Master's Thesis

Two-phase Lexical Normalization on Social Media Language

Yingying Zeng (曾莹莹)

Department of Computer Science and Engineering

Pohang University of Science and Technology

2016


POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
1986
포항공과대학교

소셜 미디어 언어를 위한 2 단계 어휘정규화

Two-phase Lexical Normalization on Social Media Language

Two-phase Lexical Normalization on Social Media Language

by

Yingying Zeng
Department of Computer Science and Engineering
Pohang University of Science and Technology

A thesis submitted to the faculty of the Pohang University of Science and Technology in partial fulfillment of the requirements for the degree of Master of Science in the Computer Science and Engineering

Pohang, Korea
6. 27. 2016
Approved by
Jong-Hyeok Lee
Academic advisor

Two-phase Lexical Normalization on Social Media Language

Yingying Zeng

The undersigned have examined this thesis and hereby certify that it is worthy of acceptance for a master's degree from POSTECH

6. 27. 2016

Committee Chair Jong-Hyeok Lee

Member Hwanjo Yu

Member Young Joo Suh

MCSE Yingying Zeng
20121185
Two-phase Lexical Normalization on Social Media Language,
소셜 미디어 언어를 위한 2 단계 어휘정규화
Department of Computer Science and Engineering , 2016, 32p, Advisor : Jong-Hyeok Lee. Text in English.

ABSTRACT

Natural Language Processing (NLP) on data from social network services (SNSs) became more difficult than before because users in SNSs shorten the words to send the message quickly and some SNSs even limit the length that users can input in one message. Therefore, lexical normalization has become a necessary step before the NLP systems process SNS data. This paper proposes a lexical normalization system that can suggest normalization candidates for an input non-standard word (NSW). The proposed system generates normalization candidates by combining phonetic substitution and letter insertion. The system uses phonetic substitution by table lookup to generate intermediate candidates and uses letter insertion on intermediate candidates to form the final candidate set. Without refering to any existing NSW and SW pairs, this system succeeded to recover most test words and reach 84.82% Top-20 recall. This result proved that NSWs can be normalized without referencing existing NSWs.

Keywords: text normalization, non-standard word (NSW), internet slang,

netspeak, candidate generation, phonetic substitution, letter insertion


POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
1986
포항공과대학교

Contents

List of Tables

List of Figures

I. Introduction

1.1 NLP on Social Network Services (SNSs)

Natural language processing (NLP) of data from social network services (SNSs) is an important problem because people frequently post and pass on information such as social activities and interests on SNSs (Fig. 1.1). Users of SNS tend to use informal text (Table 1.1) because some SNSs limit the length of message that the user can send, and because users want to reduce the typing complexity to send message quickly. However, as the NLP tasks trained usually trained on clean data, such informal text decreases the accuracy of NLP tasks such as POS tagging and noun phrase chunking [1].

Non-standard words (NSWs) do not exist in a dictionary, such as "4got" and "bday" in second example showed in Table 1.1. NSWs are usually created intentionally while unintentionally creadted NSWs, also known as misspelled words, are not the main target of lexical normalization task. [2] classifies NSWs into four categories (Table 1.2) while [3] manually classied 254 NSWs into five categories (Table 1.3). We analyze a set of 303 NSWs and classify the NSWs into five categories based on the two classification above (Table 1.4). Notice that the "slang"

Table 1.1: Examples of Informal Text

1	AM CONFUSED!! y you did that?
2	I 4got 2 tell u the bday party 2nite.

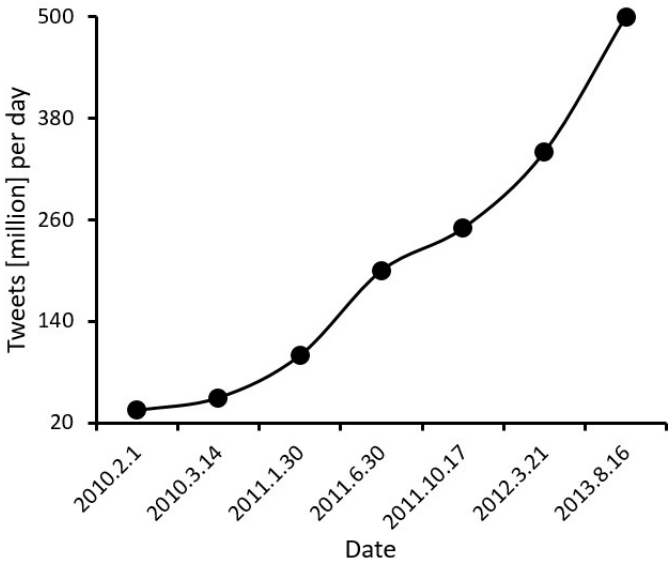

Figure 1.1: Statistics on Twitter: Tweets per day

Table 1.2: Categories of NSWs (1)

Category	Example
Phonetic writing	rite (right)
Consonantal spelling	wrk (work)
Non-conventional use of letter or number	ani1 (anyone)
Mixture	bcum (become)

in Table 1.3 category can be easily processed using dictionary lookup therefore we will not deal with that category.

Such NSWs decrease the accuracy of NLP tasks in processing SNS data. The accuracy of a well-known task, POS tagging, drops to 0.801 from 0.97 [1] and they reported that the key reason for drop of accuracy is the NSWs. Therefore, preprocessing has become necessary when the object of the task is data from SNS.

