Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 144 (2018) 105–117

Procedia Computer Science

www.elsevier.com/locate/procedia

INNS Conference on Big Data and Deep Learning 2018

Context-sensitive normalization of social media text in bahasa Indonesia based on neural word embeddings

Renny Pradina Kusumawardani*, Stezar Priansya, Faizal Johan Atletiko

Department of Information Systems, Faculty of Information and Communication Technology, Institut Teknologi Sepuluh Nopember, Surabaya 60111, Indonesia

Abstract

We present our work in the normalization of social media texts in Bahasa Indonesia. To capture the contextual meaning of tokens, we create a neural word embeddings using word2vec trained on over a million social media messages representing a mix of domains and degrees of linguistic deviations from standard Bahasa Indonesia. For each token to be normalized, the embeddings is used for generating candidates from vocabulary words. To select from among these candidates, we use a scoring combining their contextual similarity to the token as gauged by their proximity in the embeddings vector space with their orthographical similarity measured using the Levenshtein and Jaro-Winkler distances. For normalization of individual words, we observe that detecting whether a token actually represent an incorrectly spelled word is at least as important as finding the correct normalization. However, in the task of normalizing entire messages, the system achieves a highest accuracy of 79.59%, suggesting that our approach is quite promising and worthy of further exploration. Furthermore, in this paper we also discuss some observations we made on the use of the neural word embeddings in the processing of informal Bahasa Indonesia texts, especially in the social media.

Keywords: Social Media; Bahasa Indonesia; Word2Vec; Normalization; Word Embeddings; Deep Learning

* Corresponding author. Tel.: +62-31-5999944; fax: +62-31-5964965.
E-mail address: renny.pradina@gmail.com, renny@is.its.ac.id

1. Introduction

Texts in the social media offers a rich source of information and insight into current events and people's opinions. However, they are known not to follow conventional rules of languages, due to their brevity, often due to a constraint on their length, and the users' need to convey information and emotion as quickly and expressively as possible. The deviations occur not only in the grammatical structure, but notably, also in the way various words are spelled. As the variations to the standard language is often dependent on the domain, NLP models used for processing social media texts are usually trained for a specific domain and application, and perform poorly outside their original purpose. This limits the applicability of language resources developed for social media texts. Furthermore, it also limits the incorporation of NLP tools trained on standard language variety.

One method which allows a more generalized processing is to normalize social media texts into a standard form of the language, which is the aim of this paper. Specifically, we attempt to normalize tokens found in social media texts in Bahasa Indonesia to a standard word with the same semantic meaning. Since the understanding of human readers are often determined by the context words surrounding the token, we use vector representations resulting from neural word embeddings which take words in a proximity of the token into consideration. The embeddings are trained on over a million tweets from Indonesian accounts using primarily Bahasa Indonesia, representing a mix of different domains and degrees of language deviations. A standard word could only be selected as a substitute if its embedding representation has a high similarity to the token. A substitute to the token is then selected from among these contextually similar words based on its orthographical similarity to the token. From our experiments, we find that embeddings trained against the goal of modeling with CBOW gives better performance for word normalization when compared to Skip-Gram. This, in combination to other elements of our method, gives the best accuracy of 79.56% in the normalization of entire messages. However, in the task of normalizing individual tokens, it is very important to first ensure, at least to some degree, that the tokens are actually incorrectly spelled words, instead of proper nouns or words of another language.

This paper is organized as follows. After this introduction, some related works are briefly discussed, followed by a short exposition on neural word embeddings. We then discuss the architecture of our normalization system, with more in-depth discussion on the data and the learning of word embeddings, the creation of the Indonesian lexicon, and the scoring used to select from the normalization candidates. This is followed by a discussion on the performance of the system, both in normalizing individual words and in the normalization of entire messages. Finally, we discuss some conclusions of this work and suggest some venues for future work.

2. Related Works

The linguistic style adopted by users of social media makes it difficult to process the messages automatically, as they are very divergent in the degrees of conformity to the standard variant of the language [1]. Of these variations are spelling errors, ad hoc abbreviations, emoticons, ungrammatical structures and substitution of letters with other phonetically, or even visually, similar characters. This makes it difficult for general NLP tools to handle these messages [2]. Furthermore, these linguistic variations also hinder readers' understanding of the information contained within [3]. Furthermore, text normalization is vital for tasks dependent on string matching or word frequency statistic, and in some other cases, model re-training is simply insufficient for the task at hand [4].

In what it does, which is substituting tokens in the text with their standard counterparts, text normalization is akin to spelling correction. This is commonly done using some distance measures between the token and the correction, for example the Levenshtein distance (see, for example, Lhoussain, et. al. [5] for usage in combination with bi-gram models for Arabic text). There are also applications to social media texts, e.g. by Sorokin and Shavrina [6], combining Levenshtein distance with phonetic similarity. Besides these, other methods include substitution by word list and string patterns [7], and finite-state transducers [8].

