An LSTM-based Spell Checker for Indonesian Text

Damar Zaky
School of Computing
Telkom University
Bandung, Indonesia
damzaky@gmail.com

Ade Romadhony
School of Computing
Telkom University
Bandung, Indonesia
aderomadhony@telkomuniversity.ac.id

***Abstract*—A spell checker is a tool for detecting and correcting various spelling errors. While it might be trivial for humans, spell detecting and correcting can be very useful for machines, because machines could not detect spelling errors and correct them automatically. In Natural Language Processing (NLP), detecting and correcting spelling errors is a task that has been widely performed to normalize data, since most raw texts are noisy and have many spelling errors. In recent years, Long Short-term Memory (LSTM) has shown to give an extraordinary result in solving sequential problems, including spelling correction. In this paper, we propose an LSTM model that encodes input word at character level, that also uses word and POS tag contexts as features. We performed the experiment on an artificial dataset based on Indonesian Wikipedia articles that we made by simulating some artificial spelling errors at character level and tested it on real dataset, mostly are Indonesian online news articles. The evaluation on test dataset gives 83.76% accuracy.**

***Index Terms*—Indonesian spell checker, LSTM, deep learning**

I. Introduction

Spelling errors occur in many kind of Indonesian texts regardless of their formality (whether they are formal texts or informal texts). Spelling errors are very common in human-generated texts and the cause of them can vary from accidental hand or mind slips to writer's lack of spelling knowledge. For reading purpose, spell detecting and correcting might be trivial for humans since humans can naturally detect patterns easily that make them able to read and understand texts even though the texts have spelling errors, while machines cannot do it without being instructed or trained to.

In Natural Language Processing (NLP) applications, data normalization is very important [1] because it can improve the performance and the accuracy of NLP applications. By performing spell checking before other normalization task for various NLP applications such as information retrieval, machine translation, text classification, and opinion mining, spell checking can reduce out-of-vocabulary (OOV), reduce the size of bag of words representation, and produce better stemming or lemmatization result. For humans, spell checking can help when they are writing texts that must contain no mistakes (often texts in formal context) or for a better readability of texts.

Spell checking is a challenging task because there are many error possibilities. The basic approach or detecting spelling errors, is using dictionary lookup. The problem arise when there are words that are not spelling errors but detected as errors because we could not find them in the dictionary or when there are named entities that are detected as spelling errors. Another approach for correcting spelling errors is rule-based approach and the challenge lies on many error possibilities, therefore we have to define many set of rules.

There are several spell checkers for Indonesian text that uses different approaches. The first study, proposed a rule-based approach that uses similarity measure and forward reversed dictionary [2]. The next study uses machine learning-based approach which uses morphological analyzer and probability of similarity and Hidden Markov Model (HMM) for candidates ranking [3]. The most recent one also uses machine learning-based approach which uses bi-gram/coocurrence/unigram model and HMM for candidates ranking [4]. These works gave a high accuracy. However, these works still require many manually defined rules. For other languages, there are spell checker studies that use deep learning-based approach [5] [6] [7]. However, no such work has been found yet for Indonesian language. We propose a Long Short-term Memory (LSTM) based encoder decoder model which examines word by word at character level that does not require any rules to correct spelling errors. We use words characters and context feature, in which the context features consist of adjacent words and Part of Speech (POS) tags. The adjacent words are represented by the word-embedding vector.

II. Related Works

There are few previous works on spelling error detection and correction for Indonesian text [3] [4] [2], in which most of them used dictionary lookup for spelling error detection and rule-based for spelling error correction. Early works on this field proposed a solution that used similarity measure and forward reversed dictionary [2]. Statistical-based approach (Hidden Markov Model/HMM) were utilized in [Application of document spelling checker], combined with forward reversed dictionary on error correction. This work uses dictionary look-up to detect spelling error that uses lexicon resource that is also similar to how our system detect spelling errors. This work uses forward reversed dictionary and similarity measure to correct spelling errors. The forward reversed dictionary finds the error zone by sub-string matching between misspelled word and words in forward reversed dictionary, then it adds possible words from four types of errors that requires to define the four sets of edit operations rules. Then it also uses similarity measure to compare the error word with the correct word and produces a score. Then HMM is used for candidate ranking.