Table 1.3: Categories of NSWs (2)

Category	Ratio	Example
Letter&Number	2.36%	b4 (before)
Letter	72.44%	shuld (should)
Number Substitution	2.76%	4 (for)
Slang	12.20%	lol (laugh out loud)
Other	10.24%	sucha (such a)

Table 1.4: Categories of NSWs (3)

Category	Subcategory	Example
Phonetic substitution	letter substitution	rite (right)
	number substitution	2day (today)
Letter deletion	vowel deletion	wrk (work)
	duplicated consonant deletion	account (acount)
Mixture of categories		bcum (become), 2mr (tomorrow)

1.2 Lexical Normalization

Lexical normalization is the process that corrects NSWs to their standard forms. For an input sentence, a lexical normalization system aims to recover all NSWs and for each NSW, the system will output a Top-K candidate set that consists standard words (SW) that system suggests as the candidates of the NSW. For a specific NSW in the input sentence, the system generates a candidate set and for each candidate, the system will suggest a score as the conditional probability (Fig. 1.2). Usually, the score contains two parts: word level score and contextual score. Contextual score is not always considered but [3, 4] believe contextual information can help scoring the candidates.

The NSWs are not always input to the normalization system separately, this means NSWs that the system has to process should be detected first. The process to detect the NSWs is called NSW detection. NSW detection will provide the NSWs that the normalization system has to normalize. A simple solution is using dictionary lookup: a word is an NSW if the word does not exist in the dictionary. In this work, we assume that all NSWs are perfectly detected already. Therefore, our normalization system will take isolated NSWs as input.

Based on the form of standard words, the lexical normalization task consists of 1-to-1 mapping and 1-to-M mapping. 1-to-1 mapping mainly aims to recover NSWs that depend on only one SW, for example, “shuld (should)”, “enuf (enough)”. 1-to-M mapping aims to recover NSWs that depend on several words, for example, “btw (by the way)”, “sucha (such a)”. “btw (by the way)” belongs

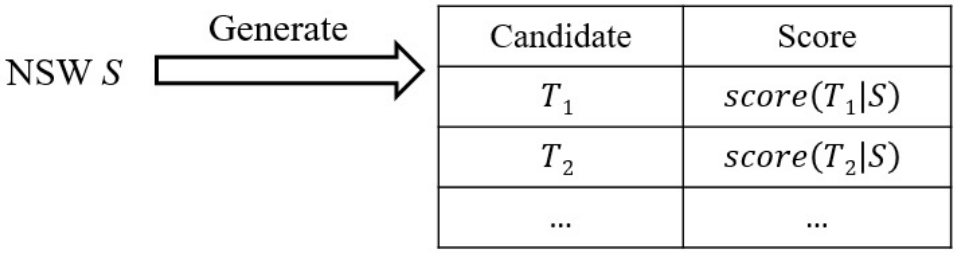

Figure 1.2: Input and Output of Lexical Normalization System

to the “slang” category we discussed before and “sucha (such a)” is grammatic noise in fact. Therefore, we will only consider 1-to-1 mapping in this work.

II. Related Work

2.1 Overview

For a lexical normalization system (Fig. 1.2), the main problems that the system has to solve are

(1) How to generate the canditates?

(2) How to score the candidates?

Many researchers tried to solve this problem using variant approaches. Several work proves that combination of different systems may get better performance [2, 5].

2.2 Lexicon Building

As a typical and classic approach in lexical normalization problem, [6] proposed a system from the most natural viewpoint of human (Fig. 2.1). [6] trained a HMM based on a (SW, NSW) corpus. For every SW in a dictionary, the system use the HMM to construct NSWs and pair them with the original SW. The collection of such (SW, NSW) pairs will form a new lexicon. With an NSW* input to the system, it will search the (SW, NSW*) paris from the lexicon. The SWs that paired with input NSW* will be considered as the candidates that system suggest.

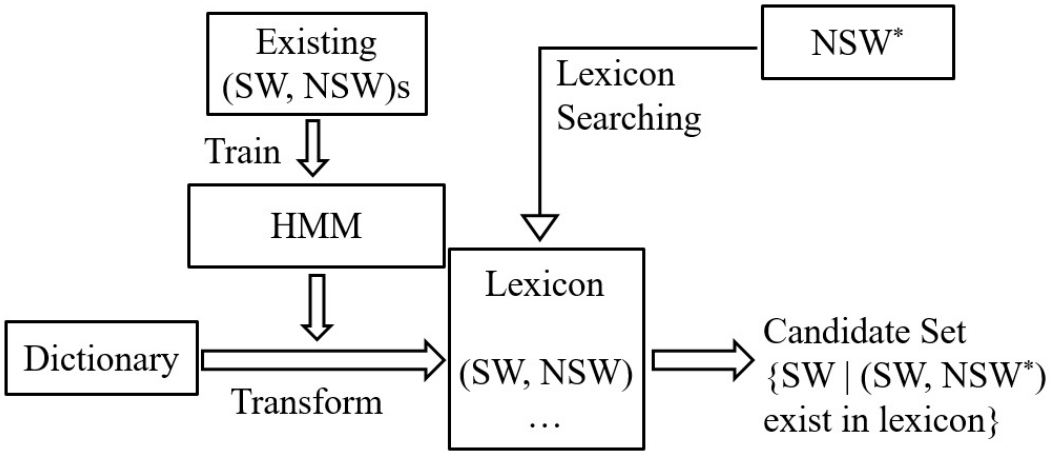

Figure 2.1: System Structure of [6]

This system used HMM and try to predict how people variate SWs. By extend the variations, the system is able to variate all SW words. As all hard work done before the NSW input, the system can return the candidate set quickly because the remaining problem is just lexicon lookup. However, the training pairs is created manually and this may cause the problem that the input NSW does not exist in the lexicon. If so, the candidate set suggest by system may be just a empty set. This kind of problem definitely exists in lexicon building systems.

Besides this work, [5, 7] uses letter transformation to generate NSWs from SWs. Instead of generating NSWs by SWs, [4] collects (SW, NSW) paris using contextual graph random walks.

