mile decision of model parameter training.

method	BLEU		
	NAG	HIR	SEN
No-transformation	72.4	63.9	57.3
Moses (train)	**80.1**	72.3	67.1
Moses (train + mr:R)	75.4	71.0	64.9
Moses (train + mr:W)	80.0	73.7	67.7
Moses (train + mo)	79.9	74.3	66.9
Moses (train + mo + mr:W)	80.0	73.3	67.8
EncDec (train)	43.3	33.9	27.6
EncDec (train + mr:R)	75.3 / 63.5	69.0 / 67.3	64.2 / 58.8
EncDec (train + mr:W)	78.6 / 78.2	74.9 / 73.5	68.0 / 67.6
EncDec (train + mo)	79.1 / 79.1	74.2 / 72.9	66.9 / 65.6
EncDec (train + mo+mr:W)	**80.1** / 79.5	**75.5** / 74.6	**68.2** / 68.1

Table 5.3 BLEU scores of normalization. "/" indicates with (left) and without (right) fine tuning. 200,000 pairs of augmented data were used.

method	BLEU		
	NAG	HIR	SEN
Moses (oracle)	80.2	75.7	68.3
Moses (best)	80.1	74.3	67.8
EncDec (oracle)	84.8	81.6	73.1
EncDec (best)	80.1	75.5	68.2

Table 5.4 Evaluation of oracle sentences

5.4 Experiments

5.4.1 Data

The dialect data we used were crowdsourced data. We first prepared the standard seed sentences, and crowd workers (dialect natives) rewrote the seed sentences as dialects. The target areas of the dialects were Nagoya (NAG), Hiroshima (HIR), and Sendai (SEN), which are major cities in Japan. Each region's data consists of 22,020 sentence pairs, and we randomly split the data into training (80%), development (10%), and test (10%). For augmented data, we used the data of Yahoo chiebukuro, which contains community QA data. Since the human-annotated data are spoken language text, we used the community QA data as close-domain data.

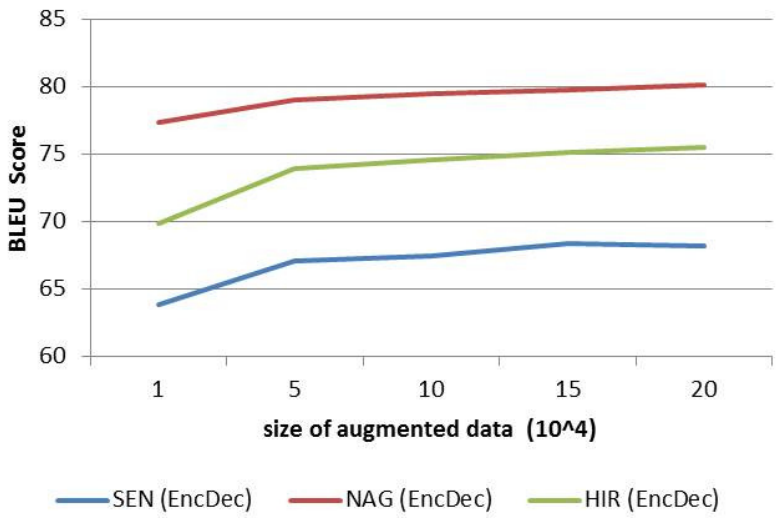

Figure 5.1 The effect of augmentation data

5.4.2 Settings

For the baseline model other than encoder-decoder models, we used Moses. Moses is a tool of training statistical machine translation and a strong baseline for the text-normalization task (Junczys-Dowmunt and Grundkiewicz, 2016). For such a task, we can ignore the word reordering; therefore, we set the distortion limit to 0. We used MERT on the development set for tuning. We confirmed that using both manually annotated and augmented data for building LM greatly degraded its final BLEU score in our preliminary experiments and used only manually annotated data as the training data of LM.

We used beam search for the encoder-decoder model (EncDec) and set the beam size to 10. When in the n beam search step, we used length normalized score $S(t, s)$, where $S(t, s)$ =log$(p(\boldsymbol{t}|\boldsymbol{s}, \theta))/|\boldsymbol{t}|$. We maximize $S(t, s)$ to find normalized sentence. We set the embedding size of the character and hidden layer to 300 and 256, respectively. We used "mrphaug (mr)" as the augmented data generated from morphological-level conversion and "mosesaug (mo)" as augmented data generated from character-level conversion (Moses). The “mr:R" and “mr:W" represent the difference in generative probability $p(m_s|m_t)$, which is used when generating augmented data; “mr:R" indicates fixed generative probability and “mr:W"

indicates weighted generative probability. For the evaluation, we used BLEU (Papineni et al., 2002), which is widely used for machine translation.

5.4.3 Results

Table 5.3 lists the normalization results. No-transformation indicates the result of evaluating input sentences without transformation. Moses achieved a reasonable BLEU score with a small amount of human-annotated data. However, the improvement of adding augmented data was limited. On the other hand, the encoder-decoder model showed a very low BLEU score with a small amount of human-annotated data. With this amount of data, the encoder-decoder model generated a sentence that was quite different from the reference. When adding augmented data, the BLEU score improved, and fine tuning was effective for all cases.

When comparing our augmented-data-generation methods, generating data according to fixed probability (mr:R) degraded the BLEU score both for Moses and the encoder-decoder model. When generating data with fixed probability, the quality of augmented data becomes quite low. However, by generating data according to generative probability (mr:W), which is estimated with training data, the BLEU score improved. This indicates that when generating data using morphological-level Conversion, it is important to take into account the generative probability. Combining “mr:W" and “mo" (train+mo+mr:W) achieves higher BLEU scores than that of other methods. This suggests that combining different types of data will have a positive effect on normalization accuracy.

When comparing three difference regions, the BLEU scores of Moses (train) and EncDec (train+mo+mr:W) for NAG (Nagoya) were the same score, while there were improvements for HIR (Hiroshima) and SEN (Sendai). It is inferred that the effect of the proposed methods for NAG were limited because the difference between input (dialect) sentences and correct (standard) sentences was small.

5.5 Discussion

The Effect of Augmented Data We examined the effect of augmented data by changing the number of augmented data. The results is shown in figure 5.1. We found that the accuracy was greatly improved by 10,000 data, and almost converged when we added about 200,000 data. To further improve the accuracy, it is necessary to create higher quality augmented data.

Oracle Analysis To investigate the further improvement on normalization accuracy, we analyzed oracle performance. We enumerated the top 10 candidates of normalized sentences from Moses and proposed method, extracted the candidates that were the most similar to the reference, and calculated the BLEU scores. Table 5.4 shows the results of oracle performance. Interestingly, the oracle performances of the encoder-decoder model with augmented data was quite high, while that of Moses was almost the same as the best score. This implies that there is room for improvement for the encoder-decoder model by just improving the decoding or ranking function.

Other text normalization task In this study, we evaluated our methods with Japanese dialect data. However, these methods are not limited to Japanese dialects because they do not use dialog-specific information. If there is prior knowledge, the combination of them will be more promising for improve normalization performance. We will investigate the effectiveness of our methods for other normalization tasks for future work.

Limitation Since our data-augmentation methods are based on human-annotated training data, the variations in the generated data depend on the amount of training data. The variations in augmented data generated with our data-augmentation methods are strictly limited within those appearing in the human-annotated training data. This essentially means that the quality of augmented data deeply relies on the amount of (human-annotated) training

data. We plan to develop more general methods that do not deeply depend on the amount of training data.

Relationship with the method of Japanese Morphological Analysis and Normalization We proposed the method of Japanese morphological analysis and normalization in Chapter3. In Chapter3, the system outputs are word segmentation, POS and normalized form. On the other hand, In this Chapter, the system output is only normalized form and it is a simpler setting. To apply the encoder-decoder model to the former setting, we need more training data. In addition, we have to consider how to handle word-level and character-level information efficiently in the encoder-decoder model.

5.6 Conclusion

We investigated the effectiveness of our augmented-data-generation methods for neural text normalization. From the experiments, the quality of augmented data greatly affected the BLEU score. Moreover, a two-step training strategy and fine tuning with human-annotated data improved this score. From these results, there is possibility to improve the accuracy of normalization if we can generate higher quality data. For future work, we will explore a more advanced method for generating augmented data.

Chapter 6

Length-controllable Abstractive Summarization by Guiding with Summary Prototype

6.1 Introduction

Neural summarization has made great progress in recent years. It has two main approaches: extractive and abstractive. Extractive methods generate summaries by selecting important sentences (Zhang, Lapata, et al., 2018; Q. Zhou et al., 2018). They produce grammatically correct summaries; however, they do not give much flexibility to the summarization because they only extract sentences from the source text. By contrast, abstractive summarization enables a more flexible summarization, and it is expected to generate more fluent and readable summaries than extractive models. The most commonly used abstractive summarization model is the pointer-generator (See, P. J. Liu, and Manning, 2017), which generates a summary word-by-word while copying words from the source text and generating words from a pre-defined vocabulary set. This model can generate an accurate summary by combining word-level extraction and generation.

