An Unsupervised and Data-Driven Approach for Spell Checking in Vietnamese OCR-scanned Texts

Cong Duy Vu HOANG & Ai Ti AW
Department of Human Language Technology (HLT)
Institute for Infocomm Research (I2R)
A*STAR, Singapore
{cdvhoang,aaiti}@i2r.a-star.edu.sg

Abstract

OCR (Optical Character Recognition) scanners do not always produce 100% accuracy in recognizing text documents, leading to spelling errors that make the texts hard to process further. This paper presents an investigation for the task of spell checking for OCR-scanned text documents. First, we conduct a detailed analysis on characteristics of spelling errors given by an OCR scanner. Then, we propose a fully automatic approach combining both error detection and correction phases within a unique scheme. The scheme is designed in an unsupervised & data-driven manner, suitable for resource-poor languages. Based on the evaluation on real dataset in Vietnamese language, our approach gives an acceptable performance (detection accuracy 86%, correction accuracy 71%). In addition, we also give a result analysis to show how accurate our approach can achieve.

1 Introduction and Related Work

Documents that are only available in print require scanning from OCR devices for retrieval or e-archiving purposes (Tseng, 2002; Magdy and Darwish, 2008). However, OCR scanners do not always produce 100% accuracy in recognizing text documents, leading to spelling errors that make the texts texts hard to process further. Some factors may cause those errors. For instance, shape or visual similarity forces OCR scanners to misunderstand some characters; or input text documents do not have good quality, causing noises in resulting scanned texts. The task of spell checking for OCR-scanned text documents proposed aims to solve the above situation.

Researchers in the literature used to approach this task for various languages such as: **English** (Tong and Evans, 1996; Taghva and Stofsky, 2001; Kolak and Resnik, 2002), **Chinese** (Zhuang et al., 2004), **Japanese** (Nagata, 1996; Nagata, 1998), **Arabic** (Magdy and Darwish, 2006), and **Thai** (Meknavin et al., 1998).

The most common approach is to involve users for their intervention with computer support. Taghva and Stofsky (2001) designed an **interactive** system (called OCRSpell) that assists users as many interactive features as possible during their correction, such as: choose word boundary, memorize user-corrected words for future correction, provide specific prior knowledge about typical errors. For certain applications requiring automation, the interactive scheme may not work.

Unlike (Taghva and Stofsky, 2001), **non-interactive** (or fully automatic) approaches have been investigated. Such approaches need pre-specified lexicons & confusion resources (Tong and Evans, 1996), language-specific knowledge (Meknavin et al., 1998) or manually-created phonetic transformation rules (Hodge and Austin, 2003) to assist correction process.

Other approaches used **supervised** mechanisms for OCR error correction, such as: statistical language models (Nagata, 1996; Zhuang et al., 2004; Magdy and Darwish, 2006), noisy channel model (Kolak and Resnik, 2002). These approaches performed well but are limited due to requiring large annotated training data specific to OCR spell checking in languages that are very hard to obtain.

Further, research in spell checking for Vietnamese language has been understudied.

Proceedings of the Workshop on Innovative Hybrid Approaches to the Processing of Textual Data (Hybrid2012), EACL 2012, pages 36–44,
Avignon, France, April 23 2012.

Hunspell−spellcheck−vn[1] & Aspell[2] are **interactive** spell checking tools that work based on pre-defined dictionaries.

According to our best knowledge, there is no work in the literature reported the task of spell checking for Vietnamese OCR-scanned text documents. In this paper, we approach this task in terms of 1) fully automatic scheme; 2) without using any annotated corpora; 3) capable of solving both non-word & real-word spelling errors simultaneously. Such an approach will be beneficial for a poor-resource language like Vietnamese.

2 Error Characteristics

First of all, we would like to observe and analyse the characteristics of OCR-induced errors in compared with typographical errors in a real dataset.

2.1 Data Overview

We used a total of 24 samples of Vietnamese OCR-scanned text documents for our analysis. Each sample contains real & OCR texts, referring to texts without & with spelling errors, respectively. Our manual sentence segmentation gives a result of totally 283 sentences for the above 24 samples, with 103 (good, no errors) and 180 (bad, errors existed) sentences. Also, the number of syllables[3] in real & OCR sentences (over all samples) are 2392 & 2551, respectively.