Other researchers employ context-dependent nature of the meaning of the word to hinge between a token and its counterpart word in the standard form of the language. Wang et. al. [4] assume and exploits context collocation tokens to derive rule-based and statistical features which are then fed into classifiers. Han, et. al. [9] exploits context similarity to build a normalization dictionary, exploring various measures such as the Kullback-Leibler (KL) divergence and the Jensen-Shannon (JS) distance, selecting from the normalization candidates based on their orthographical similarity using various edit distance. They also applied their approach to Spanish text [10], and elaborate their idea on lexical normalization further in [2].

A more recent, and still rarely explored, approach to text normalization is by using the distributed representations of words, i.e., word embeddings. The first to do so is Sridhar [11], in which distributed representation of words is learned to create a normarlization lexicon, subsequently stored as finite-state machines (FSMs). An approach more similar to ours is by Bertaglia and Nunes [3]. As in our approach, Bertaglia and Nunes uses word embeddings combined with an orthographical measure of tokens similarity. However, while they use a modified LCSR (Longest Common Sequence Ratio), our similarity scoring combines word embedding with the Levenshtein [12] and Jaro-Winkler [13] [14] distance, which emphasizes similarity at the beginning of two tokens, and is supposed to be better at comparing short strings [15] such as the ones employed in social media texts.

3. Neural Word Embeddings

In text processing, each word is usually represented by its presence or absence in a document, tf-idf, or other single scalar measure of its property relative to the document and/or a corpus. A different approach is to use a distributed representation introduced by Hinton, et. al. [16], in which each word is represented by a continuous vector. This approach gained traction by the publication of a paper by Mikolov, et. al. [17], in which they introduced techniques to learn word vectors from massively large datasets of the order of billions of words, with vocabulary size in the order of millions. Prior to this, distributed word representation has already been shown, for example by Collobert and Weston [18], to significantly improve NLP applications. However, Mikolov, et. al. introduces models which preserves linear regularities among words, a test for measuring syntactic and semantic regularities, and demonstrate that it is possible to learn such regularities with high accuracy, which is vital to their adoption by the NLP community. The vector representations of words learned by the models are often referred to as word embeddings. In the last couple of years, word embeddings has become the prevalent approach to word representations, especially in the research community where its usage dominates works in major NLP conferences.

The models, continuous-bag-of-words (CBOW) and continuous Skip-gram, learns as follows. Basically, both are a kind of language modelling with architecture similar to that of a feedforward neural network, but without the nonlinear hidden layer and where all the words share the projection layer. In CBOW the language modelling task is to predict a word from its preceding words without regard to the ordering of these words. Words following the predicted words could also be taken into consideration, thus in CBOW, both the history words and the future words could be used as the context. Unlike standard bag-of-words, each word in the context is represented by continuous vector. Conversely, in the Skip-gram model the current word is used to predict the context words. The architectures of CBOW and Skip-gram are shown in Figures 1(a) and 1(b), respectively.

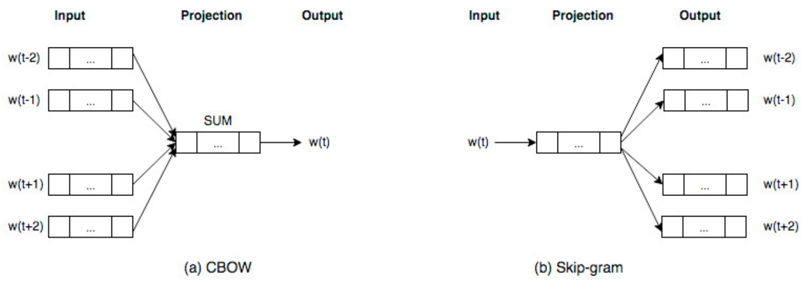

Fig. 1. Architectures of the (a) CBOW and (b) Skip-gram models [17].

A very interesting property of the embedding vectors produced with these models is that they capture the semantic similarity between words. For example, performing cosine similarity to the to pre-trained vectors trained on the Google News dataset released by the Google Word2Vec project [19] with the query word of 'france', results in ‘spain’, ‘belgium’, and ‘netherlands’, as the top three most similar words. These words and their cosine distances are shown in Table 1.

Table 1. Example of the semantic similarity captured in the embedding vectors: top three words with the largest cosine similarity to 'france' [19].

Word	Cosine distance to ‘france’
spain	0.678515
belgium	0.665923
netherlands	0.652428

Table 1 shows that although the embeddings were created without explicit inclusion of any semantic information, it is very plausible that the country France should be considered similar to Spain, Belgium, and Netherlands, since they are all European countries, members of the European Union, and have similar levels of economic prosperity. They are even located near each other. This verisimilitude of understanding of semantic meaning of words contained by the embedding vectors is derived solely from words and their contexts in various sentences. Indeed, words shall be known by the company they keep [20].

Therefore, the main idea of this work is as follows: if word embeddings could capture contextual similarities, then since alternate spellings having the same semantic meaning should be surrounded by the same words, we should be able to find the proper form of a non-formally spelled word by finding a ‘dictionary’ word which embedding vector is the closest to that word. To obtain the word embeddings, we shall train CBOW or Skip-gram models with social media text, which is known to contain many variations of word spellings. Specifically, in this work we aim to do this word normalization to social media text in Bahasa Indonesia, with texts from social media accounts which mainly uses Bahasa Indonesia.