Source Text

various types of renewable energy such as solar and wind are often touted as being the solution to the world 's growing energy crisis . but one researcher has come up with a novel idea that could trump them all - a biological solar panel that works around the clock . by harnessing the electrons generated by plants such as moss , he said he can create useful energy that could be used at home or elsewhere . a university of cambridge scientist has revealed his green source of energy . by using just moss he is able to generate enough power to run a clock -lrb- shown -rrb- . **he said panels of plant material could power** appliances in our homes . and the technology could help farmers grow crops where electricity is scarce . (...)

Reference Summary

university of cambridge scientist has revealed his green source of energy . by using just moss he is able to generate enough power to run a clock . he said panels of plant material could power appliances in our homes . and the tech could help farmers grow crops where electricity is scarce .

Outputs (K=10)

[**Extracted prototype**] he said panels of plant material could power in our
[**Abstractive summary**] panels of plant material could power appliances .

Outputs (K=30)

[**Extracted Prototype**] university of cambridge scientist has revealed his he said panels of plant material could power appliances in our homes and the technology could help farmers grow crops where is scarce
[**Abstractive summary**] university of cambridge scientist has revealed his green source of energy . he said panels of plant material could power appliances in our homes .

Figure 6.1 Output examples of our model. Our model extracts the top-K important words, which are colored red ($K = 10$) and blue ($K = 30$), as a prototype from the source text. It generates an abstractive summary based on the prototype and source texts. The length of the generated summary is controlled in accordance with the length of the prototype text.

Although the idea of controlling the length of the summary was mostly neglected in the past, it was recently pointed out that it is actually an important aspect of abstractive summarization (Yizhu Liu, Luo, and Zhu, 2018; Fan, Grangier, and Auli, 2018). In practical

applications, the summary length should be controllable in order for it to fit the device that displays it. However, there have only been a few studies on controlling the summary length. Kikuchi et al. (2016) proposed the length-controllable model that uses length embeddings. In the length embedding approach, the summary length is encoded either as an embedding that represents the remaining length at each decoding step or as an initial embedding to the decoder that represents the desired length. Yizhu Liu, Luo, and Zhu (2018) proposed a model that uses the desired length as an input to the initial state of the decoder. These previous models control the length in the decoding module using length embeddings. However, length embeddings only add length information on the decoder side. Consequently, they may miss important information because it is difficult to take into account which contents should be included in the summary for certain length constraints.

We propose a new length-controllable abstractive summarization by guiding with the prototype text. Figure 6.1 shows output examples generated by our model. Our idea is to introduce a word-level extractive module instead of length embeddings to control the summary length. Figure 6.2 compares the previous length-controllable models and the proposed model. The Yellow blocks are the modules responsible for length control. Since the word-level extractor controls which contents are to be included in the summary when a length constraint is given, it is possible to generate a summary including the important contents. Our model consists of two steps. First, the word-level extractor predicts the word-level importance of the source text and extracts important words according to the importance scores and the desired length. The extracted word sequence is used as a "prototype" of the summary; we call it the "prototype" text. Second, we use the prototype text as an additional input of the encoder-decoder model. The length of the summary is kept close to the length of the prototype text because the summary is generated by referring to the prototype text.

Another idea to improve summarization accuracy is to devise a new method of extractive-and-abstractive summarization. Extractive-and-abstractive summarization incorporates an extractive model in an abstractive model, and has achieved state-of-the-art accuracy in

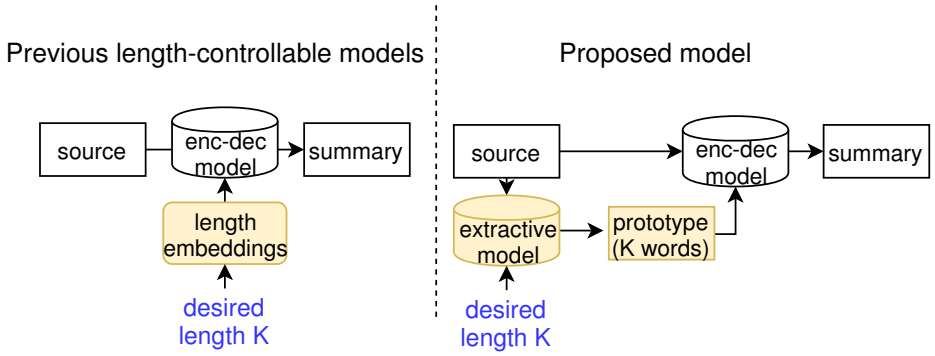

Figure 6.2 Comparison of previous length-controllable models and proposed model. Our model controls the summary length in accordance with the length of the prototype text.

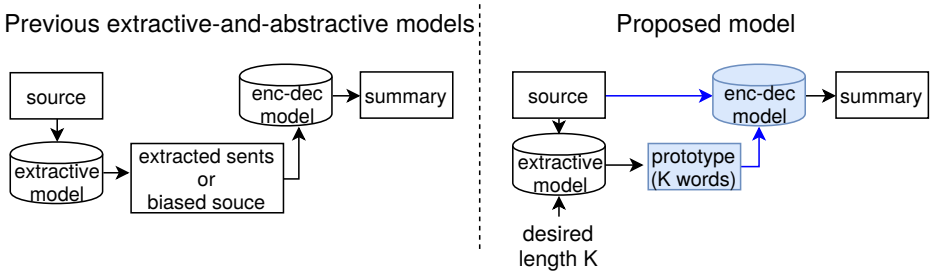

Figure 6.3 Comparison of previous extractive-and-abstractive models and proposed model. Our model jointly encodes the source and prototype texts and copies words from both texts.

abstractive summarization (Gehrmann, Deng, and A. Rush, 2018; Chen and Bansal, 2018). Since the extractive model identifies important contents to be included in the summary, the abstractive model can generate a more accurate summary by incorporating the extractive result. However, the previous methods only consider the extracted result when generating a summary and the extractive results often drop important information in the source text. On the other hand, our model generates a summary by jointly encoding the source and prototype texts and copying words from both texts. Figure 6.3 compares the previous models and the proposed model. Since the prototype text guides the summary while considering the source text, our model can generate an informative summary.

Ours is the first method that controls the summary length using an extractive module and achieves both high accuracy and length controllability in abstractive summarization.

Our contributions are summarized as follows:

- We propose a new length-controllable prototype-guided abstractive summarization model, called LPAS. Our model effectively guides the abstractive summarization using a summary prototype. Our model controls the summary length by controlling the number of words in the prototype text.
- Our model achieved state-of-the-art ROUGE scores in standard and length-controlled abstractive summarization settings on the CNN/DM and NEWSROOM datasets.
- We verified the effectiveness of BERT (Devlin et al., 2018) as a prototype extractor. This is the first study to incorporate BERT in extractive-and-abstractive summarization.

6.2 Task Definition

Our study defines length-controllable abstractive summarization as two pipelined tasks: prototype extraction and prototype-guided abstractive summarization. The problem formulations of each task are described below.

Task 1 (Prototype Extraction)**.** *Given a source text* X^C *with* L *words* $X^C = (x_1^C, \ldots, x_L^C)$ *and desired summary length* K*, the model estimates importance scores* $P^{\text{ext}} = (p_1^{\text{ext}} \ldots p_L^{\text{ext}})$ *and extracts the top-*K *important words* $X^P = (x_1^P, \ldots, x_K^P)$ *as a prototype text on the basis of* P^{ext}*. The desired summary length* K *can be set to an arbitrary value. Note that the original word order is preserved in* X^P *(*X^P *is not bag-of-words). This is because the expressions in the original text are often used as they are in the generated summaries, and it is considered that the expressions in the original document can be reused more efficiently by maintaining the original word order.*

Task 2 (Prototype-guided Abstractive Summarization)**.** *Given the source text and the extracted prototype text* X^P*, the model generates a length-controlled abstractive summary* Y =

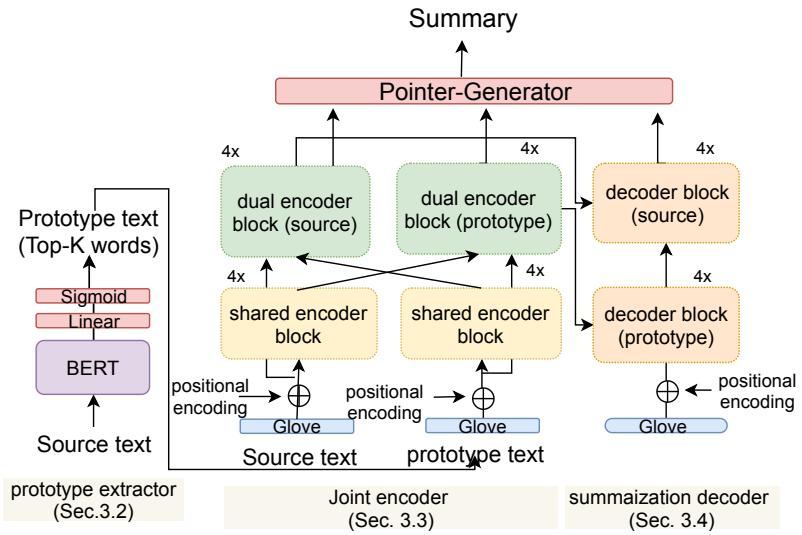

Figure 6.4 Architecture of proposed model

$(y_1, \ldots, y_T)$. *The length of summary* T *is controlled in accordance with the prototype length* K.