2.2 Error Classification

We carried out an in-depth analysis on spelling errors, identified existing errors, and then manually classified them into three pre-defined error classes. For each class, we also figured out how an error is formed.

As a result, we classified OCR-induced spelling errors into three classes:

Typographic or Non-syllable Errors (Class 1): refer to incorrect syllables (not included in a standard dictionary). Normally, at least one character of a syllable is expected misspelled.

Real-syllable or Context-based Errors (Class 2): refer to syllables that are **correct** in terms of their existence in a standard dictionary but **incorrect** in terms of their meaning in the context of given sentence.

Unexpected Errors (Class 3): are accidentally formed by unknown operators, such as: insert non-alphabet characters, do incorrect upper-/lower- case, split/merge/remove syllable(s), change syllable orders, ...

Note that errors in **Class 1 & 2** can be formed by applying one of 4 operators[4] (Insertion, Deletion, Substitution, Transposition). **Class 3** is exclusive, formed by unexpected operators. Table 1 gives some examples of 3 error classes.

An important note is that an erroneous syllable can contain errors across different classes. **Class 3** can appear with **Class 1** or **Class 2** but **Class 1** never appears with **Class 2**. For example:

– **h**oà**n** (correct) || **Hò**an (incorrect) (Class 3 & 1)

– bắ**t** (correct) || bặ**t’** (incorrect) (Class 3 & 2)

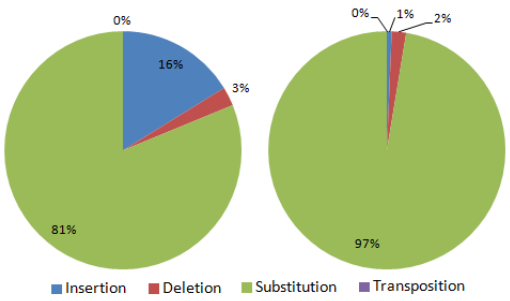

Figure 1: Distribution of operators used in **Class 1** (left) & **Class 2** (right).

2.3 Error Distribution

Our analysis reveals that there are totally 551 recognized errors over all 283 sentences. Each error is classified into three wide classes (Class 1, Class 2, Class 3). Specifically, we also tried to identify operators used in **Class 1** & **Class 2**. As a result, we have totally 9 more fine-grained error classes (1A..1D, 2A..2D, 3)[5].

We explored the distribution of 3 error classes in our analysis. **Class 1** distributed the most, following by **Class 3** (slightly less) and **Class 2**.

[1] http://code.google.com/p/hunspell-spellcheck-vi/

[2] http://en.wikipedia.org/wiki/GNU_Aspell/

[3] In Vietnamese language, we will use the word “syllable” instead of “token” to mention a unit that is separated by spaces.

[4] Their definitions can be found in (Damerau, 1964).

[5] A, B, C, and D represent for Insertion, Deletion, Substitution, and Transposition, respectively. For instance, **1A** means **Insertion** in **Class 1**.

Class	**Insertion**	**Deletion**	**Substitution**	**Transposition**[a]
Class 1	áp (correct) ‖ áip (incorrect) (“i” inserted)	không (correct) ‖ kh_ (incorrect). (“ô”, “n”, and “g” deleted)	yếu (correct) ‖ yếu (incorrect). (“y” substituted by “y”)	N.A.
Class 2	lên (correct) ‖ liên (contextually incorrect). (“i” inserted)	trình (correct) ‖ tình (contextually incorrect). (“r” deleted)	ngay (correct) ‖ ngây (contextually incorrect). (“a” substituted by “â”)	N.A.
Class 3	xác nhận là (correct) ‖ x‖nha0a (incorrect). 3 syllables were misspelled & accidentally merged.			

[a]Our analysis reveals no examples for this operator.

Table 1: Examples of error classes.

Generally, each class contributed a certain quantity of errors (38%, 37%, & 25%), making the correction process of errors more challenging. In addition, there are totally 613 counts for 9 fine-grained classes (over 551 errors of 283 sentences), yielding an average & standard deviation 3.41 & 2.78, respectively. Also, one erroneous syllable is able to contain the number of (fine-grained) error classes as follows: 1(492), 2(56), 3(3), 4(0) ((N) is count of cases).

