Non-words Spell Corrector of Social Media Data in Message Filtering Systems

Zar Zar Wint[1,3], Théo Ducros[2], Masayoshi Aritsugi[1]
[1]Computer Science and Electrical Engineering
Graduate School of Science and Technology
Kumamoto University, Kumamoto 860-8555, Japan
zarwint12@gmail.com , aritsugi@cs.kumamoto-u.ac.jp
[2]Polytech Clermont-Ferrand
University Clermont Auvergne Clermont-Ferrand, France
ducros.t64@gmail.com
[3]Department of Computer Engineering and Information Technology
Mandalay Technological University
Mandalay, Myanmar

Journal of Digital Information Management

ABSTRACT: *We develop an extended version of spell checker and corrector to check non-word errors in social media datasets, which will be used in message filtering systems especially for cyberbullying detection. We use the dictionary techniques to check words, twelve-word spell error checking and correction approaches to correct the non-word errors, and n-gram and Levenshtein distance to select the most suitable word among corrected words. If there is more than one corrected word we get from each approach, we use n-gram techniques to choose the corrected and reasonable word from the words in n-gram database. When we used the Levenshtein distance in our previous work, we found that it selected the first corrected word and it was not a reasonable one in some sentences. Therefore, we use the n-gram database in this paper.*

Subject Categories and Descriptors: [D.4.4 Communications Management]: Message sending; **[K.4.2 Social Issues] [H.3.3 Information Search and Retrieval]:** Information filtering; **[H.3.1 Content Analysis and Indexing]**: Linguistic processing

General Terms: Message Filtering Systems, Social Media, Non-word Analysis, Information Filtering

Keywords: Non-word Error, Spell Checker, Spell Corrector, Spell Checking And Correction Approaches, N-gram

Received: 19 October 2017, Revised 22 November 2017, Accepted 5 December 2017

1. Introduction

With the rising technology and life-changing inventions, people use electronic devices, such as phone, computer, tablet not only keeping the record, such as writing the diary, book, newspaper, publication but also using social media, such as writing posts, comments and chatting. Most of the word processors have a built-in spelling checker that flags the spelling mistakes and also provides an autocorrect option or suggested words option. There are two kinds of word errors in the real world text processing applications and websites: real word errors and non-word errors. Real word errors are words that can be found in the dictionary and users misspell them. Non-word errors are words that cannot be found in the dictionary. The non-word spell errors happen when users pressed the nearby key on the keyboard by accidentally, when the users spelled the words as they speak, etc. There have been many studies to check these kinds of word errors in the real world text processing applications and websites [1]. Since the 1980s, researchers have done research for spell checking and correcting in word errors [2, 3]. Kukich [4] discussed three aspects of research in her paper: (1) non word error detection; (2) isolated-word error correction; and (3) context-dependent word correction. N-gram and dictionary lookup techniques were used for non-word error detection. She described that isolated-

word (non-word) error has occurred when (1) single instances of insertions, deletions, substitutions, or transpositions have happened (2) the error word length is within one letter in length of the intended word (3) the first letter of a word is misspelled (4) strong keyboard adjacency effects have occurred and (5) strong letter frequency effects have occurred. She used (1) detection of an error; (2) generation of candidate corrections; and (3) ranking of candidate corrections to correct the non-word errors. She used traditional NLP (Natural Language Processing) and SLM (Statistical Language Modeling) for context-dependent (real) word correction. When we learned social media datasets, we found not only those kinds of word errors mentioned in [4] but also some error words had repeated characters more than three or four times. Some research considered error words and their corresponding real words were either the same word length or the word length differs just by one, and removed one character to correct the error word [4, 5, 6]. At that time, we could not consider the error word and real word length different was zero or one. Therefore, we developed our spell checker and corrector based on ten approaches to solve these kind of problems in our previous paper [7].

Our previous work [7] developed the spell checker and corrector with dictionary look up techniques, ten spell checking and correcting approaches with Levenshtein distance to check and correct the non-real words in social media datasets. We developed that to classify the bullied messages exactly. Sometimes, social media use bullied or abusive words filter in their systems, for example, Twitter uses the "mute word" to block unwanted words by the users. At that time, some users use misspelled words intentionally to bully others and avoid the filter. For example they use "bi+(h" instead of the bullied or abusive word "bitch". We have found that there are so many word errors in social media datasets. Users intentionally did most of these word errors for fun and avoiding censorship from the filter. Word errors occur in some reasons e.g., (1) Sometimes, people use characters to replace letters like "@" for "a", "0" for "o", "(" for "c", "3" for "e", "$" for "s", "+" for "t" and "!" for "i" (example: "bi+(h" instead of "bitch") and (2) Sometimes, users use repeated characters at the end of the word or middle the word like "hellooooooo" instead of "hello" and "weeeekeeend" instead of "weekend". Most of the word spell checkers and correctors rely on the dictionaries, which contain the corrected spelled words. By using the dictionaries, word spell mistakes are checked and corrected. Trying to find the word by looking in the entire dictionary can blow up in the time of complexity. There are some approaches, which have tried to reduce the time complexity of spell correction. In our proposed system, we do not look in the whole dictionary to find if the word is correct or not. We only check the word starting with the same letter as the one we are testing. Sometimes, cyberbully detectors cannot detect well the bullied words because of the intentional misspelling. We have found these kinds of misspelled words in our testing datasets. By adopting such approaches, we can investigate more precisely the characteristics of datasets, thereby allowing us to improve the detection with spell correction.

Moreover, our existing system used the Levenshtein distance [8] to find how close a dictionary word to misspelled word. We found some weaknesses in our existing system [7]: it cannot correct the nearby keyword errors and the corrected word chosen by the Levenshtein distance is not always the reasonable one. We therefore updated our system with the nearby key correcting approach and N-gram techniques to choose reasonable words among corrected words. In this paper, we present the spell checker and corrector with two new approaches, namely, Nearby Key Approach (NKA) and Character Changing Approach (CCA) and N-gram approach, to check and correct non-word errors, which have already existed in social media datasets. We intended to use this spell checker and corrector in the preprocessing step of message filtering systems for cyberbullying cases or abusive word detection. Therefore, all of the experiments used the social media datasets in this paper.