6.3 Proposed Model

6.3.1 Overview

Our model consists of three modules: the prototype extractor, joint encoder, and summary decoder (Figure 6.4). The last two modules comprise Task 2, the prototype-guided abstractive summarization. The prototype extractor uses BERT, and the joint encoder, and summary decoder uses the Transformer architecture (Vaswani et al., 2017).

Prototype extractor (§6.3.2)

The prototype extractor extracts the top-K important words from the source text.

Joint encoder (§6.3.3)

The joint encoder encodes both the source text and the prototype text.

Summary decoder (§6.3.4)

The summary decoder is based on the pointer-generator model and generates an abstractive summary by using the output of the joint encoder.

6.3.2 Prototype Extractor

Since our model extracts the prototype at the word level, the prototype extractor estimates the importance score p_l^{ext} of each word $x_l^C \in X^C$. BERT has achieved SOTA on many classification tasks, so it is a natural choice for the prototype extractor. Our model uses BERT and a task-specific feed-forward network on top of BERT. We tokenize the source text using the BERT tokenizer[1] and fine-tune the BERT model. The importance score p_l^{ext} is defined as

$$p_l^{\text{ext}} = \sigma(W_1^\top \text{BERT}(X^C)_l + b_1) \tag{6.1}$$

where BERT() is the last hidden state of the pre-trained BERT. $W_1 \in \mathcal{R}^{d_{bert}}$ and b_1 are learnable parameters. σ is a sigmoid function. d_{bert} is the dimension of the last hidden state of the pre-trained BERT.

To extract a more fluent prototype than when using only the word-level importance, we define a new weighted importance score $p_l^{\text{ext}_w}$ that incorporates a sentence-level importance score as a weight for the word-level importance score:

$$p_l^{\text{ext}_w} = p_l^{\text{ext}} \cdot p_{S_j}^{\text{ext}}, \quad p_{S_j}^{\text{ext}} = \frac{1}{N_{S_j}} \sum_{l:x_l \in S_j} p_l^{\text{ext}} \tag{6.2}$$

where N_{S_j} is the number of words in the j-th sentence $S_j \in X^C$. Our model extracts the top-K important words as a prototype from the source text on the basis of $p_l^{\text{ext}_w}$. It controls the length of the summary in accordance with the number of words in the prototype text, K.

[1] https://github.com/google-research/bert/

6.3.3 Joint Encoder

Embedding layer

First, this layer projects each of the one-hot vectors of words x_l^C (of size V) into a d_{word}-dimensional vector space with a pre-trained weight matrix $W^e \in \mathcal{R}^{d_{word} \times V}$ such as GloVe (Pennington, Socher, and Manning, 2014). Then, the word embeddings are mapped to d_{model}-dimensional vectors by using the fully connected layer, and the mapped embeddings are passed to a ReLU function. This layer also adds positional encoding to the word embedding (Vaswani et al., 2017).

Transformer encoder blocks

The encoder encodes the embedded source and prototype texts with a stack of Transformer blocks (Vaswani et al., 2017). Our model encodes the two texts with the encoder stack independently. We denote these outputs as $E_s^C \in \mathcal{R}^{d_{model} \times L}$ and $E_s^P \in \mathcal{R}^{d_{model} \times K}$, respectively.

Transformer dual encoder blocks

This block calculates the interactive alignment between the encoded source and prototype texts. Specifically, it first encodes the source and prototype texts and performs multi-head attention on the other output of the encoder stack (i.e., E_s^C and E_s^P). This component is quite similar to a decoder block. The difference between an original Transformer decoder block and this block is that this component does not use the subsequent mask for self-attention since, unlike decoder, a source text and a prototype text are given in advance.

We denote the outputs of the dual encoder stack of the source text and prototype text by $M^C \in \mathcal{R}^{d_{model} \times L}$ and $M^P \in \mathcal{R}^{d_{model} \times K}$, respectively.

6.3.4 Summary Decoder

Embedding layer

The decoder receives a sequence of words in an abstractive summary Y, which is generated through an auto-regressive process. At each decoding step t, this layer projects each of the one-hot vectors of the words y_t in the same way as the embedding layer in the joint encoder.

Transformer decoder blocks

The decoder uses a stack of decoder Transformer blocks (Vaswani et al., 2017) that perform the multi-head attention on the encoded representations of the prototype, M^P. It uses another stack of decoder Transformer blocks that perform the multi-head attention on those of the source text, M^C, on top of the first stack. The first one rewrites the prototype text, and the second one complements the rewritten prototype with the original source information. The subsequent mask is used in the stacks since this component is used in a step-by-step manner at test time. The output of the stacks is $M^S \in \mathcal{R}^{d_{model} \times T}$.

Copying mechanism

Our pointer-generator model copies the words from the source and prototype texts on the basis of the copy distributions, for efficient reuse.

Copy distributions

The copy distributions of the source and prototype words are described as follows:

$$p_p(y_t) = \sum_{k:x_k^P = y_t} \alpha_{tk}^P, \quad p_c(y_t) = \sum_{l:x_l^C = y_t} \alpha_{tl}^C$$

where α_{tk}^P and α_{tl}^C are respectively the first attention heads of the last block in the first and second stacks of the decoder.

Final vocabulary distribution

The final vocabulary distribution is described as follows:

$$
\begin{aligned}
p(y_t) &= \lambda_g p_g(y_t) + \lambda_c p_c(y_t) + \lambda_p p_p(y_t) \\
\lambda_g, \lambda_c, \lambda_p &= \mathrm{softmax}(W^v[M_t^S; c_t^C; c_t^P] + b^v) \\
c_t^C &= \sum_l \alpha_{tl}^C M_l^C, \quad c_t^P = \sum_k \alpha_{tk}^P M_k^P \\
p_g(y_t) &= \mathrm{softmax}(W^g(M_t^S) + b^g)
\end{aligned}
$$

where $W^v \in \mathcal{R}^{3 \times 3d_{model}}$, $b^v \in \mathcal{R}^3$, $W^g \in \mathcal{R}^{d_{model} \times V}$ and $b^g \in \mathcal{R}^V$ are learnable parameters.

6.4 Training

Our model is not trained in an end-to-end manner: the prototype extractor is trained first and then the encoder and decoder are trained.

6.4.1 Generating Training Data

Prototype extractor

Since there are no supervised data for the prototype extractor, we created pseudo training data like in (Gehrmann, Deng, and A. Rush, 2018). The training data consists of word x_l^C and label r_l pairs, (x_l^C, r_l) for all $x_l^C \in X^C$. r_l is 1 if x_l^C is included in the summary; otherwise it is 0. To construct the paired data automatically, we first extract oracle source sentences S^{oracle} that maximize the ROUGE-R score in the same way as in (Hsu et al., 2018). Then, we calculate the word-by-word alignment between the reference summary and S^{oracle} using a dynamic programming algorithm to consider the word order. Finally, we label all aligned words with 1 and other words, including the words that are not in the oracle sentence, with 0.

Joint encoder and summary decoder

We have to create triple data of (X^C, X^P, Y), consisting of the source text, the gold prototype text, and the target text, for training our encoder and decoder. We use the top-K words (in terms of $p_l^{\text{ext}_w}$; Eq. 6.2) in the oracle sentences S^{oracle} as the gold prototype text to extract a prototype closer to the reference summary and improve the quality of the encoder-decoder training.

How to determine the K value? K is determined using the reference summary length T. To obtain a natural summary close to the desired length, we quantize the length T into discrete bins, where each bin represents a size range. We set the size range to 5 in this study. That is, the value nearest to the summary length T among multiples of 5 is selected for K. At the time of training, by associating the length of the reference summary with the length of the prototype, the length of the prototype and that of the generated summary become close. As a result, the length of the summary can be controlled by selecting arbitrary K during the test. It is the main point of this study to select the important word according to the summary length to generate the length-controlled summary.

6.4.2 Loss Function

Prototype extractor

We use the binary cross-entropy loss, because the extractor estimates the importance score of each word (Eq. 6.1), which is a binary classification task.

$$L_{\text{ext}} = -\frac{1}{NL}\sum_{n=1}^{N}\sum_{l=1}^{L}\begin{pmatrix} r_l \log p_l^{\text{ext}} + \\ (1-r_l)\log(1-p_l^{\text{ext}}) \end{pmatrix}$$

where N is the number of training examples.

Joint encoder and summary decoder

The main loss for the encoder-decoder is the cross-entropy loss:

$$L_{\text{gen}}^{\text{main}} = -\frac{1}{NT}\sum_{n=1}^{N}\sum_{t=1}^{T}\log p(y_t|y_{1:t-1}, X^C, X^P).$$

Moreover, we add the attention guide loss of the summary decoder. This loss is designed to guide the estimated attention distribution to the reference attention.

$$L_{\text{attn}}^{\text{sum}} = -\frac{1}{NT}\sum_{n=1}^{N}\sum_{t=1}^{T}\log \alpha_{t,l(t)}^{C}$$
$$L_{\text{attn}}^{\text{proto}} = -\frac{1}{NT}\sum_{n=1}^{N}\sum_{t=1}^{T}\log \alpha_{t,l(t)}^{\text{proto}}$$

$\alpha_{t,l(t)}^{\text{proto}}$ is the first attention head of the last block in the joint encoder stack for the prototype. $l(t)$ denotes the absolute position in the source text corresponding to the t-th word in the sequence of summary words. The overall loss of the generation model is a linear combination of these three losses.

$$L_{\text{gen}} = L_{\text{gen}}^{\text{main}} + \lambda_1 L_{\text{attn}}^{\text{sum}} + \lambda_2 L_{\text{attn}}^{\text{proto}}$$

λ_1 and λ_2 were set to 0.5 in the experiments.