2.3 Dictionary Searching

[3] uses dictionary to search proper candidates of an input NSW word. The system uses two distances to make criterions for searching: character edit distance

Table 2.1: Recall and average number of candidats using different criterion

Criterion	Recall(%)	Average Candidates
$T_c \leq 2$	76.6	240
$T_p = 0$	55.4	65
$T_p \leq 2$	91.0	9694
$T_c \leq 2 \vee T_p \leq 1$	88.8	1269
$T_c \leq 2 \vee T_p \leq 2$	92.7	9515

and phonetic edit distance (Table 2.1). For each candidate, the word similarity score considered: lexical edit distance, phonemic edit distance and the longest common subsequence (LCS). This system combines the word similarity score with contextual score that is obtained from a trigram language model (LM). However, the fact is (also reported in the paper) the contextual score is not always useful since the context of an NSW word is not necessarily clean.

Moreover, this kind of approach is too näive and generates too much garbage that the system does not need at all. Even though the score will give the rank for all the candidates and maybe the real garbage is defeated in ranking, still this will slow the system down and delay the time for output.

2.4 MT-like System

The lexical normalization problem can be also considered as a machine translation (MT) problem since the goal of a lexical normalization system is to "translate" a NSW to (a) SW(s). [8] trained a lexical normalization system using MOSES, which is a famous statistitical MT training tool.

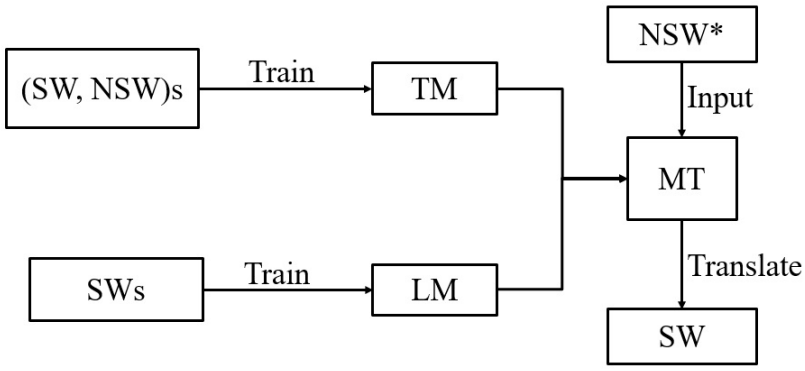

Figure 2.2: System Structure of [9]

2.4.1 Character-level MT System

MT-like systems are usually trained on word-level. Instead of word, [9] proposed a character-based PBMT system (Fig. 2.2). The translation model (TM) and language model (LM) are all trained on character-level. As we can expect, to train such a system, the training (NSW, SW) pairs must be collected manually. Beside, the character-level MT system is trained automatically and only can catches the transformation between characters.

2.4.2 Phonetic-based MT System

To overcome the shortage of [9], [10] proposes a phonetic-based MT system (Fig. 2.3) and combines it with the character-level MT system from [9]. The phonetic-based MT system first uses MT system1 to translate the NSW to its phonetic representation and then uses MT system2 to translate the phonetic symbols to standard words. The problem is that even this system combine character and phonetic information together to normalize the NSW, the MT systems still

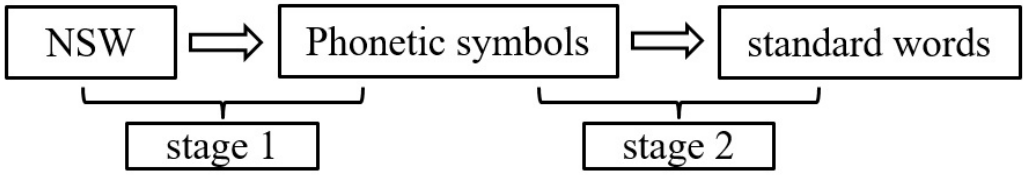

Figure 2.3: System Structure of [10]

strongly depend on the training pairs that is manually collected.

III. Proposed Method

Most of the previous systems used human created or collected training data that is a collection of existing (NSW, SW) pairs. However, this kind of collection: (1) cannot contain all existing pairs since the collection job is extremely hard and painful; (2) only can reflect relation of NSW and SW words that are already created. On the contrary, people are still creating new NSWs every day. This makes the (NSW, SW) pair corpus becomes out-of-date quickly and normalization system may fail to recover the new NSWs.

Even though the normalization system may fail sometimes, the reader still can understand the meaning of the new NSWs. This fact leads to a very interesting question: how SNS users understand the NSWs even though they do not know all existing NSWs. Based on this viewpoint, we proposed a system that do not require such (NSW, SW) pairs.

3.1 Motivation

We illustrate our motivation of proposed method by analyzing how people make NSWs case by case. We analyze two cases, letter-based variation and number substitution.

3.1.1 Letter-based Variation

shuld (should)

The variation from “should” to “shuld” can be considered in two ways: (1) letter dropping, letter “o” is dropped; (2) phonetic variation, “u” pronounced “[u]” in some words same as “ou” in “should”.

enuf (enough)

There are two variations in this example, “u (ou)” and “f (gh)”. “u (ou)” can be considered in two ways same as previous example “shuld (should)”. Variation “f (gh)” can only be considered as phonetic substitution.

3.1.2 Number Substitution

4 (for)

The variation of “4” to “for” is obvious based on phonetic similarity because the pronunciation of “4” is [fɔ:] same as “for”.

2day (today)

This example is noticeable because the pronunciation of “to” in “today” is “[t]” but the pronunciation of “2” is “[tu]”. The reason that people replace “to” by “2” is because the pronunciation of “to” is “[tu]” as a single word.

In fact, almost all number substitution is based on phonetic similarity.

3.1.3 System Preview

Based on the analysis we did on previous NSWs, we found out that all NSWs are made based on letter dropping and phonetic substitution (Table 3.1). This fact agrees with the classification (Table 1.4) and lead us to a quite simple solution from human’s viewpoint, combination of letter insertion and phonetic substitution.