4. Normalization System Architecture

The steps of by which we perform word normalization is as shown in Figure 1. Input texts are lowercased and tokenized, and for each token the normalized version is generated based on the condition which matches the token. The first one is if the token is already recognized as a correctly spelled word, or if it consists of a single character, which in our data very rarely has a meaning. In these cases, the token is returned unchanged. For determining whether the token is a legitimate word, we use a combination of several sources to come up with a quite extensive dictionary of formal Indonesian words. The process of the creation of this lexicon of formal words is elaborated further in 4.2 on Indonesian Lexicon.

When the token is neither a correct word nor trivially consists of only a character, the next check is whether the token is in a lookup table of tokens which appropriate correction word is already known. In this case, the corresponding word is returned. Basically, the function of this table is to store the mapping of tokens to words from previous correction generation processes to increase the efficiency of finding a correction for a token. Basically, this lookup table serves as a sort of memoization.

If the token does not exist in the list of previously known correction mapping, then we query the word embeddings model for candidate words most semantically similar to the token. These candidate words must be contained in the Indonesian lexicon prepared in Section 4.2, to ensure that we do not replace a non-formal word token with another non-formal word. However, sometimes the model does not generate any such replacement candidates for a token. We observe that in almost all cases, this occurs when the token is a proper noun. Obviously, in such cases, it is most appropriate to just return the token unchanged.

When the model does produce several candidates, it will then be necessary to select the best one. As described in section 4.1, the word embeddings used in this word normalization is trained on the language modelling task, and as an effect, is able to capture contextual similarity between words, which often entails semantic similarity.

However, since misspellings, both deliberate and intentional, are generally a orthographical variation to the correct version of the word, we combine the contextual similarity captured by the word embeddings with two edit-based distance measures, the Levenshtein [12] and the Jaro-Winkler [13] [14] distances, to come up with a similarity score described in section 4.3.

We then select the highest scoring word as the candidate for replacing the token. However, we found that due to the grammatical structure of Bahasa Indonesia, it is very common for two variations of a lemma to be surrounded by similar words. For example, consider the sentences:

(1) Fathimah sayang adiknya – 'Fathimah loves her brother'

(2) Fathimah disayang adiknya – 'Fathimah is loved by her brother'

Note that the first and the second sentences in Bahasa Indonesia are similar but in one word, makan and dimakan. This is a very common in Bahasa Indonesia, where an addition of prefix or/and suffix in a single word, usually the verb, changes the meaning of the sentence entirely, without the need to modify the word order or any other aspects of the sentence. Note also that the modification often simply takes the form of concatenating an affix to its appropriate place at the beginning or the end of the word, without much modification to the lemma. These are the reasons why a variation of a lemma could have the highest similarity score to another variation. In such cases, in this work we opt to use the lemma of the word, since in our data this is often the most appropriate approach.

Another interesting case is when the candidate word is a noun in the plural form. In Bahasa Indonesia, the only thing that needs to change in a sentence with a plural noun, even when the noun acts as the subject of the sentence. Thus, it should not be surprising that the plural form of a noun would be suggested as a substitute of the noun, and vice versa. In such cases, the singular lemma form would be returned.

Lastly, when the token does not fit all the above, the highest-scoring candidate word would be returned without modification as the replacement of that token.