6.5 Inference

During the inference period, we use a beam search and re-ranking (Chen and Bansal, 2018). We keep all N_{beam} summary candidates provided by the beam search, where N_{beam} is the size of the beam, and generate the N_{beam}-best summaries. The summaries are then re-ranked by the number of repeated N-grams, the smaller the better. The beam search and this re-ranking improve the ROUGE score of the output, as they eliminate candidates that contain

repetitions. For the length-controlled setting, we set the value of K to the desired length. For the standard setting, we set it to the average length of the reference summary in the validation data.

6.6 Experiments

6.6.1 Datasets and settings

Dataset

We used the CNN/DM dataset (Hermann et al., 2015), a standard corpus for news summarization. The summaries are bullet points for the articles shown on their respective websites. Following See, P. J. Liu, and Manning (2017), we used the non-anonymized version of the corpus and truncated the source documents to 400 tokens and the target summaries to 120 tokens. The dataset includes 286,817 training pairs, 13,368 validation pairs, and 11,487 test pairs. We also used the NEWSROOM dataset (Grusky, Naaman, and Artzi, 2018). NEWSROOM contains various news sources (38 different news sites). We used 973,042 pairs of data for training. We sampled 30,000 pairs for validation data, and the number of the test pairs was 106,349. To evaluate the length-controlled settings for NEWSROOM dataset, we randomly sampled 10,000 samples from the test set.

6.7 Model Configurations

We used the same configurations for the two datasets. The prototype extractor used the pre-trained BERT large model (Devlin et al., 2018). We fine-tuned BERT for two epochs. The default settings were used for the other parameters for fine-tuning.[2]. The joint encoder and summarization decoder model used pre-trained 300-dimensional GloVe embeddings. The

[2] https://github.com/google-research/bert/

model size of the Transformer d_{model} was set to 512; the Transformer had four transformer blocks for the joint encoder and summarization decoder. The number of heads was 8, and the number of dimensions of the feed-forward network was 2048. We set the dropout rate to 0.2. For optimization, we used the Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = e^{-9}$. We varied the learning rate over the training, according to Vaswani et al. (2017). We set the warm-up steps to 8000. We set the size of the input vocabulary to 100,000 and the output vocabulary to 1,000.

6.7.1 Evaluation Metrics

We used the ROUGE scores (F1), including ROUGE-1 (R-1), ROUGE-2 (R-2), and ROUGE-L (R-L), as the evaluation metrics (Lin, 2004). We used the files2rouge toolkit for calculating the ROUGE scores[3].

6.7.2 Results

Does our model outperform previous models in the standard setting?

Table 6.1 shows the results of our model and the other models that do not consider the length constraint. Models 2-7 are previous models based on the extractive-and-abstractive approach. Model 8 is the latest model that is based on a pre-trained sequence-to-sequence model. From these results, our models outperformed the models in terms of ROUGE-1, 2 and L. Although the simple Transformer-based model (LPAS w/o Prototype) had high ROUGE scores, our model improved on it by about 1.5 points.

We also examined the results of generating a summary from only the prototype (LPAS w/o Source). Here, using only the prototype, turned out to have the same accuracy as using only the source, but the model using the source and the prototype simultaneously had higher accuracy. These results indicate that our prototype extraction and joint encoder effectively

[3] https://github.com/pltrdy/files2rouge

Model	R-1	R-2	R-L
Pointer-Generator[1]	36.44	15.66	33.42
Pointer-Generator + Coverage[1]	39.53	17.28	36.38
Key information guide network[2]	38.95	17.12	35.68
Unified summarization[3]	40.68	17.97	37.13
Sentence-rewriting[4]	40.88	17.80	38.54
Bottom-Up[5]	41.22	18.68	38.34
EXCONSUMM Compressive[6]	40.9	18.0	37.4
ETADS [7]	41.75	19.01	38.89
PoDA[8]	41.87	19.27	38.54
LPAS	**42.55**	**20.09**	**39.36**
w/o Prototype	40.71	18.43	37.32
w/o Source	40.08	18.32	37.08

Table 6.1 ROUGE scores (F1) of abstractive summarization models on CNN/DM. [1](See, P. J. Liu, and Manning, 2017); [2](Chenliang Li et al., 2018); [3](Hsu et al., 2018); [4](Chen and Bansal, 2018); [5](Gehrmann, Deng, and A. Rush, 2018); [6](Mendes et al., 2019);[7](You et al., 2019); [8](L. Wang et al., 2019). LPAS w/o Prototype denotes a simple Transformer-based pointer-generator, which is our model without the prototype extractor and the joint encoder. LPAS w/o Source denotes a model that generates a summary only from the prototype text.

Model	R-1	R-2	R-L
w/ dual encoder block	41.90	19.61	38.85
w/o dual encoder block	41.46	19.24	38.30

Table 6.2 Comparison between with and without dual encoder block

incorporated the source text and prototype information and contributed to improving the accuracy.

We compare the result with and without the dual encoder block in table 6.2. Note that this result is the preliminary experiment and the details of parameter settings differ from those in Table 6.1. This result indicates that the dual encoder block contributed to improving the accuracy of summarization.

Length	Model	R-1	R-2	R-L
10	LC[1]	**19.03**	8.45	16.47
	LenEmb	18.19	**8.96**	**17.44**
	LPAS	17.43	8.87	16.78
30	LC	32.26	13.60	24.64
	LenEmb	34.01	15.51	31.43
	LPAS	**35.11**	**17.21**	**32.83**
50	LC	34.71	14.24	25.62
	LenEmb	38.66	17.17	35.49
	LPAS	**41.47**	**19.70**	**38.46**
70	LC	33.83	13.67	24.67
	LenEmb	39.57	17.38	36.22
	LPAS	**42.48**	**19.97**	**39.25**
90	LC	32.17	13.00	23.28
	LenEmb	38.51	16.79	35.24
	LPAS	**41.54**	**19.43**	**38.30**
AVG	LC	30.40	12.59	22.94
	LenEmb	33.79	15.16	31.16
	LPAS	**35.60**	**17.04**	**33.12**

Table 6.3 ROUGE scores (F1) of abstractive summarization models with different lengths on the CNN/DM dataset (10, 30, 50, 70, 90 words). AVG indicates the average ROUGE score for the five different lengths. [1](Yizhu Liu, Luo, and Zhu, 2018)

Does our model improve the ROUGE score in the length-controlled setting?

We used two types of length-controllable models as baselines. The first one is a CNN-based length-controllable model (LC) that uses the desired length as an input to the initial state of the CNN-based decoder. (Yizhu Liu, Luo, and Zhu, 2018). The second one (LenEmb) embeds the remaining length and adds them to each decoder step (Kikuchi et al., 2016). Since there are no previous results on applying LenEmb to the CNN/DM dataset, we implemented it as a Transformer-based encoder-decoder model. Specifically, we simply added the embeddings of the remaining length to the word embeddings at each decoding step. Table 6.3 shows that our model achieved high ROUGE scores for different lengths and outperformed the previous length-controllable models in most cases. Our model was about 2 points more

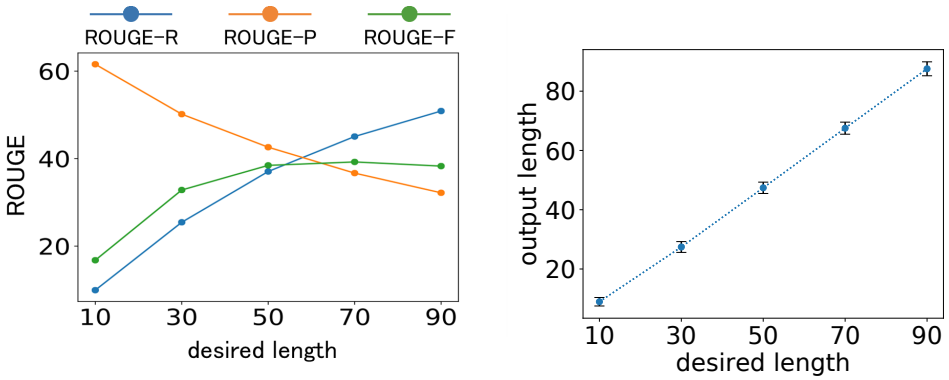

Figure 6.5 Results in the length-controlled setting on CNN/DM. a): ROUGE-L recall, precision and F scores for different lengths (left). b): Output length distribution (right).

accurate on average than LenEmb. Our model selects the most important words from the source text in accordance with the desired length. It is thus effective at keeping the important information even in the length-controlled setting. Figure 6.5a shows the precision, recall, and F score of ROUGE for different lengths. Our model maintains a high F-score around the average length (around 60 words); this indicates that it can select important information and generate stable results with different lengths.