Another study came up with morphological analyzer and probability of similarity and HMM for candidates ranking [3]. This work uses almost the same aforementioned work, but this one adds a morphological analyzer. The morphological analyzer algorithm that is used is called Tala algorithm. This morphological analyzer requires to define Indonesian word morphemes such as particles (-kah, -tah, lah, -pun), possessives (-mu, -ku, -nya), suffixes (-an, -i, -kan), and prefixes (beN-, teN-, di-, se-, ke-, -meN, peN). The morphological analyzer sheds the morphemes by particle, possessive, prefixes, and suffix orderly

The most recent study employed rule-based approach and HMM for candidates ranking that does not only correct spelling errors, but also correct grammar errors combined [4]. This work uses dictionary look-up as error detection with Kamus Besar Bahasa Indonesia (KBBI) as the lexicon. The dictionary is stored in trie data structure. The work also uses a dictionary that consists of common misspellings to correct common misspellings directly. The work's error correction goal is to correct split words, run-on words, words with certain affixes or bound morphemes, passive or imperative verbs, and words combined with dash or slash errors. For every error types, set of rules are required to be defined, such as checking the word's suffix, affix, prefix specifically for every types of errors.

All of the approaches used in previous works have limitation on rule correctness and completeness and also corpus availability. The rule based approach for spelling error correction requires manually defined rules, which is very expensive to produce valid and complete rules. The statistical approach also needs improvement on the Indonesian spelling error corpus availability, which to date there is no large available corpus.

Currently, machine learning approach especially deep learning has shown performance improvement on several NLP and NLP related tasks performance, including spell checking and correction. The work by Ghosh and Kristensson (2017) is the first one that used Deep Neural Networks for text correction [8].

Sakaguchi et al. (2017) solved the Spell Correction using semi-character Recurrent (RNN) Neural Networks on English data [9]. There is also one work of spelling correction using deep learning on low resource language [automatic spelling correction for resource-scarce language]. Deep learning based approach has advantage over rule-based and statistical-based approach, since it does not require manually defined rules and large scale dictionary. While there are several works on English and also low resource language spell checkers employing deep learning approach, to the best of our knowledge, there is no previous study in Indonesian language.

III. Proposed Solution

Our proposed solution consists of two parts: spelling error detection and spelling error correction. Initially, we performed text tokenization and POSTagging as a preprocessing step. The spelling error detection is conducted based on dictionary lookup technique. In this study we used KBBI, which is Indonesian language official dictionary, as the source for spelling error detection. The spelling error detection part checks whether the examined word exists in the dictionary, if it exists, it will be fed to LSTM model to check whether the LSTM produces an output that exists in the dictionary, if LSTM output a word that does not exist in the dictionary, the initial word will be the output. If the examined word could not be found in the dictionary, it will be detected as an error and will be processed by LSTM to be corrected, in this case, the output is whatever the LSTM model produces.