The remainder of this paper is organized as follows: In Section 2, we discuss similar research work on spell error checking and correction. In Section 3, we describe the social media datasets used to build the n-gram database and to test our proposed system. In Section 4, we present our extended proposed spell checker and corrector. In Section 5, we present experimental results using the seven social media datasets. Section 6 concludes the paper and mentions some future work.

2. Related Work

There have been many spell checkers and correctors in the real world applications. Some researchers are still trying to improve the spell checkers and correctors in every text processor by using modern techniques based on their needs.

Pande [9] presented a novel, unsupervised, and distance measure agnostic method for search space reduction in spell correction using neural character embedding. The embedding is learned by skip-gram Word2Vec training on sequences generated from dictionary words in a phonetic information retentive manner. He tried to reduce the time complexity of spell correction by reducing the search space of words over which the search for correct spelling is to be done. He also used the dictionary of correctly spelled words during the training process. It is distance measure agnostic because once the search space is reduced then any distance measure of spell correction is used. His method is used as a filter before a spell correction algorithm. His method outperforms one of the recent methods, in terms of both extent of search space reduction and success rates. Our proposed system checks the same first letter word in the dictionary. We also use the Levenshtein distance [8] to find how close a dictionary word is to the corrected misspelled word.

A language-independent spell-checker based on an enhancement of the n-gram model was presented [10]. Their spell checker proposed correction suggestions by selecting the most promising candidates from a ranked list of correction candidates that was derived based on n-gram statistics and lexical resources. Their proposed approach in an application is for keyword and semantic-based search support. They concentrated on the correction of the non-word errors with revised n-gram based approaches because the pure n-gram based approach to compute the similarity coefficient did not consider the order of the n-grams. This kind of problem can be solved by our proposed approaches.

Yannakoudakis and Fawthrop [5] found that in most cases the first letter in a misspelled word is usually correct and the misspelled and real word would be either the same length or the length differs just by one. However, in social media datasets, there are so many repeated letters in the middle of the word and at the end of the word. Hence, the word length is too much larger than the real word. This kind of problem can be solved by our proposed approaches: deletion (removing end character approach and removing middle characters approach).

Smart Spell Checker System (SSCS) can adapt itself to a particular user by using user's feedback for adjusting its behavior [6]. The result of the adjustment is manifested in a different ordering of the suggestions to the user on how a particular spelling mistake should be corrected. SSCS uses the Adaptive Software Architecture (ASA) which contains number of components called Knowledge Sources, i.e., Left/Right Character Shifter, Character Doubler, End Character Appender, Character Remover and Subsequent Character Switcher. Each Knowledge Source is responsible for generating suggestions for correcting a specific type of error. Feedback is propagated to Knowledge Sources after the user makes a selection of the correction. In response to feedback, Knowledge Sources adjust their algorithms. Similar to this system, our system also uses these kinds of knowledge sources but our system does not take back the users suggestions. We use the Levenshtein distance [8] to find how close a dictionary word is to corrected misspelled word and get the corrected word among the suggested words. Programs that do not check what you are typing as you type it are becoming increasingly annoying and the surprising number of programs that would not check your spelling at all are even worse. Despite this, editors and other tools used by programmers are least likely to offer spelling checking.

Chen et al. [11] found that pre-defined lexicons did not work well because that cannot be cover the vast amount of terms occurring across the web. Therefore, they used the web search results to improve the existing query spelling correction models solely based on query logs by leveraging the rich information on the web related to the query and its top-ranked candidate. Their system performed based on real world queries randomly sampled from search engine's daily logs. As they use the web search results to improve the existing system, we also developed our own n-gram database by using seven different social media datasets. By extracting n-gram words from the existing corpus cannot cover word errors in the social media data.

Hodge and Austin [12] developed simple, flexible, and efficient hybrid spell checking methodology based upon phonetic matching, supervised learning, and associative matching in the AURA neural system to correct the isolated word error, particularly spell checking user queries in a search engine. They integrate Hamming Distance (to overcome substitution and transposition errors) and n-gram algorithms (to match small character subsets of the query term). Our existing system also considers the phonetic approaches and our new proposed system used the n-gram approach to choose the most suitable words among the corrected words.

Bassil and Alwani [13] established context-sensitive spelling correction method for detecting and correcting non-word and real-word errors in generic computer text documents. They used Google Web 1T unigram data set to detect non-word errors, Google Web 1T unigram data set and a character-based 2-gram model to generate a list of candidate spellings for every detected error in the text and Google Web 1T 5-gram data set to perform context-sensitive error correction and select the best appropriate spelling candidates. Our proposed system used the ngram database which is developed from the seven social media datasets. Our system can only check and correct the non-word errors currently.

Liu et al. [14] developed a broad-coverage normalization system for the social media language without using the human annotations. It integrates three key components: enhanced letter transformation, visual priming, and string/phonetic similarity. Their system was evaluated on both word- and message level using four SMS and Twitter datasets. They used phoneme-, syllable-, morpheme- and word-boundary based features as similar as our system, namely, Similar Sound Approach (SSoA), Similar Shape Approach (SShA), and Nearby Key Approach (NKA). They did not consider the various sentiment related expressions, such as emoticons, interjections and acronyms. Our system considered them in the preprocessing stage.

3. Social Media Datasets

We used seven public social media datasets to build the n-gram database and used in our experiments. Five datasets have positive and negative sentences and the other two datasets have positive, negative and neutral sentences. All datasets are detailed below:

3.1. Movie Review Dataset (IMDB)

The IMDB dataset [15] is a benchmark dataset for sentiment classification. The task is to determine if the movie reviews are positive or negative. Both the training and test

sets consist of 25K reviews. We used the small dataset that include 500 positive and 500 negative sentences.

3.2. Formspring.me Dataset

Renolds et al. [16] extracted the questions and answer text from the Formspring.me data and they used the Amazon's Mechanical Turk service to determine the labels for their truth sets. From the users' posts of Formspring.me (.xml file) dataset, we extracted questions and answers to text file [17]. In this paper, we used the small dataset, which consists of 800 positive and 800 negative sentences.

