Integrated Scoring for Spelling Error Correction, Abbreviation Expansion and Case Restoration in Dirty Text

Wilson Wong, Wei Liu and Mohammed Bennamoun

School of Computer Science and Software Engineering
University of Western Australia
Crawley WA 6009
Email: {wilson,wei,bennamou}@csse.uwa.edu.au

Abstract

An increasing number of language and speech applications are gearing towards the use of texts from online sources as input. Despite such rise, not much work can be found in the aspect of integrated approaches for cleaning dirty texts from online sources. This paper presents a mechanism of **I**ntegrated **S**coring for **S**pelling error correction, **A**bbreviation expansion and **C**ase restoration (ISSAC). The idea of ISSAC was first conceived as part of the text preprocessing phase in an ontology engineering project. Evaluations of ISSAC using 400 chat records reveal an improved accuracy of 96.5% over the existing 74.4% based on the use of Aspell only.

Keywords: Spelling error correction, abbreviation expansion, case restoration, dirty text, text preprocessing, text cleaning

1 Introduction

The tasks of correcting spelling errors, expanding abbreviations and restoring cases are essential to many language and speech applications such as text and data mining (Castellanos 2003, Tang, Li, Cao & Tang 2005), and automatic or semi-automatic ontology engineering (Maedche & Volz 2001, Xu, Kurz, Piskorski & Schmeier 2002, Degeratu & Hatzivassiloglou 2002, Novacek & Smrz 2005*b*). These tasks are typically performed as part of the text preprocessing phase in these applications, and are usually known by other names such as *text cleaning* and *text normalization*. Despite the importance of these cleaning tasks, the existing applications have been relying on ad-hoc techniques (e.g. pattern matching, hand-crafted rules) designed to serve individual needs when problems arise.

Text preprocessing, especially spelling error correction, abbreviation expansion and case restoration, is beginning to attract the attention of language and speech applications as they gear towards using online sources for textual input. Examples include corporate intranet (Kietz, Volz & Maedche 2000) and documents retrieved by search engines (Cimiano & Staab 2005, Sanchez & Moreno 2005, Sombatsrisomboon, Matsuo & Ishizuka 2003, Turney 2001) for automatic or semi-automatic ontology engineering, and chat records (Castellanos 2003) and emails (Tang et al. 2005) for text and data mining. The quality of texts from such sources, in particular blogs, emails and chat records can be extremely poor. The constructions of sentences in blogs, emails and chat records are filled with spelling errors, ad-hoc abbreviations and improper casing. As the different quality of texts will pose different requirements during the preprocessing phase, dirty texts can be very demanding. With the prevalence of online sources, this *"...annoying phase of text cleaning..."*(Mikheev 2002) becomes much more important and relevant than ever before. Accordingly, an increasing number of researchers, particularly in the field of ontology engineering (Maedche & Volz 2001, Novacek & Smrz 2005*b*, Novacek & Smrz 2005*a*), has began to acknowledge the impact of the cleanliness of input on the quality of ontology. In a text mining research, Tang et al. (Tang et al. 2005) reported an improved terms extraction accuracy of 38-45% (based on F1-measure) after the input (i.e. emails) has been cleaned.

In this paper, we propose a mechanism for **I**ntegrated **S**coring for **S**pelling error correction, **A**bbreviation expansion and **C**ase restoration (ISSAC). ISSAC is built on top of the spell checker Aspell (Atkinson 2006) for automatically correcting spelling errors, expanding ad-hoc abbreviations and restoring case in dirty texts (e.g. chat records). ISSAC makes use of six weights based on different information sources, namely, original rank by Aspell, reuse factor, abbreviation factor, normalized edit distance, domain significance and general significance. Evaluations performed on four different set of chat records yield an average of 96.5% accuracy in the automatic correction of spelling errors, expansion of ad-hoc abbreviations and restoration of casing.

2 Background and Related Work

Spelling error detection and correction is the task of recognizing misspellings in texts and providing suggestions for correcting the errors. For example, detecting *"cta"* as an error and suggesting that the error be replaced with *"cat"*, *"act"* or *"tac"*. Hence, it is obvious that suggestions can only be made after detecting the errors, and more information is usually required to perform a correct replacement. There are four basic types of single-error misspelling (Chan, He & Ounis 2005, Damerau 1964), namely, insertion (e.g *"receivee"* with an extra *'e'*), deletion (e.g. *"receiv"* with the missing *'e'*), substitution (e.g. *"receiva"* where *'a'* is supposed to be *'e'*) and transposition (e.g. *"recieve"* where *'i'* and *'e'* switched). The task of abbreviation expansion deals with recognizing shorter forms of words (e.g. *"abbr."* or *"abbrev."*), acronyms (e.g. *"NATO"*) and initialisms (e.g. *"HTML"*, *"FBI"*), and expanding them to their corresponding words[1]. The many-to-many relation-

[1]Some researchers refer to this relationship as *abbreviation and definition* or *short-form and long-form*

ship between abbreviations and their corresponding expansions makes this task equally difficult. For case restoration, improper casing in words are detected and restored. For example, detecting the letter *'j'* in *"jones"* as improper and correcting the word to produce *"Jones"*.