We can also observe more about the distribution of operators that were used within each error class in Figure 1. The **Substitution** operator was used the most in both **Class 1** & **Class 2**, holding 81% & 97%, respectively. Only a few other operators (**Insertion**, **Deletion**) were used. Specially, the **Transposition** operator were not used in both **Class 1** & **Class 2**. This justifies the fact that OCR scanners normally have ambiguity in recognizing similar characters.

3 Proposed Approach

The architecture of our proposed approach (namely (VOSE)) is outlined in Figure 2. Our purpose is to develop VOSE as an unsupervised data-driven approach. It means VOSE will only use textual data (un-annotated) to induce the detection & correction strategies. This makes VOSE unique and generic to adapt to other languages easily.

In VOSE, potential errors will be detected locally within each error class and will then be corrected globally under a ranking scheme. Specifically, VOSE implements two different detectors (**Non-syllable Detector** & **Real-syllable Detector**) for two error groups of **Class 1/3** & **Class 2**, respectively. Then, a corrector combines the outputs from two above detectors based on ranking scheme to produce the final output. Currently, VOSE implements two different correctors, a **Contextual Corrector** and a **Weighting-based Corrector**. **Contextual Corrector** employs language modelling to rank a list of potential candidates in the scope of whole sentence whereas **Weighting-based Corrector** chooses the best candidate for each syllable that has the highest weights. The following will give detailed descriptions for all components developed in VOSE.

3.1 Pre-processor

Pre-processor will take in the input text, do tokenization & normalization steps. Tokenization in Vietnamese is similar to one in English. Normalization step includes: normalize Vietnamese tone & vowel (e.g. h**ò**a –> ho**à**), standardize upper-/lower- cases, find numbers/punctuations/abbreviations, remove noise characters, . . .

This step also extracts unigrams. Each of them will then be checked whether they exist in a pre-built list of unigrams (from large raw text data). Unigrams that do not exist in the list will be regarded as **Potential Class 1 & 3 errors** and then turned into **Non-syllable Detector**. Other unigrams will be regarded as **Potential Class 2 errors** passed into **Real-syllable Detector**.

3.2 Non-syllable Detector

Non-syllable Detector is to detect errors that do not exist in a pre-built combined dictionary (Class 1 & 3) and then generate a top-k list of potential candidates for replacement. A pre-built combined dictionary includes all syllables (unigrams) extracted from large raw text data.

In VOSE, we propose a novel approach that uses **pattern retrieval** technique for **Non-syllable**