3.3. Amazon Dataset

Kotzias et al. [18] selected sentences from the full Amazon dataset [19] that has a clearly positive or negative connotation, the goal was to select no neutral sentence. 500 positive and 500 negative sentences exist in this dataset.

3.4. Yelp Dataset

Kotzias et al. [18] selected random sentences from the full yelp dataset [20] that has a clearly positive or negative connotation, the goal was to select no neutral sentences. For each website, there are 500 positive and 500 negative sentences.

3.5. Twitter Dataset I

Mukherjee and Bhattacharyya [21] created an artificial dataset using hashtags. Twitter API was used to collect another set of 15,214 tweets based on hashtags. Hashtags #positive, #joy, #excited, #happy etc. were used to collect tweets bearing positive sentiments, whereas hashtags like #negative, #sad, #depressed, #gloomy, #disappointed etc. were used to collect negative tweets. Hashtag keywords were removed subsequently from the tweets. We used positive and negative tweets in our experiments of this paper.

3.6. Twitter Dataset II

The data are stored as a CSV and as a pickled pandas dataframe (Python 2.7). There are three classes in the file: hate speech, offensive_language and neither or non-offensive [22]. We extracted the tweets and put them into three different files: hate speech, offensive_language and neither or non-offensive.

3.7. Facebook Dataset

Daniel et al. [23] created dataset with annotations on two independent scales. Valence (or sentiment) represents the polarity of the affective content in a post, rated on a nine point scale from 1 (very negative) to 5 (neutral/objective) to 9 (very positive) and Arousal (or intensity) represents the intensity of the affective content, rated on a nine point scale from 1 (neutral/objective post) to 9 (very high). We used their dataset with 3 classes: Negative (valence point scale from 1 to 4), Neural (valence point scale 5) and Positive (valence point scale from 6 to 9) in our experiments of this paper.

4. Spell Checker and Corrector

There are four steps in our word spell checker and corrector: (i) preprocessing step, (ii) dictionary look up, (iii) word spell checking and correction approaches, and (iv) spell corrector as shown in Fig. 1.

4.1. Preprocessing Step

When we look at social media datasets, they include various word patterns: chat abbreviation, syntax, short-term words, emotional icons, etc. Before doing the word error checking and correction, we transform these words to ordinary words. A detail of the preprocessing step is described in our previous paper [7].

4.2. Dictionary Look Up

After the preprocessing step, we use the dictionary file to check if the word exists in the dictionary or not. The dictionary we used in the proposed system is based on a file composed of 194,438 words [24]. Detail of the dictionary look up step is described in our previous paper [7]. After looking at the dictionary file, the unknown words go through the new lists to word spell check and correct approaches.

4.3. Word Spell Checking and Correction Approaches

Our proposed system consists of total twelve word spell checking and correcting approaches: the first ten approaches were described in our previous paper [7] and two new approaches are described in this paper. The word transformation form of each approach is as shown in Table 1.

The twelve words spell checking and correction approaches are as follows:

1) Plural to Singular Approach (PSA)

2) Removing End Characters Approach (RECA)

3) Removing Middle Characters Approach (RMCA)

4) Reordering Approach (RA)

5) Similar Shape Approach (SShA)

6) Similar Sound Approach (SSoA)

7) Dividing Approach (DA)

8) Removing Extra Character Approach (RExCA)

9) Adding Character Approach (ACA)

10) Anagram Approach (AA)

11) Nearby Key Approach (NKA)

12) Changing Character Approach (CCA)

4.3.1. Nearby Key Approach (NKA)

Sometimes, user can push wrong key when he types quickly without looking at the keyboard. Most of the time the user can type right ones, but he can type keys nearby sometimes and cause misspelled words. This approach is intended to correct this kind of mistake. We employed the same method as SShA and SSoA however each letter has its own group based on an english qwerty key