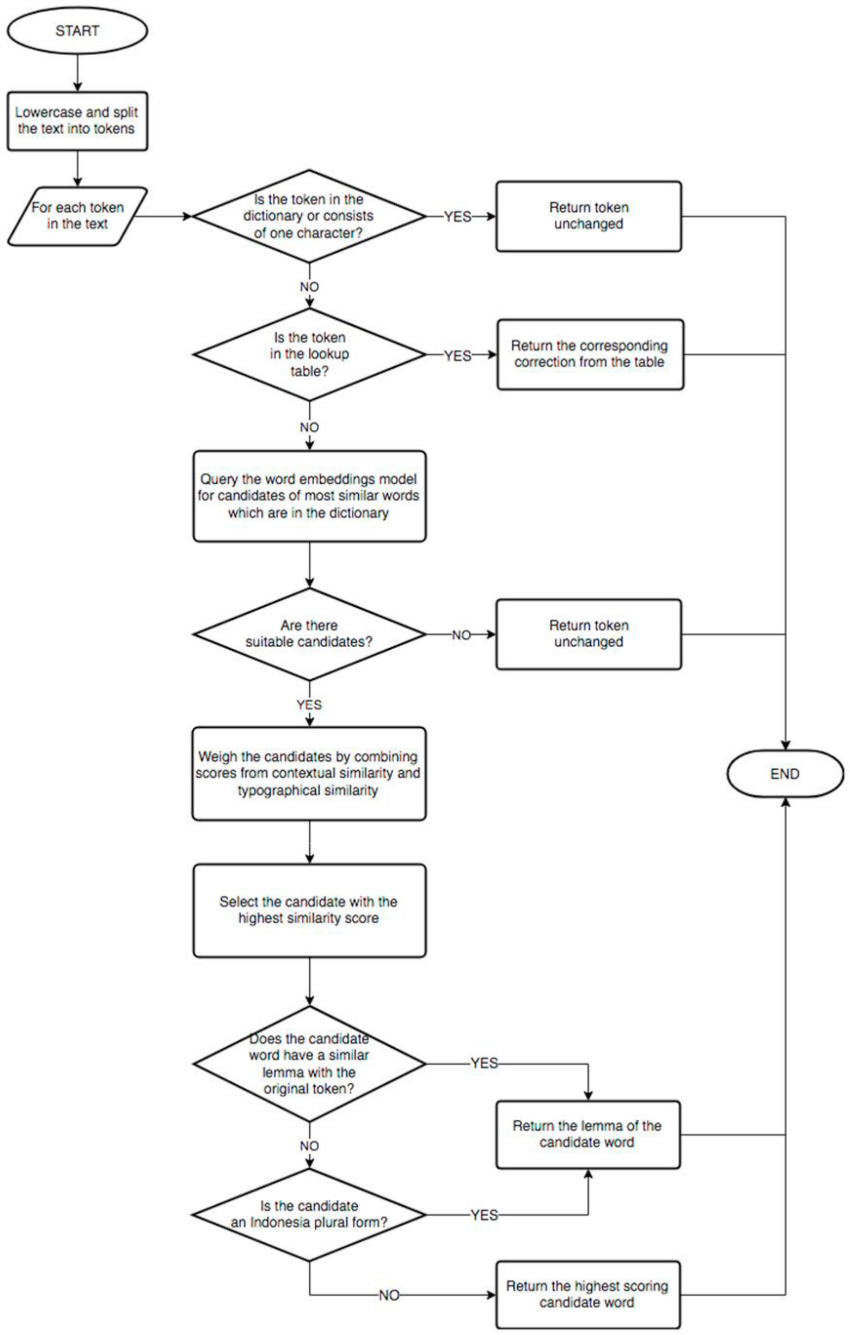
START
Lowercase and split the text into tokens
For each token in the text
Is the token in the dictionary or consists of one character?
YES
Return token unchanged
NO
Is the token in the lookup table?
YES
Return the corresponding correction from the table
NO
Query the word embeddings model for candidates of most similar words which are in the dictionary
Are there suitable candidates?
NO
Return token unchanged
YES
Weigh the candidates by combining scores from contextual similarity and typographical similarity
Select the candidate with the highest similarity score
Does the candidate word have a similar lemma with the original token?
YES
Return the lemma of the candidate word
NO
Is the candidate an Indonesia plural form?
YES
NO
Return the highest scoring candidate word
END

Fig. 2. Steps in the normalization system.

4.1. Data and learning of word embeddings

We train the word embeddings used in this work using the messages from 40 social media accounts which post mostly in Bahasa Indonesia. Of these 40 accounts, some are Twitter and the others are Facebook accounts. From the Facebook accounts, we collect posts dating from August 1, 2015 to March 25, 2017, resulting in 361,494 unique posts. From Twitter, tweets are collected from October 7, 2016 to March 25, 2017, yielding 828,391 unique tweets. Thus, our dataset contains a total of 1,189,885 unique social media messages.

This data is then used to train language models producing word embeddings, commonly referred to as word2vec. The specific word2vec implementation we use is Deeplearning4j [21]. In order to produce the best embeddings for our word normalization purposes, we vary several parameters: the architecture of the model, the training algorithm, the number of epochs and iteration, minimum word frequency, and the size of the context preceding and following the word. With the consideration that a good model should be able to retrieve the most appropriate correction within a relatively small number of candidates, and that it should be able to do so for the most frequent non-normally spelled words, we test the performance of the models on 100 most frequent, non-lexicon tokens, generating ten correction candidates for each. The scoring we use for each model is shown in Equation 1.

$$score_{model} = \sum_{1}^{100} 1 - \frac{i}{10} \quad (1)$$

where *i* is the index of the position of the correct replacement word of the token out of the ten most similar words returned by the model, having the value in the range of 0 to 9. For example, if the model returns the correct normalization as the most similar word, i will have the value of 0, thus the contribution of that token to the model's total score is one. When the model returns the correct word as the tenth most similar word, *i* will have the value of 9, resulting in a contribution of 0.1 to the total score. Table 2 shows the results of experiments on eight combinations of parameters, chosen under the heuristic of giving the best results.

Table 2. Result of experiments on various combination of parameters for word embeddings creation.