Table 3.1: Portion of letter dropping and phonetic substitution in 303 NSWs

Variation	Count
Letter dropping	122
Phonetic substitution	47
Mixture	103
Total	272

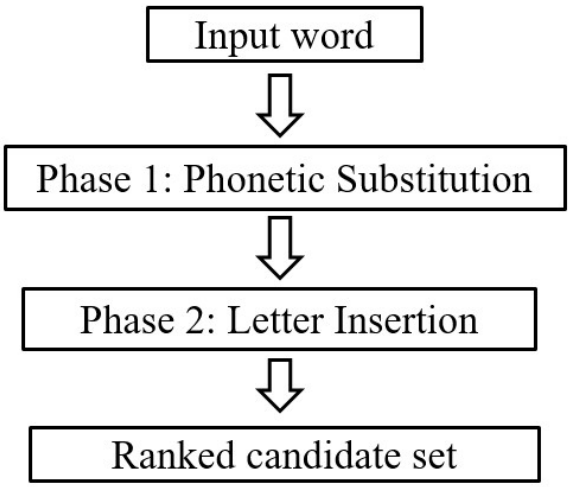

Figure 3.1: Structure of Two-phase Lexical Normalization System

3.2 Proposed System

3.2.1 Overview

We proposed a two-phase lexical normalization system (Fig. 3.1). Phase 1 is phonetic substitution and aims to substitute the letter sequences that people substituted back. Phase 2 inserts letters what dropped intentionally. The output of our system is a ranked candidate set given an input NSW.

The reason of letter insertion phase cannot come before phonetic substitution is given followed.

(1) In phonetic substitution phase, system will try to substitution the letter sequences in the NSW. If the system inserts letters first, substitution phase will replace the inserted letters by letter sequences phonetic related. If the inserted letter cannot be kept, letter insertion phase became meaningless and useless.

(2) The general solution for letter insertion is to compare the NSW with dictionary word and this make the output be an NSWs set. At the same time, the output of phonetic substitution is not necessary an set that consists of standard words because we only replace letter sequences using phonetic relation. These facts also make the order of two phases natural.

3.2.2 Phonetic Substitution

Phonetic substitution replaces letter sequences in an NSW by letter sequences related phonetically. By combining the substitution result, the system can give intermediate candidates (Table 3.2). Therefore, all we need is a letter sequence lookup table that consists letter sequences and their phonetically related letter sequences (Table 3.3). After the lookup table is generated, phonetic substitution can be simply done. We call the output candidates of phonetic substitution phase intermediate candidates.

The letter-sequence lookup table is generated using a letter-phoneme aligned parallel corpus (Fig. 3.2). Each word in the corpus is letter-phoneme aligned. For example, the alignment of word phonemic transcription pair ("enough", "[nf]") is set e-[ɪ], n-[n], ou-[ʌ], gh-[f], (Table 3.4).

Table 3.2: Example of Phonetic Substitution for Word “enuf”

NSW	Substitution Result
e	e
n	n
u	ou
f	gh

Table 3.3: Example of Letter Sequence Lookup Table

Letter-sequence	Substitution Candidates
e	e, i, ea, ...
i	i, e, ...
f	f, gh, ...
u	u, ou, a, ...

Table 3.4: Letter-phoneme alignment result for word “enough”

Letter-sequence	Phoneme
e	[ɪ]
n	[n]
ou	[ʌ]
gh	[f]

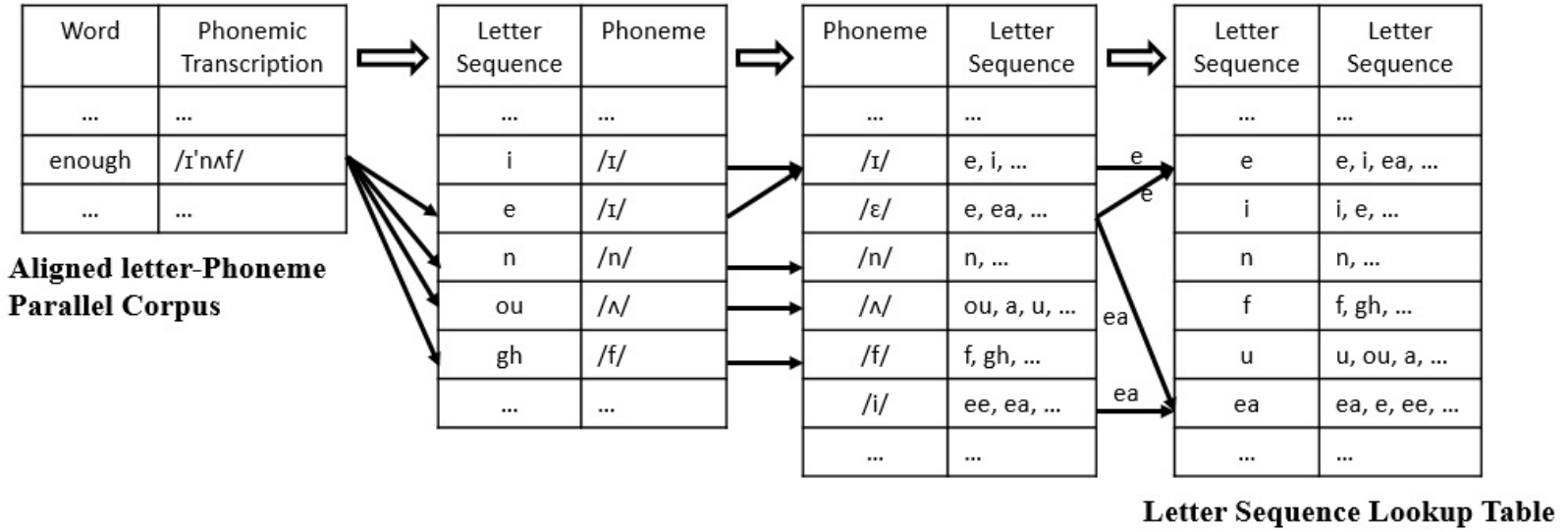

Figure 3.2: Generating Process of Letter-sequence Lookup Table

The alignment in all parallel then resulted in three sets: set L that includes unique letter sequences, set P that includes unique phonemes (possibly phoneme sequences) and set R that consists of possible relations between L and P. Then we used L, P and R to generate a lookup table that contained every letter sequence and its substitution candidates (Table 3.3). For letter sequences A and B, B can be considered as A's substitution candidate if A and B both have relations with a phoneme in P.

