Auto-correction of Consumer Generated Text in Semi-Formal Environment

Lipika Dey*, **Gargi Roy***

*TCS Innovation Labs, Delhi, {lipika.dey, roy.gargi}@tcs.com

Abstract

This paper presents a framework to identify noisy words along with their correction scheme and identification of domain specific terms for consumer generated text in semi-formal environment. Semi-formal environment is writing in a work environment such as within an enterprise to communicate with peers about work. Text produced in this environment does not contain slang, icons and shortening of words, phrases. However, these kind of text suffers from spelling mistakes, typographical errors, irregular use of punctuations and few other types of errors which are caused by the extraction procedures of the text from several applications such as email clients, web portals etc. The framework presented in this paper cleans the text to be analyzed, identifies and categorizes several kinds of error and domain specific words from unknown words based on some patterns and word distributions over the text. The correction scheme is developed with human supervision and the framework is run on huge data set with manual evaluation being satisfactory.

1. Introduction

Most of the text generated in informal text environment (non-work related communication) is noisy consisting of spelling mistakes, random abbreviation of words, random use of upper case, space and punctuations. However, work related writing generated within an enterprise/organization for internal work context, emails generated to communicate with colleagues, team mates can be considered as semi-formal text. Because, these texts generally do not contain very high degree of spelling mistakes, idiosyncrasies and arbitrary use of punctuations, spaces, and unnecessary use of upper case. Also, these kind of texts do not contain arbitrary use of symbolic icons such as smilies, shortened words and phrases such as writing '4u' instead of 'for you'. However, semi-formal texts also suffer from typographical errors, grammatical errors and more over they may contain abbreviations of domain words and several other types of errors caused by the process of text extraction from several web sources.

So, in this work, we have developed a framework for text generated in semi-formal environment with automatic rule based correction facility for identified erroneous words along with the extraction of unknown (to framework) but domain specific words from text which should not be corrected. The unknown and erroneous words are identified through matching the words against dictionary. We are not recognizing grammatically incorrect words which do exist in the dictionary, for example, ambiguity between 'form' and 'from'. Any English dictionary can be used in this framework. The framework first extracts the consumer generated text from different web sources such as email client, web portal then pre-process the text to eliminate several types of noise including errors introduced in the text extraction process. Then the text is tokenized and unknown words are segregated. The unknown words are then categorized based on some observed patterns of letters, combination of upper and lower cases in the word. Among the categorized unknown words few are marked as domain specific words which are not corrected and few are denoted as actual error to be corrected based on some insights considered and frequency distribution of the words across the text. Domain specific words are used to extend the dictionary and there by enhancing the framework's performance. The framework then adopts a rule based correction policy to generate suggestion(s) for the actual erroneous words. The correction policy is designed in such a way that it does not require much computational overhead in average case. The framework has been executed on two data sets collected from a financial organization and from an enterprise. More than 7,72,000 words are analyzed and suggestions are generated accordingly. We also manually evaluated the performance of the framework which is quite satisfactory.

The paper is organized as follows. Next section contains brief literature and third section describes the overview of the whole framework. Unknown word analysis and rule based correction policy along with evaluation is presented in section fourth and fifth respectively. Finally the paper is concluded.

2. Background

Literature consists of several study for noisy text processing and correction, the text being generated from traditional transcriptions of speech, text obtained by using OCRs on text images or text gathered from web sources such as web pages, social media, SMS etc. (Kukich, 1992) presents a nice survey on methodologies for detecting and correcting spelling errors in text. (Clark, 2003) presents a machine learning methodology using generative models and a noisy channel method for pre-processing of very noisy text. A system called ISSAC has been represented in (Wong et al., 2006) which corrects spelling errors, restores cases and expands ad-hoc abbreviations. They used an integrated scoring model for pre-processing noisy text. Work has also been done on OCR errors (Nartker et al., 2003), ASR errors (Sarma and Palmer, 2004) and output of SMS text (Choudhury et al., 2007). Opinions are mined from the noisy text by first cleaning the data with domain expertise in (Dey and Haque, 2009). Apart from text cleaning, few spelling correction methods have been proposed in (Hodge and Austin, 2001) and (Elmi and Evens, 1998) where the later presents a context based spell correction. (Mikheev, 2002) presents an approach for disambiguation of capitalized words in positions where capitalization is expected, sentence boundary disambiguation and identification of abbreviations. However, our approach cleans the text gathered from semi-formal environment and identifies several types of errors along with domain specific words using word distribution and patterns which are equally unknown to the framework. The domain words are not corrected instead used for extending dictionary and different types of errors are identified and accordingly correction policy have been developed for different types.

