AN ENSEMBLE REGRESSION APPROACH FOR OCR ERROR CORRECTION

by

Jie Mei

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
March 2017

Table of Contents

List of Tables

List of Figures

Abstract

This thesis deals with the problem of error correction for Optical Character Recognition (OCR) generated text, or *OCR-postprocessing*: how to detect error words in a text generated from OCR process and to suggest the most appropriate candidates to correct such errors. The thesis demonstrates that OCR errors are inherently more protean and volatile than handwriting or typing errors, while existing OCR-postprocessing approaches have different limitations. Through analyzing the recent development of error correction techniques, we illustrate that the compositional approach incorporating correction inferences is broadly researched and practically useful. Thus, we propose an ensemble regression approach that composite correction inferences for ranking correction candidates of complex OCR errors. On practical side, we make available a benchmark dataset for this task and conduct a comprehensive study on performance analysis with different correction inferences and ensemble algorithms. In particular, the experimental results show that the proposed ensemble method is a robust approach that is able to handle complex OCR errors and outperform various baselines.

List of Symbols Used

Constants

Z	a normalization constant
Λ	an empty symbol, $\Lambda \notin \Sigma$

Functions

g	a general math function
δ	an edit distance measure
ϕ	a correction likelihood inference function
f	a frequency lookup function
p	a probability function

Sequences

S	$= \{s_i\}_n, s.t. s_i \in \Sigma, n \geq 0$, a string
$S_{<i,j>}$	$= \{s_i, \ldots, s_j\}$, a substring of S
Σ^*	a symbol sequences of any length

Sets

$\mathbb{R}$	real numbers
Σ	finite set of symbols
Σ^+	$= \Sigma \cup \{\Lambda\}$

Acknowledgements

I would like to give sincere thanks to my supervisor Evangelos E. Milios and advisors Aminul Islam and Abidalrahman Moh'd, for their continued support and encouragement. I would like to dedicate this thesis to my parents, whose love accompanied me through every step of my way.

Chapter 1

Introduction

1.1 Problem Statement

An increasing amount of data is produced and transformed into the digital form these days, including magazines, books, and scientific articles. Using the graphic formats, like Portable Document Format (PDF) or Joint Picture Group (JPG), is a comprehensive solution for efficient digitization as well as better preserving the page layout and the graphical information (i.e., charts and figures). Since information in such formats is not machine-readable, analyzing such data relies heavily on the accuracy of Optical Character Recognition (OCR) (Doermann, 1998). However, OCR systems are imperfect and prone to errors.

Post-processing is an important step in improving the quality of OCR output, which is crucial to the success of any text analyzing system in pipeline. An OCR post-processing model attempts to detect misspellings in noisy OCR output and correct such errors to their intended representations. Many machine learning approaches (Lund and Ringger, 2009; Lund et al., 2011, 2013a,b) correct the OCR-generated errors by selecting the most appropriate correction among candidates. OCR-generated errors are more diverse than handwriting errors in many aspects (Jones et al., 1991; Kukich, 1992b). For example, $<Lanius \rightarrow Lioiiits>$ is a typical OCR error, of which is hard to guess the according correct form given the error word. Machine learning approaches incorporate different features enabling more robust candidate selection, instead of inferring from limited observations, for example, using a probabilistic-based model (Taghva and Stofsky, 2001).

While machine learning approach exhibits advantages in correcting OCR-generated

texts, two problems emerge from the existing models: First, some models (Lund and Ringger, 2009; Lund et al., 2011, 2013a,b; Kissos and Dershowitz, 2016) limit candidate suggestions from the recognition output of OCR engines. The errors unrecognized by all OCR engines are thus unable to be corrected. This issue can be problematic especially when original input suffers from degradation, for example, historical documents (Ntirogiannis et al., 2013). Secondly, another class of models (Kissos and Dershowitz, 2016) uses the frequencies of both candidate and related n-grams from corpus as features for training. Although n-gram statistics is shown to be effective in correcting real-word spelling errors (Islam and Inkpen, 2009c), training with only n-gram features does not capture the diverse nature of OCR errors and may lead to a biased model where candidates with low frequency in the corpus tend to be not selected.

1.2 Proposed Model

In this thesis, we propose an OCR post-processing error correction model that leverages different features through a learning process. Our model applies different features to avoid bias and improve the correction accuracy. To address the limitation of candidate suggestion, we enhance the scope of the candidates of an error by considering all the words available in the vocabulary within a limited Damerau-Levenshtein distance (Damerau, 1964) and then use features to narrow down the candidates number. The proposed model ranks the candidates by a regression model and shows that more than 71.2% of the errors can be corrected on a ground truth dataset. For 25.9% of the uncorrected errors, our model could provide the correction in top five suggestions.

1.3 Contributions

To sum up, our contributions are as follows:

- We design a set of OCR-specific features that quantify the likelihood of a candidate word being an error correction. These features search lexical, semantic, and contextual clues from exteral resoureces to evaluate each candidate.

- We propose an OCR post-processing model which integrates OCR-specific features in a ensemble regression approach. The evaluation result shows that the proposed model is capable of providing high quality candidates in the top suggested list.

- We make available a ground truth OCR-error dataset, which is generated from a book in Biodiversity Heritage Library. This dataset lists the mappings from OCR-generated errors to their intended representations and, as far as we know, is the first for this task to use directly for benchmark testing.

1.4 Outline

The rest of the thesis is structured as follows:

Chapter 2 We introduce the background knowledge with the focus on describing the challenges of OCR-error correction and limitations on existing approaches. We first analyze how the OCR error is different from other natural language errors, i.e. spelling error or grammar error, and why correcting such error is a hard task remain unsolved. Then, we discuss the gist and limitations of the proposed attempts on correction of OCR-errors in last two decades.

Chapter 3 We claim that two influential error correction approaches – noisy channel and similarity analysis – are essentially analogous compositional frameworks incorporating correction inferences. In this chapter, we introduce the compositive form and correction models of these two compositional frameworks. Then, through comparison and analysis, we illustrate the similarities and differences between these two frameworks.

Chapter 4 We propose and introduce a regression framework that has a stronger fitting capability than two previously introduced compositional frameworks. In addition, we describe in detail the procedure and setup of using this framework on OCR-postprocessing task.

Chapter 5 We conduct an evaluation on analyzing the performance of each step of the proposed regression framework. Then, we use some distinct output records to discuss the overall result and the framework characteristic.

Chapter 2

Background

Optical character recognition (OCR) is the process of converting images of typed, handwritten or printed text into machine-readable text, where text can be retrieved from a scanned document, an electronic signature, or signs and billboards in a landscape photo. It is a technique that widely used for digitizing printed texts so that contents can be electronically edit, searched, stored, and used in a text analytics and data mining pipline.

Although recognition techniques are ever improving, the OCR engines are less than perfect in handling real-world document image. To improve the recognition accuracy, modern OCR engines involves an intelligent post-processing step after the standard recognition procedure. Different from OCR as an image pattern recognition technique, OCR post-processing is usually considered as a natural language processing task utilizing lexical and contextual information. Although the task of correcting OCR-generated errors can be categorized as a subtask of automatic text correction, the unique generative mechanism of OCR error makes it different from and more challenging than spelling or grammar errors in detection and correction.

2.1 OCR Procedure

As shown in Figure 2.1, a typical OCR procedure involves the following four major steps:

– *Scanning*: scan the paper document and produce an electronic image. The quality of the image is determined by the document and scanner.

– *Zoning*: order the various regions of text in the image. This step may contains optional image analysis operations, including image quality assessment, text line detection, and noise removal.

– *Segmentation*: breaks the various zones into their respective components (zones are decomposed into words and words are decomposed into characters).

– *Classification*: classify the segmented images into their respective ASCII characters.

2.2 OCR-Error Characteristics

The word error rate of OCR engines, in practice, is in the range of 7-16% (Santos et al., 1992; Jones et al., 1991), which is significantly higher than the 1.5-2.5% for Handwriting (Wing and Baddeley, 1980; Mitton, 1987) and the 0.2-0.05% for the edited newswire (Pollock and Zamora, 1984; Church and Gale, 1991). The patterns of OCR-generated errors tend to vary for different OCR engines and type fonts (Kukich, 1992b). We summarize some distinct characteristics of the OCR-generated errors, which make the correction task different from and more challenging than spell and grammar error correction:

Complex non-standard edits The human-generated misspellings are character-level edits, which can be categorized into one of the following four standard types: *insertion*, *deletion*, *substitution*, and *transposition*. The majority of spell correction errors, roughly 80%, is single edit from the intended word (Damerau, 1964) and tend

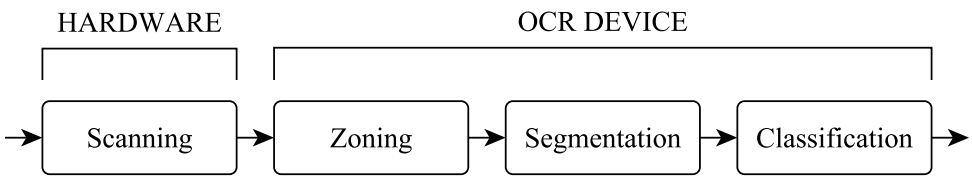


Figure 2.1: Four major processing steps in a typical OCR procedure.

to be within one length difference (Kukich, 1992b). However, a significant fraction of OCR-generated errors are not one-to-one character-level edit (e.g., *ri* → *n* or *m* → *iii*) (Jones et al., 1991).

Multi-factor error generation OCR errors are generated in different processing steps due to various factors. Taghva and Stofsky (2001) trace the errors associated with the primary OCR steps involved in the conversion process: (1) *scanning error* caused by the low paper/print quality of the original document or the pool condition of the scanning equipment. (2) *zoning error* caused by incorrect decolumnization or complex page layout. (3) *segmentation error* caused by the broken characters, overlapping characters, and nonstandard fonts in the document. (4) *classification error* caused by the incorrect mapping from segmented pixels to a single character.

Multi-source dependent The characteristics of OCR-generated errors vary according to not only human reasons (e.g., publishers or authors) but also non-human causes (e.g., text font or input quality) (Jones et al., 1991). These are especially sensitive between OCR engines. Because different OCR engines use different techniques and features for recognition leads to a different confusion probability distribution (Kukich, 1992b).

2.3 Modern Post-Processing Models

The literature of OCR post-processing research exhibits a rich family of models for correcting OCR-generated errors. The post-processing model is an integrated system, which detects and corrects misspellings of both non-word and real-word in the OCR-generated text.

Some studies view the post-processing as the initial step in a correction pipeline and involve continuous human intervention afterwards (Taghva et al., 1994; Taghva and Stofsky, 2001; Mühlberger et al., 2014). These models are designed to reduce the human effort in correcting errors manually. Taghva et al. (1994) integrate dictionaries

and heuristics to correct as many OCR errors as possible before these are given to human correctors. Their subsequent work, Taghva and Stofsky (2001), records the previous human corrections to update the underlying Bayesian model for automatic correction. Mühlberger et al. (2014), as an extreme case, build a full-text search tool to retrieve all occurrences of original images given a text query, which fully relies on the user to validate and correct the errors.

One direction of work ensembles outputs from multiple OCR engines for the same input and selects the best word recognition as the final output (Klein et al., 2002; Cecotti and Belayd, 2005; Lund and Ringger, 2009; Lund et al., 2011, 2013a,b). Klein et al. (2002) show that combining complementary results from different OCR models leads to a better output. Lund et al. (2011) demonstrate that the overall error rate decreases with the addition of different OCR models, regardless of the performance of each added model. Lund et al. (2013a) use machine learning techniques to select the best word recognitions among different OCR outputs. Lund et al. (2013b) apply both OCR recognition votes and lexical features to train a Conditional Random Field model and evaluate the test set in a different domain. While such models have proved useful, they select words only among OCR model recognitions and are blind to other candidates. Besides, they require the presence of the original OCR input and effort of multiple OCR processing.

Another class of post-processing models abstracts from OCR engines and leverages statistics from external resources (Bassil and Alwani, 2012; Kissos and Dershowitz, 2016). Kissos and Dershowitz (2016) use three n-gram statistical features extracted from three million documents to train a linear regressor for candidate ranking. Bassil and Alwani (2012) make use of the frequencies in the Google Web 1T n-gram corpus (Brants and Franz, 2006) for candidate suggestion and ranking. Candidates suggested from these models are not restricted to exist in OCR recognitions. However, existing methods make use of solely n-gram frequencies without knowing the characteristics of OCR errors and are, thus, bias to select common words from the n-gram corpus.

Chapter 3

Compositional Correction Frameworks

Most of the techniques for correcting OCR errors spring from the experience of general error correction. To improve the state-of-the-art in OCR-postprocessing, we should look at the development and evolution of the correction models in a bigger scope.

The research for automatic text correction techniques has a history more than a half century, where a large number of methods have been proposed. Throughout this broad literature, we can see some correction techniques have been continuously investigated and extended by many works in the last century (Kukich, 1992b). These chains of researches have built solid foundations for error correction, and the main idea of these techniques have kept influencing the leading developments in this research domain.