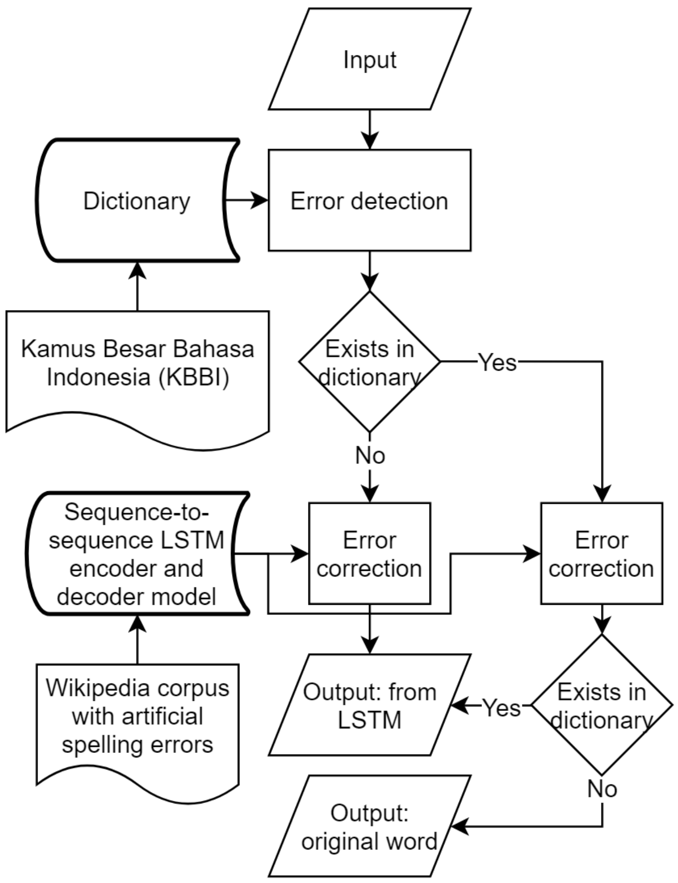

Fig. 1. System architecture

To improve the model, the model uses a simple rule where any capitalized words are assumed as named entities and are not detected as spelling error. We use sequence-to-sequence model that uses LSTM as encoder and decoder as the method in spelling error correction part. Beside word/token character, we used word and POSTag context as the feature, where the word context is represented by the word embedding vector.

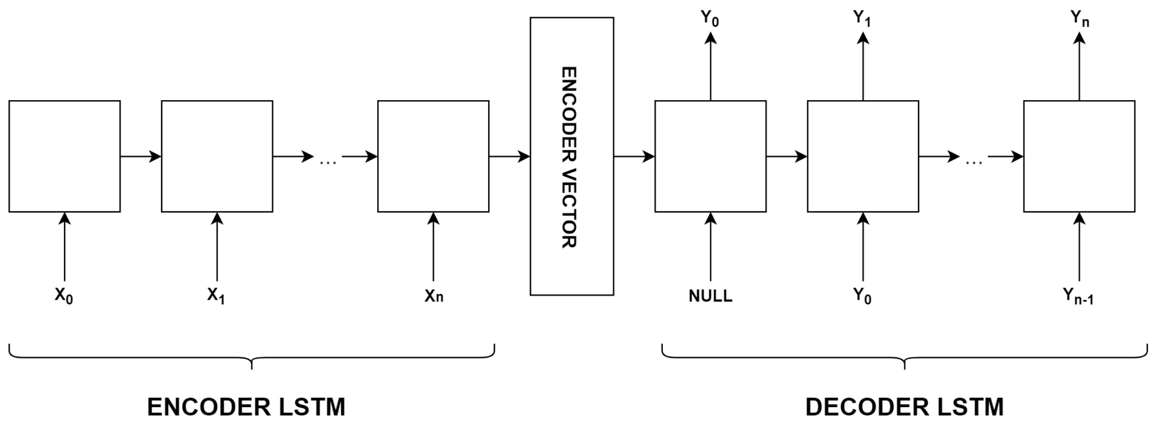

Fig. 2. LSTM Model architecture

A. Model Architecture

In this work, we use LSTM as an encoder and decoder on sequence-to-sequence model, which examines word by word at character level. One feed-forward pass of this model examines one word. For each time step, character at the time step is encoded to one-hot vector. Previous word and next word is also used as feature, those words are represented by embedding vector by summing their respective word vector and divide it by the number of context words that are used. Other feature that is used is POS tag of previous and next word that is also encoded to one-hot vector, previous word POS tag one-hot vector and next POS tag one-hot vector will be concatenated. For every time step, those vectors are concatenated to produce a feature vector, so every time step input in one forward-pass has different character one-hot vector (unless they are the same character) but same context word-embedding vector and context POS tags vector.

In encoder, the input that is used is feature vector that consists of character one-hot vector, word embedding vector and POS tags one-hot vector. In decoder however, the input is only the character one-hot vector, this also applies for the output, so each time step only produces character vector as output.

Fig. 2 is the architecture of the model. x_n is the feature vector that is the concatenation of character one-hot vector at time step, word embedding vector and POS tags one-hot vector n. y_n is an output vector at time step n that only consists of character vector. n is the length of the longest word in the data train. So before the training phase, we first have to define the maximum word length based on the data train. We then use this to transform every word that has length less than the maximum length to be padded to maximum length by replacing the rest of the length with an asterisk (this process is performed after adding the artificial spelling errors). For example, the longest word in the data train is word ”berkewarganegaraanb” that has length of 19, suppose there is a word ”gimansa” that has length of 7, this word is then transformed to ”gimansa************”.

B. Features

The feature vector, which is used by our model, consists of:

Character one-hot vector is a mapping vector of characters that are capital and non-capital letters, an asterisk, a '\t' and a hyphen (for plural words like: ”ibu-ibu”), this is because Indonesian words only have alphabetic characters. While a '\t' indicates the start and an asterisk indicates the end of sequence. Therefore, the length of the character vector vocabulary is 55.