Most of the work in detection and correction of spelling errors, and expansion of abbreviations are carried out separately. The single-error misspellings, together with other more complex errors (Kukich 1992), have given rise to and shaped a wide range of techniques since 1960s. Two of the most studied classes of techniques for detecting and correcting spelling errors are *minimum edit distance* and *similarity key*. The idea of minimum edit distance techniques began with Damerau (Damerau 1964) and Levensthein (Levenshtein 1966). Damerau-Levenshtein distance is the minimal number of insertions, deletions, substitutions and transpositions needed to transform one string into the other. For example, to change *"wear"* to *"beard"* will require a minimum of two operations, namely, a substitution of *'w'* with *'b'*, and an insertion of *'d'*. Many variants were developed subsequently such as the algorithm by Wagner & Fischer (Wagner & Fischer 1974). The second class of techniques is similarity key. The main idea behind similarity key techniques is to map every string into a key such that similarly spelled strings will have identical keys (Kukich 1992). Hence, the key, computed for each spelling error, will act as a pointer to all similarly spelled words (i.e. suggestions) in the dictionary. One of the earliest implementation is the SOUNDEX system by Odell & Russell (Odell & Russell 1918, Odell & Russell 1922). SOUNDEX maps a word into a key consisting of its first letter followed by a sequence of numbers. For every of the remaining letter l, a number is assigned according to the rules:

0 if $l \in \{A, E, I, O, U, H, W, Y\}$

1 if $l \in \{B, F, P, V\}$

2 if $l \in \{C, G, J, K, Q, S, X, Z\}$

3 if $l \in \{D, T\}$

4 if $l \in \{L\}$

5 if $l \in \{M, N\}$

6 if $l \in \{R\}$

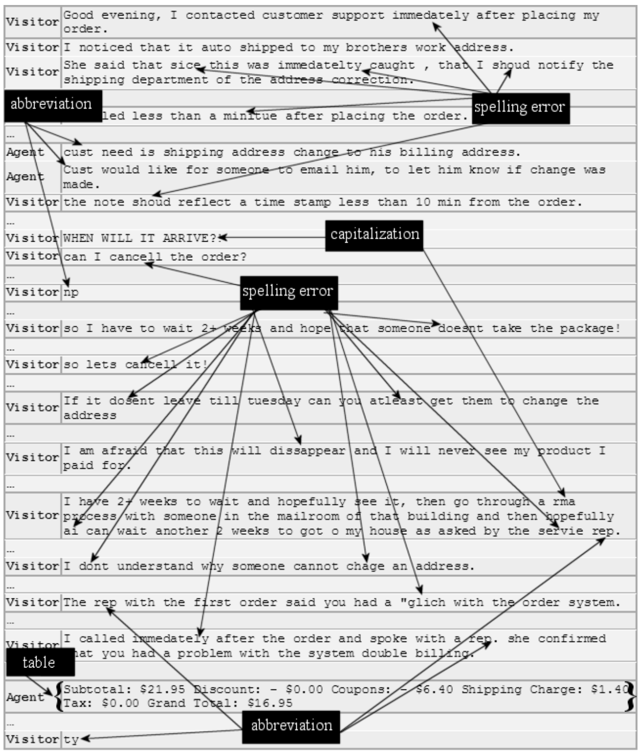
Zeros are eliminated and repeated numbers are collapsed. For example, $wear \rightarrow w006 \rightarrow w6$ and $ware \rightarrow w060 \rightarrow w6$. Since then, many improved variants were developed such as the Metaphone and the Double-metaphone algorithm by Philips (Philips 1990), Daitch-Mokotoff Soundex (Lait & Randell 1993) and others (Holmes & McCabe 2002).

Many work on detecting and expanding abbreviations are conducted in the realm of named-entity recognition and word-sense disambiguation. Due to the intensive use of abbreviations in medical texts, most researches were initiated and tested in the medical domain. Schwartz & Hearst (Schwartz & Hearst 2003) presented a simple two-stage process for extracting abbreviations and their definitions. The first stage involves the extraction of all abbreviations and definition candidates based on the adjacency to parentheses. Stage two identifies the correct definition out of the many candidates for each abbreviation. The identification is based on two criteria: the correct definition must appear in the same sentence, and the correct definition should have no more than $min(|A|+5, |A|*2)$ words where $|A|$ is the number of characters in an abbreviation A. Park & Byrd (Park & Byrd 2001) presented an algorithm based on rules and heuristics for extracting definitions for abbreviations from texts. Some of the rules and heuristics employed for this purpose include syntactic cues, priority of rules, distance between abbreviation and definition, word casing, and the number of words and stopwords in the definition. Pakhomov (Pakhomov 2001) proposed a semi-supervised approach that employs a hand-crafted table of abbreviations and their definitions for training a maximum entropy classifier. Last but not least, Sproat et al. (Sproat, Black, Chen, Kumar, Ostendorf & Richards 2001) have undertaken a project that attempts to address deficiencies in existing aproaches for various aspects in abbreviation expansion. In particular, the project focuses on the difficulty of normalizing text from online sources such as newsgroups.