Meanwhile, the recent development of correction models in past two decades witness a tendency of inference composition, where a correction model combines multiple correction techniques as inferences to improve the overall performance. Compositing measures is serverd as an performance optimization technique and have been proved to be effective in estimating text semantic similarity (Bär et al., 2012; Mei et al., 2016). We categorize recently developed models into two major compositional frameworks: noisy channel and similarity analysis. Beside the different normalization restrictions to the inferences, we argue that the composition strategy of two frameworks are analogous. Indeed, a correction model built upon noisy channel can be represented in similarity analysis framework and vice versa.

In this chapter, we will review two compositional frameworks in error correction

with the focus on their compositive form, correction applications, and applied inferences. With above knowledge, we made a comparison and illustrate the similarities and differences between these two frameworks.

3.1 Noisy Channel

The noisy channel, first introduced by Shannon (1948a,b) in his famous information theory, is a message communication framework modeling the information distortion during transmission. Shown in Figure 3.1, this framework simulates the information generation, distortion, and observation separately, and models each procedure in a probabilistic manner. Using Bayes' rule, predicting the original information relies only on the product of two independent models: the *channel model* and the *source model*.[1]

$$\begin{aligned}\hat{x} &= \arg\max_x p(x|y) \\ &= \arg\max_x \frac{p(y|x)\ p(x)}{p(y)} \\ &= \arg\max_x \underbrace{p(y|x)}_{\substack{\text{channel}\\ \text{model}}}\ \underbrace{p(x)}_{\substack{\text{source}\\ \text{model}}}\end{aligned} \tag{3.1}$$

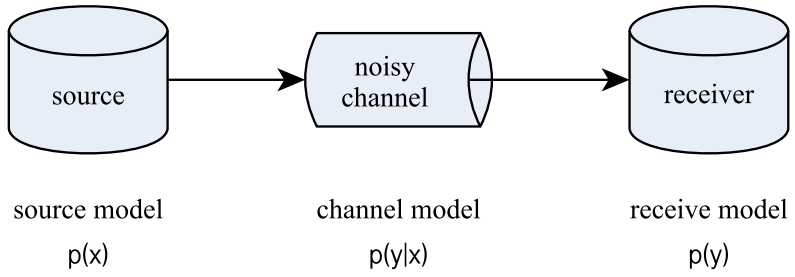


Figure 3.1: A simple diagram of the noisy channel model. The information transmisstion is separated into three procedures controlled by different models:

- The source model produce a message x of probability $p(x)$.
- The channel model transmit and modify the message into another message y with probability $p(y|x)$.
- The receive model get the message y with the observation probability $p(y)$.

[1] source model is also called language model in the literature of NLP.

The intuitive simplification and straightforward decomposition make the noisy channel a robust theoretic framework widely used in spell correction, question answering, speech recognition, and machine translation. This framework allows different inferences to be merged into one correction model, where the collaborated inferences are carried out in the product space

$$\hat{x} = \arg\max_{x \in \mathcal{X}} \prod_{y_i}^{Y} p(y_i|x)\ p(x). \tag{3.2}$$

There are two types of collaboration semantics: first, a channel model can further decompose into submodels that handle different parts of the correction. For example, Segaran and Hammerbacher (2009) applied different probabilistic inferences for different edit operations in a Damerau-Levenshtein edit transformation. Second, different noisy channels that using different correction inference can be used combinatorially. For example, combining character edit probabilities with contextual word n-gram probabilities (Pirinen et al., 2012).

3.1.1 Error Correction Models

Kernighan et al. (1990) made an early attempt to use noisy channel upon Damerau-Levenshtein distance measures (Damerau, 1964) in spell correction task. Given a word list corrected from external resources, he searched candidates that differ from the typo by one single edit operation: insertion, deletion, substitution, or reversal. The probability of each character and edit operation are computed as the source and channel model. Later models considered candidates that require multiple edit operations transforming from the typo, where the channel model is the product of edit operations in the transformation. Pirinen et al. (2010) wrapped the noisy channel into a finite-state transducer, which transition weights are trained from word frequencies. Brill and Moore (2000) suggested that a correction should be inferred by not only the characters that were modified in the edit operation but also the context defined by the neighboring characters. For example, to correct a misspelling word

confidant, model should use $p(ant|ent)$ rather than $p(a|e)$. Therefore, he proposed an improved noisy channel model that partitions the error word into non-overlapping substrings and estimates the channel model of each paired partition. The channel model of the entire string is the product of channel models from all partitions. Whitelaw et al. (2009) and Cucerzan and Brill (2004) applied this improved model on web data. In addition, Cucerzan and Brill (2004) proposed an iterative correction process, where the correction model is used in each iteration. This iterative process terminates when the correction model predicts the input string as the correction.

Context words are the major inference in correcting real word errors, since character mismatch is not the reason for making such errors. Mays et al. (1991) use trigram language model to approximate the probability of a sentence, where error word is substituted by a candidate. The sentence probability is used as the source model to infer the coherence of using such candidate in a specific context. Pirinen et al. (2012) considered the part of speech (POS) tags of the preceding words in a hidden Markov model in estimating $p(x)$. He also experimentally shows that using context improves the performance of noisy channel in both English and Finnish.

3.1.2 Correction Inferences

Correction inference refers to a quantitative measure that supports decision making in the correction task. Some influential inferences are extensively researched and well developed in the early literature (Kukich, 1992b). Although these inferences are no longer used as a standalone correction model, they are components widely adopted in modern correction models under compositional frameworks.

In Table 3.1, we summarize the correction inferences used in models that have been described previously in building the noisy channel, where each type of inference is annotated as ϕ with subscripts. $\prod_Y$ indicates that the inference for objectives $y \in Y$ is ensembled in the product space, where Y can be either E for edit transformation, W for word string, or C for preceding context words.

Table 3.1: The correction inferences adopted by models under noisy channel framework using Eqn. 3.2.

Model	channel model	source model
Kernighan et al. (1990)	ϕ_{edit}	ϕ_{wp}
Mays et al. (1991)	ϕ_{uni}	$\prod_C \phi_{\text{w3g}}$
Brill and Moore (2000)	$\prod_W \prod_E \phi_{\text{edit}}$	ϕ_{wp}
Cucerzan and Brill (2004)	$\prod_W \prod_E \phi_{\text{edit}}$	ϕ_{wp}
Segaran and Hammerbacher (2009)	$\prod_E \phi_{\text{edit}}$	ϕ_{wp}
Whitelaw et al. (2009)	$\prod_W \prod_E \phi_{\text{edit}}$	ϕ_{wp}
Pirinen et al. (2010)	$\prod_E \phi_{\text{edit}}$	ϕ_{wp}
Pirinen et al. (2012)	$\prod_E \phi_{\text{edit}}$	ϕ_{hmm}

Inferences are categorized based on their dependencies, regardless of the additional transformation or smoothing function on their representation. For example, the source model $p(x)$ is $\frac{f(x)+0.5}{N}$ in Kernighan et al. (1990), $(\frac{f(x)}{N})^{\lambda}$ in Whitelaw et al. (2009), and $-\log(\frac{f(x)}{N})$ in Pirinen et al. (2010). These representations refer to the same type of inference that relies on the unigram frequency in a corpus. Here, we brief the general form of the inference types that has been used:

Weighted elementary edit operation ϕ_{edit}, depend on the frequency of related character unigram and bigrams. Given an error word $S = \{s_i\}_n$, the probability of Damerau-Levenshtein edit operations is given by:

$$\phi_{\text{edit}} \propto \begin{cases} \frac{f_{\text{add}}(<s_i,c>)}{f(<s_i,c>)} & \text{if insert a character } c \text{ after } i\text{-th position} \\ \frac{f_{\text{del}}(<s_{i-1},s_i>)}{f(s_i)} & \text{if delete the character at } i\text{-th position} \\ \frac{f_{\text{sub}}(<s_i,c>)}{f(s_i)} & \text{if substitute the character at } i\text{-th to a different character } c \\ \frac{f_{\text{tan}}(<s_{i-1},s_i>)}{f(<s_{i-1},s_i>)} & \text{if transpose two characters at } i\text{-th and } (i{+}1)\text{-th position} \end{cases}$$

where $f(\cdot)$ indicates the frequency count in the training data, and function with subscripts refers to the frequency count in the corresponding confusion matrix. A confusion matrix records how many times different types of edit operations appears in a character bigram in the dataset. For example, $f_{\text{add}}(<$

$a, b >$) indicates the number of character bigram $< a, b >$ exists in the dataset, where b is a inserted character in the transformation from an error word to its correction. With an exception of Kernighan et al. (1990) that selected candidates with only one edit operation to the error word, other models take the product of the weight of multiple edit operations in a string transformation. Cucerzan and Brill (2004) iteratively partitioned the error string into non-overlapping substrings and estimated weighted edit transformation on each substring.

Uniform error rate ϕ_{uni}, proposed by Mays et al. (1991), is a simple approximation without statistics from a training dataset. It set up a fixed error rate $(1-\alpha)$ for any word and the probability for any candidate is uniformly distributed among all candidates with Damerau-Levenshtein edit distance 1. Given a set of candidates $\mathcal{C}$ that is 1 distance to y, the channel model is

$$p(y|x) \propto \begin{cases} \alpha & \text{if } x = y \\ \frac{1-\alpha}{|\mathcal{C}|} & \text{if } x \in \mathcal{C} \\ 0 & \text{otherwise.} \end{cases}$$

Word probability ϕ_{wp}, given by $p(x) \propto \frac{f(x)}{N}$, where N is the size of the training data or an external corpus. Besides using smoothing techniques to deal with the out-of-vocabulary word, Whitelaw et al. (2009) take the λ-th power of the word probability to adjust the degree of dependency on the word popularity. For example, with the increasing of λ, the correction model will bias to select the candidate with high frequency.

n-gram Language Model ϕ_{wng}, approximates the probability of a word sequence $\{x_i\}_m$ as n-th order Markov model, that is

$$p(\{x_i\}_m) \propto \prod_{j \in [n,m]} p(x_j | x_{j-(n-1)}, \ldots, x_{j-1}).$$

Mays et al. (1991) used the sentence probability measured by trigram language model as the source model.

POS hidden Markov model ϕ_{hmm}, assumes that the word probability is determined by a Markov process with POS tags as the hidden states, that is

$$p(x_i) \propto p(x_i|t_i)p(t_i|t_{i-1}, t_{i-2}, \dots),$$

where x_i is the i-th word in a text and t_i is the according POS tag of the i-th word.

3.2 Confidence Analysis

Confidence analysis is a quantitative approach that rate the correction confidence of candidate words. A confidence measure, deduce from one or more correction inferences, usually assigns each candidate a non-negative score. Existing correction models of this approach, listed in Table 3.2, usually combine multiple confidence measures and take the weighted sum as the final score:

$$\hat{x} = \arg\max_{x \in \mathcal{X}} \sum_{i}^{n} w_i \phi_i(x, y), \tag{3.3}$$

where $\phi_i(x, y)$ for $i \in n$ is one of n confidence measures that evaluate candidate x to error y.

3.2.1 Error Correction Models

Islam and Inkpen (2009a,b,c) proposed a type of confidence models that uses contextual and lexical inferences for correcting real-word errors. One applied confidence measure considers the popularity of using a candidate in the given context. It substitutes the error word by candidate words and constructs word trigrams with a candidate and neighboring words in the text. The confidence of a candidate with

according trigrams t_1 is calculated as $\frac{f(t_i)}{\max_{t \in T} f(t)}$, where $f(\cdot)$ is the frequency of a trigram in Google Web 1T n-gram corpus (Brants and Franz, 2006) and T is a set of trigrams constructed by different candidates of this error word. The other measure uses the weighted sum of four modified LCS measures to evaluate the lexical similarity between a candidate and the error word.

Islam and Inkpen (2011) described another correction model that rates the generated candidate texts, where each candidate text is a candidate correction for the entire input text. This model combines three measures: two pairwise text similarity measures that calculate the degree of word alignment and misalignment; and a text popularity measure that considers the frequencies of the consisting word 5-grams in Google Web 1T n-gram corpus.