Does our model generate a summary with the desired length?

Figure 6.5b shows the relationship between the desired length and the output length. The x-axis indicates the desired length, and the y-axis indicates the average length and standard deviation of the length-controlled output summary. The results show that our model properly controls the summary length. This controllable nature comes from the training procedure. When training our encoder-decoder, we set the number of words K in the prototype text according to the length of the reference summary; therefore, the model learns to generate a summary that has a similar length to the prototype text.

	R-1	R-2	R-L
Lead3	40.3	17.7	36.6
Bottom-Up (top-3 sents)[1]	40.7	18.0	37.0
Bottom-Up (word)[1]	42.0	15.9	37.3
NeuSum[2]	41.6	19.0	38.0
BertSum[3]	43.25	20.24	**39.63**
HIBERT[4]	42.37	19.95	38.83
LPAS-ext			
- top-3 sents	41.48	19.23	37.76
- Top-K words	**44.79**	**20.59**	38.12

Table 6.4 ROUGE scores (F1) of our prototype extractor (LPAS-ext) on CNN/DM. [1](Gehrmann, Deng, and A. Rush, 2018); [2](Q. Zhou et al., 2018); [3](Yang Liu, 2019); [4](Zhang, Wei, and M. Zhou, 2019)

How good is the quality of the prototype text?

To evaluate the quality of the prototype, we evaluated the ROUGE scores of the extracted prototype text. Table 6.4 shows the results. In the table, LPAS-ext (top-3 sents) means the top-three sentences were extracted using $p^{\text{ext}}_{S_j}$. Interestingly, ROUGE-1 and ROUGE-2 scores of the LPAS-ext (Top-K words) were higher than those of the sentence-level extractive models. It indicates that word-level LPAS-ext is effective at finding not only important words (ROUGE-1), but also important phrases (ROUGE-2). Also, we can see from Table 6.1 that whole LPAS improved the ROUGE-L score of LPAS-ext. This indicates that our joint encoder and summary decoder generate more fluent summaries with the help of the prototype text.

Does our abstractive model improve if the quality of the prototype is improved?

We evaluated our model in the following two settings in order to analyze the relationship between the quality of the abstractive summary and that of the prototype. In the gold-length setting, we only gave the gold length K to the prototype extractor. In the gold sentences + the gold-length setting, we gave the gold sentences S^{oracle} and gold length (see 6.4.1). Table 6.5 shows the results. These results indicate that selecting the correct number

	R-1	R-2	R-L
Average length	42.55	20.09	39.36
Gold length	43.23	20.46	40.00
Gold sentences + Gold length	46.68	23.52	43.41

Table 6.5 ROUGE scores (F1) of abstractive summarization models with gold settings on the CNN/DM dataset.

	R-1	R-2	R-L
Lead3 [1]	32.02	21.08	29.59
pointer-generator [1]	27.54	13.32	23.50
LPAS			
K = average length	39.24	27.20	35.84
K = domain length	**39.79**	**27.85**	**36.48**
LPAS (w/o Prototype)	38.48	26.99	35.30

Table 6.6 ROUGE scores (F1) of proposed models on NEWSROOM dataset. [1](Grusky, Naaman, and Artzi, 2018)

of words in the prototype improves the ROUGE scores. In this study, we simply selected the average length when extracting the prototype for all examples in the standard setting; however, there will be an improvement if we adaptively select the number of words in the prototype for each source text. Moreover, the ROUGE score largely improved in the gold sentence and gold-length settings. This indicates that the quality of the generated summary will significantly improve by increasing the accuracy of the extractive model.

Is our model effective on other datasets?

To verify the effectiveness of our model on various other summary styles, we evaluated it on a large and varied news summary dataset, NEWSROOM. The results of the NEWSROOM dataset at standard settings are shown in Table 6.6. To consider differences in summary length between news domains, we evaluated our model in the average length and domain-level average length (denoted as domain length) settings. We used the validation dataset to calculate the average length. The results indicate that our model had significantly higher

Length	Model	R-1	R-2	R-L
10	LenEmb	**22.99**	13.42	21.45
	LPAS	22.80	**13.91**	**21.59**
30	LenEmb	37.49	25.67	34.26
	LPAS	**39.22**	**27.33**	**35.95**
50	LenEmb	36.91	25.50	33.86
	LPAS	**38.57**	**27.07**	**35.44**
70	LenEmb	33.52	23.02	30.90
	LPAS	**35.29**	**24.72**	**32.62**
90	LenEmb	30.04	20.49	27.80
	LPAS	**31.53**	**22.03**	**29.30**
AVG	LenEmb	32.19	21.62	29.66
	LPAS	**33.48**	**23.01**	**30.98**

Table 6.7 ROUGE scores (F1) of abstractive summarization models with different lengths on the NEWSROOM dataset.

ROUGE scores compared with the official baselines and outperformed our baseline (LPAS w/o Prototype). They also indicate that our model is effective on datasets containing text in various styles. Moreover, we found that considering the domain length has positive effects on the ROUGE scores. This indicates that our model can easily reflect the differences in summary length among various styles. Table 6.7 and Figure 6.6 show the results in the length-controlled setting for NEWSROOM. Our model achieved higher ROUGE scores than those of LenEmb. From Figure 6.6a, we can see that the F-value of the ROUGE score was highest around 30 words. It is because the average word number is about 30 words. Moreover, Figure 6.6b shows that our model also acquired the length control capability for the dataset with various styles.

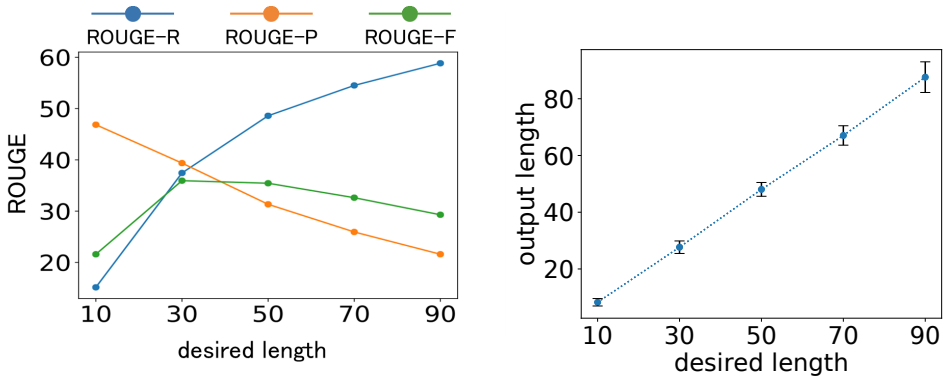

Figure 6.6 Results in the length-controlled setting on NEWSROOM. a): ROUGE-L recall, precision and F scores for different lengths (left). b): Output length distribution (right).

6.8 Examples of the Prototype text and Generated Summary

Table 6.8 and Table 6.9 list some summary examples. We can see that our prototype extractor picks up the important words and our encoder-decoder outputs fluent sentences by using the Prototype text. Moreover, from these results, we can see that the output of the prototype extractor contains many ungrammatical phrases whereas the outputs of our encoder-decoder are relatively grammatical.

6.9 Related Work and Discussion

Length control for summarization

Kikuchi et al. (2016) were the first to propose using length embedding for length-controlled abstractive summarization. Fan, Grangier, and Auli (2018) also used length embeddings at the beginning of the decoder module for length control. Yizhu Liu, Luo, and Zhu (2018) proposed a CNN-based length-controllable summarization model that uses the desired length as an input to the initial state of the decoder. Takase and Okazaki (2019) introduced positional encoding that represents the remaining length at each decoder step of Transformer-

Source text (truncated)	a florida community has voiced objection to plans of a forensic research ‘ body farm ’ in which human bodies will be left in florida elements for extended periods of time . the body farm , which would be located in lithia , is a joint project between the university of south florida institute of anthropology and the hillsborough county sheriff ’s office . it has the potential to offer forensic scientists and law-enforcement investigators a chance to see what happens to the bodies when they are left in florida elements . five to ten bodies from usf ’s after life body donation program would be buried or placed on two acres of the walter c. heinrich practical training center in lithia , which is a 230-acre plot of land bordered by a landfill and county property . scroll down for video . university of south florida , in a joint project with the hillsborough county sheriff ’s office , have proposed building a ‘ body farm ’ in lithia for research into how florida elements affect cadavers . if the research is approved by the hillsborough county commission , the information gathered at the site could help investigators solve crimes and cold cases . hillsborough county commissioner stacy white has taken a stand against the facility , while state attorney general pam bondi and hillsborough county sheriff david gee have supported the project .
Reference summary	the body farm would be located in lithia in hillsborough county . it is a joint project between the university of south florida institute of anthropology and the hillsborough county sheriff ’s office . residents fear it will bring unwanted predators and a strong odor to area . usf associate professor erin kimmerle said the project could help investigators solve cold cases and other crimes . there are more than 500 cold cases in hillsborough , pasco and pinellas counties - areas close to the research facility .’
Prototype text (10 words)	farm university of south five bodies would be of south
Generated summary	body farm would be located in lithia .
Prototype text (30 words)	the body farm is project university of south florida of and five to ten bodies would be buried on training center in lithia university of south florida body farm elements
Generated summary	the body farm is a joint project between the university of south florida institute of anthropology and the hillsborough county sheriff ’s office .
Prototype text (50 words)	the body farm would be is a joint project the university of south florida institute of anthropology and the hillsborough county ’s five to ten bodies from would be buried or on two acres training center in lithia university of south florida ’s ‘ body farm ’ in florida elements
Generated summary	the body farm is a joint project between the university of south florida institute of anthropology and the hillsborough county sheriff ’s office . five to ten bodies would be buried or placed on two acres of the walter c. heinrich practical training center in lithia .