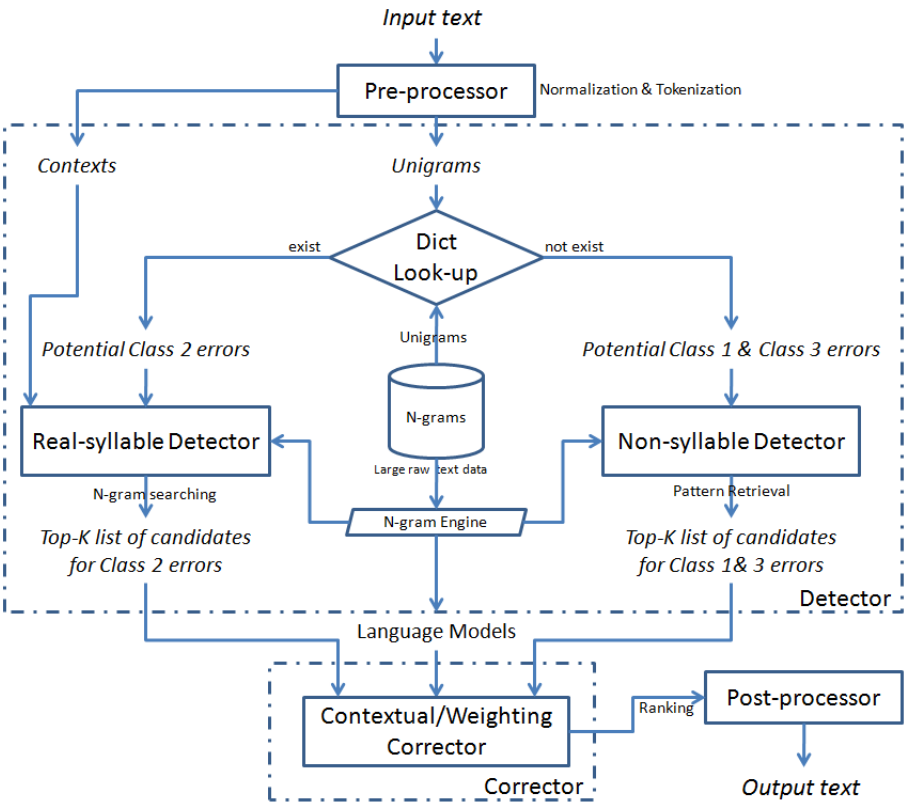

Figure 2: Proposed architecture of our approach

Detector. This approach aims to retrieve all n-gram patterns (n can be 2,3) from textual data, check approximate similarity with original erroneous syllables, and then produce a top list of potential candidates for replacement.

We believe that this approach will be able to not only handle errors with arbitrary changes on syllables but also utilize contexts (within 2/3 window size), making possible replacement candidates more reliable, and more semantically to some extent.

This idea will be implemented in the **N-gram Engine** component.

3.3 Real-syllable Detector

Real-syllable Detector is to detect all possible real-syllable errors (Class 2) and then produce the top-K list of potential candidates for replacement. The core idea of **Real-syllable Detector** is to measure the cohesion of contexts surrounding a target syllable to check whether it is possibly erroneous or not. The cohesion is measured by counts & probabilities estimated from textual data.

Assume that a K-size contextual window with a target syllable at central position is chosen.

$s_1\ s_2\ \cdots\ [\mathbf{s_c}]\ \cdots\ s_{K-1}\ s_K$ (K syllables, s_c to be checked, K is an experimental odd value (can be 3, 5, 7, 9).)

The cohesion of a sequence of syllables s_1^K biased to central syllable s_c can be measured by one of three following formulas:

Formula 1:

$$cohesion_1(s_1^K) = \log(P(s_1^K)) = \log(P(s_c) * \prod_{i\neq c, i=1}^{K} P(s_i|s_c)) \quad (1)$$

Formula 2:

$$cohesion_2(s_1^K) = count_{exist?}(s_{c-2}s_{c-1}s_c, s_{c-1}s_cs_{c+1}, s_cs_{c+1}s_{c+2}, s_{c-1}s_c, s_cs_{c+1}) \quad (2)$$

Formula 3:

$$cohesion_3(s_1^K) = count_{exist?}(s_{c-2} * s_c, s_{c-1}s_c, s_c * s_{c+2}, s_cs_{c+1}) \quad (3)$$

where:

– $cohesion(s_1^K)$ is cohesion measure of sequence s_1^K.

$-$ $P(s_c)$ is estimated from large raw text data computed by $\frac{c(s_c)}{C}$, whereas $c(s_c)$ is unigram count and C is total count of all unigrams from data.
$-$ $P(s_i|s_c)$ is computed by:

$$P(s_i|s_c) = \frac{P(s_i, s_c)}{P(s_c)} = \frac{c(s_i, s_c, |i-c|)}{c(s_c)} \quad (4)$$

where:
$-$ $c(s_i, s_c, |i-c|)$ is a distance-sensitive count of two unigrams s_i and s_c co-occurred and the gap between them is $|i-c|$ unigrams.

For **Formula 1**, if $cohesion(s_1^K) < T_c$ with T_c is a pre-defined threshold, the target syllable is possibly erroneous.

For **Formula 2**, instead of probabilities as in **Formula 1**, we use counting on existence of n-grams within a context. It's maximum value is 5. **Formula 3** is a generalized version of Formula 2 (the wild-card "*" means any syllable). It's maximum value is 4.

N-gram Engine. The N-gram Engine component is very important in VOSE. All detectors & correctors use it.

Data Structure. It is worthy noting that in order to compute probabilities like $c(s_i, s_c, |i-c|)$ or query the patterns from data, an efficient data structure needs to be designed carefully. It MUST satisfy two criteria: 1) **space** to suit memory requirements 2) **speed** to suit real-time speed requirement. In this work, **N-gram Engine** employs inverted index (Zobel and Moffat, 2006), a well-known data structure used in text search engines.