Params	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8
Architecture	Skip-gram	CBOW	Skip-gram	CBOW	Skip-gram	CBOW	Skip-gram	CBOW
Training algorithm	Negative sampling	Hierarchical softmax	Negative sampling	Hierarchical softmax	Negative sampling	Hierarchical softmax	Negative sampling	Hierarchical softmax
# of epochs	1	1	10	10	1	1	10	10
# of iters	1	1	10	1	5	5	3	2
Min. word frequency	5	5	5	10	5	5	5	5
Context size	5	5	5	5	100	100	10	5
Score	4.5	17.5	4.5	23.5	5.4	24.9	2.9	25

From Table 2, it is quite apparent that the CBOW model gives the best result for our word normalization task, almost regardless of other factors such as the number of epochs and the size of the context. Therefore, for the following experiments CBOW will be used, with values of other parameters in accordance to Model 8 in Table 2, since this model produces the highest score, albeit the rather insignificant improvement contributed by these other parameter values when compared to the model architecture used.

4.2. Indonesian Lexicon

To avoid replacing an incorrectly spelled word with another word of non-standard spelling, or worse, an already correctly spelled word with other word, a lexicon of standard words is necessary. The first obvious source for such words is the Kamus Besar Bahasa Indonesia (KBBI), the official dictionary of the Indonesian language, a work of the Center for Language Development of the Indonesian Ministry of Education and Culture, which records the formal version of words. However, it is possible in Bahasa Indonesia to generate other formal words by adding prefixes, infixes, and suffixes, according to some rules. In fact, this is one of the most important means by which words in Bahasa Indonesia gain different yet related meanings to the lemma. The KBBI does not record all of these variants.

On the other hand, the addition of affixes of words is not a simple matter of concatenating them to the lemma. Therefore, to at least be able to capture common variants of words, we use Kateglo [22], which is an online dictionary, thesaurus, and glossarium of Bahasa Indonesia, with the source of the 3rd and 4th edition of the KBBI and public contribution. Additionally, we also add entries from the Indonesian Wiktionary [23], primarily to obtain derivations of words in common use. Lastly, to get more of those derivations, we programmatically generate word derivations by adding affixes. However, these derived words might not be correct, or be used by native speakers of Bahasa Indonesia. To check whether these words are at the very least commonly occurring in naturally generated texts, we use the "did you mean" feature of Google Translate [24], wherein the suggested variant, instead of the automatically generated word derivation, will be added to the lexicon when such suggestion is given by the feature. From these sources, the lexicon contains 82,642 words. Additionally, it also has 5,205 mappings of common informal words to their formal version. These mappings comes from several websites, such as ebahasaindonesia [25] and kbbi.web.id [26]. All these comprise sources for the lexicon, which is then indexed using Apache Solr [27].

4.3. Candidates Similarity Scoring

While the context of a token gives important clues as to what the token means, and word embeddings could be exploited to find words that are syntactically and semantically similar even for irregularly spelled words, for the actual task of word normalization we need to find the actual word meant by the author of the text. For this purpose, the letters in a token is a very important clue to what that correct word is. Therefore, in selecting the candidates proposed by the word embeddings model, we employ measures of orthographical similarity. In our work, we use a combination of the Levenshtein [12] and the Jaro-Winkler [13] [14] distance. The Levenshtein distance is based on the number of edits (insertions, deletions, and substitions) needed to transform one of the strings under comparison to the other. Additionally, the Jaro-Winkler distance is included since it is supposedly good for comparison of short strings, therefore it is expected to be suitable for handling tokens which are irregular abbreviations of a formal word, which we often see in social media texts. Our experiments confirm that the addition of the Jaro-Winkler distance indeed increases the correctness of the replacement candidate selection.

However, since these distances are measures of dissimilarity, we normalize both of this measures to 0 to 1, then subtract that from 1 to get similarity scores. This is shown in Equation 2, where the subscript distance denotes either the Levenshtein or the Jaro-Winkler distance.

$$similarity_{distance} = 1 - normalized_{distance} \tag{2}$$

We then combine these similarity scores with a score reflecting the similarity of the candidate word to the query token calculated using Equation 3.

$$similarity_{WEcandidate} = 1 - \frac{i}{10} \tag{3}$$

Similar to Equation 2, in Equation 3 i is the index of the candidate word out of the 10 words most similar to the query token generated by the embeddings model, with 0 denoting the most similar word and 9 the tenth most similar.

Combining Equations 2 and 3 using Equation 4 produces the final similarity score, as follows.

$$score_{candidate} = 0.5 * similarity_{WEcandidate} + 0.25 * similarity_{Levenshtein} + 0.25 * similarity_{JaroWinkler} \tag{4}$$

Finally, having calculated this score for each of the 10 top candidates generated by the word embeddings model, we select the one with the highest score as the normalization for the query token.

5. Performance of the System

We test our word normalization method under two scenarios: first, how well the resulting system performs normalization on individual tokens not found in our Indonesian lexicon. Second, given complete messages, we measure whether it correctly treats each token in the message. Note that for both scenarios, we use the settings of Model 8 in Section 4.1.

5.1. Normalization of individual words

For the first test, we filter all non-lexicon tokens in the corpus and sort them based on their frequency. Testing is then performed on 1000 most occurring tokens by comparing the replacement proposed by the system, with the correct normalization manually produced. Table 3 shows examples of the normalization results.