Word Embedding is a vector representation of a text. In word embedding, every word has a vector representation WV. Word embedding can capture context of a word in a text, as well as similarity between one word to another. This means that similar words will have a similar vector form. In this work, every word has word embedding feature vector of adjacent words WEV_c^n where c is the number of previous and next adjacent words of n^{th} word in a sentence. The word embedding feature vector can be obtained by using below formula:

$$WEV_c^n = \frac{\sum_{k=-c}^{n-1} WV_k + \sum_{k=n+1}^{c} WV_k}{c * 2} \tag{1}$$

To retrieve the word embedding, we train using word2vec with Wikipedia dataset. The word embedding vector that we trained has length of 400.

Part of Speech (POS) Tag is a one-hot vector that consists of POS tags. The vocabulary of the POS tag vector based on our training model is 21. The length of POS tag vector is (POS tag vocabulary)*2 that in our work is 42 (this is to differentiate the vector for previous POS tag and next POS tag). In this work, we use word's adjacent POS tag to be used as feature.

The Character one-hot vector feature is the main feature that is used for all the models that are evaluated, while POS tag and word2vec features are additional features for capturing context that are used in some models.

Table I shows an example of feature vector of one word that is examined in one forward-pass for the model that uses all

TABLE I
TIME STEP FEATURE VECTOR FOR WORD "SUKA" THAT THE CONTEXT IS WORD "AKU" AND "SUSU"

	's'	**'u'**	**'k'**	**'a'**
Character vector	[0,0,1,0]	[0,0,0,1]	[0,1,0,0]	[1,0,0,0]
Word embedding vector	[-0.2,0.2,0.1]	[-0.2,0.2,0.1]	[-0.2,0.2,0.1]	[-0.2,0.2,0.1]
POS tag vector	[1,0,0,0,0,1]	[1,0,0,0,0,1]	[1,0,0,0,0,1]	[1,0,0,0,0,1]
Feature vector	[0,0,1,0,-0.2,0.2,0.1,1,0,0,0,0,1]	[0,0,0,1,-0.2,0.2,0.1,1,0,0,0,0,1]	[0,1,0,0,-0.2,0.2,0.1,1,0,0,0,0,1]	[1,0,0,0,-0.2,0.2,0.1,1,0,0,0,0,1]

the three features. Suppose the vocabulary of character in the whole training data is only 'a', 'k' 's', and 'u', where 'a' is mapped as [1,0,0,0], 'k' is mapped as [0,1,0,0], 's' is mapped as [0,0,1,0], and 'u' is mapped as [0,0,0,1]. Suppose the length of the word embedding vector of word2vec model is three. Suppose the vocabulary of POS tags in the whole training data is "PRP", "VB", "NN", where "PRP" is mapped as [1,0,0], "VB" is mapped as [0,1,0], and "NN" is mapped as [0,0,1]. Now suppose the sentence that is fed to the model is "aku suka susu" and the POS tag for respective word is "PRP", "VB", "NN" and the word vectors of respective word is [-0.2,0.3,-0.2], [-0.5,-0.5,-0.2], [-0.2,0.1,0.4]. Now suppose the model is examining the second word that is "suka" at time step 's', so the character one-hot vector is [0,0,1,0], the word embedding vector is $\frac{[-0.2,0.3,-0.2]+[-0.2,0.1,0.4]}{2}$ = [-0.2,0.2,0.1], and the POS tag one-hot vector is [1,0,0] $\cdot$ [0,0,1] = [1,0,0,0,0,1]. Now concatenate all those vectors so the feature vector for time step 's' is [0,0,1,0,-0.2,0.2,0.1,1,0,0,0,0,1].

IV. EXPERIMENTS

A. Dataset

The dataset that we use for training our model is an artificial dataset that is based on random Indonesian Wikipedia articles by simulating artificial spelling errors at character level. Before simulating artificial spelling errors, we have to make sure that the corpus is clean from any spelling errors so that the model would not consider spelling errors as a correct spelling, because of that, we pre-process the data. First, we tokenize the words into tokens, then we count the frequency of each word. After counting the frequency, we POS tag every words.

After that, the corpus is transformed from raw text into an array of object. The object consists of main word, previous and next word embedding vectors, and the previous and next word POS tag. Then, since we assume that words that occurrence is less than 20 is a spelling error, we remove objects that main word occurs less than 20 so that the model would not get spelling errors as label.