Schaback and Li (2007) and Bao et al. (2011) introduced two models that learned weighting scheme using Support Vector Machine (Joachims, 1999). Kissos and Dershowitz (2016) proposed an OCR correction model that learned weights using logistic regression. Benefitting from the learning approach, these models are flexible in distributing weights among confidence measures. Thus, these two models separated the same type of correction inference into different confidence measures, where each measure with a different focus. For example, Schaback and Li (2007) adopted three measures of context bigram coocurrence popularity, where two use $\frac{f(x_i,x_{i+1})}{f(x_i)}$ with different neighboring words and the last one takes the product of the previous two. To correct errors in email, Bao et al. (2011) made three different binary measure for identifying whether a candidate word exists in "Subject", "From", or "X-From" field. [2]

3.2.2 Correction Inferences

Inferences are categorized by the statistics used in the calculation and are annotated as ϕ with subscripts, including ϕ_{edit} uses character edits, $\phi_{\text{w}n\text{g}}$ uses word n-gram statistics (e.g. ϕ_{w1g}, ϕ_{w2g}), ϕ_{ph} uses phonetic information, ϕ_{lex} uses lexicons, and

[2] More to read: (Gao et al., 2010; Sun et al., 2010; Flor)

Table 3.2: Weighting schemes and correction inferences adopted by models under noisy confidence analysis framework using Eqn. 3.3.

model	weighting scheme	correction inferences
Schaback and Li (2007)	SVM	$\phi_{\text{w1g}} + \sum \phi_{\text{w2g}} + \sum \phi_{\text{ctxt}}$
Islam and Inkpen (2009a,b,c)	huristics	$\sum \phi_{\text{LCS}} + \phi_{\text{w3g}}$
Gao et al. (2010)	averaged	$\sum \phi_{\text{LCS}} + \phi_{\text{w3g}}$
Islam and Inkpen (2011)	huristics	$\phi_{\text{w5g}} + \sum \phi_{\text{ctxt}}$
Bao et al. (2011)	SVM	$\phi_{\text{edit}} + \phi_{\text{ph}} + \phi_{\text{w1g}} + \sum \phi_{\text{lex}}$
Kissos and Dershowitz (2016)	logistic regression	$\phi_{\text{edit}} + \phi_{\text{w1g}} + \sum \phi_{\text{w2g}}$

ϕ_{ctxt} uses statistics of context words (excluded by ϕ_{wng}). $\sum$ indicates that there are multiple measures of the same inference type are applied in the model.

3.3 Framework Comparison

A compositional correction frameworks that combine correction inferences are broadly researched in recent literature. Noisy channel interprets the likelihood under probabilistic and decomposes using Bayes's rule, while the confidence analysis summates different similarity measures between the error and candidate words. The intuitive and straightforward decompositions of both frameworks benefit the modeling in many aspects for the error correction task.

First of all, the correction model in this approach is easy to build from data. For the noisy channel, given a specific environment, for example handwriting documents or OCR-generated texts, the behavior of error generation $p(y|x)$ is fixed, thus can be directly estimated by a set of erroneous observations. The source model $p(x)$ is unconditional so that can be estimated from unlabeled data set. On the other hand, confidence analysis relies on independent correction inferences, where inferences are similarity measures that computed from corpus statistics, such as, word or n-gram frequencies.

Secondly, different correction inferences can be merged into one model, where the collaborated inferences are carried out in the summation or production space.

There are two types of collaboration semantics: first, a channel model can further decompose into submodels that handle different parts of the correction. For example, Kernighan et al. (1990) applied different probabilistic inferences for different edit operations in a Damerau-Levenshtein edit transformation. Second, noisy channels using different correction inference can be used combinatory. For example, combining character edit probabilities with contextual word n-gram probabilities (Pirinen et al., 2012).

It is easy to observe that the compositional forms of two frameworks are analogous. The compositional form of the noisy channel, in Eqn. 3.2, is able to be transformed to summation in a log space

$$\hat{x} = \arg\max_{x \in \mathcal{X}} \big(\sum_{y_i}^{Y} \log p(y_i|x) + \log p(x) \big). \tag{3.4}$$

Comparing with another summation form, which is Eqn. 3.3 for confidence analysis, there are some differences in compositing correction inferences. We analyze merit and demerit along with the differences between two frameworks as follows:

– Noisy channel restricts the adopted inferences to be probability measures, while confidence analysis accepts any quantitative measure. For some correction inferences, this difference may only lead to different normalization factors. For example, an inference given by word frequency in a corpus is $\frac{f(x)}{Z}$, where Z can be any variable to use in confidence analysis, but ought to be the corpus size in order to be a probability measure and adopted in the noisy channel. More importantly, this restriction rejects some inferences. For example, probabilities of edit operations in an edit transformation is a common channel model, however, the unit-cost edit distance value is hardly being directly considered.

– confidence analysis involves weighting schemes, while noisy channel does not. This feature let confidence analysis be more expressive in reasoning a selection decision by composition. It is also a reason why confidence analysis accepts inferences that do not directly infer a correction probability – a weak inference

will have lower weight and vice versa – the contribution to the decision making is being considered in the weighting scheme. In this regard, noisy channel is an over-simplification since inferences are equally weighted.

On practical side, the distinction between two frameworks is getting blurred on some proposed correction models (Gao et al., 2010; Bao et al., 2011). Although claimed to use noisy channel framework, some models are actually beyond its paradigm and involve above features from confidence analysis.

Chapter 4

Proposed Model

In the previous chapter, we illustrated that the inference composition is an influential correction approach, which is easy to build from data and to expand with multiple inferences. Comparative analysis of two compositional frameworks shows that using a weighting scheme is more plausible in reasoning contributions among generalize inferences, which are quantitative measures of lexical or contextual similarity and their importance to the decision making are varied.

For OCR postprocessing task, the generation of OCR error are complex, and hence the effectiveness of some widely adopted inferences is ambiguous. For example, the error $\langle$`da}'` $\rightarrow$ `day`$\rangle$ caused by segmentation issue during OCR procedure, which incorrectly separate "`y`" into "`}'`". It takes two dependent edit operations. However, the correlation of these two operations cannot be considered in simple composition. Simply extend the edit operation into character n-gram is likely to get overfit to the training data.

Another issue is the limit amount of the labeled OCR error records. It is problematic when correction models rely on inferences from error patterns and some error patterns are not observed in the training data. Such case is likely to happen considering the protean natural of OCR errors.

In this chapter, we propose an ensemble regression approach for correction OCR errors. Ensemble method is the learning algorithm that combines multiple predictors to improve the overall prediction result. Empirically, ensembles tend to yield better performance when there is high diversity among the models (Kuncheva and Whitaker, 2003; Sollich and Krogh, 1996) and to reduce the generalization error. Different from confidence analysis models that learn a weighing scheme among features, the

ensemble method is adopted to learn to rank the candidates for an error according to the confidence as an intended correction. Given these problems that make OCR correction hard to be modeled directly, the proposed approach uses inferences that describe the different aspects of lexical or contextual features on candidates. Consider the limited amount of data that makes supervised correction models less accurate, the training process involves not only the correct candidates but also the rejected ones. Figure 4.1 shows the overview of the proposed model. The details of each processing steps will be introduced in the following sections.

4.1 Error Detection

Error detection step identifies errors in the tokenized text, which is the first step in the correction procedure. Since a correct word will not proceed to the further correction steps, we want to set a weak detection restriction to filter only highly confident words. We rely on the n-gram frequency to determine the correctness of a word. A word is detected as an error if any one of the following conditions does not fulfill.

- Consider a common word is less likely to be an error word, the 1-gram frequency of a word should be greater than a frequency threshold. The frequency threshold varies with different word length.
- A word is likely to be correct if this word with its context occurs in other places. We use a sliding window to construct n-gram contexts for a word. The frequency of one of the context in the n-gram corpus should be greater than a frequency threshold.

4.2 Candidate Search

We select a candidate set for each error, which contains all the words in the vocabulary within a limited number of character modifications. To be specific, let Σ be the

symbol set, $\mathcal{L} \in \Sigma^*$ be a language lexicon. The candidate set for a detected error w_e is:

$$\{\ w_c\ |w_c \in \mathcal{L}, dist(w_c, w_e) \leq \delta\}, \tag{4.1}$$

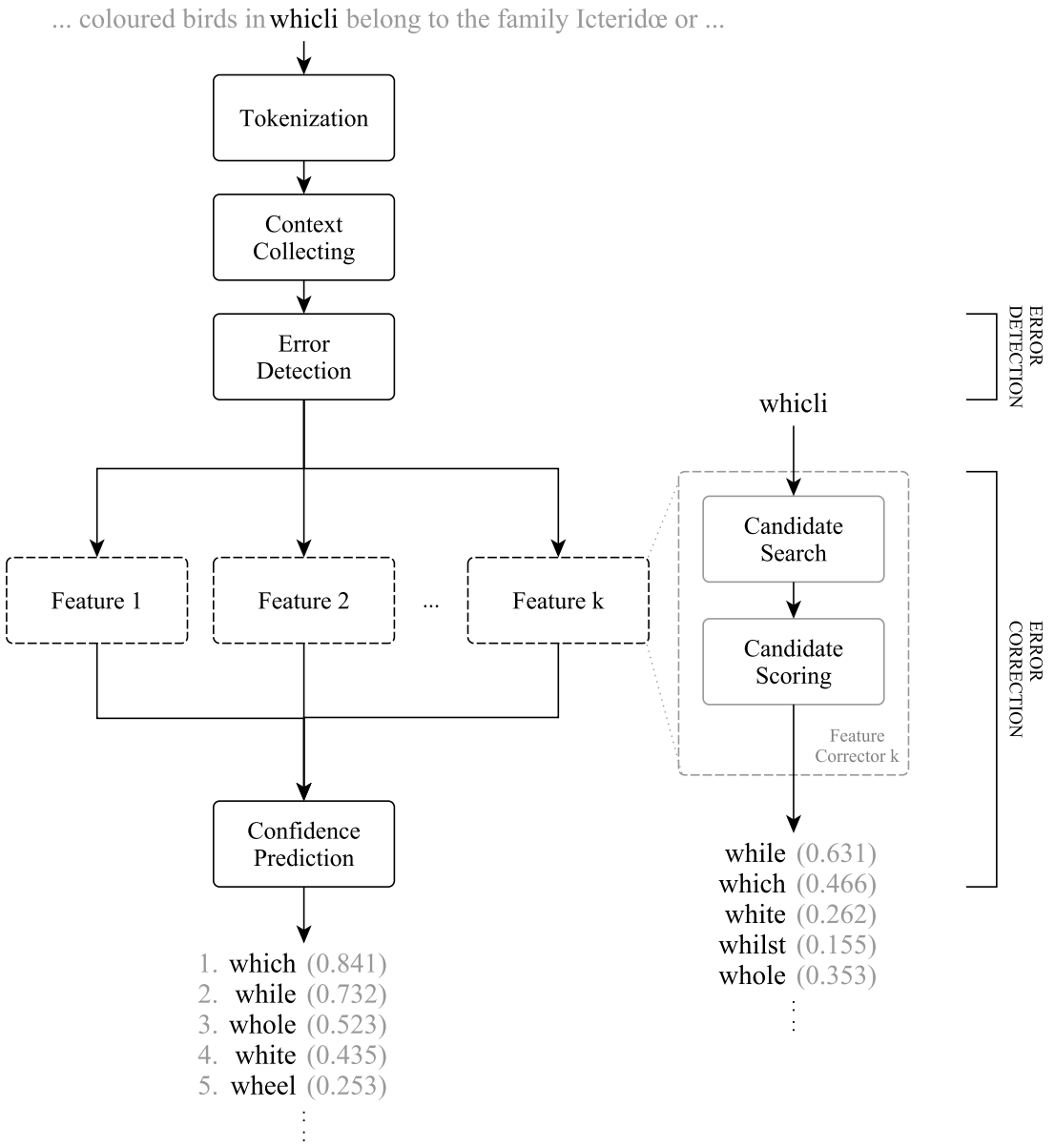


Figure 4.1: Proposed model overview.

where $dist(*)$ is the minimum edit distance and δ is a distance threshold. Damerau-Levenshtein distance (Damerau, 1964), which is used in most of the spell correction models for locating the candidates, considers all four character-level editing types. Since transposition errors are common in human-generated text but rarely occur in the OCR-generated text, we apply Levenshtein distance (Levenshtein, 1966), which uses a simpler operation set without transposition.

4.3 Feature Scoring

We discuss different features that can be used to quantify the likelihood of a candidate word being an error correction. From the mathematical point of view, each candidate evaluation feature is a function

$$\phi : \Sigma^* \times \Sigma^* \rightarrow \mathbb{R} \tag{4.2}$$

that receives two strings – an error word and a candidate word – and produces a quantitative degree of likelihood that the given candidate is an expected correction.

With the focus on evaluating correction candidates, we modify and adopt developed measures from correction inferences used in the error correction literature as well as some closely related NLP tasks:

Approximate String Mathing The general goal of this task is to evaluate the degree of lexical differences between two sequential patterns. A typically usage scenario compares a corrupted sequence with a correct seqeunce, and model the divergence as the minimum effort to align two sequences. The application of this task including detecting mutations in DNA sequences (Sellers, 1980; Needleman and Wunsch, 1970; Sankoff and Kruskal, 1983; Altschul et al., 1990; Myers, 1991, 1994; Waterman, 1995; Yap et al., 1996; Gusfield, 1997), correcting error for natural language texts (Wagner and Fischer, 1974; Nesbit, 1986; Owolabi and McGregor, 1988; Kukich, 1992b; Zobel and Dart, 1996; French

et al., 1997; Gonnet and Baeza-Yates, 1991), recovering signals that transmitted via noisy channels (Levenshtein, 1965, 1966; Vintsyuk, 1968; Dixon and Martin, 1979), and many more mentioned in Sankoff and Kruskal (1983) as well as Kukich (1992b).