In the context of ontology engineering and other related areas such as text mining, spelling errors correction, abbreviations expansion and case restoration (Mikheev 2002, Lita, Ittycheriah, Roukos & Kambhatla 2003, Mikheev 1999) are mainly carried out as part of the text preprocessing (i.e. text cleaning, text filtering, text normalization) phase. A review by Wong et al. (Wong, Liu & Bennamoun 2006) shows that many existing systems require the input texts to be clean and hence, the techniques for extracting plain text from various format, correcting spelling errors and expanding abbreviations becomes unnecessary. Ontology engineering approaches such as Xu et al. (Xu et al. 2002), Text-to-Onto (Maedche & Volz 2001), OntoStruct (Degeratu & Hatzivassiloglou 2002) and OLE/BOLE (Novacek & Smrz 2005*b*, Novacek & Smrz 2005*a*) are the few exceptions. In a text mining approach for extracting topics from chat records, Castellanos (Castellanos 2003) presented a very comprehensive list of techniques for text preprocessing. In addition to the common text preprocessing tasks, the approach employs a thesaurus, constructed using the Smith-Waterman algorithm (Smith & Waterman 1981), for correcting spelling errors and identifying abbreviations. Tang et al. (Tang et al. 2005) presented a cascaded approach for cleaning emails prior to any text mining processing. The approach is composed of four passes including non-text filtering, paragraph normalization, sentence normalization, and word normalization.

3 Scoring Mechanism

The idea of ISSAC was initially conceived as part of an ontology engineering project that uses multiple online sources (i.e. media articles and chat records) with varying cleanliness. In particular, the use of chat records as input has required us to place more effort during the text preprocessing phase. Figure 1 highlights the various spelling errors, ad-hoc abbreviations and improper casing that occur very frequently in chat records which are not present in clean text. ISSAC provides an integrated approach for simultaneously correcting spelling errors, expanding abbreviations and restoring cases without expert participation. Along the same line of thought, Clark (Clark 2003) defended that *"...a unified tool is appropriate because of certain specific sorts of errors"*. To illustrate this idea, consider the error word *"cta"*. Do we immediately take it as a spelling error and correct it as *"cat"*, or is it a problem with the letter casing, which makes it a probable acronym? Hence, it is obvious that the problems of spelling error, abbreviation and letter casing are inter-related to a certain extent. ISSAC provides an integrated approach for solving the three problems (i.e. spelling error correction, abbreviation expansion and case restoration) in the following ways:



Visitor	Good evening, I contacted customer support immedately after placing my order.
Visitor	I noticed that it auto shipped to my brothers work address.
Visitor	She said that sice this was immedatelty caught , that I shoud notify the shipping department of the address correction.
	led less than a minitue after placing the order.
...	
Agent	cust need is shipping address change to his billing address.
Agent	Cust would like for someone to email him, to let him know if change was made.
Visitor	the note shoud reflect a time stamp less than 10 min from the order.
...	
Visitor	WHEN WILL IT ARRIVE?!
Visitor	can I cancell the order?
...	
Visitor	np
...	
Visitor	so I have to wait 2+ weeks and hope that someone doesnt take the package!
...	
Visitor	so lets cancell it!
...	
Visitor	If it dosent leave till tuesday can you atleast get them to change the address
...	
Visitor	I am afraid that this will dissappear and I will never see my product I paid for.
...	
Visitor	I have 2+ weeks to wait and hopefully see it, then go through a rma process with someone in the mailroom of that building and then hopefully ai can wait another 2 weeks to got o my house as asked by the servie rep.
...	
Visitor	I dont understand why someone cannot chage an address.
...	
Visitor	The rep with the first order said you had a "glich with the order system.
...	
Visitor	I called immedately after the order and spoke with a rep. she confirmed that you had a problem with the system double billing.
Agent	{Subtotal: \$21.95 Discount: - \$0.00 Coupons: - \$6.40 Shipping Charge: \$1.40} {Tax: \$0.00 Grand Total: \$16.95}
...	
Visitor	ty

Figure 1: Example of spelling errors, ad-hoc abbreviations and improper casing in a chat record

- Spelling error correction: The foundation for correcting spelling errors is the spell checker Aspell (Atkinson 2006). Whenever a word is considered as an error, Aspell will provide a list of suggestions for correction. This list is the key component of ISSAC. The ability of ISSAC to find the correct replacement will be dependent on the quality of the suggestions.

- Abbreviation expansion: The foundation for expanding abbreviations is the online abbreviation dictionary `www.stands4.com`. When the scoring process takes place and the corresponding expansions for potential abbreviations are required, `www.stands4.com` is consulted. A copy of the expansion is stored in a local *abbreviation dictionary* for future reference. The expansions for potential abbreviations will be added to the suggestion list produced by Aspell. Later, the various weights in ISSAC will help in determining if the error (i.e. potential abbreviation) is an actual abbreviation and that replacement should be done using the corresponding expansion.

- Case restoration: There are two ways of restoring cases using ISSAC. The first employs the inherent capability of Aspell to recognize proper nouns without appropriate casing as errors. This will allow such words to be thrown into ISSAC for restoration. The second way is based on the heuristic that valid words rarely appear as acronyms, and those who do will not fit into the surrounding context. For example, consider the phrase with improper casing, *"shipping TIME frame"*. Appearing as an independent word, *"TIME"* has an equally likely chance of being a word (with improper casing) or an acronym for *"Timed Interactive Multimedia Extensions"*[2]. When the neighbouring words *"shipping"* and *"frame"* are taken into considerations, then the probability of *"TIME"* being an acronym becomes significantly less. Based on this idea, words with all uppercase letters are first turned into lowercase and then automatically disambiguated using ISSAC.