Table 3. Examples of token normalization by the system.

Input token	Proposed correction	Correct normalization	*English meaning*	Correction status	Similarity score			
					Embeddings	Levenshtein	Jaro-Winkler	Overall
karna	karena	karena	*because*	correct	1	0.833	0.961	0.949
ujan	hujan	hujan	*Rain*	correct	1	0.800	0.933	0.933
dri	dari	dari	*From*	correct	1	0.75	0.925	0.919
adlh	adalah	adalah	*am/is/are*	correct	1	0.667	0.911	0.894
lalin	lalim	lalu lintas	*Traffic*	incorrect	0.9	0.800	0.920	0.880
one	monel	one	*One*	incorrect	1	0.600	0.867	0.867

Out ouf the 1000, we found that the system normalizes correctly on 441 tokens and incorrectly on the rest. Analyzing the errors, we found several causes. First, many of these errors occur when the tokens normalized are actually proper nouns – while they are obviously not in the lexicon of Indonesian words, these should be left as they are. It is therefore necessary to integrate a way for distinguishing proper nouns to misspelled words. A simple NER which focuses on orthographical features might be useful in this case. We could also hypothesize that when modelled at the character level, names have different distributions to words, therefore learning and integrating character level embeddings might be useful.

Second, as shown in Table 3 for the token 'lalin', the model currently is unable to handle tokens which are abbreviations of two words or more. Where 'lalin' is an informal abbreviation for the phrase 'lalu lintas' – traffic, it is normalized to 'lalim' – despotic, a legitimate Indonesian word of a totally unrelated meaning.

Third, the system also fails when the token is a foreign word, most frequently English, for example the word 'one' in Table 3. To handle such cases, it might be necessary to integrate a way for recognizing foreign words, perhaps by incorporating the likelihood that the token is Indonesian or not into the model. It is interesting to consider whether the distributional properties of character level embeddings in different languages could be a helpful clue in distinguishing tokens of different languages.

However, we should like to note that there is a strong correlation between the similarity score of a candidate to the token, to it being the correct normalization of the token. For example, when sorted according to their overall similarity scores, out of the 100 tokens which proposed normalization has the highest similarity score, only four are incorrect, corresponding to an accuracy of 96%. We observe that the likelihood of tokens being incorrectly normalized increases with the decrease of the similarity score of their highest scoring correction candidate. This suggests that in addition to correcting for the three reasons for errors discussed, adding more data into the dataset might also be helpful, since this would allow the model to find more appropriate candidates to be considered.

5.2. Normalization of entire messages

We measure the normalization accuracy of the system to complete messages to understand its performance more comprehensively, since it is as important for the system to recognize which tokens needs normalization and vice versa, and to treat them accordingly. Therefore, we input 1000 random messages and compare the output of each token to the manual correction. Additionally, we observe that the word or token returned by the system in some cases might vary when different threshold values are used; i.e., when we require that a candidate must have at least some minimal similarity score to the input token in order to be considered. Therefore, we also note the influence of varying the value of this threshold. The result is summarized in Table 4.

Table 4. Result on entire message normalization under the influence of various threshold values.

Category	Score Threshold					
	T_65	T_70	T_75	T_80	T_85	T_90
Success	**15432**	**15438**	**15412**	**15400**	**15307**	**15140**
• Token is found in the Indonesian lexicon	14385	14385	14385	14385	14385	14385
• A mapping of token to a formal word exists in the current lookup table	560	560	560	560	560	560
• The model + similarity weighting finds the token's correct normalization	487	493	467	455	362	195
Failure	**3972**	**3966**	**3992**	**4004**	**4097**	**4264**
• The model + similarity weighting fails to generate a normalization	481	1246	2372	3103	3482	3782
• The generated normalization is a lexicon word, but is incorrect	3381	2574	1402	617	281	108
• The generated normalization is both incorrect and is not a lexicon word	110	146	218	284	334	374
Accuracy	**79.53%**	**79.56%**	**79.43%**	**79.37%**	**78.89%**	**78.03%**

For all threshold values, the accuracy is about 78 to 79.5%, with the highest of 79.56% achieved at a similarity threshold of 70%. As it happens, it appears that most tokens in our test set are already in correct spelling form, thus emphasizing the importance of having a comprehensive way of distinguishing between tokens which need normalizing and those which are not, in this case by having a thorough lexicon.

As for threshold values, we see from Table 4 that up until a certain point, lower threshold value corresponds to higher accuracy. We observe that lowering the threshold value decreases the chance that the system would fail to generate a normalization candidate. However, the quality of the candidates also decreases, shown by the larger number of incorrect normalization. For our test data, at a threshold of 65% the benefit of generating more candidates has been entirely offset by the loss caused by incorrect normalization. Conversely, increasing the threshold reduces the chance of getting incorrect normalization, but making it more difficult to generate candidates, that in some cases no normalization candidates might be generated at all. The balance between these counteracting influences in our test data is achieved with the threshold value of 70%, at which the system performance is maximal. While the issue of having more candidates v.s. candidate quality seems to be inevitable, it will indeed be interesting to observe whether increasing the size of the training data would increases the likelihood of the system to find the appropriate normalization candidate, thus increasing the overall performance of the system.