Word Similarity This task evaluates the degree of semantic relatedness between two words. The models of this task infer pairwise word similarities from lexical (Kondrak, 2005; Islam and Inkpen, 2008), semantic (Strube and Ponzetto, 2006; Gabrilovich and Markovitch, 2007), and contextual corelations (Finkelstein et al., 2002; Islam and Inkpen, 2006; Jarmasz and Szpakowicz, 2012; Islam et al., 2012).

These adopted measures calculate the lexical, semantic, or contextual relatedness between two strings. Note that some measures proposed in the above tasks that makes use of the semantic relations are not suitable for this problem. For example, Jarmasz and Szpakowicz (2012) represent the given word as a TF-IDF (Salton and McGill, 1986) vector of the closely related concepts in Wikipedia and estimate pairwise word similarity using cosine measure (Zobel and Moffat, 1998). This measure is not useable for representing error word and such semantic clue is irrelevent to evaluate correction candidate.

To increase the adaptiveness of feature values in learning models (Juszczak et al., 2002; Mohamad and Usman, 2013), we linearly rescale each feature value into range $[0, 1]$

$$\phi(S_1, S_2) = \frac{g(S_1, S_2) - g_{\min}}{g_{\max} - g_{\min}}, \tag{4.3}$$

where $g(\cdot)$ and $\phi(\cdot)$ are the feature values before and after the rescaling procedure. $g_{\min}$ and g_{max} are the minimum and maximum values computed using this feature in a processing batch. For feature function that has $g_{\min} \equiv 0$, we can simplify Eqn. 4.3

as

$$\phi(S_1, S_2) = \frac{1}{Z_f} g(S_1, S_2), \tag{4.4}$$

where $Z_f = g_{\max}$ is a feature specific constant. We do not apply standarization since the feature values, from either a string similarity measure or a string distance measure, are not Gaussian distributed. Considering the long tail distribution of word (n-gram) distribution, for measures that take advantage of aggregated frequencies from corpus, the feature values are mapped to the log space with add-one smooth before rescaling.

$$\begin{aligned} \phi(S_1, S_2) &= \frac{\log g(S_1, S_2) - \log g_{\min}}{\log g_{\max} - \log g_{\min}} \\ &\approx \log \frac{g(S_1, S_2) + 1}{g_{\max}}. \end{aligned} \tag{4.5}$$

We classify the candidate evaluation features into three families: *edit distance* family, *character n-gram* family, and *contextual word n-gram* family. The measure classification is base on the principles in similarity/dissimilarity computation, where edit distance family considers the single character modification, character n-gram family makes uses of the string subsequences, and contextual word n-gram family infers correction confidence from neighboring words. Table 5.4 lists all the measures that are considered in this thesis. We will elaborate each of these three families on the following subsections.

4.3.1 Edit Distance Family

Edit distance family refers to a set of string simlarity/dissimilarity measures that take advantage of edit operations. An edit operations is a character level modification, which contains four types – *insertion*, *deletion*, *substitution* one character into another, *transposition* between two consequtive characters – different measures allow only a subset of edit operations to be used, which are listed in Table 4.2. Some measures calculate the standard edit distance between two strings – the minimum

Table 4.1: Overview of applied features in feature families

Feature Family	Feature Measure	Score Function
Edit Distance	Hamming Distance	Eqn. 4.6
	Longest Common Subsequence Distance	Eqn. 4.6
	Levenshtein Distance	Eqn. 4.6
	Damerau-Levenshtein Distance	Eqn. 4.6
	Optimal String Alignment	Eqn. 4.6
	Jaro-Winkler Distance	Eqn. 4.7
	Longest Common Subsequence Similarity	Eqn. 4.8
Character n-Gram	n-gram Jaccard Coefficient	Eqn. 4.9
	q-Grams Distance	Eqn. 4.10
	n-Grams Distance	Eqn. 4.11
Contextual Word n-Gram	Language popularity	Eqn. 4.12
	Lexicon existence	Eqn. 4.13
	Context Coherence	Eqn. 4.14
	Approximate Context Matching	Eqn. 4.14

number of edit operations required to transfrom one string to another[1] – including Hamming distance, longest common subsequence distance, Levenshtein distance, Damerau-Levenshtein distance, and Optimal String Alignment. The scoring function of these standard edit distance measures is

$$\phi_{\text{dist}}(S_1, S_2) = \frac{\delta_{\text{dist}}(S_1, S_2)}{Z_{\text{dist}}}, \tag{4.6}$$

while other measures use distinct formulas to capture lexical characteristics in different apects.

Hamming Distance (Hamming, 1950) is the edit distance allowing only substitution operation and $|S_1| = |S_2|$. The Hamming distance was developed for error detecting and correcting codes. It is used to define some important notions in disciplines including information theory, coding theory, and cryptography.

[1]See Appendix A.1 for the definition of edit distance measuring.

Table 4.2: List of applied feature measures and according edit operation sets. The measure name that annotated with * indicates its compuatation procedure does not follow the standard edit distance.

Feature Measure	Edit Operation Set
Hamming Distance	subsubsection
Jaro-Winkler Distance*	transposition
Longest Common Subsequence Distance	insertion, deletion
Longest Common Subsequence Similarity*	insertion, deletion
Levenshtein Distance	subsubsection, insertion, deletion
Damerau-Levenshtein Distance	subsubsection, insertion, deletion, transposition
Optimal String Alignment	subsubsection, insertion, deletion, transposition

Hamming distence is limited in comparing strings with the equal length. Thus, it is not able to be applied as a metric for general string comparison.

Longest Common Subsequence Distance (Navarro, 2001) is the edit distance allowing insertion and deletion operations. Longest common subsequence (LCS) (Sankoff, 1972) algorithm measure the length of the ordered character pairs between two strings. LCS distance is a distance metrics based on LCS, which measures the number of unpaired characters.

Levenshtein Distance (Levenshtein, 1966) is the edit distance allowing substitution, insertion, and deletion operations. Levenshtein distance was first proposed in the context of binary error-correcting codes, which allows three types of edit operations: *reversal*, *deletion*, and *insertion*. Wagner and Fischer (1974) generalized this work with finite alphabet, w.l.o.g, *reversal* operation became *substitution* which denotes a character changes into another one.

Damerau-Levenshtein Distance (Damerau, 1964) is the edit distance allowing substitution, insertion, deletion, and transposition operations. Compare with Levenshtein distance, Damerau-Levenshtein distance considers *transposition* operation and is able to includes 80% of human misspellings within distance 1 (Damerau, 1964).

Optimal String Alignment (Boytsov, 2011) add an addition restriction to Damerau-Levenshtein Distance that there allows only one edit operation performed on a substring.

Jaro-Winkler Jaro distance (Jaro, 1989) is a metric that suitable for measuring phrases such as person or place names

$$\phi_{\text{Jaro}}(S_1, S_2) = \frac{1}{3}(\frac{m}{|S_1|} + \frac{m}{|S_2|} + \frac{m-t}{m}), \text{if } m > 0 \text{ or } 0 \text{ otherwise,}$$

where t refers to the number of transposition operation between two strings and m is the number of aligned characters. The character alignment is defined as two identical characters from two strings s_1 and s_2 respectively and the difference of character index is no more than $\lfloor \frac{max(|s_1,s_2)}{2} \rfloor - 1$. Jaro-Winkler distance (Jaro, 1990) is a variation of Jaro distance (Jaro, 1989), which is favourable to string pairs with common prefix

$$\phi_{\text{Jaro-Winkler}}(S_1, S_2) = \phi_{\text{Jaro}}(S_1, S_2) + l \cdot p \cdot (1 - \phi_{\text{Jaro}}(S_1, S_2)) \tag{4.7}$$

where l is the length of the common prefix and p is a constant scaling factor which is set to 0.1 in the proposed paper.

LCS Similarity (Allison and Dix, 1986) (LCS) is an alternative approach than edit distance in matching similar strings. There are variations of LCS: *Normalized Longest Common Subsequence (NLCS)*, which take into account the length of both the shorter and the longer string for normalization.

$$\phi_{\text{NLCS}}(S_1, S_2) = \frac{2 \cdot \phi_{\text{LCS}}(S_1, S_2)^2}{|S_1| + |S_2|}.$$

Normalized Maximal Consecutive Longest Common Subsequence (MCLCS), which limits the common subsequence to be consecutive. There are three types of modifications with different additional conditions: $NLCS_1$ and $NLCS_n$ use the subsequences starting at the first and the n-th character, respectively;

$NLCS_z$ takes the subsequences ending at the last character. They apply the same normalization as $NLCS$.

$$\phi_{\text{NMNLCS}_1}(S_1, S_2) = \frac{2 \cdot \phi_{\text{MCLCS}_1}(S_1, S_2)^2}{|S_1| + |S_2|}$$
$$\phi_{\text{NMNLCS}_n}(S_1, S_2) = \frac{2 \cdot \phi_{\text{MCLCS}_n}(S_1, S_2)^2}{|S_1| + |S_2|}$$
$$\phi_{\text{NMNLCS}_z}(S_1, S_2) = \frac{2 \cdot \phi_{\text{MCLCS}_z}(S_1, S_2)^2}{|S_1| + |S_2|}.$$

We apply the measure proposed in (Islam and Inkpen, 2009b) for scoring, which takes the weighted sum of the above LCS variations:

$$\begin{aligned}\phi_{\text{LCS-Sim}}(S_1, S_2) \; &= \alpha_1 \cdot \phi_{\text{NLCS}}(S_1, S_2) + \alpha_2 \cdot \phi_{\text{NMNLCS}_1}(S_1, S_2) \\ &+ \alpha_3 \cdot \phi_{\text{NMNLCS}_n}(S_1, S_2) + \alpha_4 \cdot \phi_{\text{NMNLCS}_z}(S_1, S_2). \end{aligned} \tag{4.8}$$

4.3.2 Character n-Gram Family

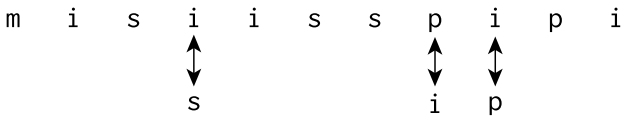
Another family of string comparison measures represents string as a list of overlapping character n-grams, which is also called the string n-gram profile and annotated as

$$S_{<n>} = \{S_{<i,i+n-1>}\}_{m-n+1}$$

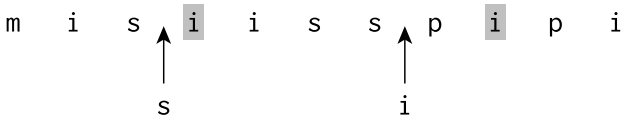
for string S of length m. For example, the bigram profile for string `family` is $\{\texttt{fa}, \texttt{am}, \texttt{mi}, \texttt{il}, \texttt{ly}\}$. Measures in this family evaluate the lexical similarity between strings using their n-gram profiles. Figure. 4.4 gives an examples for each introduced measure.

n-Gram Jaccard Jaccard coefficient (Jaccard, 1908) is a widely used similarity measures between finite sets. We use this coefficient to evaluate the similarity between string n-gram profiles

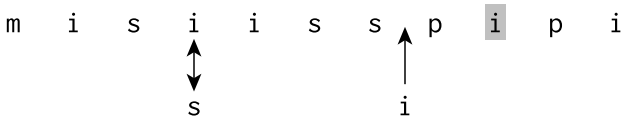
$$\phi_{\text{n-Jac}}(S_1, S_2) = \frac{|S_{1<n>} \cap S_{2<n>}|}{|S_{1<n>} \cup S_{2<n>}|}. \tag{4.9}$$



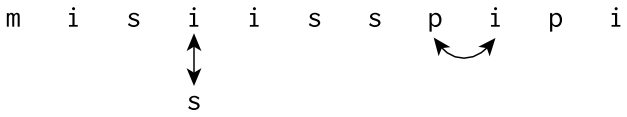
(a) An example edit transformation leading to Hamming distance (Hamming, 1950).



(b) An example edit transformation leading to LCS distance (Navarro, 2001).



(c) An example edit transformation leading to Levenshtein distance (Levenshtein, 1966).



(d) An example edit transformation leading to Damerau-Levenshtein distance (Damerau, 1964).