Table 6.8 Example of prototype texts and generated summaries in CNN/DM dataset.

Source text	a man sought on a felony arrest warrant unsuccessfully used a motorhome to escape alaska state troopers in a high speed chase caught on video . troopers say 49-year-old eligah christian was taken into custody friday after mashing the bulky vehicle into several patrol cars . earlier friday , an officer spotted christian driving a 2014 motorhome near wasilla . he was being sought on a $ 100,000 warrant on charges of scheming to defraud , 15 counts of theft and 21 counts of issuing bad checks . high speed chase : the driver of this motorhome took alaska state troopers on a high speed chase outside sarah palin 's home of wasilla on friday . fugitive : eligah christian , 49 , took off when police tried to pull him over . he was being sought on a $ 100,000 warrant on charges of scheming to defraud , 15 counts of theft and 21 counts of issuing bad checks . the officer turned on the patrol car lights and siren , but christian failed to stop and started driving recklessly , reports ktuu . ‘ christian failed to yield to emergency lights and siren and began to recklessly travel south on church road towards the parks highway where he turned northbound , ’ a trooper wrote . troopers deployed spike strips to stop the motorhome , but the vehicle struck patrol cars as the suspect tried turning around . ‘ he began to turn the motorhome around striking several patrol vehicles , ’ a trooper continued . ‘ christian was stopped at a residence off pittman road and was placed into custody . ’ christian was charged with felony reckless driving and criminal mischief . he remained jailed sunday . brought to a halt : troopers deployed spike strips to stop the motorhome , but the vehicle struck patrol cars as the suspect tried turning around .
Reference Summary	eligah christian , 49 , was taken into custody friday after his unsuccessful bid to elude capture near sarah palin 's home of wasilla . christian was being sought on a $ 100,000 warrant on charges of scheming to defraud , 15 counts of theft and 21 counts of issuing bad checks .
Prototype text (10 words) Generated summary	eligah christian was the into patrol to defraud 15 theft eligah christian was taken into custody friday .
Prototype text (30 words) Generated summary	49-year-old eligah christian was taken into custody friday after the vehicle into several patrol cars $ 100,000 scheming to defraud , 15 counts of theft and 21 issuing bad checks eligah christian , 49 , sought on $ 100,000 warrant on charges of scheming to defraud , 15 counts of theft and 21 counts of issuing bad checks .
Prototype text (50 words) Generated summary	troopers 49-year-old eligah christian was taken into custody friday after mashing the bulky vehicle into several patrol cars driving a wasilla on a $ 100,000 warrant on charges of scheming to defraud , 15 counts of theft and 21 counts of issuing bad checks felony reckless driving and criminal mischief eligah christian , 49 , was being sought on a $ 100,000 warrant on charges of scheming to defraud , 15 counts of theft and 21 counts of issuing bad checks . christian was charged with felony reckless driving and criminal mischief . he remained jailed sunday .

Table 6.9 Example of prototype texts and generated summaries in CNN/DM dataset.

based encoder-decoder model. It is almost equivalent to the model LenEmb we implemented. These previous models use length embeddings for controlling the length in the decoding module, whereas we use the prototype extractor for controlling the summary length and to include important information in the summary.

Neural extractive-and-abstractive summarization

Hsu et al. (2018), Gehrmann, Deng, and A. Rush (2018) and You et al. (2019) incorporated a sentence- and word-level extractive model in the pointer-generator model. Their models weight the copy probability for the source text by using an extractive model and guide the pointer-generator model to copy important words. Chenliang Li et al. (2018) proposed a keyword guided abstractive summarization model. Chen and Bansal (2018) proposed a sentence extraction and re-writing model that trains in an end-to-end manner by using reinforcement learning. Cao et al. (2018) proposed a search and rewrite model. Mendes et al. (2019) proposed a combination of sentence-level extraction and compression model. The idea behind these models is word-level weighting for the entire source text or sentence-level re-writing. On the other hand, our model guides the summarization with length-controllable prototype text by using the prototype extractor and joint encoder. Utilizing extractive results to control the length of the summary is a new idea.

Large-scale pre-trained language model

BERT (Devlin et al., 2018) is a new pre-trained language model that uses bidirectional encoder representations from Transformer. BERT has performed well in many natural language understanding tasks such as the GLUE benchmarks (A. Wang et al., 2018) and natural language inference (Williams, Nangia, and Bowman, 2018). Yang Liu (2019) used BERT for sentence-level extractive summarization model. Zhang, Wei, and M. Zhou (2019) trained a new pre-trained model that considers document-level information for sentence-level extractive summarization. L. Wang et al. (2019) proposed sequence-to-sequence based pre-trained

model and incorporated it in an abstractive summarization model. We used BERT for the word-level prototype extractor and verified the effectiveness of using a BERT in the word-level extractive module.

Reinforcement learning for summarization

Reinforcement learning (RL) is a key summarization technique. RL can be used to optimize non-differential metrics or multiple non-differential networks. Narayan, Cohen, and Lapata (2018) and Dong et al. (2018) used RL for extractive summarization. For abstractive summarization, Paulus, Xiong, and Socher (2017) used RL to mitigate the exposure bias of abstractive summarization. Chen and Bansal (2018) used RL to combine sentence-extraction and pointer-generator models. Our model achieved high ROUGE scores without RL. In future, we may incorporate RL in our models to get a further improvement.

6.10 Conclusion

We proposed a new length-controllable abstractive summarization model. Our model consists of a word-level prototype extractor and a prototype-guided abstractive summarization model. The prototype extractor identifies the important part of the source text within the length constraint, and the abstractive model is guided with the prototype text. This characteristic enabled it to achieve a high ROUGE score in standard summarization tasks. Moreover, our prototype extractor ensures the summary will have the desired length. Experiments with the CNN/DM dataset and the NEWSROOM dataset show that our model outperformed previous models in standard and length-controlled settings. In future, we would like to incorporate a pre-trained language model in the abstractive model to build a higher quality summarization model.

Chapter 7

Conclusion

7.1 Summary

In this dissertation, we focused on three problems of converting sentences to readable sentences: word-level normalization and morphological analysis, sentence-level normalization, document-level summarization.

For the first problem, word-level normalization morphological analysis, we proposed a novel joint estimation method of word normalization and morphological analysis in Chapter 3. We automatically generate both character-level and word-level normalization candidates to expand the word lattice and use discriminative methods to formulate a cost function. Experimental results show that the proposed method achieved higher accuracy and recall for word segmentation, POS tagging and normalization than those of conventional systems.

In addition, in Chapter 4, we proposed a novel method for automatically extracting pairs of a variant word and its normal form from unannotated text on the basis of a pair-wise similarity approach. We incorporated the acquired variant-normalization pairs into Japanese morphological analysis. The experimental results show that our method can extract widely covered variants from large Twitter data and improve the recall of normalization without degrading the overall accuracy of Japanese morphological analysis.

For the second problem, sentence-level normalization, we proposed simple but effective methods of data augmentation for encoder-decoder-based neural normalization models in Chapter 5. In this study, we propose two methods for generating augmented data. The experimental results with Japanese dialect normalization indicate that our methods are effective for an encoder-decoder model and achieve higher BLEU score than that of baselines. We also investigated the oracle performance and revealed that there is sufficient room for improving an encoder-decoder model.

For the third problem, document-level summarization, propose a new prototype-guided length-controllable abstractive summarization model in Chapter 6. We incorporate a word-level extractive module in the encoder-decoder model instead of length embeddings. Our model first extracts a prototype text and generates a summary by jointly encoding and copying words from both the prototype text and source text. Since the prototype text is a guide to both the contents and length of the summary, our model can generate an informative and length-controlled summary. Experiments with the CNN/Daily Mail dataset and the NEWSROOM dataset show that our model outperformed previous models in standard and length-controlled settings.

7.2 Future Directions

7.2.1 An Unified Model of Normalization and Summarization Tasks

This dissertation deals with the normalization of noisy words and sentences and summarization of long sentences. In the future, the combined task of normalization and summarization, such as summarizing long texts that contain noisy expressions, is also an essential topic for practical application. For example, there is a need for meeting summarization. In this task, input texts contain a lot of spoken language and automatic speech recognition errors, and it is a challenging task than previous tasks.