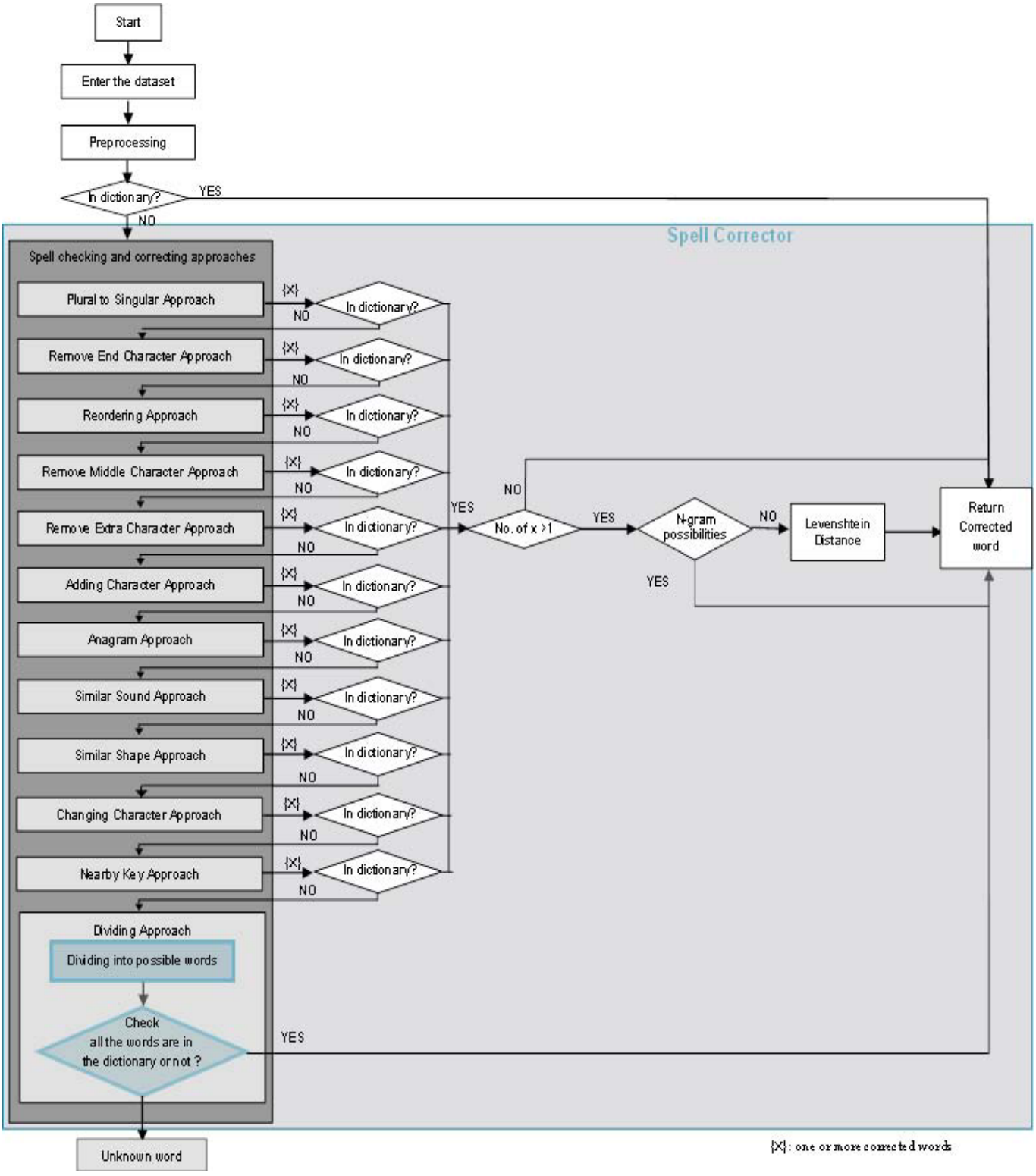

Figure 1. Flowchart of our Spell Checker and Corrector

board. For example, "*q*" group is {"*q*", "*w*", "*a*"}, " *r* " group is {"*e*", "*r*", "*t*", "*d*", "*f*"} and "*g*" group is {"*f*", "*g*", "*h*", "*r*", "*t*", "*y*", "*v*", "*b*" }. In this paper, we consider only the same row nearby key, that is, "*q*" group is {"*q*", "*w*"}, "*r*" group is {"*e*", "*r*"} and "*g*" group is {"*f*", "*g*", "*h*"}. We decided to do so because most of the possible words given by the round nearby keys are misspelled words and computation time takes long.

4.3.2. Changing Character Approach (CCA)

To understand a word, it is not necessary to have all letters as long as we have enough clues to guess it. One way to proceed is to change only a letter consequently as shown in Figure 2. For instance, we found out "bxtch" for "bitch" but "*x*" was not the only character used. Moreover, it was sometimes a random choice that is why we have to try any character.

Category	Approaches	Error Words	Transformed Words	Corrected words
Deletion	PSA	puppys	**puppy**	puppy
	RECA	Treeee	**Tree**, Tre	Tree
	RMCA	saaaaafeeeely	saafeely, saafely, safeely, **safely**	safely
	RExCA	faike	fike, **fake**, faie, faik	fake
Insertion	ACA	nes	anes, naes, neas, nesa, bnes, nbes, nebs, nesb, cnes, nces, necs, nesc, dnes, ndes, neds, nesd, enes, nees, nees, nese, fnes, nfes, nefs, nesf, gnes, nges, negs, nesg, hnes,nhes, nehs, nesh, ines, nies, neis, nesi, jnes, njes, nejs, nesj, knes, nkes, neks, nesk, lnes, nles, nels, nesl, mnes, nmes, nems, nesm, nnes, nnes, nens, nesn, ones, noes, neos, neso, pnes, npes, neps, nesp, qnes, nqes, neqs, nesq, rnes, nres, ners, nesr, snes, nses, ness, ness,tnes, ntes, **nets**, **nest**, unes, nues, neus, nesu, vnes, nves, nevs, nesv, wnes, nwes, news, nesw, xnes, nxes, nexs, nesx, ynes, nyes, neys, nesy, znes, nzes, nezs, nesz	nets, nest
Transformation	SSoA	noyse	noyse, neoyse, nouyse, neouyse, nuyse, neuyse, **noise**, neoise, nouise, neouise, nuise, neuise, noyze, neoyze, nouyze, neouyze, nuyze, neuyze, noize, neoize, nouize, neouize, nuize, neuize	noise
	SShA	nei	nci, nei, noi, nai, ncj, nej, noj, naj, ncl, nel, nol, nal, nct, **net, not**, nat, ncf, nef, nof,naf	net, not
	CCA	yxs	yas, ybs, ycs, yds, **yes**, yfs, ygs,yhs, yis, yjs, yks, yls, yms, yns, yos, yps, yqs, yrs, yss, yts, yus, yvs, yws, yxs, yys, yzs, yxa, yxb, yxc, yxd,yxe, yxf, yxg, yxh, yxi, yxj, yxk,yxl, yxm, yxn, yxo, yxp, yxq, yxr, yxs, yxt, yxu, yxv, yxw, yxx, yxy, yxz	yes
	NKA	pusj	pusj, pysj, pisj, puaj, pyaj, piaj, pudj, pydj, pidj, **push**, pysh, pish, puah, pyah,piah, pudh, pydh, pidh, pusk, pysk, pisk, puak, pyak, piak, pudk, pydk, pidk	push
Separation	DA	nooneknew	**no one knew**	no one knew
Reordering	RA	taht	**that**, that, thta	that
	AA	mxedi	mxedi, mxeid, mxdei, mxdie, mxied, mxide, mexdi, mexid, medxi, medix,meixd, meidx, mdxei, mdxie, mdexi, mdeix, mdixe, mdiex, **mixed**, mixde, miexd, miedx,midxe, midex	mixed

Table 1. Word Transformation for Each Approach. Ten approaches were described in detail in our previous paper [7] and two new approaches, namely, Nearby Key Approach (NKA) and Changing Character Approach (CCA) are discussed in this paper