Figure 4.2: Examples of edit distance family. Distance calculation is performed between `mississippi` and `misiisspipi`. Different edit operations are notated as follows:

- a one-way arrow indicates an insertion operation for an character to a position;
- a shaded character indicates a deletion operation on this character;
- a two-way arrow that points to a character in string and a character outside indicates a substitution operation between two characters;
- a two-way arrow that points to two successive characters in string indicates a transposition operation between two characters.

q-Grams Distance q-grams distance (Ukkonen, 1992) is a distance measure that calculates the total number of unmatching n-grams in both string n-gram profiles. We scale the q-grams distance with a function specific normalization factor, $Z_{\text{n-qGrams}}$, which is the maximum q-grams distance in the processing batch. This feature function is

$$\phi_{\text{n-qGrams}}(S_1, S_2) = \frac{1}{Z_{\text{n-qGrams}}} |S_{1<n>} \cup S_{2<n>}|. \tag{4.10}$$

n-Gram Distance Kondrak (2005) introduced an extension of Levenshtein distance that each edit operation modifies n consequtive characters instand of a single character. To enphasize the initial characters in computation, Kondrak (2005) affix $n-1$ null-symbols to the head of the orignal string, which increases the occurrence of initial characters in n-gram profile. There are three proposed cost functions to measure edit cost with n-grams:

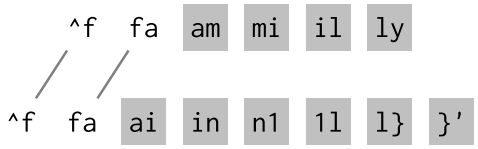
- *binary* cost gives 1 to a pair of exactly matched n-gram pairs and 0 otherwise;
- *positional* cost gives $\frac{1}{n}$ to each matching character pair in the given n-gram pairs and 0 otherwise, where character pairs are constructed with characters of the same index from both n-grams.
- *comprehensive* cost is the same as positional cost, except character pair can be constructed with character of different indices.

Kondrak (2005) normalize the transformation cost by the lenght of the longer input string, which breaks the triangle inequality. Thus, we get the feature value by scaling the minimum transformation cost

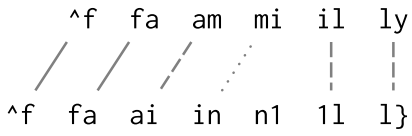
$$\phi_{\text{n-nGrams}}(S_1, S_2) = \frac{1}{Z_{\text{n-nGrams}}} \min_{E_{S_1,S_2} \in \mathcal{E}_{S_1,S_2}} \gamma_E(E_{S_1,S_2}). \tag{4.11}$$

	1l	am	ai	fa	il	in	ly	l}	mi	n1	}'
family	0	1	0	1	1	0	1	0	1	0	0
fain1l}'	1	0	1	1	0	1	0	1	0	1	1

(a) Example using Jaccard Similarity Coefficient (Jaccard, 1908) to compare two string n-gram profiles. The binary value indicates the existance of a character bigram in a string. The bigram Jaccard similarity of these two strings is $1/11 \approx 0.09$.



(b) Example alignment of q-gram distance (Ukkonen, 1992). A solid link donotes a matched bigram pair. Unmatched bigrams are highlighted. The q-gram distance ($q = 2$) of this two strings is 10.



binary	0 + 0 + 1 + 1 + 1 + 1 + 1 + 1 =	6
positional	0 + 0 + .5 + 1 + 1 + .5 + .5 + 1 =	4.5
comprehensive	0 + 0 + .5 + .5 + 1 + .5 + .5 + 1 =	4

(c) Example alignment of n-gram distance (Kondrak, 2005). Symbol ˆ denotes a null-character prefix that increase the weight of the heading character in distance calculation. There are three line types indicating different categories of bigram matchings:

- a solid link denotes a fully matched bigram pair,
- a dash line denotes a positionally partially matched pair, and
- a dot line denotes a partially matched pair that the matching character are not in the same position of crossponding bigrams.

Without normalization, the *binary*, *positional*, and *comprehensive* n-gram distance ($n = 2$) of these two strings are 6, 4.5, and 4, respectively.

Figure 4.3: Examples of character n-gram family. Distance calculation is performed between `family` and `fain1l}'` using their string bigram profiles.

4.3.3 Contextual word n-Gram Family

In addition to the lexical relations between two given words, we also evaluate candidate with external resources using the word itself and its neighboring words. Features in this family quantify the likelihood of using a candidates in scenario, with respect to the langage in general, professional domains, or a specific usage case. Their functions, as a simplified form of Eqn. 4.2, takes only the candidate word as input

$$\phi : \Sigma^* \to \mathbb{R}.$$

There are two types of external resources being used: lexicon and n-gram corpus. Lexicon, interchangeably with "dictionary" and "word list" in the literature, is a list of language's words. n-gram corpus is a list of word n-gram, $n \in \mathbb{R}$, that each n-gram has a observed frequency counts. For all features that make use of n-gram statistics in this these, we adopt Web 1T 5-gram corpus (Brants and Franz, 2006). Web 1T 5-gram corpus, contributed by Google Inc., contains English word n-grams ranging from unigrams to five-grams which are generated from approximately 1 trillion word tokens of text from publicly accessible web pages.

Language popularity We evaluate the likelihood of using a candidate in English language. With the English word distribution estimated by unigram frequencies, the language popularity feature is

$$\phi_{\text{lang}}(S) = \frac{f(S)}{Z_{\text{lang}}}, \tag{4.12}$$

where $c(\cdot)$ is a function that finds the frequency count of the given unigram.

Lexicon existence We use domain lexicons to detect the existence of an candidate in a specific subject matter. This feature helps to capture domain specific

terminologies, which may not be popular in general usage.

$$\phi_{\text{lex}}(S) = \begin{cases} 1 & \text{if } S \text{ exists in the lexicon} \\ 0 & \text{otherwise} \end{cases} \tag{4.13}$$

Context Coherence An appropriate correction candidate should be coherent in context. Using word n-gram for context analysis is a broadly researched approach in correcting real-word errors (Islam and Inkpen, 2009b). Given an error word w_e in a text, we have its n-gram contexts $\mathcal{G}$ constructed using a sliding window (see Figure 4.4). To score a candidate w_c of this error, we first substitute the error word from each of its n-gram contexts by such candidate and create a new set of contexts $\mathcal{G}_c$. Let $\mathcal{C}$ be all candidates suggested for w_e, and $freq_n(\cdot)$ be the n-gram frequency, which gives 0 to a non-existing n-gram. The score function is given as:

$$\phi(S) = \frac{\sum_{\mathbf{c} \in \mathcal{G}_c} f_n(\mathbf{c})}{\max_{\text{w}'_c \in \mathcal{C}} \{\sum_{\mathbf{c}' \in \mathcal{G}'_c} f_n(\mathbf{c}')\}} \tag{4.14}$$

Approximate Context Matching A context with longer n-gram size defines a more specific use case for a given word, where its existence in the corpus shows higher confidence for a candidate. In general, an n-gram corpus has limited coverage for all possible n-grams in the language, especially for the emerging words in the language. Candidates of a rare word can barely be suggested from its contexts because of the limited coverage in the n-gram corpus. We deal with such issue by relaxing the context matching condition to allow one mismatching context word. For example, in Figure 4.4, we consider only the first 5-gram context given "which" be the candidate. We need the frequency of "brightly coloured birds in which" for computing exact context popularity. As for the relaxed context popularity, we need to sum up the frequencies of four types of 5-grams: "* coloured birds in which", "brightly * birds in which", "brightly coloured * in which", and "brightly coloured birds * which", where

* matches any valid unigram. The scoring function is the same as the exact context matching (Eq. 4.14), except for the candidate set and the context set are larger in the relaxed case.

4.4 Candidate Ranking

We formulate the confidence prediction task as a regression problem. Given candidate feature scores, we predict the confidence of each candidate being a correction for the error word. The confidence is used for ranking among candidates of one error.

To train a regressor for correction, we label candidate features with 1 if a candidate is the intended correction, or 0 otherwise. The training data contains candidates from different errors, and there are more candidates labeled 0 than 1. To deal with the unbalanced nature of the candidates, we weight the samples when computing the training loss

$$loss(\mathcal{D}) = \sum_{e \in \mathcal{E}} \sum_{c \in \mathcal{C}_e^{\mathcal{F}}} w_c \cdot loss(\mathbf{x}_c, y_c). \tag{4.15}$$

We count the number of samples with label 1 and 0, respectively. Then, we use the ratio to weight for samples labeled 1, and 1 for samples labeled 0.

We applied different ensamble methods and evaluated their performance in the next Chapter.

Figure 4.4: Examples of exact and relaxed word 5-gram context generation. ∗ denotes a related word position.

Chapter 5

Evaluation

To better describe the error sample, we use the annotation <$w_t \rightarrow w_e$> to represent the intended word w_t being recognized as the error word w_e.

5.1 Experimental Setup

5.1.1 OCR Dataset

We made available a dataset with 2910 OCR-generated errors along with the ground truth and OCR text for benchmark testing[1]. The OCR text was generated from the book titled "Birds of Great Britain and Ireland (Volumn II)" (Butler et al., 1907) and made publicly available by the Biodiversity Heritage Library (BHL) for Europe[2] using Tesseract 3.0.2[3]. The ground truth text is based on an improved OCR output[4] and adjusted manually to match with the original content of the whole book.[5]

This source image data of the book contains 460 page-separated files, where the the main content is included in 211 pages. Table 5.1 shows the OCR performance, measured by precision and recall, indicating a typical OCR output with high error rate. The statistical analysis on these errors confirmes my inductive arguments in Section 2.2 and demonstrates that correction for this dataset is a complicated task: First of all, some complex errors that has large edit distance away from their correct from in ground truth. The distribution of error words with respect to edit distance shown in Table. 5.2. Secondly, this book combines different font types and layouts in

[1] https://github.com/jmei91/MiBio-OCR-dataset
[2] http://www.biodiversitylibrary.org/item/35947#page/13/mode/1up
[3] https://github.com/tesseract-ocr/tesseract
[4] http://www.bhle.eu/en/results-of-the-collaboration-of-bhl-europe-and-impact
[5] See Appendix B for the preprocessing details in generating the evaluation dataset.

main text, which example shown in Figure. 5.1, leading to erroneous results caused by multiple reasons. More importantly, the correction of some types of errors, for example errors on punctuation or onomatopoeia word in Table. 5.3, can hardly been inferred from lexical or contextual informations.

5.1.2 Evaluation Setup

Walker and Amsler (2014) claim that the lexicon from a published dictionary has limited coverage on newswire vocabulary, and vice versa. Thus, we construct a language lexicon with unigrams in the Google Web 1T n-gram corpus[6]. This corpus contains the frequencies of unigrams (single words) to five-grams, which is generated from approximately 1 trillion word tokens extracted from publicly accessible Web pages. Its unigram corpus is filtered with the frequency no less than 200. We use five-grams in Google Web 1T corpus on contextual features.

For lexicon existence feature, we use three lexicons to build three features instances: (1) *Wikipedia entities* extracted from article names in Wikipedia. This feature gives credit to common terminologies. (1) *Terminology list* that contains a comprehensive terminologies from different domains. (2) *Biodiversity terminologies* collected from biodiversity digital library to capture the domain specific terms, which may not be contained in Wikipedia.

The proposed model receives OCR-generated plain text as input. We apply the Penn Treebank tokenization with the additional rules from Google[7] to tokenize the input text. This tokenization method is consistent with the Google Web 1T n-gram corpus. The frequency and existence of rarely hyphenated words can be poorly estimated using external resources. Thus we split the hyphenated word by the internal hyphen.

Experimentally, we filter tokens of the following types after tokenization: (1) *punctuations*; (2) *numeric tokens*, which contains only numeric characters (i.e., 0-9); (3) *common English words*. We apply a lexicon of frequent English words for

[6] https://catalog.ldc.upenn.edu/LDC2006T13
[7] https://catalog.ldc.upenn.edu/docs/LDC2006T13/readme.txt

Table 5.1: Prcision and recall of OCR on evaluation image files, where measures are defined as:

$$\text{OCR precision} = \frac{\text{number of correct items}}{\text{number of items in OCR output}}$$

$$\text{OCR recall} = \frac{\text{number of correct items}}{\text{number of items in ground truth}}$$