Now that the dataset is clean and tagged, main word in every objects in the dataset is transformed into a spelling error with rules as follows:

This paper's goal is to detect and correct four types of error that are:

1. Character addition (e.g "pekerjaan" ⇒ "pekedrjaan" where after character 'e' is added character 'd' that is one of its neighbor characters)
2. Character deletion (e.g "dalam" ⇒ "dalm" where second character 'a' is deleted from the word)
3. Character substitution (e.g "kerugian" ⇒ "kerguian" where character 'u' and 'g' are swapped)
4. Character transformation (e.g "tulang" ⇒ "tulsng" where character 'a' is replaced by 's' that is one of its neighbor characters)

Where the transformed character candidate's position is chosen randomly and evenly. The added and transformed characters are the neighbor characters of either its previous or next characters in QWERTY keyboard since Indonesian standard keyboard is QWERTY keyboard.

For testing the model, the dataset that we use are mostly obtained from various Indonesian news portals, including VIVA news, Kompas.com, Detik.com, Tribunnews.com, and several student documents. There are 103 selected documents where each of them have at least one spelling error. The selected documents consist of all four types of errors where the proportion is 28 character addition errors, 56 character deletion errors, 10 substitution errors, and 23 transformation errors. Other than those four types, there are run-on words error and split words error type that are not counted as errors, because the goal of this paper is to only detect and correct the four aforementioned error types.

The selected documents are pre-processed similarly to the training dataset, the difference is that the removing words that occurence is less than 20 and the artificial noise transformation is excluded in the process.

B. Experiment Methodology

To measure the performance of our models in this paper, we use two kinds of recall and precision measures, that are recall incorrect, recall correct, precision incorrect, and precision correct, and Overall Performance Measure [10]. This shows that this is similar to binary classification where the classes are correct and incorrect.

The evaluation is calculated based on the number of True Positive (TP) where correct words become correct, True Negative (TN) where incorrect words become correct, False Negative (FN) where correct words become incorrect, and

TABLE II
EXPERIMENT RESULT OF ALL LSTM MODELS

	TN	TP	FP	FN	Recall		Precision		Accuracy	Addition		Deletion		Substitution		Transformation	
					inco-rrect	cor-rect	inco-rrect	cor-rect		cor-rect	inco-rrect	cor-rect	inco-rrect	cor-rect	inco-rrect	cor-rect	inco-rrect
Character + POS (1 neighbor)	95	29375	22	2320	81.20%	92.68%	3.93%	99.93%	92.64%	25	1	45	12	9	1	17	7
Character + w2v (2 neighbors)	73	25870	44	5825	62.39%	81.62%	1.24%	99.83%	81.55%	20	6	34	23	8	2	11	13
Character	88	30283	29	1412	75.21%	95.55%	5.87%	99.90%	95.47%	23	3	38	19	9	1	18	6
Character + w2v (1 neighbor)	98	30024	19	1671	83.76%	94.73%	5.54%	99.94%	94.69%	25	1	45	12	9	1	19	5
Character + POS (2 neighbors)	95	30320	22	1375	81.20%	95.66%	6.46%	99.93%	95.61%	24	2	46	11	9	1	17	7
Character + w2v + POS (1 neighbor)	59	30857	58	838	50.43%	97.36%	6.58%	99.81%	97.18%	13	13	33	24	8	2	5	19

False Positive (FP) where incorrect words become incorrect [10].

C. Result and Analysis

Table II shows the accuracies of all the LSTM models. All the models are tested using the same dataset that is 103 documents. In the 103 documents, there is a total of 117 spelling errors out of 31812 words. From table II, we can conclude that LSTM model that uses Character, w2v (with one neighbor), and POS tag (with one neighbor) gives the highest accuracy, which is 97.18%. It also shows that without adding additional features that are POS tag and word2vec, the model can get a high accuracy. As for the incorrect words-only correction, it is shown that using character feature with word2vec feature vector result in the highest recall incorrect that is 83.76%. Among the four error types, transformation error type is the hardest type to be corrected as the average accuracy of transformation type is the lowest that is 60%.