We use CMU pronunciation dictionary as the source of letter-phoneme aligned parallel corpus. The CMU pronunciation dictionary consists of 133,746 entries and each entry has form "dictionary - D IH K SH AH N EH R IY", notice that [D], [IH] are the phonetic symbols in CMU dictionary. However, there is no information about the corresponding relation between letters and phonemes and accuracy of automatic letter-phoneme alignment (Fig. 3.3) system only reached 75.52% [11]. We aligned 4716 most frequently used words manually and extended the alignment to whole CMU dictionary using heuristic recurrent algorithm. The automation of letter-phoneme alignment achieved 77.63% accuracy on word level that is lower than our expectation. Therefore, the parallel corpus for lookup table generation only contains 4726 words we aligned manually in this work.

3.2.3 Letter Insertion

Unlike the example in Table 3.2, not all NSWs can be recovered only using phonetic substitution. The best intermediate candidate we can get from "plz" is "plse" (Table 3.5) because there is no phonetic base for the missing letters "ea".

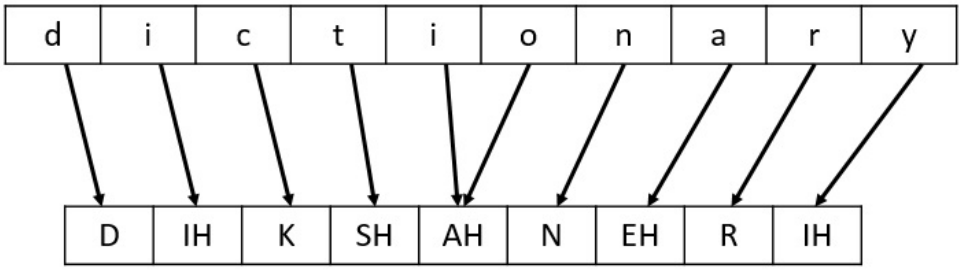

Figure 3.3: Example of Letter-to-Phoneme Alignment of Word “dictionary”

Table 3.5: Example of Phonetic Substitution for Word “plz”

NSW	Substitution Result
p	p
l	l
z	se

Therefore, we need to insert missed letters as we mentioned before.

In letter insertion, we check the letters that the system has to insert to make the intermediate word to be words in dictionary (Table 3.6) and form the final candidate set. The final candidate set is formed by all dictionary words that the number of inserted letters $\leq k$. For example, for intermediate candidate “plse”, k=2 candidate set contains “please”, “pulse”, “pulsed” but “pleased” since “pleased” need three letter insertions.

Table 3.6: Process of Letter Insertion

Intermediate word	Dictionary word	Inserted letter	# of inserted letters
Ex) plse	...	...	...
	please	e, a	2
	plesed	e, a, d	3
	...	...	...

We have to mention that we didn't use any restriction on the position of insertion nor types of insertion. Types of insertion consists vowel and consonant. There is no evidence to restrict the type since both of vowel and consonant can be deleted. Moreover, we need to analyze the regularity of deleted letter position otherwise the rule without any evidence will only limit the performance of the system.

3.2.4 Scoring

We combined two score to give the candidates the final scores: word-level similarity score, Twitter corpus score. Given an NSW $S = s_1 \dots s_n$, for each candidate T,

$$Score(T \mid S) = P(T' \mid S) * F(T)$$

where $T' = t'_1 \dots t'_n$ is the intermediate candidate associated with T and,

$$P(T' \mid S) = \prod_{i=1}^{n} P(t'_i \mid s_i).$$

$P(T' \mid S)$ is word-level similarity score and $F(T)$ is related to frequency of T in a Twitter corpus.