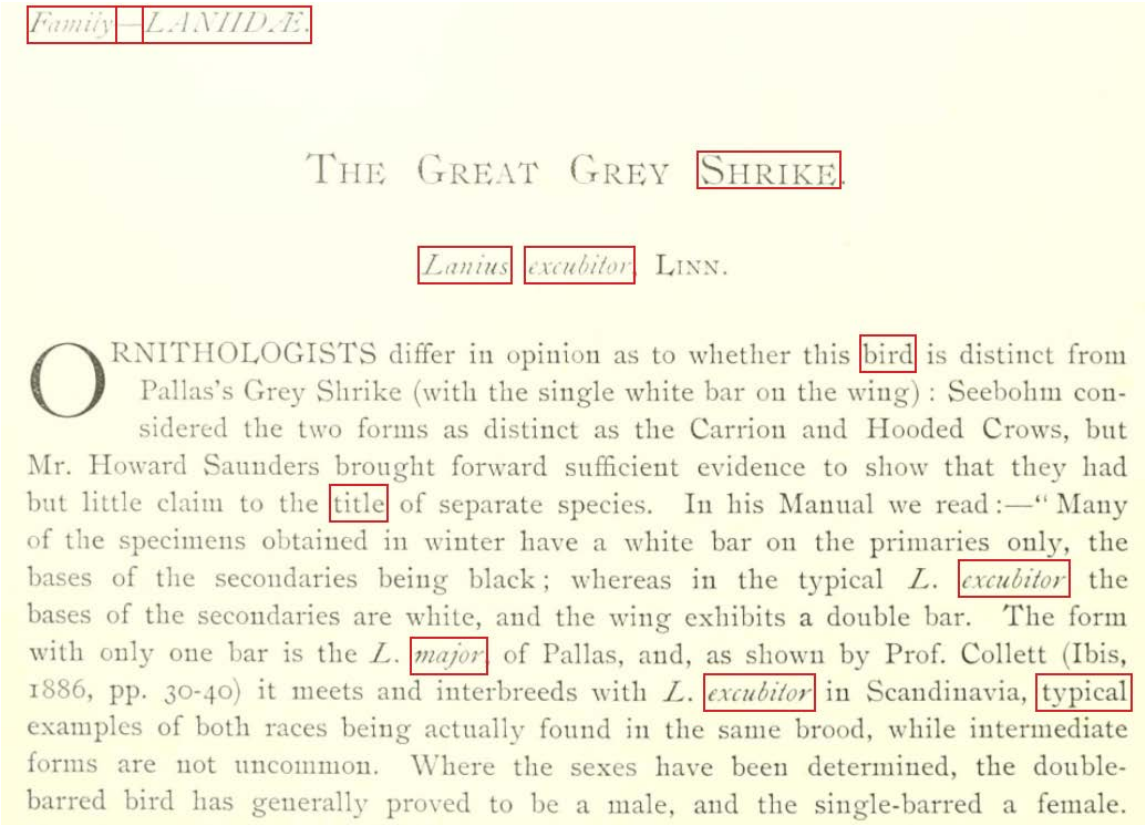
Measure	Character-wise	Word-wise
Precision	6563 / 497275 = 1.28%	2904 / 100033 = 6.36%
Recall	6563 / 492584 = 1.29%	2904 / 98097 = 6.49%

Table 5.2: The Levenshtein edit distance distribution of errors in the experimental dataset.

Edit distance	Error Statistics		Sample Error	
	Number	Percentage	Intended Word	Error Word
1	872	30.03%	galbula	ga/bula
2	1397	47.49%	yellowish	jˆellowish
3	298	10.26%	bents	Ijcnts
4	155	5.34%	my	ni}’
5	74	2.55%	Lanius	Lioiiits
6	57	1.96%	minor	)iii>iof
7	27	0.93%	garrulus	f;ay>///us
8	16	0.55%	curvirostra	iUi’7’iyosira
9	7	0.24%	Nucifraga	Aiiii/rutˆd
≥ 10	19	0.65%	pomeranus	poiiui-iVtiis
total	2904	100%		

Table 5.3: The type distribution of error in the experimental dataset.

Type	Error Statistics		Sample Error	
	Number	Percentage	Intended Word	Error Word
puntutaion	122	4.20%	-	ˆ
onomatopoeia	61	2.10%	füid	fnid
unicode error	11	0.38%	Zippammer	Zippa≫n>icr
unicode word	237	8.16%	ni}’	my



Family—*LANIIDÆ*.

THE GREAT GREY SHRIKE.

Lanius excubitor, LINN.

ORNITHOLOGISTS differ in opinion as to whether this bird is distinct from Pallas's Grey Shrike (with the single white bar on the wing) : Seebohm considered the two forms as distinct as the Carrion and Hooded Crows, but Mr. Howard Saunders brought forward sufficient evidence to show that they had but little claim to the title of separate species. In his Manual we read :—" Many of the specimens obtained in winter have a white bar on the primaries only, the bases of the secondaries being black ; whereas in the typical *L. excubitor* the bases of the secondaries are white, and the wing exhibits a double bar. The form with only one bar is the *L. major* of Pallas, and, as shown by Prof. Collett (Ibis, 1886, pp. 30-40) it meets and interbreeds with *L. excubitor* in Scandinavia, typical examples of both races being actually found in the same brood, while intermediate forms are not uncommon. Where the sexes have been determined, the double-barred bird has generally proved to be a male, and the single-barred a female.

(a)

```
Faiuiiv^LAXIILKl-:.

The Great Grey Shrh^e.

Lauius txciibilor, LiNN.

ORNITHOLOGISTS differ in opinion as to whether this biixl is distinct from
Pallas's Grey Shrike (with the single white bar on the wing) : Seebohm con-
sidered the two forms as distinct as the Carrion and Hooded Crows, but
Mr. Howard Saunders brought forward sufficient evidence to show that they had
but little claim to the title of separate species. In his Manual we read : - " Many
of the specimens obtained in winter have a white bar on the primaries only, the
bases of tlie secondaries being black ; whereas in the typical L. cxcubitor the
bases of the secondaries are white, and the wing exhibits a double bar. The form
with only one bar is the L. viajor, of Pallas, and, as shown by Prof. Collett (Ibis,
1886, pp. 30-40) it meets and interbreeds with L. exaibiior in Scandinavia, t3'pical
examples of both races being actually found in the same brood, while intermediate
forms are not uncommon. Where the sexes have been determined, the double-
barred bird has generally proved to be a male, and the single-barred a female.
```

(b)

Figure 5.1: A image segment (a) and its according OCR-generated text (b) of the evaluation dataset. The recognition errors are highlighted in red.

Table 5.4: Overview of applied evaluation measures and their according feature families

Family	Feature
LCS Distance	
Levenshtein Distance	
Optimal String Alignment	
LCS Similarity	
Jaro-Winkler	
n-gram Jaccard Coefficient	size = {2, 3, 4, 5}
q-Grams Distance	size = {2, 3, 4, 5}
n-Grams Distance	size = {2, 3, 4, 5} type = {binary, positional, comprehensive}
Language popularity	
Lexicon existence	lexicon = {Wikipedia, Term, Biodiversity}
Context Coherence	size = {2, 3, 4, 5}
Approximate Context Matching	size = {2, 3, 4, 5}

filtering. For each candidate being suggested to a potential error word, the proposed model computes all feature scores and searches contexts in a web-scale n-gram corpus. Although efficiency optimization techniques (Mei et al., 2015; Kou et al., 2015) have been adopted to accelerate the computational procedure, the computation is still computationally expensive and thus requiring necessary trade-offs between accuracy and efficiency. The accuracy of the system will increase with more relaxed filtering conditions on English words, for example, filtering only English stop words or even no filtering, but the computation time increases as the trade-off. Similarly for reducing the candidate detection time in Eq. 4.1, we set the maximum Levenshtein distance δ for candidate search to be 3.

Table 5.5: Confusion matrix for error detection

		Model Detection		
		Error	**Correct**	**total**
Ground Truth Correctness	**Error**	2457 (TN)	241 (FP)	2698
	Correct	1273 (FN)	80523 (TP)	81794
	total	3730	80764	

5.2 Detection Evaluation

5.2.1 Detection Recall

We evaluate error detection as a recall oriented task, which focus on finding all possible errors. In all error correction techniques, an undetected error will not get into the correction phase.

We report the confusion matrix for error detection in Table 5.5. The proposed model achieves 91.07% detection recall. There are considerable number of ture-positive errors, which are correct words but detected as errors. When using this type of errors for training or testing, We use the word itself as the intended word for each error. The correction results regarding to all types of errors are reported in Section 5.3.

5.2.2 Word Boundary Detection Recall

For tokenizing the noisy text, any tokenization approach is inevitably involved in the common word boundary problem (Kukich, 1992b), the correct boundary of the errors are not properly identified, in both human-generated (Kukich, 1992a) and OCR-generated text (Jones et al., 1991). Such problem can be caused by the splitting (e.g., <*spend*→*sp end*>) and merging (e.g., <*in form*→*infrom*>) mistakes. It

Table 5.6: The number of detected errors and recall of bounded and unbounded detections

Detection Category	Number	Recall
Bounded	1995	73.94%
Unbounded	462	17.12%
Total (True-Positive)	2457	91.07%

is especially problematic in OCR-generated text, where words containing characters are recognized as punctuation and are thus splitted by the tokenization heuristics. Most error detection and correction techniques define token as character sequence separated by white space characters (e.g., blanks, tabs, carriage returns, etc.) (Kukich, 1992b), which do not split the error token by punctuations. However, this approach cannot distinguish between true punctuation and misrecognized trailing punctuation (e.g., $<family \rightarrow famil\} \hat{}>$).

An error may be "partially" detected if an overlapped but non-identical character sequence is treated as an error. For example, if an error $<spend \rightarrow sp\ end>$ is detected as *end*, *sp* will exists in the context and candidate edit distance will be computed with *end* instead of *sp end*. We call this "partially" detected case as a success *unbounded* detection, where the correct recognition of the character sequence as success *bounded* detection. Unbounded detection can potentially be corrected, but it has inaccurate features scores that will influence the correction accuracy. In addition, there may exist multiple unbounded errors detected for one ground truth error, because of the splitting mistakes. For every ground truth error, We count at most one successful unbounded detection. Our model achieves the 73.51% bounded detection recall and 90.51% total detection recall (i.e., sum of bounded and unbounded detection) shown in Table 5.6.

Table 5.7: The number and the percentage of errors, where correction exists among the top 10 candidates of any applied feature.

Detection Category	Correct Candidates		
	Number	Percentage Among All	Percentage in Search Scope
Bounded	1540	77.19%	84.71%
Unbounded	108	23.38%	47.58%
True-Positive	1648	67.07%	78.66%
False-Positive	1273	100.00%	100.00%
Total	2627	66.15%	78.00%

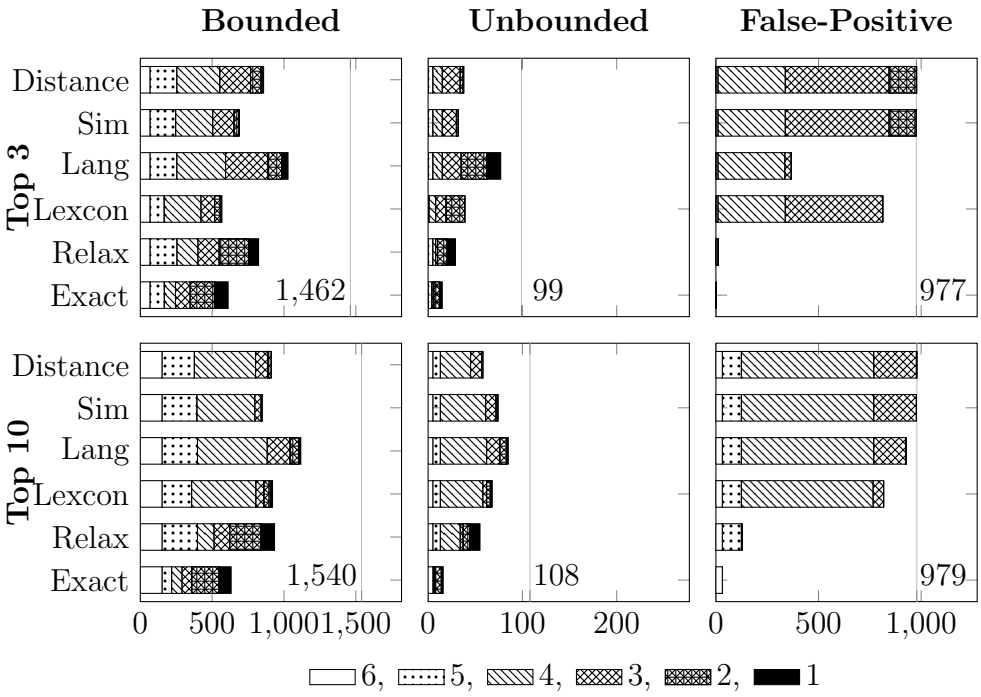


Figure 5.2: The distinctiveness of features in locating error corrections. A bar of a feature represents the number of error corrections located by this feature. The color of a bar indicates the number of features that locates these errors. i.e., white bar indicates a portion of error corrections located by all the features, while black bar indicates error corrections located by only one feature.

5.3 Correction Evaluation

5.3.1 Selected Features Evaluation

We want to study the contribution of features to candidate suggestion. We first explore how well the scoring functions could rank the intended words to the top without predicting by the regressor. For each detected error, we construct a candidate set containing top 3 candidates scored by each feature and check whether this candidate set contains a correction. Note that candidate search scope is limited by the number of edit distance δ (in Eq. 4.1 by default), thus the intended words w_t for w_e cannot be found if $dist_{lev}(\mathrm{w}_t, \mathrm{w}_e) > \delta$. Results are shown in Table 5.7. The model could locate most of the correction in top candidates with the collaboration of all applied features. We observe that performance varies drastically for bounded and unbounded errors, presumably because the feature score for unbounded errors is inaccurate (e.g., the split part of a splitting error is counted as the context word in context search).

To get a better intuition for the contribution of individual feature, we plot the distinctiveness of the located error corrections by each feature in Figure 5.2. For bounded detected errors, context-based features are able to locate some distinctive corrections, which can rarely be found by other features. In addition, top candidates suggested from approximate context matching feature show better coverage than ones from context coherence feature. On the other hand, the other four features are important for false-positive errors, where context-based features provide little help.