3. Overview of the framework

The framework consists of several modules which perform assigned tasks. The overview of the framework is depicted

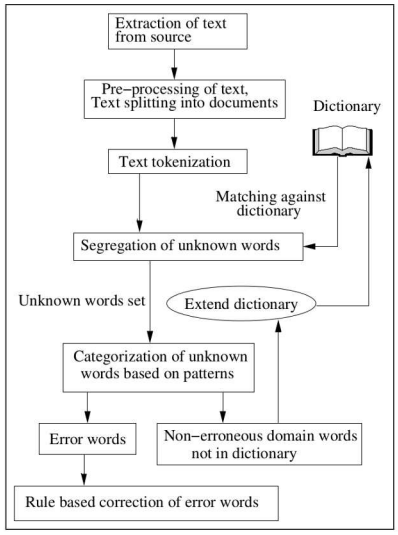

Figure 1: Overview of the framework.

in Figure 1. At first consumer generated texts are extracted through *import* facility of the source applications where the text has been generated. Once text is imported it is then pre-processed which is first splitting the texts into documents according to their time stamps, unique identifiers and other meta information such as 'To', 'From', 'Subject', 'Body' headers in case of email data. The pre-processing also includes identifying and cleaning of noises embedded in the text and introduced by the *import* process. Details about each task performed by the framework is discussed in the next section.

In next phase, the cleaned text is tokenized and from the tokens word tokens are extracted and matched against the dictionary to construct a set of unknown words (which are not contained in dictionary). Once unknown words are segregated, they are categorized to error words which need correction and domain specific words which should not be corrected. The recognized domain specific words are also used to extend the existing dictionary to enhance the performance of error identification. The categorization of unknown words are done using observed patterns and word distributions (with human supervision) and then the error words are corrected using a rule based correction policy.

4. Analysis of unknown words

4.1. Pre-processing and segregating unknown words

The collected text has several kinds of errors including typographical error, import error and human error. Typographical error includes spelling mistakes, missing full stop between sentences, missing space after punctuations. These kinds of errors are shown in the following text snippet from the original data highlighting the error in italics. "account for the *entrepreneurThe* entrepreneur gets a professional online shop..One saves a *seperate* bookkeeping system and the hassle..to college tuition for our kids or a skiing *holiday.What* if a customer could..". Other than typographical errors, several other kinds of errors are introduced when the text is extracted from different web sources through application specific import facility which often includes programming code fragment, omits few line breaks, omits/misplaces spaces etc. Following are text snippets showing import error in italics. " .. company by a 'face to face' *skyping-, video-* , or other life* *conference..Succession Planning Issue•*   * European Clients that enjoyed.." Although the text is generated in semi-formal environment, the text may contain few basic smilies for work purpose, such as "..App (ratingstars, ☺☹, send email to their emailadress to grant opportunity to reply."

In case of analyzing email data, the framework removes signatures by identifying few markers in each line followed by lines containing words such as 'Regards', 'Best Regards', 'rgds', 'Thanks', 'Thanks and Regards', 'Best regards' etc. These markers includes 'Ph:-', 'Email:-', 'mailto:', 'Cell:-', 'Website:' etc. These markers may occur in different order in each line. Many a times signatures contain symbol pattern such as "=====−−=====" followed by extra information generated by few types of mail clients. Those cases are also considered for identification and removal of signatures from email data.