[2]Source from `http://www.stands4.com/bs.asp?st=time&SE=1`

Prior to the scoring, each sentence in the input text (e.g. chat record) is tokenized to obtain a list of words $T = \{t_1, ...t_w\}$ which will be fed into Aspell. For each word e that Aspell considers as erroneous, a list of ranked suggestions S is produced. The list S is generated by Aspell based on the Metaphone algorithm (Philips 1990) and near-miss strategy by its predecessor Ispell (Kuenning 2006). Initially, $S = \{s_{1,1}, ..., s_{n,n}\}$ is an ordered list of n suggestions where $s_{j,i}$ is the j^{th} suggestion with rank i (smaller i indicates higher rank). If e appears in the abbreviation dictionary, the list S is augmented by appending all the corresponding m expansions at the front of S as additional suggestions, all with rank 1. In addition, the error word e is appended at the end of S with rank $n+1$. These augmentations result in an extended list $S = \{s_{1,1}, ..., s_{m,1}, s_{m+1,1}, ..., s_{m+n,n}, s_{m+n+1,n+1}\}$, which is a combination of m suggestions from the abbreviation dictionary (if e is a potential abbreviation), n suggestions by Aspell, and the error word e itself. Placing the error word e back into the list of possible replacements serves one purpose: to ensure that if no better replacement is available, we keep the error word e as it is. Once the extended list S is obtained, each suggestion $s_{j,i}$ is re-ranked using ISSAC based on a combination of weights, including the existing rank i. The other weights include the reuse factor RF, abbreviation factor AF, normalized edit distance NED, domain significance DS and general significance GS. The attempt to measure the significance of suggestions also takes into account the neighbouring words l (i.e. word to the left) and r (i.e. word to the right) of error e. This is based on the assumption that *"...errors are isolated and surrounded by clean context that can be used to correct the errors"* (Clark 2003). The new score for each suggestion $s_{j,i}$ is defined as

$$\begin{aligned} NS(s_{j,i}) &= i^{-1} + NED(e, s_{j,i}) \\ &\quad + RF(e, s_{j,i}) + AF(s_{j,i}) \\ &\quad + DS(l, s_{j,i}, r) + GS(l, s_{j,i}, r) \end{aligned}$$

The different weights are defined in the following subsections.

3.1 Normalized Edit Distance

$NED(e, s_{j,i}) \in (0, 1]$ is the *normalized edit distance* obtained by normalizing the value of the minimum edit distance between error e and suggestion $s_{j,i}$. The normalized edit distance is defined as

$$NED(e, s_{j,i}) = \frac{1}{ED(e, s_{j,i}) + 1}$$

The minimum edit distance ED between two words is obtained using the Wagner-Fischer algorithm (Wagner & Fischer 1974). Wagner-Fischer distance is preferred over other metrics because it covers multi-error misspellings (Kukich 1992) and also accounts for transpositions in addition to insertions, deletions and substitutions. NED provides higher importance to suggestions that are lexically similar to the error. During the evaluation, this weight has shown to be useful especially for maintaining proper nouns that

Aspell considers as errors. For example, Aspell mistakenly identified *"Carrollton"*[3] as an error and suggested the replacement *"Carillon"*. Similarly, Aspell suggested that *"iPod"* be replaced with *"pod"*.

3.2 Reuse Factor and Abbreviation Factor

$RF(e, s_{j,i}) \in \{0,1\}$ is the boolean *reuse factor* for providing more weight to suggestion $s_{j,i}$ that has been previously used for correcting error e. This weight is to ensure consistency for correcting similar errors. Certain spelling errors in chat records are repetitive in nature. For example, the word *"received"* and *"receive"* are often misspelled as *"recieved"* and *"recieve"* respectively. Out of the 2016 spellings errors in an evaluation of 400 chat records, there are about 40 occurrences of *"recieve"* and its variants. The reuse factor is obtained through a lookup into a *history list* that consists of previous corrections. $RF(e, s_{j,i})$ will provide factor 1 if the error e has been previously corrected with $s_{j,i}$ and 0 otherwise.

$AF(s_{j,i}) \in \{0,1\}$ is the *abbreviation factor* for denoting that $s_{j,i}$ is a potential abbreviation. A lookup into the *abbreviation dictionary*, $AF(s_{j,i})$ will yield factor 1 if suggestion $s_{j,i}$ is found to exists in the dictionary and 0 otherwise. The abbreviation dictionary is automatically updated on demand using `www.stands4.com`.

3.3 Domain Significance

$DS(l, s_{j,i}, r) \in [0,1]$ measures the *domain significance* of suggestion $s_{j,i}$ based on its appearance in the domain corpora. The domain significance weight is inspired by the *TF-IDF* (Robertson 2004) measure which is commonly used in Information Retrieval. In addition to individual occurrence, the frequency of appearance of $s_{j,i}$ together with its neighbouring words l and r is also taken into consideration. The weight is defined as

$$DS(l, s_{j,i}, r) = \frac{f_{s_{j,i}} + f_{ls_{j,i}r}}{\sum_{k=1}^{m+n+1}(f_{s_{k,i}} + f_{ls_{k,i}r})} e^{-\frac{ND_{s_{j,i}}}{ND}}$$

where $f_{s_{j,i}}$ is the frequency of occurrence of suggestion $s_{j,i}$ in the domain corpora and $f_{ls_{j,i}r}$ is the frequency of occurrence of suggestion $s_{j,i}$ together with neighbours l and r in the domain corpora. $\sum_{k=1}^{m+n+1}(f_{s_{k,i}} + f_{ls_{k,i}r})$ is the sum of the frequencies of occurrences of all individual suggestions, and of the frequencies of occurrences of all suggestions together with neighbours l and r in the domain corpora. ND is the total number of documents in the domain corpora and $ND_{s_{j,i}}$ is the number of documents in the domain corpora that contain suggestion $s_{j,i}$. The significance of $s_{j,i}$ will increase proportionally to the number of times the suggestion appears in the domain corpora. Raising e to the power of $-\frac{ND_{s_{j,i}}}{ND}$ has similar effect as the traditional *IDF* (Robertson 2004), namely, as an offset to suggestions that occur too frequently.