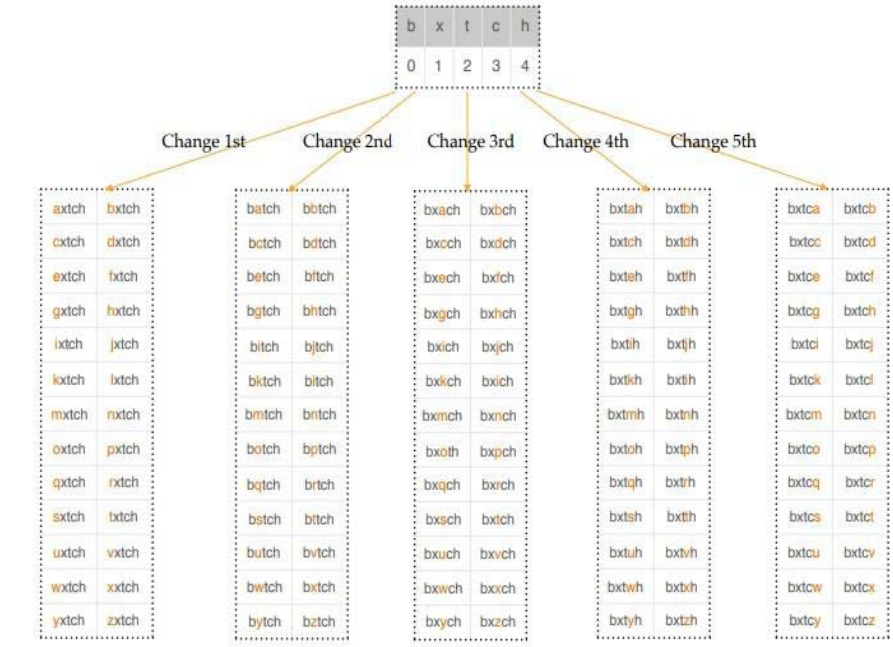

Figure 2. Changing Character Approach

4.4. Spell Corrector

The proposed spell corrector is designed to correct words that have two or more characters. Moreover, in the social media datasets, there are hashtags, pseudonyms, websites links and numbers and we escape these to check and correct. If there are more than one corrected word in one approach, the proposed system uses n-gram approach and Levenshtein distance to choose suitable word. When we used Levenshtein distance in our previous work, we found that some of the word errors could not be corrected to the suitable one. Therefore, we used bigram approach in this paper.

4.4.1. Bigram Approach

We use the seven social media datasets to extract the bigrams words. We used 23K bi-gram words in our proposed system. We used bigram approach to choose the suitable words as shown in Figure 3.

The steps of the bigram approach are as follow:

Step I: If more than one corrected word are found in the dictionary files, check the word position in the sentence or possibilities of word

Step II: If the word is at the first position, go to Step VII.

Step III: If it is not, combine it with the left word

Step IV: Check it's combination is in the Bigram Files or not.

Step V: If bigram word is found, go to Step VIII.

Step VI: If it is not, go to Step VII.

Step VII: Use Levenshtein distance to choose the corrected one.

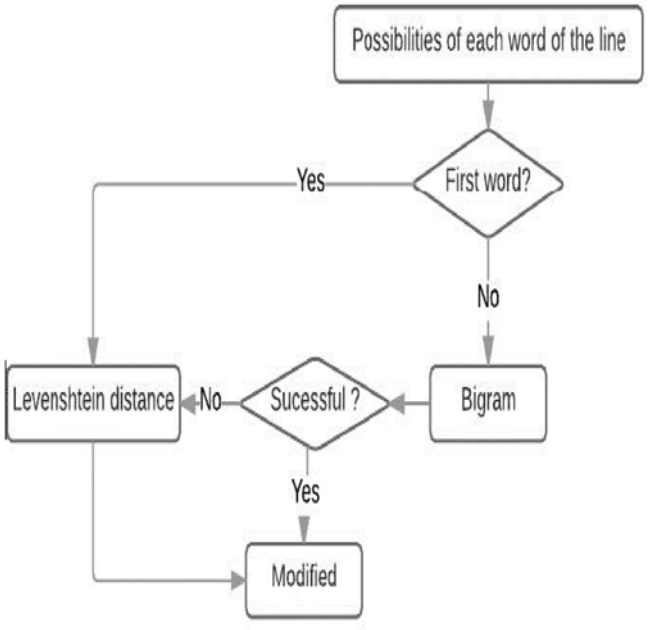

Figure 3. Flowchart of Choosing Suitable Word

Step VIII: Modified the word with chosen word.

5. Experimental Results and Discussions

The experiments of the proposed system and performance results are discussed in this section. We use the seven social media datasets in our experiments as mentioned in Section 3. There are three parts in this section: experiment I is checking and correcting the word errors from the social media datasets, experiment II is choosing the corrected words by using Levenshtein distance and the bigram approach, and experiment III is using the corrected datasets in four different text classifiers.

5.1. Experiment I: Checking and correcting word errors (Levenshtein distance)

We used seven social media datasets with small data files in this experiment. Table 2 shows the word correcting rate for datasets with two classes (positive/no bullied and negative/bullied) and Table 3 presents the word correcting rate for datasets with three classes (positive, negative and neutral).

According to these experimental results, our new spell checker and corrector gave better word correcting rate than our previous spell checker and corrector [7] in every dataset. The average correcting rate for all datasets was 92.7%. Among all the datasets, we got the best correcting rate in Amazon dataset and the worst correcting rate in Movie dataset.

This was because most of the uncorrected words in Movie dataset were names of people or movies. Most of the uncorrected words in Yelp dataset were names of food too. Our spell checker and corrector checked unknown or error word by using dictionary file. It recorded these names of people or food or movies as the unknown words and could not correct them.

We also found that some of the uncorrected words were combinations of two words. The former word of combined word was misspelled, e.g., "peoplbefore" instead of "people before". Our spell checker and corrector tried to correct this kind of word error with Dividing Approach (DA) (described in our previous paper [7]). However, DA checked the former word "peopl" of the combined word "peoplbefore" and "peopl" could not be found in dictionary. Therefore, DA left it as the uncorrected word "peoplbefore".