5.3.2 Ranking Method Selection

We report candidate ranking performance of different regression models in Table 5.8. The same training and testing dataset, described in Section 5.3, are used for all models.

Given the upperbound of correction rate within edit distance three is 78% (in Table 5.7), all regressors achieve good results. As can be seen, ensemble methods, like

Table 5.8: The percentage of errors, where correction exists among top 1, 3, 5, and 10 candidates suggested by Support Vector regressor (SVR), Rigid Linear regressor (RL), Multiple Layer Perceptron with rectified linear unit (MLP.ReLU), Random Forest (RF), Randoized Decision Trees (RDT), and AdaBoost.R2

Ranking Model	All Errors			
	P@1	P@3	P@5	P@10
RL	0.5637	0.6817	0.7020	0.7313
SVR	0.6060	0.6060	0.7113	0.7230
MLP.ReLU + BFGS	0.6095	0.7025	0.7283	0.7604
AdaBoost.R2 + DT	0.6125	0.6125	0.7113	0.7347
RF	0.6827	0.6827	0.7378	0.7662
RDT	0.7126	0.7126	0.8023	0.8635

Random Forest and AdaBoost, are more robust than others in suggesting appropriate candidates.

5.3.3 Error Correction Model Comparsion

We compares the proposed correction approach with various baseline correction models in Table 5.9. Aspell[8] is an open-source spell checker preinstalled in Linux distributions, which has different modes (ultra, fast, normal, bad-spellers) in seeking different balance points between runing speed and candidate coverage. We use Aspell version 0.60.7 with updated dictionaries from SCOWL[9] Version 2017.01.22 (Spell Checker Oriented Word Lists). Although Aspell contains a comprehensive dictionary, it suggests candidates only within a very limited Damerau-Levenshtein distance, which is suitable for correcting spelling errors. Segaran and Hammerbacher (2009) is a noisy channel correction model and Kissos and Dershowitz (2016) is one of the most recent developed correction model using confidence analysis framework. The same training and testing dataset, described in Section 5.3, are used for all these models. To make a fair comparison, the same tokenization and candidate sets are given to all models as

[8] http://aspell.net/
[9] http://wordlist.aspell.net/

Table 5.9: Correction models comparsion

Error Correction Model	All Errors			
	P@1	P@3	P@5	P@10
Aspell(bad-spellers)	0.2364	0.2364	0.2364	0.2364
Aspell(normal)	0.2403	0.2403	0.2403	0.2403
Aspell(fast|ultra)	0.2583	0.2583	0.2583	0.2583
Segaran and Hammerbacher (2009)	0.5581	0.5581	0.5581	0.5581
Kissos and Dershowitz (2016)	0.6071	0.7145	0.7378	0.7516
proposed approach (RDT)	0.7126	0.7126	0.8023	0.8635

input and models are only responsible for ranking candidates for correction. The results show that the proposed ensemble regression approach significantly outperforms other models.

5.3.4 Overall Performance

We take the following steps to build a training dataset: First, we construct a candidate set for each error containing top 3 candidates scored by each feature. Then, we select a subset of errors, whose intended word exists in the candidate set. Finally, we randomly select 80% errors and use their candidates sets for training.

We train multiple AdaBoost regressors with different settings and apply 10-fold cross-validation to select the best setting for evaluating the rest of the errors. We report the correction results regarding different error categories in Table 5.10. *P@n* represents precision at top n candidate suggestions, which calculate the ratio of the existence of intended words in top n candidates. The proposed model rank the candidates by a regression model and show that more than 71.26% of the errors can be corrected. For 80.23% of the uncorrected errors, our model could provide the correction in the top five suggestions.

Table 5.10: The percentage of errors in the proposed OCR dataset, where correction exists among top 1, 3, 5, and 10 candidates suggested by the proposed model.

Error Categories	P@1	P@3	P@5	P@10
Bounded	0.7369	0.7710	0.8628	0.9405
Unbounded	0.6637	0.6417	0.7220	0.7823
True-Positive	0.7095	0.7025	0.7838	0.8604
False-Positive	0.7971	0.7145	0.8338	0.8942
Total	0.7126	0.7126	0.8023	0.8635

Chapter 6

Conclusion

We introduce a statistical learning model for correcting OCR-generated errors. By integrating different features in a ensemble regression process, the correction model able to select candidates that are similar to the error, suitable for the domain, and coherent to the context. The evaluation results show that the proposed approach can correct 71.3% of the errors and could provide a correction on top three suggestions for 31.2% of the uncorrected errors, which significantly outperforms various baselines. i.e., by suggesting five candidate corrections for each error, our model can correct 80.23% of the error cases in a theoretical correction upper bound of 87.78%.

Bibliography

L Allison and T I Dix. A bit-string longest-common-subsequence algorithm. *Inf. Process. Lett.*, 23(6):305–310, December 1986.

Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3): 403 – 410, 1990.

Zhuowei Bao, Benny Kimelfeld, and Yunyao Li. A graph approach to spelling correction in domain-centric search. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 905–914, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

Daniel Bär, Chris Biemann, Iryna Gurevych, and Torsten Zesch. Ukp: Computing semantic textual similarity by combining multiple content similarity measures. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, SemEval '12, pages 435–440, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

Youssef Bassil and Mohammad Alwani. Context-sensitive spelling correction using google web 1t 5-gram information. *Computer and Information Science*, 5(3):37–48, 2012.

Leonid Boytsov. Indexing methods for approximate dictionary searching: Comparative analysis. *J. Exp. Algorithmics*, 16:1.1:1.1–1.1:1.91, May 2011.

Thorsten Brants and Alex Franz. Web 1t 5-gram corpus version 1 ldc2006t13, 2006. Philadelphia: Linguistic Data Consortium.

Eric Brill and Robert C. Moore. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pages 286–293, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

Arthur G. Butler, William Frohawk, Frederick, and H. Grönvold. *Birds of Great Britain and Ireland. by Arthur G. Butler ; illustrated by H. Grönvold and F.W. Frohawk. Order Passeres*, volume 2. Hull; Brumby & Clarke, 1907.

Hubert Cecotti and Abdel Belayd. Hybrid ocr combination approach complemented by a specialized icr applied on ancient documents. In *Proceedings of the Eighth International Conference on Document Analysis and Recognition*, ICDAR '05, pages 1045–1049, Washington, DC, USA, 2005. IEEE Computer Society.

Kenneth W. Church and William A. Gale. Probability scoring for spelling correction. *Statistics and Computing*, 1(2):93–103, 1991.

Silviu Cucerzan and Eric Brill. Spelling correction as an iterative process that exploits the collective knowledge of web users. 2004.

Fred J. Damerau. A technique for computer detection and correction of spelling errors. *Commun. ACM*, 7(3):171–176, March 1964.

N. Rex Dixon and Thomas B. Martin. *Automatic Speech and Speaker Recognition.* John Wiley & Sons, Inc., New York, NY, USA, 1979.

David Doermann. The indexing and retrieval of document images. *Comput. Vis. Image Underst.*, 70(3):287–298, June 1998.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. *ACM Trans. Inf. Syst.*, 20(1):116–131, January 2002.

Michael Flor. Four types of context for automatic spelling correction.

James C. French, Allison L. Powell, and Eric Schulman. Applications of approximate word matching in information retrieval. In *Proceedings of the Sixth International Conference on Information and Knowledge Management*, CIKM '97, pages 9–15, New York, NY, USA, 1997. ACM.

Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 1606–1611, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

Jianfeng Gao, Xiaolong Li, Daniel Micol, Chris Quirk, and Xu Sun. A large scale ranker-based system for search query spelling correction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 358–366, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

G. H. Gonnet and R. Baeza-Yates. *Handbook of Algorithms and Data Structures: In Pascal and C (2Nd Ed.).* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1991.

Dan Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge University Press, New York, NY, USA, 1997.

R. W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160, April 1950.

A. Islam and D. Inkpen. Real-word spelling correction using google web 1t n-gram with backoff. In *Natural Language Processing and Knowledge Engineering, 2009. NLP-KE 2009. International Conference on*, pages 1–8, Sept 2009a.

Aminul Islam and Diana Inkpen. Second order co-occurrence pmi for determining the semantic similarity of words. In *Proceedings of the International Conference on Language Resources and Evaluation*, Genoa, Italy, 2006. Citeseer.

Aminul Islam and Diana Inkpen. Semantic text similarity using corpus-based word similarity and string similarity. *ACM Trans. Knowl. Discov. Data*, 2(2):10:1–10:25, July 2008.

Aminul Islam and Diana Inkpen. Real-word spelling correction using google web 1tn-gram data set. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 1689–1692, New York, NY, USA, 2009b. ACM.

Aminul Islam and Diana Inkpen. 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, 2009c. Association for Computational Linguistics.

Aminul Islam and Diana Inkpen. *Advances in Artificial Intelligence: 24th Canadian Conference on Artificial Intelligence, Canadian AI 2011, St. John's, Canada, May 25-27, 2011. Proceedings*, chapter Correcting Different Types of Errors in Texts, pages 192–203. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

Aminul Islam, Evangelos Milios, and Vlado Kešelj. Text similarity using google tri-grams. In Leila Kosseim and Diana Inkpen, editors, *Advances in Artificial Intelligence: 25th Canadian Conference on Artificial Intelligence, Canadian AI 2012, Toronto, ON, Canada, May 28-30, 2012. Proceedings*, pages 312–317, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

P. Jaccard. *Nouvelles recherches sur la distribution florale*. Bulletin de la Société vaudoise des sciences naturelles. Impr. Réunies, 1908.

Mario Jarmasz and Stan Szpakowicz. Roget's thesaurus and semantic similarity. *CoRR*, abs/1204.0245, 2012.

Matthew A. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.

Matthew A. Jaro. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. pages 354–359, 1990.

Thorsten Joachims. Advances in kernel methods. chapter Making Large-scale Support Vector Machine Learning Practical, pages 169–184. MIT Press, Cambridge, MA, USA, 1999.

M. A. Jones, G. A. Story, and B. W. Ballard. Interating multiple knowledge sources in a bayesian ocr post-processor. *International Journal on Document Analysis and Recognition*, pages 925–933, 1991.

P Juszczak, D Tax, and Robert PW Duin. Feature scaling in support vector data description. In *Proceedings of the 8th Annual Conf. of the Advanced School for Computing and Imaging*, pages 95–102. Citeseer, 2002.

Mark D. Kernighan, Kenneth W. Church, and William A. Gale. A spelling correction program based on a noisy channel model. In *Proceedings of the 13th Conference on Computational Linguistics - Volume 2*, COLING '90, pages 205–210, Stroudsburg, PA, USA, 1990. Association for Computational Linguistics.

I. Kissos and N. Dershowitz. Ocr error correction using character correction and feature-based word classification. In *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*, pages 198–203, April 2016.

Shmuel T Klein, M Ben-Nissan, and M Kopel. A voting system for automatic ocr correction. August 2002.

Grzegorz Kondrak. N-gram similarity and distance. In *Proceedings of the 12th International Conference on String Processing and Information Retrieval*, SPIRE'05, pages 115–126, Berlin, Heidelberg, 2005. Springer-Verlag.

X. Kou, J. Mei, Z. Yao, A. Rau-Chaplin, A. Islam, A. Moh'd, and E. Milios. Efficient parallelization of the google trigram method for document relatedness computation. In *2015 44th International Conference on Parallel Processing Workshops*, pages 98–104, Sept 2015.

Karen Kukich. Spelling correction for the telecommunications network for the deaf. *Commun. ACM*, 35(5):80–90, May 1992a.

Karen Kukich. Techniques for automatically correcting words in text. *ACM Comput. Surv.*, 24(4):377–439, December 1992b.

Ludmila I. Kuncheva and Christopher J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Mach. Learn.*, 51 (2):181–207, May 2003.

V. Levenshtein. Binary codes capable of correcting spurious insertions and deletions of ones. *Problems of Information Transmission*, 1:8–17, 1965.

V. I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, February 1966.

W. B. Lund, D. D. Walker, and E. K. Ringger. Progressive alignment and discriminative error correction for multiple ocr engines. In *2011 International Conference on Document Analysis and Recognition*, pages 764–768, Sept 2011.

William B. Lund and Eric K. Ringger. Improving optical character recognition through efficient multiple system alignment. In *Proceedings of the 9th ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '09, pages 231–240, New York, NY, USA, 2009. ACM.

William B. Lund, Douglas J. Kennard, and Eric K. Ringger. Combining multiple thresholding binarization values to improve ocr output. In *Proc. SPIE*, volume 8658, pages 86580R–86580R–11, 2013a.

William B. Lund, Eric K. Ringger, and Daniel D. Walker. How well does multiple ocr error correction generalize? In *Proc. SPIE*, volume 9021, pages 90210A–90210A–13, 2013b.