3.4 General Significance

$GS(l, s_{j,i}, r) \in [0,1]$ measures the *general significance* of suggestion $s_{j,i}$ based on its appearance in the general collection (e.g. Goggle). The purpose of general significance is similar to that of the domain significance. The weight is defined as

$$GS(l, s_{j,i}, r) = \frac{NG_{ls_{j,i}r}}{\sum_{k=1}^{m+n+1} NG_{ls_{k,i}r}} e^{-\frac{NG_{s_{j,i}}}{\sum_{k=1}^{m+n+1} NG_{s_{k,i}}}}$$

where $NG_{ls_{j,i}r}$ is the number of documents in the general collection that contains suggestion $s_{j,i}$ within the neighbours l and r, and $NG_{s_{j,i}}$ is the number of documents in the general collection that contains suggestion $s_{j,i}$ alone. This weight is especially useful for two reasons. Firstly, this weight will provide due consideration for the use of proper names such as *"iPod"* and *"XBox"* that are not part of Aspell's dictionary. Secondly, the contemporary use of English in areas like business and computing has given rise to various entirely new words that results from combinations of existing ones. General collection of documents such as `www.google.com` is the best candidate when considering the spelling for new words.

4 Evaluation and Results

Evaluations are conducted using chat records provided by `247Customer.com`[4]. As a provider of customer lifecycle management services, the chat records by `247Customer.com` offer a rich source of domain information in a natural setting (i.e. conversations between customers and agents). Consequently, these chat records are filled with spelling errors, ad-hoc abbreviations, improper casing and many other problems that are considered as intolerable by many of the existing language and speech applications. Consequently, these chat records become the ideal source for evaluating the scoring mechanism presented in this paper. Four sets of test data, each comes in an XML file of 100 chat records, are employed for evaluations. Each XML file has an average of 10,000 words. The chat records and the Google search engine constitutes the domain corpora and general collection respectively while GNU Aspell version 0.60.4 (Atkinson 2006) is employed for detecting errors and generating suggestions. Four evaluations are performed, one for each set based on the steps described in Algorithm 1.

Determining whether $cISSAC_{u,r}$ and $cAspell_{u,r}$ is a correct replacement for $error_{u,r}$ is a delicate process that must be performed manually. To illustrate, consider the error *"itme"*. It is difficult to automatically determine whether *"itme"* should be replaced with *"time"* or *"item"*. Without neighbouring words, both replacements are of equally likely nature. Appearing as an error (*"shipping itme frame"*) in the first evaluation, Aspell's first choice for $error_{1,r} = itme$ is $cAspell_{1,r} = item$, while ISSAC ranked $cISSAC_{1,r} = time$ as the replacement. It is only proper for us to rate the replacement by Aspell as wrong given the error's appearance in the context of *"shipping"* and *"frame"*. In another error (*"Chad amateau <"* where $<$ is the end-of-sentence character) in the third evaluation, Aspell suggested *"amateur"* as the most ideal replacement while ISSAC suggested *"Amateau"*. As there is no such word as *"Amateau"* in the dictionary, we would be tempted to immediately rate the suggestion by Aspell as correct if we did not take into consideration the fact that *"Chad Amateau"* is a proper name.

The evaluation of the errors and replacements are conducted in an integrated manner. The errors are not classified into spelling errors, ad-hoc abbreviations and improper casing. For example, should the error *"az"* (*"AZ"* is the abbreviation for the state of