6. Conclusion and Future Work

In this paper, we have described our work in normalizing social media texts in Bahasa Indonesia. We use neural word embeddings to contextually understand tokens to generate formal versions of that token, from which the most orthographically similar candidate is selected as the normalization. The main idea is despite the irregularity of word spelling used in social media texts, the use of word embeddings allows tokens to be mapped to their correctly spelled versions by considering the context surrounding them. We found that in comparison to Skip-gram, CBOW models performs better in generating normalization candidates, with negligible contribution of other factors such as number of training epoch and iterations, and the size of the context. In selecting from the normalization candidates, we combined the similarity score from the word embeddings with normalized scores from the Levenshtein and the Jaro-Winkler distances of the candidates to the token, capturing the orthographical similarity between the candidates and the token.

In normalizing 1000 most occurring non-lexicon words, the system correctly normalized 441 tokens and incorrectly on the remaining 559. However, many of these errors did not occur due to the system's failure in selecting the appropriate correct form of the input token. Instead, in many cases the query token is either a proper noun or a foreign word, which should not be normalized at all. Additionally, when the input token is an informal abbreviation, the system would erroneously generate a single-word normalization. Indeed, when the input token is an informal variant of an Indonesian word, the system successfully assigns the correct normalization as the replacement of the word. This underlines the importance of being able to distinguish cases in which normalization is the appropriate response to formalize a token which is not in the reference lexicon. We would like to note that the weighting combination approach between word embeddings and orthographical features seem to work quite well, in that higher scores do correspond to the selection of the appropriate normalization – for example, amongst the top 100 highest scoring normalizations, only four are incorrect.

In the task of normalizing entire messages, the system achieved the highest accuracy of 79.56%, under a similarity threshold of 70%. We observed that lowering the required similarity threshold of candidates would result in higher accuracy, up until a point when the benefit of getting more normalization candidates is surpassed by the risk of generating incorrect normalization, which is prevented when higher threshold values are imposed.

To the best of our knowledge, this work is the first employing word embeddings for normalization of social media text in Bahasa Indonesia. The prevailing popular approach in such cases is to use manually created word list and string pattern rules [7]. With this work, we seek to provide a more comprehensive solution to normalizing the endlessly varying usage of word spellings in the social media, especially in Bahasa Indonesia, and additionally, to understand the challenges in doing so. Some that we have discussed and we consider as promising venues for future works are as follows. First, we note that CBOW models give superior performance when compared to the Skip-gram in our normalization case. Indeed, it might be the case that CBOW is simply more similar how human thinks in terms of inferring the meaning of tokens, i.e., from its surrounding contexts, instead of Skip-gram, which predicts the context instead of the token itself. However, this also mean that CBOW has a smoothing effect over distributional information, which is beneficial for smaller dataset. Thus, we should like to note whether CBOW retains its superiority over Skip-gram on a large dataset.

Having more data could also potentially addresses the issue of candidate generation. We note that the reason of why lowering the similarity threshold results in higher accuracy is because a lower threshold allows more candidates to be generated. Thus, it might not be an issue of the threshold value itself, but whether the data has more alternative words to be considered. It is therefore very reasonable to suppose that more data, which means more words, and additionally, more context information, would result in better normalization performance.

The problem of recognizing proper nouns as one of the cases in which word normalization is not appropriate could potentially be solved by using a simple Named-Entity Recognition, probably with a more emphasize on case features. Furthermore, it is interesting to see whether incorporating character-level embeddings might be helpful in distinguishing proper nouns from other types of tokens, since intuitively, in many languages names are often formed from distinctive sequences of characters which are quite different to those forming words. Similarly, character-level embeddings may also be useful in recognizing whether a token is more likely to be an Indonesian word or not. Note that since we work with social media data, the use of character-level embeddings is particularly appealing, since both the grammatical structure and the word spellings often do not adhere to standard language rules, thus thwarting the work of systems developed under the assumption of standard language. An example of a work which uses such character-level embeddings for language identification is by Jaech, et. al. [28].

Lastly, in solving the problem of mapping tokens which are abbreviation of phrases, we could take the approach of Srihard [11], in which the training data is modified so that compound words could be treated as a single token; e.g., "i love you too" is replaced with "i_love_you_too". However, as Srihard has pointed out, this approach requires a list of such phrases. Since unsupervised phrase induction rarely capture phrases from high frequency function word, it might be necessary to obtain such list by manual tagging, which is challenging due to the perpetually growing and changing nature of phrase abbreviations used in the social media. Alternatively, we could explore the use of embeddings which consist not only of a single word, but of multiple words. However, since doing so might potentially increase the size of the vocabulary exponentially, it might be necessary to impose some limits, for example the number of the words in the phrase, or a minimal frequency the phrase must appear in the corpus.