Some of the uncorrected words were similar sound errors, e.g., "becuz" instead of "because". In this word error "becuz", social media user used "uz" instead of "ause". Our spell checker and corrector with Similar Sound Approach (SSoA) " (described in our previous paper [7]) was able to correct some similar sound errors but it considered one or two letter words similar sounds, e.g., "k" instead of "ke" and "d" instead of "th". So this kind of error words were left as uncorrected words too.

File	Twitter I		Formspring.me		Amazon		Movie		Yelp	
	Positive	Negative	No Bullied	Bullied	Positive	Negative	Positive	Negative	Positive	Negative
Number of words	90234	79523	18174	19578	4718	5074	14563	14770	4948	5522
Hashtags	9516	11073	0	0	1	0	0	0	0	0
Pseudo	3532	3488	0	2	0	0	0	0	0	0
Web	396	923	10	7	0	0	0	0	0	0
Number of unknown words	3547	2868	592	2346	123	110	298	348	82	50
Corrected by Old Approaches	3242 91.40%	2573 89.71%	526 90.37%	2114 92.01%	116 93.64%	103 94.31%	243 81.54%	276 79.54%	72 87.81%	44 88%
Corrected by New Approaches (NKA & CCA)	3333 **93.02%**	2673 **93.96%**	541 **93%**	2189 **95.30%**	119 **96.75%**	107 **97.27%**	254 **85.24%**	302 **86.78%**	76 **92.68%**	46 **92%**
Average Correcting rate	**93.49%**		**94.15%**		**97.01%**		**86.01%**		**92.34%**	

Table 2. The Correcting Rate for Datasets with Two Classes. In these results, we got the best correcting rate in Amazon Negative dataset and the worst correcting rate in Movie Negative dataset. The correcting rate increased at least 1.62% by using new approaches compared to the old approaches

File	Facebook			Twitter II		
	Positive	Negative	Neutral	Hate Speech	Offensive	Neither
Number of words	16910	7258	14155	15386	50386	15917
Hashtags	4	1	2	160	1131	255
Pseudo	0	0	1	1252	3311	1249
Web	0	0	0	85	803	199
Number of unknown words	791	274	696	826	2517	867
Corrected by Old Approaches	701 88.62%	253 92.34%	596 84.20%	729 92.89%	2237 88.88%	760 87.66%
Corrected by New Approaches (NKA & CCA)	739 **93.43%**	265 **96.72%**	628 **90.23%**	749 **95.41%**	2314 **91.94%**	780 **89.97%**
Average Correcting rate	**93.46%**			**92.44%**		

Table 3. Comparison of Correcting Rates for Social Media Datasets with Three Classes. In these experimental results, we got the best correcting rate in Facebook Negative dataset and the worst correcting rate in Twitter II Neither dataset. The correcting rate increased at least 2.31% by using new approaches compared to the old approaches

5.2. Experiment II: Choosing the corrected words by using Levenshtein distance and Bigram

According to the experimental results, the corrected words chosen by bigram approach were more reasonable than by Levenshtein distance. Some of the examples of corrected and chosen words are shown in Table 4.

Approaches	Original Words	Corrected Words in Dictionary Files	Chosen by using Levenshtein Distance	Chosen by Levenshtein Distance & BiGram
RECA	dooo	doo, do	doo	do
	toooo	too, to	too	to
RMCA	tirrred	tirred, tired	tirred	tired
	goood	good, god	good	god
ACA	re	rue, rea, red, rec, ref, ree, reh, reg, rae, rei, ren, rem, rep, reo, ret, res, pre, rev, rex, rew, rez, roe, are, cre, ere, gre, ire, ore, ure, rye are rue	are	rue
	gon	gong, gonk, agon, gone, goon, goan, gown	gong	gone
	thi	thig, this, thir, thio, thin, thai, tich, tshi	thig	this
	evn	evan, envy, even	evan	even
	moring	moringa, mooring, morning, smoring, morling	moringa	morning
SShA	hehehe	hake, haka, hooke, hooka, hebe, hobo, hoke, haboob, haha	hebe	haha
	mommom	marron, marner, manner, marram, manor, manon, manna, meme, memo, meno, mene, mormon, marne, mero, mere, mama, mano, mana, mane, mermen, mamma, merman, moner, moron, mara, morne, mare, marc, morra, memnon, morro, mammon, mammer, mome, mono, mona, mora, more, moro, momma	mormon	mama
	werrkkkkk	wank, wark, womb, work	wank	work
CCA	bxtch	botch, batch, butch, bitch	botch	bitch

Table 4. Corrected Words Chosen by Levenshtein Distance and Bigram

Levenshtein distance chose the corrected word, which has the least Levenshtein distance value differ from the original word. Moreover, Levenshtein distance chose the first word when the corrected words had the same Levenshtein distance value. Therefore, the chosen word was not always reasonable one.

Some example sentences for corrected words are as follows:

For the error word "**dooo**"-
Original Sentence – "YOU CAN **DOOO** IT!"
Chosen by Levenshtein distance – "YOU CAN **DOO** IT!"
Chosen by Bigram and Levenshtein distance – "YOU CAN **DO** IT!"

In this example, our approaches corrected the error word "dooo" to "doo" and "do". Levenshtein distance value of word "doo" differ from original word "dooo" is "1" and Levenhstein distance value of word "do" differ from original word "dooo" is "2". At this time, Levenshtein distance chose the less Levenshtein distance word "doo" as corrected word and the sentence was not the meaningful one. When we used bigram approach to choose the corrected word from these two words, bigram approach chose the word "do" as the corrected word and we got the meaningful sentence.

For the error word "**tirrred**"-

Original Sentence – "so **tirrred**.. to behave? or not to behave....."

Chosen by Levenshtein distance – "so **tirred**.. to behave? or not to behave....."

Chosen by Bigram and Levenshtein distance – "so **tired**.. to behave? or not to behave....."

This example is similar to the first one.

For the error word "**werrkkkkk**"

Original Sentence – "So tired. Nap and then up for a bit for school **werrkkkkk**."