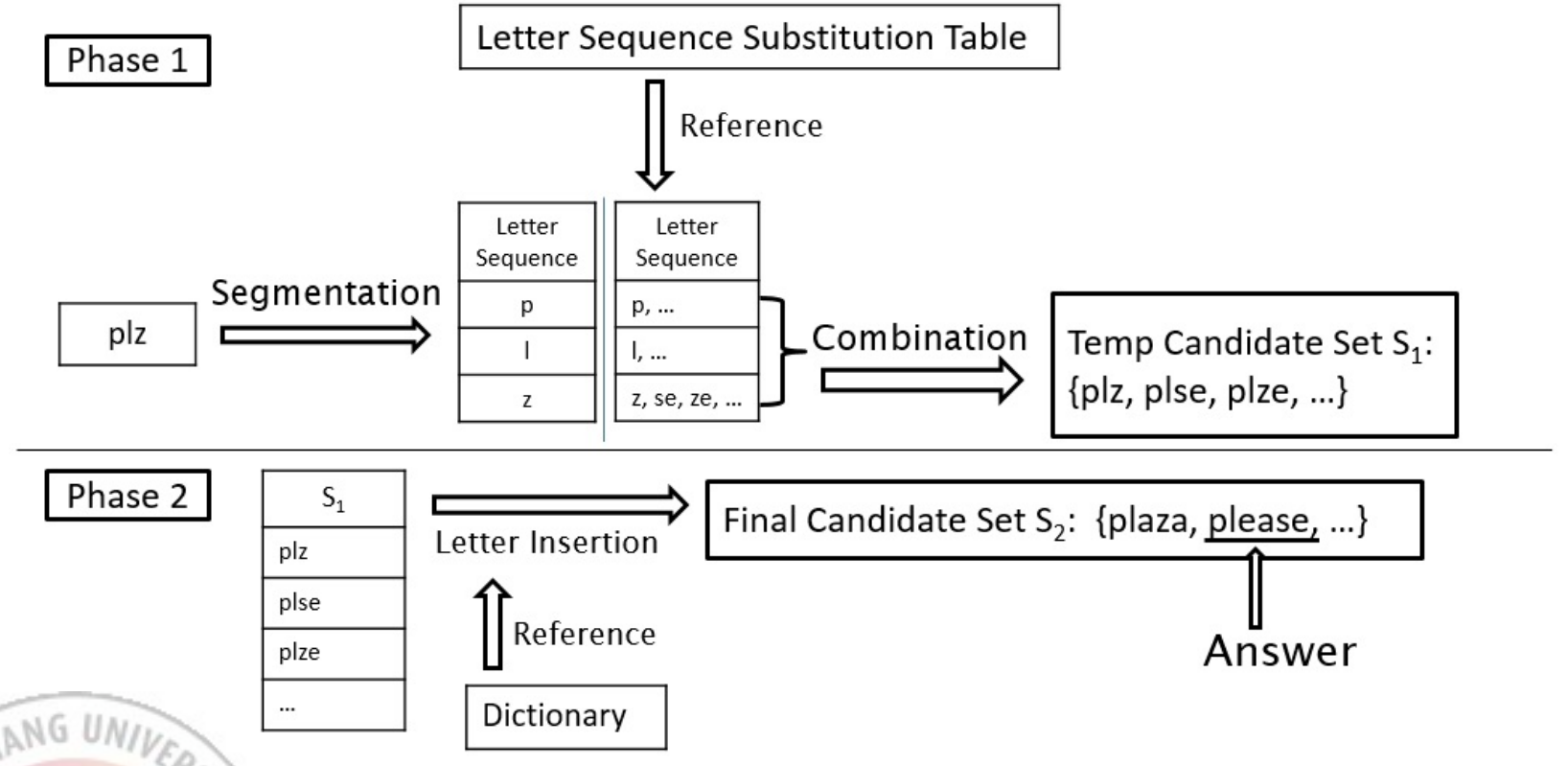

Figure 3.4: Example for Two-phase Lexical Normalization System on NSW "plz"

IV. Experiment

4.1 Test Data Description

We use the test dataset created by [6]. This dataset contains 303 words and is the most popular used test dataset (Table 4.1). Testing on this data set will make the comparison between our system and previous systems easier. This dataset contains 272 words that our system is targeting on, categories "letter dropping", "phonetic substitution" and "mixture". Else words are in categories: "slang" that our system do not need to precess as we discussed before; same for category "misspelled"; and test data error, such as NSW "fnds" should have correspond standard form "finds" not "found" and "6 (six)" do not need normalization at all.

The input of phonetic substitution is a segmented word. Therefore, the system has to segment the input NSW into letter-sequences. We called this process letter chunking (Table 4.2) which has similar purpose with [11].

Table 4.1: Statistics of 303 Test Words

Category	Count	Example
Letter dropping	122	accnt (account)
Phonetic substitution	47	enuf (enough)
Mixture	103	ansa (answer)
Slang	4	brekkie (breakfast)
Misspelled	11	frist (first)
Test data error	15	fnds (found), 6 (six)

Table 4.2: Letter Chunking Result for Word "account". '0' means the letter is not the boundary of a letter chunk and '1' means opposite.

Letter	A	C	C	O	U	N	T
Boundary	1	0	1	0	1	1	1

We trained a SVM classifier to search the letter chunk boundary used features such as letter, vowel or consonant, previous chunk boundary information. The classifier used a human annotated training set contains 4700 words and achieved 91.4% training accuracy and 89.92% test accuracy on character level. Since the classifier did not reach the accuracy that we expected and we do not want to propagate the error to the normalization system, we will use human annotation at this point and combine the classifier with the system in the future.

4.2 Experiment Result

4.2.1 Result

We evaluated performance of our system using accuracy in Top-K ($K = 1, 3, 10, 20$). As the process that we introduced in phase letter insertion, we used ten different insert criterion I for insertion, $I = 0 - 9$. Moreover, based on two different kinds of scores, word-level similarity score and Twitter corpus score, we ranked the candidates in three different ways: word-level similarity score only, Twitter corpus score only and combination of two scores (Table 4.3, 4.4, 4.5).

Coverage means the existence of correct answer in final candidate set and average number of candidate is the average number of candidate in final candidate sets of 303 words. Moreover, it is worth to mention that in evaluation only

Table 4.3: System Accuracy(%) in Top-K candidate set using word-level similarity score only

I	**Top-1**	**Top-3**	**Top-10**	**Top-20**	**Coverage**	**Average # of candidates**
0	31.68	49.17	42.90	50.17	50.83	8
1	51.16	60.73	71.95	75.25	76.90	65
2	62.05	74.59	82.18	83.83	85.15	252
3	65.02	77.89	84.82	86.14	87.46	623
4	66.34	79.21	87.13	88.78	91.09	1140
5	66.01	79.54	88.78	91.09	92.08	1714
6	66.67	80.20	88.78	91.75	92.74	2256
7	67.33	80.86	89.77	92.08	93.07	2690
8	67.33	81.19	90.10	92.08	93.07	2992
9	67.33	81.52	90.10	92.08	93.07	3180

Table 4.4: System Accuracy(%) in Top-K candidate set using Twitter corpus score only