[3] A city in the state of Texas. Source from `http://www.ci.carrollton.tx.us/`

[4] `http://www.247customer.com/`

Algorithm 1 Evaluation of ISSAC

```
1: input four sets of chat records $CR_1$, $CR_2$, $CR_3$, $CR_4$
2: for each set of chat records $CR_u$ do
3:   initialize $EVA_u$, an array of triplets ($error_{u,r}$,$cISSAC_{u,r}$,$cAspell_{u,r}$) where $error_{u,r}$ is the $r^{th}$ error in the $u^{th}$ evaluation, $cISSAC_{u,r}$ is the correction proposed by ISSAC for the $r^{th}$ error in the $u^{th}$ evaluation, and $cAspell_{u,r}$ is the first suggestion proposed by Aspell for the $r^{th}$ error in the $u^{th}$ evaluation
4:   for each sentence in $CR_u$ do
5:     Tokenize the sentence to produce a set of words $T = \{t_1, ..., t_w\}$
6:     for each word $t \in T$ do
7:       if $t$ consists of all uppercase then
8:         Turn all letters in $t$ to lowercase
9:       else if $t$ consists of all digits then
10:        continue with next term
11:      end if
12:      Feed $t$ to Aspell
13:      if $t$ is identified as error by Aspell then
14:        initialize $NSC$, an array of new scores for all suggestions for error word $t$
15:        A set of $n$ suggestions for word $t$, $S = \{s_{1,1}, ..., s_{n,n}\}$ is generated by Aspell
16:        Append error $t$ at the end of $S$
17:        Perform lookup in the abbreviation dictionary and retrieve all corresponding $m$ expansions for $t$
18:        Append the $m$ expansions at the front of $S$
19:        The additional suggestions will produce an extended list $S = \{s_{1,1}, ..., s_{m,1}, s_{m+1,1}, ..., s_{m+n,n}, s_{m+n+1,n+1}\}$
20:        for each suggestion $s_{j,i} \in S$ do
21:          Execute ISSAC to obtain the new score $NS(s_{j,i})$
22:          Push $NS(s_{j,i})$ into $NSC$
23:        end for
24:        Sort $NSC$ in descending order
25:        Form the triplets ($t$,$NSC_1$,$s_{m+1,1}$) where $NSC_1$ is the first element in the sorted $NSC$ (i.e. the replacement proposed by ISSAC) and $s_{m+1,1}$ is the first suggestion by Aspell
26:        Push the triplets ($t$,$NSC_1$,$s_{m+1,1}$) into the array $EVA_u$
27:      else
28:        continue with next term
29:      end if
30:    end for
31:  end for
32:  for each triplets ($error_{u,r}$, $cISSAC_{u,r}$, $cAspell_{u,r}$) in $EVA_u$ do
33:    if $cISSAC_{u,r}$ is the correct replacement for $error_{u,r}$ then
34:      Rate $cISSAC_{u,r}$ as 1
35:    else
36:      Rate $cISSAC_{u,r}$ as 0
37:    end if
38:    if $cAspell_{u,r}$ is the correct replacement for $error_{u,r}$ then
39:      Rate $cAspell_{u,r}$ as 1
40:    else
41:      Rate $cAspell_{u,r}$ as 0
42:    end if
43:  end for
44: end for
45: Count the number of $cISSAC_{u,r}$ and $cAspell_{u,r}$ rated as 1 for all the four evaluations $EVA_{u=1}$, $EVA_{u=2}$, $EVA_{u=3}$ and $EVA_{u=4}$
```

Table 1. Accuracy of Aspell and ISSAC across four evaluations

	Evaluation 1, $EVA_{u=1}$	Evaluation 2, $EVA_{u=2}$	Evaluation 3, $EVA_{u=3}$	Evaluation 4, $EVA_{u=4}$	Average
number of correct replacements using ISSAC, $cISSAC_{u,r}=1$	97.06%	97.07%	95.92%	96.20%	96.56%
number of correct replacements using Aspell, $cAspell_{u,r}=1$	74.61%	75.94%	71.81%	75.19%	74.39%

"Arizona") in the context of *"Glendale az <"* be considered as an abbreviation or improper casing? The boundaries between the different types of dirtiness that occur in real-world texts, especially those from online sources, are not clear. This is the main reason behind the increasing number of efforts that attempt to provide techniques to handle various dirtiness in an integrated manner (Tang et al. 2005, Mikheev 2002, Sproat et al. 2001, Clark 2003). After a careful evaluation of all replacements suggested by Aspell and by ISSAC for all 2016 errors, we discovered a promising improvement in accuracy using the latter. The accuracy is obtained by dividing the number of errors with correct replacement by the total number of errors identified by Aspell. As shown in Table 1, the use of the first suggestion by Aspell as replacement yields an average of 74.4%. With the addition of the various weights that form ISSAC, an average increase of 22% was achieved, resulting to an improved accuracy of 96.5%.

5 Discussion

The list of suggestions and the initial ranks provided by Aspell are integral parts of ISSAC. The achievement of an average of 74.4% accuracy by Aspell itself, given the extremely poor nature of the texts shows the existing strength of the Metaphone algorithm and near-miss strategy. The further average increase of 22% in accuracy demonstrates the potential of the combined weights with regard to spelling error correction and other related areas. In the course of analyzing the remaining 3.5% of errors which have been wrongfully replaced, we have discovered several interesting points as explained below.

Firstly, half of the errors with wrong corrections are actually manifestations of certain inadequacies that ISSAC has inherited from Aspell. In other words, the accuracy of correction by ISSAC is bounded by the coverage of the list of suggestions S produced by Aspell. About 2% of wrong replacements is due to the absence of the correct replacement from the list of suggestions produced by Aspell. For example, the error *"dotn"* in the context of *"i dotn have"* was wrongfully replaced by both Aspell and ISSAC as *"do-tn"* and *"do tn"* respectively. After a look into the evaluation log, we realized that the correct replacement *"don't"* was not in S. In another case, error *"everyhitng"* (as in *"over everyhitng again"*) was wrongfully replaced with *"overhung"* and *"everyhitng"* by Aspell and ISSAC respectively. In such cases, there is no way for ISSAC to propose the correct replacement except that error e is an abbreviation. If e is an abbreviation, then the correct replacement (i.e. expansion) would have made its way into the extended list S as one of the first m suggestions $\{s_1, ..., s_m\}$ (i.e. all possible expansions for potential abbreviation e).

Secondly, the use of the two immediate neighbouring words l and r to inject more contextual consideration into domain and general significance has contributed to the large portion of the increase in accuracy. This claim is based on the result of a separate evaluation similar to those presented in the previous section with one exception: we omit the domain DS and general GS significance from ISSAC. A stunning drop of accuracy was observed, with an average of only 77%. Despite the contribution of l and r to the overall performance of ISSAC, it is by no means foolproof. About 1% out of the total errors with wrong replacement are due to two flaws related to l and r.