Pattern Retrieval. After detecting potential errors, both **Non-syllable Detector** and **Real-syllable Detector** use **N-gram Engine** to find a set of possible replacement syllables by querying the textual data using 3-gram patterns ($s_{c-2}s_{c-1}[\mathbf{s_c^*}]$, $s_{c-1}[\mathbf{s_c^*}]s_{c+1}$, and $[\mathbf{s_c^*}]s_{c+1}s_{c+2}$) or 2-gram patterns ($s_{c-1}[\mathbf{s_c^*}]$, $[\mathbf{s_c^*}]s_{c+1}$), where $[\mathbf{s_c^*}]$ is a potential candidate. To rank a list of top candidates, we compute the weight for each candidate using the following formula:

$$weight(s_i) = \alpha \times Sim(s_i, s_c^*) + (1-\alpha) \times Freq(s_i) \quad (5)$$

where:
$-$ $Sim(s_i, s_c^*)$ is the string similarity between candidate syllable s_i and erroneous syllable s_c^*.
$-$ $Freq(s_i)$ is normalized frequency of s_i over a retrieved list of possible candidates.
$-$ α is a value to control the weight biased to string similarity or frequency.

In order to compute the string similarity, we followed a combined weighted string similarity (CWSS) computation in (Islam and Inkpen, 2009) as follows:

$$\begin{aligned} Sim(s_i, s_c^*) &= \beta_1 \times NLCS(s_i, s_c^*) \\ &+ \beta_2 \times NCLCS_1(s_i, s_c^*) + \beta_3 \times NCLCS_n(s_i, s_c^*) \\ &+ \beta_4 \times NCLCS_z(s_i, s_c^*) \end{aligned} \quad (6)$$

where:
$-$ β_1, β_2, β_3, and β_4 are pre-defined weights for each similarity computation. Initially, all β are set equal to $1/4$.
$-$ $NLCS(s_i, s_c^*)$ is normalized length of longest common subsequence between s_i and s_c^*.
$-$ $NCLCS_1(s_i, s_c^*)$, $NCLCS_n(s_i, s_c^*)$, and $NCLCS_z(s_i, s_c^*)$ is normalized length of maximal consecutive longest common subsequence between s_i and s_c^* starting from the first character, from any character, and from the last character, respectively.
$-$ $Sim(s_i, s_c^*)$ has its value in range of $[0, 1]$.

We believe that the CWSS method will obtain better performance than standard methods (e.g. Levenshtein-based String Matching (Navarro, 2001) or n-gram based similarity (Lin, 1998)) because it can exactly capture more information (beginning, body, ending) of incomplete syllables caused by OCR errors. As a result, this step will produce a ranked top-k list of potential candidates for possibly erroneous syllables. In addition, **N-gram Engine** also stores computation utilities relating the language models which are then provided to **Contextual Corrector**.

3.4 Corrector

In **VOSE**, we propose two possible correctors:

Weighting-based Corrector

Given a ranked top-K list of potential candidates from **Non-syllable Detector** and **Real-syllable Detector**, **Weighting-based Corrector** simply chooses the best candidates based on their weights (Equation 5) to produce the final output.

Contextual Corrector

Given a ranked top-K list of potential candidates from **Non-syllable Detector** and **Real-syllable Detector**, **Contextual Corrector** globally ranks the best candidate combination using language modelling scheme.

Specifically, **Contextual Corrector** employs the language modelling based scheme which chooses the combination of candidates $(s_1^n)^*$ that makes $PP((s_1^n)^*)$ maximized over all combinations as follows:

$$(s_1^n)^*_{best} = \arg\max_{(s_1^n)^*} PP((s_1^n)^*) \quad (7)$$

where: $PP(.)$ is a language modelling score or perplexity (Jurafsky and Martin, 2008; Koehn, 2010).

In our current implementation, we used Depth-First Traversal (DFS) strategy to examine over all combinations. The weakness of DFS strategy is the explosion of combinations if the number of nodes (syllables in our case) grows more than 10. In this case, the speed of DFS-based **Contextual Corrector** is getting slow. Future work can consider **beam search** decoding idea in Statistical Machine Translation (Koehn, 2010) to adapt for **Contextual Corrector**.

3.5 Prior Language-specific Knowledge

Since **VOSE** is an unsupervised & data-driven approach, its performance depends on the quality and quantity of raw textual data. VOSE's current design allows us to integrate prior language-specific knowledge easily.

Some possible sources of prior knowledge could be utilized as follows:

− **Vietnamese Character Fuzzy Matching** - In Vietnamese language, some characters look very similar, forcing OCR scanners mis-recognition. Thus, we created a manual list of highly similar characters (as shown in Table 2) and then integrate this into VOSE. Note that this integration takes place in the process of string similarity computation.