7.2.2 Integration with Unsupervised Language Models

BERT (Devlin et al., 2018) is a pre-trained language model that uses bidirectional encoder representations of the transformer model. BERT has performed well in many natural language understanding tasks such as the GLUE benchmarks (A. Wang et al., 2018) and natural language inference (Williams, Nangia, and Bowman, 2018). Recently, an encoder-decoder based pre-learning model has appeared, such as BART (Lewis et al., 2019) and T5 (Raffel et al., 2019). These models also use the transformer structure, and they extend the idea of BERT to the encoder-decoder model. Several studies showed that a simple fine-tuned model of these pre-trained models improves the accuracy of many language generation tasks. We can also improve the accuracy of our models by incorporating these pre-trained models.

REFERENCES

Aw, AiTi et al. (2006). "A Phrase-based Statistical Model for SMS Text Normalization". In: *COLING/ACL*, pp. 33–40.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014). "Neural Machine Translation by Jointly Learning to Align and Translate". In: *CoRR*.

— (2015). "Neural Machine Translation by Jointly Learning to Align and Translate". In: *ICLR*.

Bisani, Maximilian and Hermann Ney (2008). "Joint-sequence Models for Grapheme-to-phoneme Conversion". In: *Speech Commun.* Pp. 434–451.

Cao, Ziqiang et al. (2018). "Retrieve, Rerank and Rewrite: Soft Template Based Neural Summarization". In: *ACL*, pp. 152–161.

Chen, Yen-Chun and Mohit Bansal (2018). "Fast Abstractive Summarization with Reinforce-Selected Sentence Rewriting". In: *ACL*, pp. 675–686.

Cho, Kyunghyun et al. (2014). "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *EMNLP*, pp. 1724–1734.

Collins, Michael (2002). "Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms". In: *IJCNLP*, pp. 1–8.

Cook, Paul and Suzanne Stevenson (2009). "An Unsupervised Model for Text Message Normalization". In: *the Workshop on Computational Approaches to Linguistic Creativity*, pp. 71–78.

Devlin, Jacob et al. (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR*.

Dong, Yue et al. (2018). "BanditSum: Extractive Summarization as a Contextual Bandit". In: *EMNLP*, pp. 3739–3748.

Fan, Angela, David Grangier, and Michael Auli (2018). "Controllable Abstractive Summarization". In: *NMT@ACL*, pp. 45–54.

Gehrmann, Sebastian, Yuntian Deng, and Alexander Rush (2018). "Bottom-Up Abstractive Summarization". In: *EMNLP*, pp. 4098–4109.

Grusky, Max, Mor Naaman, and Yoav Artzi (2018). "Newsroom: A Dataset of 1.3 Million Summaries with Diverse Extractive Strategies". In: *ACL*, pp. 708–719.

Han, Bo and Timothy Baldwin (2011). "Lexical Normalisation of Short Text Messages: Makn Sens a #Twitter". In: *ACL:HLT*, pp. 368–378.

Han, Bo, Paul Cook, and Timothy Baldwin (2012). "Automatically Constructing a Normalisation Dictionary for Microblogs". In: *EMNLP-CoNLL*, pp. 421–432.

— (2013). "Lexical Normalization for Social Media Text". In: *ACM Trans. Intell. Syst. Technol.* 5:1–5:27.

Hassan, Hany and Arul Menezes (2013). "Social Text Normalization using Contextual Graph Random Walks". In: *ACL*, pp. 1577–1586.

Hermann, Karl Moritz et al. (2015). "Teaching Machines to Read and Comprehend". In: *NIPS*, pp. 1693–1701.

Hsu, Wan-Ting et al. (2018). "A Unified Model for Extractive and Abstractive Summarization using Inconsistency Loss". In: *ACL*, pp. 132–141.

Huang, Fei (2015). "Improved Arabic Dialect Classification with Social Media Data". In: *EMNLP*, pp. 2118–2126.

Ikeda, Taishi, Hiroyuki Shindo, and Yuji Matsumoto (2017). "Japanese Text Normalization with Encoder-Decoder Model". In: *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*.

Jiang, Wenbin, Haitao Mi, and Qun Liu (2008). "Word Lattice Reranking for Chinese Word Segmentation and Part-of-speech Tagging". In: *ICCL*, pp. 385–392.

Junczys-Dowmunt, Marcin and Roman Grundkiewicz (2016). "Phrase-based Machine Translation is State-of-the-Art for Automatic Grammatical Error Correction". In: *EMNLP*.

Kaji, Nobuhiro and Masaru Kitsuregawa (2013). "Efficient Word Lattice Generation for Joint Word Segmentation and POS Tagging in Japanese". In: *IJCNLP*, pp. 153–161.

— (2014). "Accurate Word Segmentation and POS Tagging for Japanese Microblogs: Corpus Annotation and Joint Modeling with Lexical Normalization". In: *EMNLP*, pp. 99–109.

Kikuchi, Yuta et al. (2016). "Controlling Output Length in Neural Encoder-Decoders". In: *EMNLP*, pp. 1328–1338.

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". In: *ICLR*.

Koehn, Philipp et al. (2007). "Moses: Open Source Toolkit for Statistical Machine Translation". In: *ACL on Interactive Poster and Demonstration Sessions*.

Kudo, T. (2005). "MeCab : Yet Another Part-of-Speech and Morphological Analyzer". In: *http://mecab.sourceforge.net/*.

Kudo, Kaoru Yamamoto, and Yuji Matsumoto (2004). "Applying conditional random fields to Japanese morphological analysis". In: *EMNLP*, pp. 230–237.

Kurohashi, Sadao et al. (1994). "Improvements of Japanese morphological analyzer JUMAN". In: *The International Workshop on Sharable Natural Language Resources*, pp. 22–38.

Lewis, Mike et al. (2019). "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension". In: *arXiv e-prints*. arXiv: 1910.13461.

Li, Chen and Yang Liu (2012). "Improving Text Normalization using Character-Blocks Based Models and System Combination". In: *COLING*, pp. 1587–1602.

— (2014). "Improving Text Normalization via Unsupervised Model and Discriminative Reranking". In: *ACL 2014 Student Research Workshop*, pp. 86–93.

Li, Chenliang et al. (2018). "Guiding Generation for Abstractive Text Summarization Based on Key Information Guide Network". In: *ACL*, pp. 55–60.

Lin, Chin-Yew (2004). "ROUGE: A Package for Automatic Evaluation of summaries". In: *ACL*.

Ling, Wang et al. (2013). "Paraphrasing 4 Microblog Normalization". In: *EMNLP*, pp. 73–84.

Liu, Fei, Fuliang Weng, and Xiao Jiang (2012). "A Broad-Coverage Normalization System for Social Media Language". In: *ACL*, pp. 1035–1044.

Liu, Fei, Fuliang Weng, Bingqing Wang, et al. (2011). "Insertion, Deletion, or Substitution? Normalizing Text Messages without Pre-categorization nor Supervision". In: *ACL*, pp. 71–76.

Liu, Yang (2019). "Fine-tune BERT for Extractive Summarization". In: *CoRR* abs/1903.10318. arXiv: 1903.10318.

Liu, Yizhu, Zhiyi Luo, and Kenny Zhu (2018). "Controlling Length in Abstractive Summarization Using a Convolutional Neural Network". In: *EMNLP*, pp. 4110–4119.

Luong, Thang, Hieu Pham, and Christopher D. Manning (2015). "Effective Approaches to Attention-based Neural Machine Translation". In: *EMNLP*.

Machery, W, F J Och, and I Uszkoreit J Thayer (2008). "Lattice-based Minimum Error Rate Training for statistical Machine Translation". In: *EMNLP*, pp. 725–734.

Maekawa, Kikuo et al. (2014). "Balanced corpus of contemporary written Japanese". In: *Language Resources and Evaluation*, pp. 345–371.

Mendes, Afonso et al. (2019). "Jointly Extracting and Compressing Documents with Summary State Representations". In: *NAACL*, pp. 3955–3966.

Mikolov, Tomas, Wen-tau Yih, and Geoffrey Zweig (2013). "Linguistic Regularities in Continuous Space Word Representations". In: *NAACL*, pp. 746–751.

Mohri, Mehryar, Fernando Pereira, and Michael Riley (2002). "Weighted Finite-State Transducers in Speech Recognition". In: *Computer Speech Language*, pp. 69–88.

Narayan, Shashi, Shay B. Cohen, and Mirella Lapata (2018). "Ranking Sentences for Extractive Summarization with Reinforcement Learning". In: *NAACL*, pp. 1747–1759.

Oka, Teruaki et al. (2011). "Handling Orthographic Variations in Morphological Analysis for Near-Modern Japanese (in Japanese)". In: *JSAI*.

Papineni, Kishore et al. (2002). "BLEU: A Method for Automatic Evaluation of Machine Translation". In: *ACL*.

Paulus, Romain, Caiming Xiong, and Richard Socher (2017). "A Deep Reinforced Model for Abstractive Summarization". In: *CoRR* abs/1705.04304. arXiv: 1705.04304.

Pennell, Deana and Yang Liu. "A Character-Level Machine Translation Approach for Normalization of SMS Abbreviations". In: *IJCNLP*, pp. 974–982.

— (2011). "A Character-Level Machine Translation Approach for Normalization of SMS Abbreviations". In: *IJCNLP*, pp. 974–982.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning (2014). "Glove: Global vectors for word representation". In: *EMNLP*.