- In the first one, the neighbouring words themselves are not correctly spelled. For example, the error *"iberal"* (in the context of *"morel iberal return"*) is incorrectly replaced by both Aspell and ISSAC. This is due to the low values of DS and GS which fail to capture the actual significance of the correct replacement (i.e. *"liberal"*) with respect to the erroneous left word *"morel"*. Nonetheless, as shown by the low percentage of such flaw, this problem is not drastic. An error *"gto"* (in the context of *"lookin gto buy"*) was correctly replaced with *"to"* by ISSAC even though the left word *"lookin"* is erroneous.

- In the second case, the left and right words are inadequate. This is especially true when the errors to be corrected are located at the start and end of sentence. For example, there are no left and right words for the error *"winsted"* in the context of *"> winsted <"*. Such phenomena are the same as not using contextual information when attempting to correct an error. In such cases, the most popular suggestion $s_{j,i} \in S$ in both domain and general collection will triumph. Even with one or both neighbouring words present, incorrect replacement is also possible due to the indiscriminative nature of the neighbours. In the example *"both ocats <"*, the left word *"both"* does not provide much clue as to adequately discriminate between suggestions such as *"coats"*, *"cats"* and *"acts"*. Such neighbouring words are in sharp contrast to better ones such as *"wood"* as in *"the mindi wood"*.

Lastly, the remaining 0.5% can be seen as anomalies where ISSAC does not apply. There are two cases for these anomalies:

- Similar to throwing a dice, the first group of anomalies is characterized by the equally likely nature of all the possible replacements. For example, in the context *"Janice cheung <"*, the left word is correctly spelled and has adequately confined the suggestions to proper names even though the right word is absent. In addition, the correct replacement *"Cheung"* is present as a suggestion $s_{j,i} \in S$. Despite all these, both Aspell and ISSAC decided to replace *"cheung"* with *"Cheng"*. A look into the evaluation log reveals that the surname *"Cheung"* is as common as *"Cheng"*. In such cases, the probability of replacing e with the correct replacement is c^{-1} where c is the number of suggestions with approximately same $NS(s_{j,i})$.

- The second group of anomalies are due to contrasting value of certain weights, especially NED and i^{-1}, that causes wrong replacements to be made. For example, in the case *"cannot chage an"*, ISSAC replaced the error *"chage"* with *"charge"* instead of *"change"*. All the other weights for *"change"* are comparatively higher (i.e. DS and GS) or the same (i.e. RF, NED and AF) as *"charge"*. Such inclination indicates that *"change"* is the most proper replacement given the various cues. Nonetheless, the original rank by Aspell for *"charge"* is *i=1* while *"change"* is *i=6*. As smaller i indicates higher rank, the inverse of the original rank by Aspell i^{-1} results in the plummeting of the combined weight for *"change"*.

6 Conclusion and Future Work

As more and more language and speech applications open up to the use of online sources, the need to handle dirty texts becomes inevitable. Regardless of whether we acknowledge this fact, the quality of output and the proper functioning of such applications are, to a certain extent, dependent on the cleanliness of the input texts. Most of the existing techniques for correcting spelling errors, expanding abbreviations and restoring cases are studied separately. We, along with an increasing number of researchers, have acknowledged the fact that many errors in texts are composite in nature. As we have demonstrated during our evaluation and discussion in this paper, many errors are difficult to be classified as either spelling errors, ad-hoc abbreviations or improper casing. In this paper, we presented ISSAC, an integrated scoring mechanism that builds upon the famous spell checker Aspell for simultaneously providing solution to spelling errors, abbreviations and improper casing. ISSAC combines weights based on various information sources, namely, original rank by Aspell, reuse factor, abbreviation factor, normalized edit distance, domain significance and general significance. Four evaluations conducted over 40,000 words have demonstrated a promising accuracy at an average of 96.5%.

Even though the idea for ISSAC was first motivated and conceived within the paradigm of ontology engineering, we see great potential in further improvements and fine-tuning for a wide range of uses, especially in language and speech applications. We hope that an integrated approach such as ISSAC will pave the way for more research in providing a complete solution for text preprocessing (i.e. text cleaning) in general. At this moment, the various factors and weights are considered as equally important in this version of ISSAC. Depending on the applications or dirtiness of the texts, changes to the existing weights or even adding new weights may be necessary. We are planning to vary the importance of the different weights and fine-tune ISSAC to possibly improve the remaining 3.5% flaws pointed out in the discussion section. In addition, we have plans to extend the evaluation of ISSAC using other types of data such as news articles from online media sites. Last but not least, the assessment of the scalability of ISSAC in terms of runtime and accuracy using much larger datasets will also be explored.

Acknowledgement

This research was supported by the Australian Endeavour International Postgraduate Research Scholarship, and the Research Grant 2006 by the University of Western Australia. The authors would like to thank `247Customer.com` for providing the evaluation data. Gratitude to the developer of GNU Aspell, Kevin Atkinson.

References

Atkinson, K. (2006), 'Gnu aspell 0.60.4', http://aspell.sourceforge.net/.

Castellanos, M. (2003), Hotminer: Discovering hot topics on the web, *in* M. Berry, ed., 'Survey of Text Mining', Springer-Verlag.