− **English Words & Vietnamese Abbreviations Filtering** - In some cases, there exist English words or Vietnamese abbreviations. VOSE may suggest wrong replacements for those cases. Thus, a syllable in either English words or Vietnamese abbreviations will be ignored in VOSE.

4 Experiments

4.1 Baseline Systems

According to our best knowledge, previous systems that are able to simultaneously handle both non-syllable and real-syllable errors do not exist, especially apply for Vietnamese language. We believe that VOSE is the first one to do that.

No.	Character	Similar Characters
1	a	{á ạ à ả â ấ ậ ầ}
2	e	{ẽ ê ế ề} + **{c}**
3	i	{ỉ ĩ} + **{l}**
4	o	{ò ơ ờ ở ỡ}
5	u	{ũ ư ự ừ ữ}
6	y	{ý ỵ}
7	d	{đ}

Table 2: Vietnamese similar characters.

4.2 N-gram Extraction Data

In VOSE, we extracted ngrams from the raw textual data. Table 3 shows data statistics used in our experiments.

4.3 Evaluation Measure

We used the following measure to evaluate the performance of VOSE:

- For **Detection**:

$$DF = \frac{2 \times DR \times DP}{DR + DP} \quad (8)$$

Where:

− DR (Detection Recall) = the fraction of errors correctly detected.

− DP (Detection Precision) = the fraction of detected errors that are correct.

− DF (Detection F-Measure) = the combination of detection recall and precision.

- For **Correction**:

$$CF = \frac{2 \times CR \times CP}{CR + CP} \quad (9)$$

Where:

− CR (Correction Recall) = the fraction of errors correctly amended.

− CP (Correction Precision) = the fraction of amended errors that are correct.

− CF (Correction F-Measure) = the combination of correction recall and precision.

4.4 Results

We carried out our evaluation based on the real dataset as described in Section 2. In our evaluation, we intend:

− To evaluate whether VOSE can benefit from addition of more data, meaning that VOSE is actually a data-driven system.

− To evaluate the effectiveness of language modelling based corrector in compared to weighing

No	Dataset	NumOfSents	Vocabulary	N-grams			
				2-gram	3-gram	4-gram	5-gram
1	DS1	1,328,506	102,945	1,567,045	8,515,894	17,767,103	24,700,815
2	DS2[a]	2,012,066	169,488	2,175,454	12,610,281	27,961,302	40,295,888
3	DS3[b]	283	1,546	6,956	9,030	9,671	9,946
4	DS4[c]	344	1,755	6,583	7,877	8,232	8,383

[a] includes DS1 and more
[b] annotated test data (not included in DS1 & DS2) as described in Section 2
[c] web contexts data (not included in others) crawled from the Internet

Table 3: Ngram extraction data statistics.

based corrector.
− To evaluate whether prior knowledge specific to Vietnamese language can help VOSE.

The overall evaluation result (in terms of detection & correction accuracy) is shown in Table 4. In our experiments, all VOSE(s) except of VOSE 6 used contextual corrector (Section 3.4). Also, **Real-syllable Detector** (Section 3.3) used Equation 3 which revealed the best result in our pre-evaluation (we do not show the results because spaces do not permit).

We noticed the tone & vowel normalization step in **Pre-processor** module. This step is important specific to Vietnamese language. VOSE 2a in Table 4 shows that VOSE using that step gives a significant improvement (vs. VOSE 1) in both detection & correction.

We also tried to assess the impact of language modelling order factor in VOSE. VOSE using 3-gram language modelling gives the best result (VOSE 2a vs. VOSE 2b & 2c). Because of this, we chose 3-gram for next VOSE set-ups.

We experiment how data addition affects VOSE. First, we used bigger data (DS2) for ngram extraction and found the significant improvement (VOSE 3a vs. VOSE 2a). Second, we tried an interesting set-up in which VOSE utilized ngram extraction data with annotated test data (Dataset DS3) only in order to observe the recall ability of VOSE. Resulting VOSE (VOSE 3b) performed extremely well.

As discussed in Section 3.5, VOSE allows integrated prior language-specific knowledge that helps improve the performance (VOSE 4). This justifies that statistical method in combined with such prior knowledge is very effective.