I	**Top-1**	**Top-3**	**Top-10**	**Top-20**	**Coverage**	**Average # of candidates**
0	36.63	46.20	50.17	50.50	50.83	8
1	41.25	57.43	70.96	74.26	76.90	65
2	34.32	53.47	69.64	76.90	85.15	252
3	32.01	48.18	66.34	75.25	87.46	623
4	31.02	46.20	65.68	73.93	91.09	1140
5	30.36	45.87	66.01	73.60	92.08	1714
6	30.36	45.21	66.34	72.61	92.74	2256
7	30.36	44.88	65.68	72.28	93.07	2690
8	30.36	44.88	65.68	72.28	93.07	2992
9	30.36	44.88	65.68	72.28	93.07	3180

Table 4.5: System Accuracy(%) in Top-K candidate set using combination of two scores

I	**Top-1**	**Top-3**	**Top-10**	**Top-20**	**Coverage**	**Average # of candidates**
0	42.90	46.86	50.50	50.83	50.83	8
1	60.73	67.66	74.59	75.58	76.90	65
2	67.66	74.92	78.55	80.53	85.15	252
3	64.69	73.60	79.54	83.17	87.46	623
4	64.03	71.29	80.86	84.82	91.09	1140
5	64.03	71.62	81.19	84.49	92.08	1714
6	62.05	71.29	81.19	84.82	92.74	2256
7	62.38	71.29	81.52	84.49	93.07	2690
8	62.71	71.29	81.19	84.49	93.07	2992
9	62.38	70.96	81.19	84.49	93.07	3180

Table 4.6: Average Number of Candidates in Top-K Set for $I = 4$ only using word similarity score

K	Accuracy(%)	Candidates
1	66.34	81
3	79.21	183
10	87.13	332
20	88.78	447

using word-level similarity, our system output not only K candidates in Top-K set (Table 4.7. The reason is that our word-level similarity only consider phonetic similarity and we did not use any score when inserting letters. This makes candidates that from same intermediate candidate have same score.

We also test each phase separately to check the ability of each phase (Table **??**). "Ph1" is the system only use phonetic substitution, "Ph2" is the system only use letter insertion and "Ph1+2" is the combined system. The combined system

Table 4.7: Average Number of Candidates in Top-K Set for $I = 4$ only using word similarity score

I	**Top-1**	**Top-3**	**Top-10**	**Top-20**	**Coverage**	**Average # of candidates**
Ph1	42.9	46.86	50.5	50.83	50.83	8
Ph2	50.17	62.38	64.69	65.02	65.35	141
Ph1+2	**64.03**	**71.29**	**80.86**	**84.82**	**91.09**	**1140**

Table 4.8: Comparison with previous work. "*" means the system uses (NSW, SW) corpus

System	Top-1	Top-3	Top-10	Top-20
Baseline: Jazzy Spell Checker ([12])	49.86	53.13	54.78	55.44
(Choudhury et al., 2007)*[6]	59.9	-	84.3	88.7
(Han and Baldwin, 2011)[3]	75.4	-	-	-
(Penell and Liu, 2011)*[9]	60.39	74.58	75.57	75.57
(Li and Liu, 2012)*[10]	61.96	75.33	81.75	83.11
Our system	**62.05**	**71.29**	**81.19**	**85.15**

improved two phases a lot. By checking the coverage of two separated phases, we can understand well that only use phonetic substitution or letter insertion cannot solve normalization problem completely.

4.2.2 Result Analysis

Our system performed not bad compared to previous system while most of them use human collected (NSW, SW) corpus (Table 4.8). Moreover, compare to [3], our system can generate much less candidates even we use different test set: [3] generates 9515 candidates and reaches 92.7% coverage; our system generates 2256 candidates and reaches 92.74% coverage when $I = 6$. Our system focused

Table 4.9: Recovered Statistic for each Category in $I = 6$

Category	**Count**	**Recovered**	**Negative Example**
Letter dropping	122	118	def (definitely)
Phonetic substitution	47	42	ovawise(otherwise)
Mixture	103	101	thru (through)
Slang	4	3	brekkie (breakfast)
Typing Error	11	6	frist (first)
Test Data Error	15	7	fnds (found), 6 (six)

mainly on "letter dropping", "phonetic substitution", "mixture" and recovered most words in these three categories (Table 4.9). For words that our system failed to recover: (1) "def (definitely)" need more that 6 insertions and $I > 6$ systems can recover it; (2) "ovawise (otherwise)" is variated based on pronunciation of German; (3) for NSW "thu", candidate "through" did exist in the candidate set but the score is too low to make the candidate into the Top-20 set. Some words in categories that we are not focus on are recovered because the variation is based on pronunciation substitution and letter dropping.

V. Discussion & Conclusion

We developed a system that uses a combination of phonetic substitution and letter insertion to suggest normalization candidates for an input NSW.

Most of the previous work, such as statistical approaches [13, 14, 15, 16] and machine translation approaches [2, 9, 10, 17, 8, 18], strongly depend on training data that is a collection of existing NSWs together with their correct interpretations. However, to collect such data is an extremely difficult task and people create new NSWs frequently. Therefore, the previous systems are weakened by defective training corpora and may fail to normalize new noise words. Distinct from previous work, our system recovered most of the test words without reference to any of the existing noise words. The only resource that our system used is the existing pronunciation dictionary. Without referring any existing NSW, our system achieved 84.82% accuracy in Top-20 set. With a little harm to the recall of the system, our system succeed to recover most test words especially the categories we are targeting on.

Our system was constructed basically from the viewpoint of how humans understand NSWs. However, even humans may guess the wrong answer if only an isolated noise word given (no contextual information). Instead, human choose the answer based on the context of the NSWs as some previous system did [3, 4]. The output candidate set that our system generated contains all possible

normalization candidates for the input NSW bu only considering the word level similarity. Combined with contextual information, we can combine the contextual score with our current score and improve our system to "translate" a sentence to a regular one that is processable in other natural language processes such as machine translation and information retrieval.

Our system proved that without reference to any previous NSW, an NSW still can be normalized to its canonical form. This work will also be useful in building corpora of normalization candidates of NSWs.