Chosen by Levenshtein distance – "So tired. Nap and then up for a bit for school **wank**."

Chosen by Bigram and Levenshtein distance – "So tired. Nap and then up for a bit for school **work**."

In this example, our approaches corrected the error word "werrkkkkk" to "wank", "wark", "womb", and "work". Levenshtein distance values of all corrected words were same. Therefore, Levenshtein distance chose the first word "wank" as the corrected word and the sentence was not the meaningful one. When we used bigram approach to choose the corrected word from these four words, we got the meaningful sentence.

5.3. Experiment III: Using the corrected files in Text classifiers

We used five of the seven social media datasets in this experiment. We used four text classifiers to test how much higher accuracy rate could be got by using our corrected dataset than the original one.

We used Convolutional Neural Network (CNN) [25], LSTM [26], BiLSTM [27] and CNN-LSTM [28] text classifiers in our experiment. We made 16 different tasks (4 different data files with 4 different classifiers) for each dataset in this experiment and results are shown in Table 5.

According to the results, most of the classifiers got the best accuracy rate by using the corrected datasets except Amazon dataset with BiLSTM classifier. The best accuracy rate got by using the corrected files of five datasets in each classifier is shown in Figure 4.

6. Conclusions

In this paper, we discussed the spell checker and corrector to correct the various kinds of misspelled words, which are included in social media datasets, e.g., Twitter and Formspring.me. We added the new approaches, namely, nearby key approach (NKA) and changing character approach (CCA) and we used Bigram approach to choose reasonable words among corrected words got from dictionary files. As we mentioned in the introduction part, there are two kinds of word errors: non-word and real word error. Our system got **92.7%** average correcting rate overall the datasets and the best one was **97.01%** for amazon dataset. As we expected, our new approaches could correct more words than old approaches and chose the reasonable word by using bigram. We considered just only the non-words errors in this paper and we did not count on the real word errors. According to our experimental results, we got the worst correcting rate on Movie dataset. We analyzed on

		CNN	LSTM	BiLSTM	CNN LSTM
Twitter I	Original	84.93%	81.33%	84.27%	88.8%
	Old Approaches	86.53%	80.54%	81.73%	87.47
	New Approaches	86.93%	84.4%	82.27%	88.93%
	With bi-gram	85.87%	81.73%	86%	90%
FS	Original	94%	48.13%	87.75%	69.63%
	Old Approaches	92.5%	48.13%	88.25%	85.5%
	New Approaches	91.43%	48.48%	85.71%	88.94%
	With bi-gram	94.13%	48.13%	87.13%	89%
Movie	Original	69.63%	55.37%	61.13%	67.13%
	Old Approaches	72.88%	55.75%	71.25%	72.25%
	New Approaches	66.75%	63.88%	60.12%	67.88%
	With bi-gram	66.75 %	63.75%	65.13%	70%
Amazon	Original	76%	71.2%	77.2%	72.2%
	Old Approaches	80%	77.6%	72.2%	78.4%
	New Approaches	78.6%	76.4%	75.6%	80%
	With bi-gram	81%	77.8%	72.8%	73.4%
yelp	Original	83.4%	75.8%	73.2%	81.6%
	Old Approaches	82.6%	76.4%	74.4%	77%
	New Approaches	88%	83.6%	81.2%	83%
	With bi-gram	86.8%	83.2%	78.4%	80.80%

Table 5. Accuracy Rate for Text Classification of Corrected Datasets

uncorrected words of all datasets, and found that our system cannot classify well on movie name, food name, etc. We will use POS tagging in our future work to update our system. As future work, we will upgrade our system for real word errors checking and correcting. We will also use some new approaches to correct reverse words, combined word errors and similar sound errors.

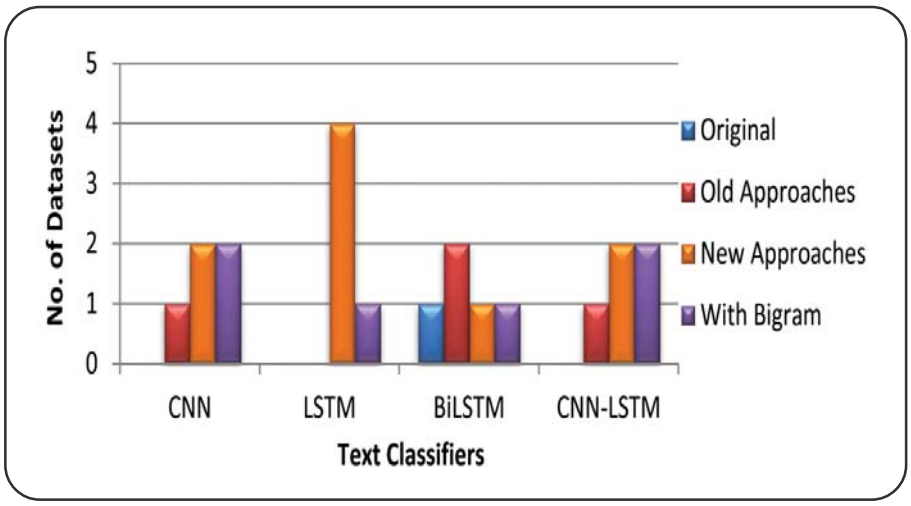

Figure 4. Best Accuracy Rate Got by Using the Corrected Files of Five Datasets in Each Classifier

References

[1] http://www.onlinecorrection.com

[2] Peterson, J. L(1980) Computer Programs for Detecting and Correcting Spelling Errors, *Communications of the ACM,* 23 (12) 676-687.

[3] Peterson, J. L (1986). A note on undetected typing errors, *Communications of the ACM*, 29 (7) 633-637.

[4] Kukich, K(1992) Techniques for automatically correcting words in text, ACM Computing Surveys (CSUR), 24 (4) 377-439 , 1992

[5] Yannakoudakis, E. J., Fawthrop, D(1983). An intelligent spelling error corrector, *Information Processing and Management,* 19 (1) 101-108.