Acknowledgements

The authors would like to thank the reviewers for their invaluable inputs. This work is funded by Institut Teknologi Sepuluh Nopember (ITS) under research contract number 779/PKS/ITS/2017.

References

[1] A. Ritter, C. Colin and B. Dolan, "Unsupervised Modelling of Twitter Conversations," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Los Angeles, 2010.

[2] B. Han, P. Cook and T. Baldwin, "Lexical Normalization for Social Media Text," *ACM Transactions on Intelligent Systems and Technology (TIST),* vol. IV, no. 5, 2013.

[3] T. F. C. Bertaglia and M. d. G. V. Nunes, "Exploring Word Embeddings for Unsupervised User-generated Content Normalization," in *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, Osaka, 2016.

[4] A. Wang, M.-Y. Kan, D. Andrade, T. Onishi and K. Ishikawa, "Chinese Informal Word Normalization: an Experimental Study," in *IJCNLP*, Nagoya, 2013.

[5] A. S. Lhoussain, G. Hissam and Y. Abdellah, "Adaptating the Levenshtein Distance to Contextual Spelling Correction," *International Journal of Computer Science and Application,* vol. XII, no. 1, pp. 127-133, 2015.

[6] A. Sorokin and T. Shavrina, "Automatic Spelling Correction for Russian Social Media Texts," in *22nd International Conference on Computational Linguistics*, Moscow, 2016.

[7] A. Purwarianti, A. Andhika, A. F. Wicaksono, I. Afif and F. Ferdian, "InaNLP: Indonesia Natural Language Processing Toolkit," in *Proceeding of the 2016 International Conference on Advanced Informatics: Concepts, Theory, and Application (ICAICTA)*, George Town, 2016.

[8] J. Porta and J.-L. Sancho, "Word Normalization in Twitter Using Finite-state Transducers," in *Tweet-Norm@ SEPLN*, Madrid, 2013.

[9] B. Han, P. Cook and T. Baldwin, "Automatically constructing a normalisation dictionary for microblogs," in *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, Jeju, 2012.

[10] B. Han, P. Cook and T. Baldwin, "unimelb: Spanish Text Normalisation," in *Proceedings of the Tweet Normalization Workshop co-located with 29th Conference of the Spanish Society for Natural Language Processing (SEPLN 2013)*, Madrid, 2013.

[11] V. K. R. Sridhar, "Unsupervised Text Normalization Using Distributed Representations of Words and Phrases," in *Proceedings of NAACL-HLT 2015*, Denver, 2015.

[12] V. I. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions, and Reversals," *Soviet Physics-Doklady,* vol. 10, no. 8, pp. 707-710, 1966.

[13] M. A. Jaro, "Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida," *Journal of the American Statistical Association,* vol. 84, pp. 414-420, 1989.

[14] W. E. Winkler, "String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage," 1990.

[15] W. W. Cohen, P. Ravikumar and S. E. Fienberg, "A Comparison of String Metrics for Matching Names and Records," in *Proceedings of the KDD Workshop on Data Cleaning and Object Consolidation*, Washington DC, 2003.

[16] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning Representations by Back-propagating Errors," *Nature,* vol. 323, pp. 533-536, 1986.

[17] T. Mikolov, K. Chen, G. Corrado and J. Dean, "Efficient Estimation of Word Representations in Vector Space," in *ICLR 2013*, Scottsdale, 2013.

[18] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*, Helsinki, 2008.

[19] "Word2vec," Google Inc., 2013. [Online]. Available: https://code.google.com/archive/p/word2vec/.

[20] J. R. Firth, "A Synopsis of Linguistic Theory, 1930-1955," *Studies in Linguistic Analysis,* Vols. Special Volume, Philological Society, pp. 1-32, 1957.

[21] "Deeplearning4j: Open-source distributed deep learning for the JVM, Apache Software Foundation License 2.0," Deeplearning4j Development Team, [Online]. Available: http://deeplearning4j.org.

[22] Kateglo, "Kamus, Tesaurus, dan Glosarium Bahasa Indonesia," Kateglo, [Online]. Available: http://kateglo.com/.

[23] Wiktionary, "Wiktionary Bahasa Indonesia, Kamus Bebas Berbahasa Indonesia," Wiktionary, [Online]. Available: https://id.wiktionary.org.

[24] Google, "Google Translate," Google, [Online]. Available: https://translate.google.com/.

[25] E. Setiawan, “KBBI Online versi 2.0 (unofficial),” [Online]. Available: https://kbbi.web.id/.

[26] “KBBI Daring (official),” Badan Pengembangan dan Pembinaan Bahasa, Kementrian Pendidikan dan Kebudayaan Republik Indonesia, 2016. [Online]. Available: https://kbbi.kemdikbud.go.id/.

[27] Apache Solr, [Online]. Available: https://lucene.apache.org/solr/.

[28] A. Jaech, G. Mulcaire, S. Hathi, M. Ostendorf and N. A. Smith, “Hierarchical character-word models for language identification,” in *Proceedings of The Fourth International Workshop on Natural Language Processing for Social Media*, Austin, 2016.