LSTM models that use surrounding words POS tag or word2vec feature vector are proven to be able to capture context since they are able to correct words, such as ”musi” to ”musik” where the sentence is ”...materi teori musi dasar serta...”, and to ”musim” where the sentence is ”...dilatihnya pada musi pertama ini...”. We realize that all of our model failed on correcting run-on words (”disini” ⇒ ”di sini”) and split words (”di atur” ⇒ ”diatur”) since our model only examine word by word and those types of errors are not included in the final pre-processed artificial training data. Our models also failed on detecting and correcting words that are not Indonesian or are unknown in the training data that have result as ”external” ⇒ ”enternal”, ”hazard” ⇒ ”hasar”, ”shake” ⇒ ”hake”, ”monitoring” ⇒ ”monitorong”, etc. where first words are correct English words, and ”ketidakstabilan” ⇒ ”ketidaktadialan”, ”penyesaran” ⇒ ”penyebaran”, ”terpetakan” ⇒ ”terletakan”, ”diimbau” ⇒ ”dimbana”, etc. where first words never occurred in the pre-processed training data, but they are detected as incorrect and incorrectly corrected.

That happens because most non-Indonesian words occur a lot less than Indonesian words on the raw training data, where they are excluded from the final pre-processed training data and they are never seen by the models, thus the models could not learn from those data. Even if they are included in the final pre-processed training data, they are probably have a different morphological patterns than Indonesian words, thus the models cannot easily correct them. If the models see unknown words, it is likely that they would not be able to correct those words unless those words have some similarities with other words that rules pattern can be learned by the model.

Even though the models have used a simple rule to exclude capitalized words that might have solved to recognize named entities, but this means that those models are also assuming that any capitalized words including words that are exactly next to period signs are named entities. For example, a word ”Karenanya”, ”Menyadari”, ”Mulailah”, ”Putranya”, etc. are valid Indonesian words, since they are next to period signs, it is normal for them to be capitalized, but this makes the models consider them as named entities, thus if they are incorrectly spelled, they would not be detected and corrected.

V. CONCLUSION AND FUTURE WORK

The spelling checker in this paper is able to detect and correct spelling errors. Since the main purpose of the models is to correct incorrect words, our main evaluation metrics is the recall incorrect. Our Character + w2v (1 neighbor) gives the highest accuracy that is 83.76%. With the given results, we conclude that using LSTM with character as time step feature and additional non-time step features can detect and correct spelling errors in Indonesian language. By using more data for training our models, we believe that they will perform better. By using named entity recognition, it should also be able to recognize that named entities should not be evaluated, thus the models can give better accuracies.

References

[1] Y. Belinkov and Y. Bisk, “Synthetic and natural noise both break neural machine translation,” 11 2017.

[2] M. Kamayani, R. Reinanda, S. Simbolon, M. Soleh, and A. Purwarianti, “Application of document spelling checker for bahasa indonesia,” 01 2011.

[3] Moch Yusup Soleh and A. Purwarianti, “A non word error spell checker for indonesian using morphologically analyzer and hmm,” in *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics*, July 2011, pp. 1–6.

[4] A. Fahda and A. Purwarianti, “A statistical and rule-based spelling and grammar checker for indonesian text,” in *2017 International Conference on Data and Software Engineering (ICoDSE)*, Nov 2017, pp. 1–6.

[5] S. Sooraj, M. K, M. Kumar, and S. Kp, “Deep learning based spell checker for malayalam language,” *Journal of Intelligent & Fuzzy Systems*, vol. 34, pp. 1427–1434, 03 2018.

[6] C. Whitelaw, B. Hutchinson, G. Y. Chung, and G. Ellis, “Using the web for language independent spellchecking and autocorrection,” in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*, ser. EMNLP ’09. Stroudsburg, PA, USA: Association for Computational Linguistics, 2009, pp. 890–899. [Online]. Available: http://dl.acm.org/citation.cfm?id=1699571.1699629

[7] J. Gudmundsson and F. Menkes, “Swedish natural language processing with long short-term memory neural networks : A machine learning-powered grammar and spell-checker for the swedish language,” 2018.

[8] S. Ghosh and P. Ola Kristensson, “Neural networks for text correction and completion in keyboard decoding,” 09 2017.

[9] K. Sakaguchi, K. Duh, M. Post, and B. Van Durme, “Robsut wrod reocginiton via semi-character recurrent neural network,” 08 2016.

[10] R. Hedin, “Spell checker in cet designer,” 2016.