References

[1] A. Ritter, S. Clark, and O. Etzioni, *Named entity recognition in tweets: an experimental study*, in Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2011, pp. 1524-1534.

[2] C. Kobus, F. Yvon, and G. Damnati, *Normalizing SMS: are two metaphors better than one?*, in Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1, 2008, pp. 441-448.

[3] B. Han and T. Baldwin, *Lexical normalisation of short text messages Makn sens a twitter*, in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1, 2011, pp. 368-378.

[4] H. Hassan and A. Menezes, *Social Text Normalization using Contextual Graph Random Walks*, in ACL (1), 2013, pp. 1577-1586.

[5] F. Liu, F. Weng, and X. Jiang, *A broad-coverage normalization system for social media language*, in Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1, 2012, pp. 1035-1044.

[6] M. Choudhury, R. Saraf, V. Jain, A. Mukherjee, S. Sarkar, and A. Basu, *Investigation and modeling of the structure of texting language*, Interna-

tional Journal of Document Analysis and Recognition (IJDAR), vol. 10, pp. 157-174, 2007.

[7] F. Liu, F. Weng, B. Wang, and Y. Liu, *Insertion, deletion, or substitution?: normalizing text messages without pre-categorization nor supervision*, in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2, 2011, pp. 71-76.

[8] T. Schlippe, C. Zhu, J. Gebhardt, and T. Schultz, *Text normalization based on statistical machine translation and internet user support*, in INTERSPEECH, 2010, pp. 1816-1819.

[9] D. Pennell and Y. Liu, *A Character-Level Machine Translation Approach for Normalization of SMS Abbreviations*, in IJCNLP, 2011, pp. 974-982.

[10] C. Li and Y. Liu, *Normalization of Text Messages Using Character-and Phone-based Machine Translation Approaches*, in INTERSPEECH, 2012, pp. 2330-2333.

[11] S. Jiampojamarn, G. Kondrak, and T. Sherif, *Applying Many-to-Many Alignments and Hidden Markov Models to Letter-to-Phoneme Conversion*, in HLT-NAACL, 2007, pp. 372-379.

[12] M. Idzelis, *Jazzy: The java open source spell checker*, ed, 2005.

[13] A. AiTi, Z. Min, Y. PohKhim, F. ZhenZhen, and S. Jian, *Input normalization for an english-to-chinese sms translation system*, in The Tenth Machine Translation Summit, 2005.

[14] P. Cook and S. Stevenson, *An unsupervised model for text message normalization*, in Proceedings of the workshop on computational approaches to linguistic creativity, 2009, pp. 71-78.

[15] Z. Xue, D. Yin, B. D. Davison, and B. Davison, *Normalizing Microtext*, Analyzing Microtext, vol. 11, p. 05, 2011.

[16] Y. Yang and J. Eisenstein, *A Log-Linear Model for Unsupervised Text Normalization*, in EMNLP, 2013, pp. 61-72.

[17] D. Contractor, T. A. Faruquie, and L. V. Subramaniam, *Unsupervised cleansing of noisy text*, in Proceedings of the 23rd International Conference on Computational Linguistics: Posters, 2010, pp. 189-196.

[18] A. Aw, M. Zhang, J. Xiao, and J. Su, *A phrase-based statistical model for SMS text normalization*, in Proceedings of the COLING/ACL on Main conference poster sessions, 2006, pp. 33-40.

[19] M. S. Akhtar, U. K. Sikdar, and A. Ekbal, *IITP: Hybrid Approach for Text Normalization in Twitter*, ACL-IJCNLP 2015, p. 106, 2015.

[20] N. Desai and M. Narvekar, *Normalization of noisy text data*, Procedia Computer Science, vol. 45, pp. 127-132, 2015.

[21] S. Gouws, D. Hovy, and D. Metzler, *Unsupervised mining of lexical variants from noisy text*, in Proceedings of the First workshop on Unsupervised Learning in NLP, 2011, pp. 82-90.

[22] S. Jiampojamarn and G. Kondrak, *Letter-phoneme alignment: An exploration*, in Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, 2010, pp. 780-788.

[23] M. Kaufmann and J. Kalita, *Syntactic normalization of twitter messages*, in International conference on natural language processing, Kharagpur, India, 2010.

[24] P. Wang and H. T. Ng, *A Beam-Search Decoder for Normalization of Social Media Text with Application to Machine Translation*, in HLT-NAACL, 2013, pp. 471-481.

[25] C. Zhang, T. Baldwin, H. Ho, B. Kimelfeld, and Y. Li, *Adaptive Parser-Centric Text Normalization*, in ACL (1), 2013, pp. 1159-1168.

Acknowledgements

First, I would give my greatest gratitude to my advisor, Professor Jong-Hyeok Lee, for all the kindness he has given me. With quite weak background of language processing or even computer science, he encouraged me to continue study and trusted me. I have learned not only knowledge but also great attitude towards life and society. Whenever I had trouble or hesitate, he was always there as a good advisor and a good friend.

Secondly, I would like to thank all the members in my lab. You guys are always there to offer your help. I hope you guys can live the life you just want happily.

At last, my gratitude would go to my family, for your support and understanding all the time.

Curriculum Vitae

Name : Zeng Yingying

Education

2006. 9. – 2010. 7. Department of Information and Computing Science, University of Science and Technology Bejing (B.S.)

2010. 9. – 2012. 7. Department of Mathematics, Pohang University of Science and Technology (M.S.)

2012. 9. – 2016. 7. Department of Computer Science and Engineering, Pohang University of Science and Technology (M.S.)

Publication

- Zeng Yingying, Jonggu Kim, Jong-Hyeok Lee, *Graphemic-Phonemic Candidate Generation in Lexical Normalization on Social Media Language* , Korea Computer Congress, Jeju/Korea (June 29 - July 1, 2016)


POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
1986
포항공과대학교