Raffel, Colin et al. (2019). "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *arXiv e-prints*. arXiv: 1910.10683.

Rush, Alexander M., Sumit Chopra, and Jason Weston (2015). "A Neural Attention Model for Abstractive Sentence Summarization". In: *EMNLP*.

Saito, Itsumi et al. (2014). "Morphological Analysis for Japanese Noisy Text based on Character-level and Word-level Normalization". In: *COLING*, pp. 1773–1782.

Sajjad, Hassan, Kareem Darwish, and Yonatan Belinkov (2013). "Translating Dialectal Arabic to English". In: *ACL*, pp. 1–6.

Sasaki, Akira et al. (2013). "Normalization of Text in Microblogging Based on Machine Learning (in Japanese)". In: *JSAI*.

Sasano, Ryohei, Sadao Kurohashi, and Manabu Okumura (2013). "A Simple Approach to Unknown Word Processing in Japanese Morphological Analysis". In: *IJCNLP*, pp. 162–170.

See, Abigail, Peter J. Liu, and Christopher D. Manning (2017). "Get To The Point: Summarization with Pointer-Generator Networks". In: *ACL*, pp. 1073–1083.

Sennrich, Rico, Barry Haddow, and Alexandra Birch (2016). "Improving Neural Machine Translation Models with Monolingual Data". In: *ACL*.

Sittichai, Jiampojamarn, Kondrak Grzegorz, and Sherif Tarek (2007). "Applying Many-to-Many Alignments and Hidden Markov Models to Letter-to-Phoneme Conversion". In: *NAACL*, pp. 372–379.

Takase, Sho and Naoaki Okazaki (2019). "Positional Encoding to Control Output Sequence Length". In: *NAACL*, pp. 3999–4004.

Vaswani, Ashish et al. (2017). "Attention is All you Need". In: *NIPS*, pp. 5998–6008.

Vinyals, Oriol, Meire Fortunato, and Navdeep Jaitly (2015). "Pointer Networks". In: *NIPS*, pp. 2692–2700.

Wang, Alex et al. (2018). "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding". In: *BlackboxNLP@EMNLP*, pp. 353–355.

Wang, Liang et al. (2019). "Denosing based Sequence-toSequence Pre-training for Text Generation". In: *EMNLP*.

Williams, Adina, Nikita Nangia, and Samuel R. Bowman (2018). "A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference". In: *NAACL-HLT*, pp. 1112–1122.

Xie, Ziang et al. (2016). "Neural Language Correction with Character-Based Attention". In: *CoRR*.

Yang, Yi and Jacob Eisenstein (2013). "A Log-Linear Model for Unsupervised Text Normalization". In: *EMNLP*, pp. 61–72.

You, Yongjian et al. (2019). "Improving Abstractive Document Summarization with Salient Information Modeling". In: *ACL*, pp. 2132–2141.

Yuan, Zheng and Ted Briscoe (2016). "Grammatical error correction using neural machine translation". In: *NAACL*.

Zhang, Xingxing, Mirella Lapata, et al. (2018). "Neural Latent Extractive Document Summarization". In: *EMNLP*. Association for Computational Linguistics, pp. 779–784.

Zhang, Xingxing, Furu Wei, and Ming Zhou (2019). “HIBERT: Document Level Pre-training of Hierarchical Bidirectional Transformers for Document Summarization”. In: *ACL*, pp. 5059–5069.

Zhou, Qingyu et al. (2018). “Neural Document Summarization by Jointly Learning to Score and Select Sentences”. In: *ACL*, pp. 654–663.

List of Publications

Journals

1. Itsumi Saito, Kugatsu Sadamitsu, Hisako Asano, Yoshihiro Matsuo, Morphological Analysis for Japanese Noisy Text based on Extraction of Character Transformation Patterns and Lexical Normalization, 自然言語処理, pp. 297-314, 2017.

International Conferences (Refereed)

1. Itsumi Saito, Kugatsu Sadamitsu, Hisako Asano, Yoshihiro Matsuo, Morphological Analysis for Japanese Noisy Text Based on Character-level and Word-level Normalization, in Proceedings of COLING, pp.1773-1782, 2014.

2. Itsumi Saito, Kyosuke Nishida, Kugatsu Sadamitsu, Kuniko Saito, Junji Tomita, Automatically Extracting Variant-Normalization Pairs for Japanese Text Normalization, in Proceedings of IJCNLP, pp.937-946, 2017.

3. Itsumi Saito, Jun Suzuki, Kyosuke Nishida, Kugatsu Sadamitsu, Satoshi Kobashikawa, Ryo Masumura, Yuji Matsumoto, Junji Tomita, Improving Neural Text Normalization with Data Augmentation at Character- and Morphological Levels, in Proceedings of IJCNLP, pp.257-262, 2017.

Other Publications (Refereed)

1. Itsumi Saito, Kyosuke Nishida, Hisako Asano, Junji Tomita, Commonsense Knowledge Base Completion and Generation, in Proceedings of CoNLL, pp.141-150, 2018.

2. Kyosuke Nishida, Itsumi Saito, Atsushi Otsuka, Hisako Asano, and Junji Tomita, Retrieve-and-Read: Multi-task Learning of Information Retrieval and Reading Comprehension, in Proceedings of CIKM, pp. 647-65, 2018.

3. Atsushi Otsuka, Kyosuke Nishida, Itsumi Saito, Hisako Asano and Junji Tomita, Specific Question Generation for Reading Comprehension, in Proceedings of RCQA@AAAI, 2019.

4. Kyosuke Nishida, Itsumi Saito, Kosuke Nishida, Kazutoshi Shinoda, Atsushi Otsuka, Hisako Asano and Junji Tomita, Multi-style Generative Reading Comprehension, in Proceedings of ACL, pp.2273-2284, 2019.

5. Kosuke Nishida, Kyosuke Nishida, Masaaki Nagata, Itsumi Saito, Atushi Otuka, Hisako Asano and Junji Tomita, Answering while Summarizing: Multi-task Learning for Multi-hop QA with Evidence Extraction, in Proceedings of ACL, pp.2335-2345, 2019.

6. Yasuhito Ohsugi, Itsumi Saito, Kyosuke Nishida, Hisako Asano, and Junji Tomita, A Simple but Effective Method to Incorporate Multi-turn Context with BERT for Conversational Machine Comprehension, in Proceedings of NLP4ConvAI@ACL, 2019.

Other Publications (Non-Refereed)

1. Itsumi Saito, Kyosuke Nishida, Kosuke Nishida, Atsushi Otsuka, Hisako Asano, Junji Tomita, Hiroyuki Shindo, Yuji Matsumoto, Length-controllable Abstractive Summarization by Guiding with Summary Prototype, arXiv:2001.07331.

Domestic Conferences (Non-Refereed)

1. 斉藤いつみ, 貞光九月, 浅野久子, 松尾義博, 正規-崩れ文字列アライメントと文字種変換を用いた崩れ表記正規化に基づく日本語形態素解析, pp.777-780, NLP2014.

2. 斉藤いつみ, 貞光九月, 浅野久子, 松尾義博, 崩れ表記語の生成確率を用いた表記正規化と形態素解析, pp.51-54, NLP2015.

3. 斉藤いつみ, 貞光九月, 浅野久子, 松尾義博, web上のテキストからの表記揺れ語獲得, pp.55-58, NLP2016.

4. 斉藤いつみ, 鈴木潤, 貞光九月, 西田京介, 齋藤邦子, 松尾義博, 擬似データの事前学習に基づくencoder-decoder型日本語崩れ表記正規化, pp.585-588, NLP2017.

5. 斉藤いつみ, 西田京介, 浅野久子,富田準二, フレーズ知識補完と生成の同時学習, pp.951-954, NLP2018.

6. 大塚淳史, 西田京介, 斉藤いつみ, 浅野久子，富田準二, 質問の意図を特定するニューラル質問生成モデル, DEIM2018. **[優秀論文賞]，[優秀インタラクティブ賞]**

7. 西田京介, 斉藤いつみ, 大塚淳史, 浅野久子, 富田準二, 回答スタイルを制御可能な生成型機械読解, pp.963-966, NLP2018 **[最優秀論文賞]**

8. 斉藤いつみ, 西田京介, 大塚淳史, 西田光甫, 浅野久子, 富田準二, クエリ・出力長を考慮可能な文書要約モデル, pp.585-588, NLP2019. **[最優秀ポスター賞]**

9. 西田京介, 斉藤いつみ, 西田光甫, 篠田一聡, 大塚淳史, 浅野久子, 富田準二, 回答スタイルを制御可能な生成型機械読解, pp.17-20, NLP2019 **[優秀論文賞]**

10. 西田光甫, 西田京介, 永田昌明, 大塚淳史, 斉藤いつみ, 浅野久子, 富田準二, 抽出型要約との同時学習による回答根拠を提示可能な機械読解, pp.25-28, NLP2019

11. 大塚淳史, 西田京介, 斉藤いつみ, 西田光甫, 浅野久子, 富田準二, 問い返し可能な質問応答：読解と質問生成の同時学習モデル, DEIM2019. **[優秀インタラクティブ賞]**