Eric Mays, Fred J. Damerau, and Robert L. Mercer. Context based spelling correction. *Inf. Process. Manage.*, 27(5):517–522, September 1991.

Jie Mei, Xinxin Kou, Zhimin Yao, Andrew Rau-Chaplin, Aminul Islam, Abidalrahman Moh'd, and Evangelos E. Milios. Efficient computation of co-occurrence based word relatedness. In *Proceedings of the 2015 ACM Symposium on Document Engineering*, DocEng '15, pages 43–46, New York, NY, USA, 2015. ACM.

Jie Mei, Aminul Islam, and Evangelos Milios. Dalgtm at semeval-2016 task 1: Importance-aware compositional approach to short text similarity. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 765–770, San Diego, California, June 2016. Association for Computational Linguistics.

Roger Mitton. Spelling checkers,spelling correctors and the misspellings of poor spellers. *Inf. Process. Manage.*, 23(5):495–505, September 1987.

Ismail Bin Mohamad and Dauda Usman. Standardization and its effects on k-means clustering algorithm. *Research Journal of Applied Sciences, Engineering and Technology*, 6(17):3299–3303, 2013.

Günter Mühlberger, Johannes Zelger, and David Sagmeister. User-driven correction of ocr errors: Combining crowdsourcing and information retrieval technology. In *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage*, DATeCH '14, pages 53–56, New York, NY, USA, 2014. ACM.

Eugene W Myers. An overview of sequence comparison algorithms in molecular biology. Technical report, Max-Planck Institute for Molecular Cell Biology & Genetics, 1991.

Eugene W. Myers. Algorithmic advances for searching biosequence databases. In Sándor Suhai, editor, *Computational Methods in Genome Research*, pages 121–135. Springer US, Boston, MA, 1994.

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

Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443 – 453, 1970.

John C Nesbit. The accuracy of approximate string matching algorithms. *J. Comput. Based Instruct.*, 13(3):80–83, August 1986.

Konstantinos Ntirogiannis, Basilios Gatos, and Ioannis Pratikakis. Performance evaluation methodology for historical document image binarization. *IEEE Trans. Image Processing*, 22(2):595–609, 2013.

O. Owolabi and D. R. McGregor. Fast approximate string matching. *Softw. Pract. Exper.*, 18(4):387–393, April 1988.

Tommi Pirinen, Krister Lindén, et al. Finite-state spell-checking with weighted language and error models. In *Proceedings of LREC 2010 Workshop on creation and use of basic lexical resources for less-resourced languages*, 2010.

Tommi Pirinen, Miikka Silfverberg, Krister Linden, et al. Improving finite-state spell-checker suggestions with part of speech n-grams. In *Computational Linguistics and Intelligent Text Processing 13th International Conference, CICLing 2012*, 2012.

Joseph J. Pollock and Antonio Zamora. Automatic spelling correction in scientific and scholarly text. *Commun. ACM*, 27(4):358–368, April 1984.

Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

D. Sankoff and J.B. Kruskal. *Time warps, string edits, and macromolecules: the theory and practice of sequence comparison*. Addison-Wesley Pub. Co., Advanced Book Program, 1983.

David Sankoff. Matching sequences under deletion/insertion constraints. *Proceedings of the National Academy of Sciences*, 69(1):4–6, 1972.

Paulo J. Santos, Amy J. Baltzer, Albert N. Badre, Richard L. Henneman, and Michael S. Miller. On handwriting recognition system performance: Some experimental results. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 36(4):283–287, 1992.

Johannes Schaback and Fang Li. Multi-level feature extraction for spelling correction. In *IJCAI-2007 Workshop on Analytics for Noisy Unstructured Text Data*, pages 79–86, 2007.

T. Segaran and J. Hammerbacher. *Beautiful Data: The Stories Behind Elegant Data Solutions*. Theory in practice. O'Reilly Media, 2009.

Peter H Sellers. The theory and computation of evolutionary distances: Pattern recognition. *Journal of Algorithms*, 1(4):359 – 373, 1980.

C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, July 1948a.

C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(4):623–656, Oct 1948b.

Peter Sollich and Anders Krogh. Learning with ensembles: How overfitting can be useful. *Advances in neural information processing systems*, pages 190–196, 1996.

Michael Strube and Simone Paolo Ponzetto. Wikirelate! computing semantic relatedness using wikipedia. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, pages 1419–1424. AAAI Press, 2006.

Xu Sun, Jianfeng Gao, Daniel Micol, and Chris Quirk. Learning phrase-based spelling error models from clickthrough data. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 266–274, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

Kazem Taghva and Eric Stofsky. Ocrspell: an interactive spelling correction system for ocr errors in text. *International Journal on Document Analysis and Recognition*, 3(3):125–137, 2001.

Kazem Taghva, Julie Borsack, and Allen Condit. Expert system for automatically correcting ocr output. volume 2181, pages 270–278, 1994.

Esko Ukkonen. Approximate string-matching with q-grams and maximal matches. *Theoretical Computer Science*, 92(1):191 – 211, 1992.

T. K. Vintsyuk. Speech discrimination by dynamic programming. *Cybernetics*, 4(1): 52–58, 1968.

Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *J. ACM*, 21(1):168–173, January 1974.

Donald E Walker and Robert A Amsler. The use of machine-readable dictionaries in sublanguage analysis. *Analyzing Language in Restricted Domains: Sublanguage Description and Processing*, page 69, 2014.

M.S. Waterman. *Introduction to Computational Biology: Maps, Sequences and Genomes*. Chapman & Hall/CRC Interdisciplinary Statistics. Taylor & Francis, 1995.

Casey Whitelaw, Ben Hutchinson, Grace Y. Chung, and Gerard 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*, EMNLP '09, pages 890–899, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

Alan M Wing and Alan Baddeley. *Spelling errors in handwriting: a corpus and a distributional analysis.*, pages 251–285. Academic Press, 1980.

Tieng K. Yap, Ophir Frieder, and Robert L. Martino. *High Performance Computational Methods for Biological Sequence Analysis*. Kluwer Academic Publishers, Norwell, MA, USA, 1st edition, 1996.

Justin Zobel and Philip Dart. Phonetic string matching: Lessons from information retrieval. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '96, pages 166–172, New York, NY, USA, 1996. ACM.

Justin Zobel and Alistair Moffat. Exploring the similarity space. *SIGIR Forum*, 32 (1):18–34, April 1998.

Appendix A

Theories and Definitions

A.1 Edit Distance

Definition A.1 (elementary edit operation)**.** *An elementary edit operation* (x, y) *that converts* x *to* y *is an ordered pair of characters* $\forall x, y \in \Sigma^+, (x, y) \neq (\Lambda, \Lambda)$.

Remark 1 (insertion, deletion, and substitution operation). Given definition A.1, we can directly formalize three types of elementary edit operations:

- *insertion* $\{(x, y)|x = \Lambda\}$,
- *deletion* $\{(x, y)|y = \Lambda\}$.
- *substitution* $\{(x, y)|x \neq \Lambda, y \neq \Lambda\}$,

Remark 2 (transposition operation). The *transposition* operation, first introduced by Damerau (1964) and used in many edit distance metrics, that exchanges two adjacent characters is not an elementary edit operation. Transposition operation is not an elementary, i.e. atomic, edit operation. We define transposition as an composed edit operation that consists of elementary operations that swap two adjacent characters, for example, two substitution operations.

Definition A.2 (edit transformation)**.** *An edit transformation* E_{S_1,S_2} *that converts* S_1 *to* S_2 *is an sequence of edit operations*

$$E_{S_1,S_2} = \{(x_i, y_i)\}_m. \tag{A.1}$$

There are infinite number of transformations available between two strings. We denote $\mathcal{E}_{S_1,S_2}$ as the set of all edit transformations that make a valid transposition from S_1 to S_2.

Table A.1: List of applied edit distance measures and according edit operation sets.

Distance Metrics	Edit Operation Set
Hamming Distance	substitution
Longest Common Subsequence Distance	insertion, deletion
Levenshtein Distance	subsubsection, insertion, deletion
Damerau-Levenshtein Distance	subsubsection, insertion, deletion, transposition

Definition A.3 (transformation cost)**.** *Let* $\gamma : \Sigma^+ \times \Sigma^+ \to \mathbb{R}$ *be a cost function that transform an edit operation into a nonnegative real number. The cost of an edit transformation,* γ_E*, is the cumulative sum of all edit operations in this transformation*

$$\gamma_E(E_{S_1,S_2}) = \sum_{(x_i,y_i)\in E_{S_1,S_2}} \gamma(x_i, y_i). \tag{A.2}$$

Definition A.4 (edit distance)**.** *The edit distance* $\delta(S_1, S_2)$ *is the minimum cost required to transform* S_1 *to* S_2

$$\delta(S_1, S_2) = \min_{E_{S_1,S_2}\in\mathcal{E}_{S_1,S_2}} \gamma_E(E_{S_1,S_2}). \tag{A.3}$$

Remark 3 (unit-cost edit distance)*.* Many applications simplify the definition of edit distance as the minimum number of edit operations required for a transformation between two strings. Edit distance metrics follow this simplification is called unit-cost edit distances in the literature. Without loss of generality, we can achieve this simplification by adopting $\gamma \equiv 1$ for all edit operations.

For feature functions of this family, we apply the *unit-cost* edit distances and rescale using Eqn. 4.3. Different distance measures allow only a subset of edit operations to be used, which are listed in Table A.1.

Appendix B

Evaluation Dataset Preprocessing

The dataset we used in evaluation are generated from two OCR outputs for book "Birds of Great Britain and Ireland (Volumn II)" (Butler et al., 1907). One version is generated from the standard BHL-Europe recognition workflow, which OCR technique is based on Tesseract 3.0.2[1]. Another version is generated primarily in the Abbyy Fine Reader 10 workflow with improvements developed within the IMPACT project[2]. We parsed the page separated content from the former version as the input text for evaluating OCR post-processing task and modified the latter version to get the ground truth content. For analysis, we also generated a list of ground truth error by comparing the input and the ground truth content.

Given that the ground truth is also generated from the OCR workflow, the errors need to be corrected to match the original book content. In addition, the content lines in the ground truth text should be aligned with the input text. We adopted the following rules in generating the ground truth text:

- The footnotes and page numbers are removed. Consider inferences using contextual words are used in many correction models, the purpose of removing those texts is to keep the content fluency.

- Characters with multiple representations are standardized. There are different unicode representations indicating the same character. For example, `U+00B0` and `U+02DA` both look similar to the degree sign. It is hard to identify which one is the correct recognition result to the original image. In order to reduce the confusion in further error identification and correction, we selected one of

[1] https://github.com/tesseract-ocr/tesseract
[2] http://www.bhle.eu/en/results-of-the-collaboration-of-bhl-europe-and-impact

Table B.1: Character substitutions in preprocessing the ground truth.

Type	Standard representation	Substituted representations
Latin ligature	ff	U+FB00
	fi	U+FB01
	fl	U+FB02
	ffi	U+FB03
	ffl	U+FB04
punctuation	-	U+2010, U+2014, U+2015
	’	U+2018, U+2019
	"	U+201C, U+201D
special modifier	° (U+00B0)	U+00B0, U+02DA

the representations as the substitution for all other representations. Table B.1 lists all the substitutions in preprocessing the ground truth.

To generate the error list, we adopted the following rules in identifying the errors in aligned contents from input and the ground truth:

– The punctuations in the ground truth text are separate tokens in error identification. For example, an aligned string pair “fFrinˆHluurJ” in input text aligns with “(Fringillinæ)” in the ground truth text. According to this rule, this aligned pair is separated into three errors: $<$ f $\rightarrow$) $>$, $<$ FrinˆHluurJ $\rightarrow$ Fringillinæ $>$, and $<$ J $\rightarrow$) $>$. The above example shows the simplification that this rule may involve: when segmenting an OCR-generated string into substrings that match with tokens in the ground truth text, the seperating positions are approximated manually to make the best guess. In another example, where we have two aligned strings “countr}ˆ” and “country,”, the according

Figure B.1: The comparison between U+00B0 and U+02DA.

errors are < countr}ˆ → country > and < → , >.

– The aligned two characters are different representations of the same character is not treated as an error.

– Two ASCII subsititution of unicode characters are allowed: (æ, ae) and (Æ, AE). Note that the dataset is generated from a biodiversity book, which contains terminologies with non-English characters, for example, *Corvidæ* or *ORIOLIDÆ*. We accept these two ASCII subsititutions in order to match the original terminologies to their English counterparts.

– The aligned two words with different cases is not treated as an error. Observed that the standard BHL-Europe recognition workflow is tend to lowercase the non-heading characters in some entirely capitalized words. Thus, we do not categorized this type of mismatchings as error. Such change in capitalization form is also hard to detect by human readers with only input text, where page layout is eliminated.

– The extra whitespaces between tokens are allowed. It is also observed that the standard BHL-Europe recognition workflow sometimes generate extra whitespaces between tokens. We do not categorize this type of mismatch as error unless the inserted whichspace leads to a splitting or merging error.