[6] Seth, D., Kokar, M.M(2001). SSCS: A Smart Spell Checker System Implementation Using Adaptive Software Architecture, *In*: Laddaga, R., Shrobe, H., Robertson, P.(2003) Self-Adaptive Software: Applications, IWSAS 2001. Lecture Notes in Computer Science, 2614. Springer, 2003.

[7] Wint, Z. Z., Ducros, T., Aritsugi, M (2017). Spell Corrector to Social Media Datasets in Message Filtering Systems, *In:* Proc. of International Conference on Digital Information Management (ICDIM 2017), p. 215-221, Sept. 2017.

[8] Levenshtein, V. I (1996). Binary codes capable of correcting deletions, insertions, and reversals, Soviet Physics Doklady, 10 707-710, Feb. 1966.

[9] Pande, H(2017). Effective search space reduction for spell correction using character neural embeddings, *In:* Proc. of the 15th Conference of the European Chapter of the Association for Computational Linguistics, p. 170-174, Apr. 2017.

[10] Ahmed, F., Luca, E. W. D., Nurnberger, A(2009). Revised N-Gram based Automatic Spelling Correction Tool to Improve Retrieval Effectiveness, Polibits., 39-48, 2009.

[11] Chen, Q., Li, M., Zhou, M(2007). Improving Query Spelling Correction Using Web Search Results, *In:* Proc. of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007), p. 181-189, Jun. 2007

[12] Hodge, V. J., Austin, J(2003). A Comparison of Standard Spell Checking Algorithms and a Novel Binary Neural Approach, *IEEE Transactions on Knowledge and Data Engineering,* 15 (5) 1073- 1081.

[13] Bassil, Y., Alwani, M(2012). Context-sensitive Spelling Correction Using Google Web 1T 5-Gram Information, *Computer and Information Science* 5 (3) 37-48

[14] Liu, F., Weng, F., Jiang, X(2012). A Broad-Coverage Normalization System for Social Media Language, *In:* Proc. of the 50th Annual Meeting of the Association for Computational Linguistics (ACL), p. 1035-1044, July. 2012

[15] Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., Potts, C(2011). Learning word vectors for sentiment analysis, *In:* Proc. of The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, (ACL 2011*)*. p. 142-150, Jun. 2011.

[16] Reynolds, K., Kontostathis, A., Edwards, L(2011). Using Machine Learning to Detect Cyberbullying, *In:* Proc. of the 2011 10th Conference on Machine Learning and Applications Workshops, V.. 2, p.241-244, Dec. 2011.

[17] Wint, Z. Z., Aritsugi, M(2016). Cyberbullying Detection System for Social Network Sites*, In:* Proc. of AUN/SEED-Net Regional Conference for Computer and Information Engineering (RCCIE 2016), p.283-288, Oct.2016

[18] Kotzias, D., Denil, M., Freitas, N. D., Smyth, P (2015). 'From Group to Individual Labels using Deep Features', *In:* Proc. of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KD*D '15),* p. 597-606, Aug. 2015.

[19] McAuley, J., Leskovec, J (2013). Hidden factors and hidden topics: understanding rating dimensions with review text, *In:* Proc. of the 7th ACM conference on Recommender systems, p. 165-172, Oct. 2013.

[20] Yelp dataset challenge http://www.yelp.com/dataset_challenge

[21] Mukherjee, S., Bhattacharyya, P(2012). Sentiment Analysis in Twitter with Lightweight Discourse Analysis, *In:* Proc. of the 24th International Conference on Computational Linguistics (COLING).

[22] Davidson, T., Warmsley, D., Macy, M., Weber, I(2017). Automated Hate Speech Detection and the Problem of Offensive Language, *In:* Proc. of the 11th International AAAI Conference on Web and Social Media(ICWSM '17), p. 512-515.

[23] Daniel, P.P., Schwartz, H.A., Park, G., Eichstaedt, J.C., Kern, M., Ungar, L., Shulman, E.P(2016). Modelling Valence and Arousal in Facebook Posts, *In:* Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA). NAACL, 2016.

[24] http://www.gwicks.net/dictionaries.htm

[25] Kim, Y(2014). Convolutional Neural Networks for Sentence Classification, *In:* Proc. of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014), .

[26] Bertero, D., Fung, P(2016). A Long Short-Term Memory Frame-work for Predicting Humor in Dialogues, *In:* Proc. of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016), p. 130-135, Jun. 2016.

[27] Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H., Xu, B(2016). Text Classification Improved by Integrating Bidirectional LSTM with Two-dimensional Max Pooling, *In:* Proc. of COLING 2016, p.3485-3495, Dec. 2016.

[28] Zhou, C., Sun, C., Liu, Z., Lau, F.C.M(2015). A C-LSTM Neural Network for Text Classification, CoRR, Nov. 2015.

Author biographies

Zar Zar Wint has been a doctorate student in Computer Science at Kumamoto University, Japan since 2015. She is working on "Spell checking and correction" and "Message filtering for cyberbullying cases". She got her BE(IT) in 2002 from Mandalay Technological University and ME(IT) in 2004 from Yangon Technological University, Myanmar. She served as a researcher at Material Science and Material Engineering Department from 2006 to 2011 and UTYCC from 2011 to 2014. Her research interests are NLP and machine learning. She was the member of ASEAN Cyber University Project from 2012 to 2014. She is a student member of IEEE.

Théo Ducros is currently undertaking his last year of engineering school in mathematics and modelling that he is completing by an international master 2 in computer science at University of Clermont Auvergne. After growing up and graduating from high school in the southwest of France, he moved to Clermont-Ferrand where he followed the preparation course in both sciences and languages. He had the opportunity to discover Japan and work on this spell checker and corrector system for his fourth-year internship at Kumamoto University from May to July 2017.

Masayoshi Aritsugi received his B.E. and D.E. degrees in computer science and communication engineering from Kyushu University, Japan, in 1991 and 1996, respectively. From 1996 to 2007, he was with the Department of Computer Science, Gunma University, Japan. Since 2007, he has been a Professor at Kumamoto University, Japan. His research interests include database systems and parallel/distributed data processing. He is a senior member of IEICE and IPSJ, and a member of ACM, IEEE, and DBSJ.