Chan, S., He, B. & Ounis, I. (2005), An in-depth survey on the automatic detection and correction of spelling mistakes, *in* 'Proceedings of the 5th Dutch-Belgian Information Retrieval Workshop (DIR)'.

Cimiano, P. & Staab, S. (2005), Learning concept hierarchies from text with a guided agglomerative clustering algorithm, *in* 'Proceedings of the Workshop on Learning and Extending Lexical Ontologies with Machine Learning Methods', Bonn, Germany.

Clark, A. (2003), Pre-processing very noisy text, *in* 'Proceedings of the Workshop on Shallow Processing of Large Corpora at Corpus Linguistics'.

Damerau, F. (1964), 'A technique for computer detection and correction of spelling errors', *Communications of the ACM* **7**(3), 171–176.

Degeratu, M. & Hatzivassiloglou, V. (2002), Building automatically a business registration ontology, *in* 'Proceedings of the ACM Annual National Conference on Digital Government Research'.

Holmes, D. & McCabe, C. (2002), Improving precision and recall for soundex retrieval, *in* 'Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC)'.

Kietz, J., Volz, R. & Maedche, A. (2000), Extracting a domain-specific ontology from a corporate intranet, *in* 'Proceedings of the 4th Conference on Computational Natural Language Learning', Lisbon, Portugal.

Kuenning, G. (2006), 'International ispell 3.3.02', http://ficus-www.cs.ucla.edu/geoff/ispell.html.

Kukich, K. (1992), 'Technique for automatically correcting words in text', *ACM Computing Surveys* **24**(4), 377– 439.

Lait, A. & Randell, B. (1993), An assessment of name matching algorithms, Technical report, University of Newcastle upon Tyne.

Levenshtein, V. (1966), 'Binary codes capable of correcting deletions, insertions, and reversals', *Soviet Physics Doklady* **10**(8), 707–710.

Lita, L., Ittycheriah, A., Roukos, S. & Kambhatla, N. (2003), truecasing, *in* 'Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics', Japan.

Maedche, A. & Volz, R. (2001), The ontology extraction & maintenance framework: Text-to-onto, *in* 'Proceedings of the IEEE International Conference on Data Mining', California, USA.

Mikheev, A. (1999), A knowledge-free method for capitalized word disambiguation, *in* 'Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics'.

Mikheev, A. (2002), 'Periods, capitalized words, etc.', *Computational Linguistics* **28**(3), 289–318.

Novacek, V. & Smrz, P. (2005*a*), Bole - a new bio-ontology learning platform, *in* 'Proceedings of the Workshop on Biomedical Ontologies and Text Processing'.

Novacek, V. & Smrz, P. (2005*b*), Ole - a new ontology learning platform, *in* 'Proceedings of the International Workshop on Text Mining'.

Odell, M. & Russell, R. (1918), U.s. patent numbers 1,261,167. U.S. Patent Office, Washington, D.C.

Odell, M. & Russell, R. (1922), U.s. patent numbers 1,435,663. U.S. Patent Office, Washington, D.C.

Pakhomov, S. (2001), Semi-supervised maximum entropy based approach to acronym and abbreviation normalization in medical texts, *in* 'Proceedings of the 40th Annual Meeting on Association for Computational Linguistics'.

Park, Y. & Byrd, R. (2001), Hybrid text mining for finding abbreviations and their definitions, *in* 'Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)'.

Philips, L. (1990), 'Hanging on the metaphone', *Computer Language Magazine* **7**(12), 38–44.

Robertson, S. (2004), 'Understanding inverse document frequency: On theoretical arguments for idf', *Journal of Documentation* **60**(5), 503–520.

Sanchez, D. & Moreno, A. (2005), Automatic discovery of synonyms and lexicalizations from the web, *in* 'Proceedings of the 8th Catalan Conference on Artificial Intelligence'.

Schwartz, A. & Hearst, M. (2003), A simple algorithm for identifying abbreviation definitions in biomedical text, *in* 'Proceedings of the Pacific Symposium on Biocomputing (PSB)'.

Smith, T. & Waterman, M. (1981), 'Identification of common molecular subsequences', *Journal of Molecular Biology* **147**(1), 195–197.

Sombatsrisomboon, R., Matsuo, Y. & Ishizuka, M. (2003), Acquisition of hypernyms and hyponyms from the www, *in* 'Proceedings of the 2d International Workshop on Active Mining', Japan.

Sproat, R., Black, A., Chen, S., Kumar, S., Ostendorf, M. & Richards, C. (2001), 'Normalization of non-standard words', *Computer Speech & Language* **15**(3), 287–333.

Tang, J., Li, H., Cao, Y. & Tang, Z. (2005), Email data cleaning, *in* 'Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining'.

Turney, P. (2001), Mining the web for synonyms: Pmi-ir versus lsa on toefl, *in* 'Proceedings of the 12th European Conference on Machine Learning (ECML)', Freiburg, Germany.

Wagner, R. & Fischer, M. (1974), 'The string-to-string correction problem', *Journal of the ACM* **21**(1), 168–173.

Wong, W., Liu, W. & Bennamoun, M. (2006), Progress and open problems in ontology engineering from text. Submitted to the Journal of Web Semantics.

Xu, F., Kurz, D., Piskorski, J. & Schmeier, S. (2002), An domain adaptive approach to automatic acquisition of domain relevant terms and their relations with bootstrapping, *in* 'Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC)', Canary island, Spain.