Specifically, for each error in test data, we crawled the web sentences containing contexts in which that error occurs (called web contexts). We added such web contexts into ngram extraction data. With this strategy, we can improve the performance of VOSE significantly (VOSE 5), obtaining the best result. Again, we've proved that more data VOSE has, more accurate it performs.

The result of VOSE 6 is to show the superiority of VOSE using contextual corrector in compared with using weighting-based corrector (VOSE 6 vs. VOSE 4). However, weighting-based corrector has much faster speed in correction than contextual corrector which is limited due to DFS traversal & language modelling ranking.

Based on the above observations, we have two following important claims:
− First, the addition of more data in ngram extraction process is really useful for VOSE.
− Second, prior knowledge specific to Vietnamese language helps to improve the performance of VOSE.
− Third, contextual corrector with language modelling is superior than weighting-based corrector in terms of the accuracy.

4.5 Result Analysis

Based on the best results produced by our approach (VOSE), we recognize & categorize cases that VOSE is currently unlikely to detect & correct properly.

Consecutive Cases (Category 1)

When there are 2 or 3 consecutive errors, their contexts are limited or lost. This issue will affect the algorithm implemented in VOSE utilizing the contexts to predict the potential replacements. VOSE can handle such errors to limited extent.

Merging Cases (Category 2)

In this case, two or more erroneous syllables are accidentally merged. Currently, VOSE cannot

Set-up	Detection Accuracy			Correction Accuracy			Remark
	Recall	Precision	F1	Recall	Precision	F1	
VOSE 1	0.8782	0.5954	0.7097	0.6849	0.4644	0.5535	w/o TVN + 3-LM + DS1
VOSE 2a	0.8782	**0.6552**	**0.7504**	**0.6807**	**0.5078**	**0.5817**	w/ TVN + 3-LM + DS1
VOSE 2b	0.8782	0.6552	0.7504	0.6744	0.5031	0.5763	w/ TVN + 4-LM + DS1
VOSE 2c	0.8782	0.6552	0.7504	0.6765	0.5047	0.5781	w/ TVN + 5-LM + DS1
VOSE 3a	**0.8584**	**0.7342**	**0.7914**	**0.6829**	**0.5841**	**0.6296**	w/ TVN + 3-LM + DS2
VOSE 3b	0.9727	0.9830	0.9778	0.9223	0.9321	0.9271	w/ TVN + 3-LM + DS3
VOSE 4	0.8695	0.7988	0.8327	0.7095	0.6518	0.6794	VOSE 3a + PK
VOSE 5	**0.8674**	**0.8460**	**0.8565**	**0.7200**	**0.7023**	**0.7110**	VOSE 4 + DS4
VOSE 6	0.8695	0.7988	0.8327	0.6337	0.5822	0.6069	VOSE 4 but uses WC

Table 4: **Evaluation results**. Abbreviations: **TVN** (Tone & Vowel Normalization); **N-LM** (N-order Language Modelling); **DS** (Dataset); **PK** (Prior Knowledge); **WC** (Weighting-based Corrector).

handle such cases. We aim to investigate this in our future work.

Proper Noun/Abbreviation/Number Cases (both in English, Vietnamese) (Category 3)

Abbreviations or proper nouns or numbers are unknown (for VOSE) because they do not appear in ngram extraction data. If VOSE marks them as errors, it could not correct them properly.

Ambiguous Cases (Category 4)

Ambiguity can happen in:
– cases in which punctuation marks (e.g. comma, dot, dash, . . .) are accidentally added between two different syllable or within one syllable.
– cases never seen in ngram extraction data.
– cases relating to semantics in Vietnamese.
– cases where one Vietnamese syllable that is changed incorrectly becomes an English word.

Lost Cases (Category 5)

This case happens when a syllable which is accidentally lost most of its characters or too short becomes extremely hard to correct.

Additionally, we conducted to observe the distribution of the above categories (Figure 3). As can be seen, Category 4 dominates more than 70% cases that VOSE has troubles for detection & correction.

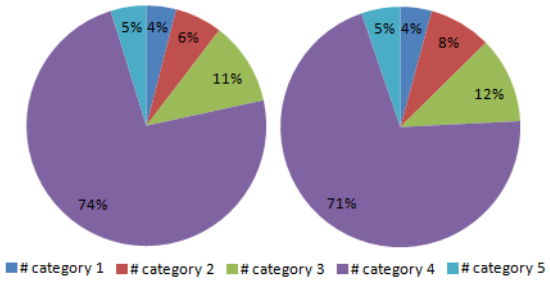

Figure 3: Distribution of categories in the result of VOSE 4 (left) & VOSE 5 (right).

5 Conclusion & Future Work

In this paper, we've proposed & developed a new approach for spell checking task (both detection and correction) for Vietnamese OCR-scanned text documents. The approach is designed in an unsupervised & data-driven manner. Also, it allows to integrate the prior language-specific knowledge easily.

Based on the evaluation on a real dataset, the system currently offers an acceptable performance (best result: detection accuracy 86%, correction accuracy 71%). With just an amount of small n-gram extraction data, the obtained result is very promising. Also, the detailed error analysis in previous section reveals that cases that current system **VOSE** cannot solve are extremely hard, referring to the problem of semantics-related ambiguity in Vietnamese language.

Further remarkable point of proposed approach is that it can perform the detection & correction processes in **real-time** manner.

Future works include some directions. First, we should crawl and add more textual data for n-gram extraction to improve the performance of current system. More data **VOSE** has, more accurate it performs. Second, we should investigate more on categories (as discussed earlier) that VOSE could not resolve well. Last, we also adapt this work for another language (like English) to assess the generalization and efficiency of proposed approach.

References

Fred J. Damerau. 1964. A technique for computer detection and correction of spelling errors. *Commun. ACM*, 7:171–176, March.

Victoria J. Hodge and Jim Austin. 2003. A comparison of standard spell checking algorithms and a novel binary neural approach. *IEEE Trans. on Knowl. and Data Eng.*, 15(5):1073–1081, September.

Aminul Islam and Diana Inkpen. 2009. Real-word spelling correction using google web it 3-grams. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, EMNLP '09, pages 1241–1249, Stroudsburg, PA, USA. Association for Computational Linguistics.

Daniel Jurafsky and James H. Martin. 2008. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall, second edition, February.

Philipp Koehn. 2010. *Statistical Machine Translation*. Cambridge University Press.

Okan Kolak and Philip Resnik. 2002. Ocr error correction using a noisy channel model. In *Proceedings of the second international conference on Human Language Technology Research*, HLT '02, pages 257–262, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 296–304, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Walid Magdy and Kareem Darwish. 2006. Arabic ocr error correction using character segment correction, language modeling, and shallow morphology. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 408–414, Stroudsburg, PA, USA. Association for Computational Linguistics.

Walid Magdy and Kareem Darwish. 2008. Effect of ocr error correction on arabic retrieval. *Inf. Retr.*, 11:405–425, October.

Surapant Meknavin, Boonserm Kijsirikul, Ananlada Chotimongkol, and Cholwich Nuttee. 1998. Combining trigram and winnow in thai ocr error correction. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 2*, ACL '98, pages 836–842, Stroudsburg, PA, USA. Association for Computational Linguistics.

Masaaki Nagata. 1996. Context-based spelling correction for japanese ocr. In *Proceedings of the 16th conference on Computational linguistics - Volume 2*, COLING '96, pages 806–811, Stroudsburg, PA, USA. Association for Computational Linguistics.

Masaaki Nagata. 1998. Japanese ocr error correction using character shape similarity and statistical language model. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 2*, ACL '98, pages 922–928, Stroudsburg, PA, USA. Association for Computational Linguistics.

Gonzalo Navarro. 2001. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, March.

Kazem Taghva and Eric Stofsky. 2001. Ocrspell: an interactive spelling correction system for ocr errors in text. *International Journal of Document Analysis and Recognition*, 3:2001.

Xian Tong and David A. Evans. 1996. A statistical approach to automatic ocr error correction in context. In *Proceedings of the Fourth Workshop on Very Large Corpora (WVLC-4*, pages 88–100.

Yuen-Hsien Tseng. 2002. Error correction in a chinese ocr test collection. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '02, pages 429–430, New York, NY, USA. ACM.

Li Zhuang, Ta Bao, Xioyan Zhu, Chunheng Wang, and S. Naoi. 2004. A chinese ocr spelling check approach based on statistical language models. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 5, pages 4727 – 4732 vol.5.

Justin Zobel and Alistair Moffat. 2006. Inverted files for text search engines. *ACM Comput. Surv.*, 38, July.