The human error occurred due to some activities done on the text. In our data, some chunk of text data have been generated through copying multiple piece of text written by several people in the work context. During the copy-paste and merging actions of different text pieces contained in excel cells, errors occur as missing space or punctuations such as "design a wind turbine that generates both electricity recovers energy from the sun and produces *waterBudget:* to start with conception and tests 100.000€..". For tokenization of such noisy text we used twitter nlp tokenizer (O'Connor et al., 2010; Owoputi et al., 2012; Owoputi et al., 2013) as it handles several types of noise such as splitting at punctuations, spaces, not splitting at '-', handle missing space between proper sentences. Further details on twitter npl can be found in the given references.

However, as twitter nlp handles issues of text generated in informal environment consisting of emoticons, symbols, it is not fully usable for semi-formal environment. For example, during tokenization of the text snippet "..they need and they *want.The* simplicity of this..", twitter nlp outputs *'t.T'* as tokens and there by effecting tokenization of surrounding words. Similarly tokenization of "Huge prospects *databaseAssets:Database:*tokenization.." contains ':D' as a token (because this is character symbol of a smiley icon) and instead of recognizing 'databaseAssets', 'Database' as tokens it gives tokens as 'databaseAssets', ':D' and 'atabase'. It also recognizes icons as tokens. Hence, we have handled such cases by identifying and introducing according characters in the text before tokenization to avoid such scenarios such as introduces a space character after ':', 't.' 'o.', '■' so that when the tokenizer splits at the space desired tokens for an semi-formal environment are obtained. However, texts are not split at '-', '_' as two words concatenated with '-' essentially indicates one word in English language and we assume the words concatenated with '_' are of names of different entities such as file name. Existing code snippets (import error) are also cleaned from text such as removing ' ', '=>' and replacing them with space for proper tokenization.

After tokenization, the tokens of special characters are discarded and word tokens (excluding tokens of day and month names) are further analyzed for detecting unknown words which include domain specific words and error words. Unknown words are obtained by matching the words against dictionary and if a word is not contained in the dictionary then it is marked as unknown.

4.2. Patterns of unknown words

Once the unknown word set is obtained, the words are analyzed to identify error words and possible domain specific words using observed patterns. Unknown words are categorized into five different types based on their patterns which are as follows.

Type-1: Unknown word having all lower case letters, named as *All_Lower*. Example includes 'simillar', 'demopgraphic', 'featue' etc.

Type-2: Unknown word having all upper case letters, named as *All_Upper*, for example, 'WOW', 'WIFI', 'BENEFIT' etc.

Unknown word having a combination of lower and upper case letters, named as *Multi_Caps*. This type of unknown words are further categorized into three groups which are as follows.

Type-3: Unknown word having single upper case letter followed and preceded by lower case letters, named as *Single_Upper_in_Middle*. This type of unknown words have lower case letters in beginning an end positions. Examples include 'bothExample', 'situationWe', 'clientsManage' etc.

Type-4: Unknown word consisting of an upper case letter in the starting position, for example, 'Facebook', 'Google', 'Kinect' etc. This type of unknown words are called *Initial_Upper*.

Type-5: Unknown word comprising of random mixing up of multiple upper case and lower case letters are called *Multi_Caps* and example includes, 'PayPal', 'WhatsApp', 'BankOnLine', 'momentsWOWs', 'PRODUCTSEveryone', 'SMEs' etc.

The procedure to identify different types of unknown words based on their pattern is depicted in Algorithm 1.

4.3. Identifying errors and domain specific words

To identify the domain specific words over actual erroneous words, the distribution of the unknown words across the text along with their patterns are considered with few intuitions. The intuitions considered are as follows.

I_1 : *Same error does not occur uniformly*

I_2 : *When words are written in upper case in semi-formal environment that implies special significance/importance*

I_3 : *When multiple versions of spelling of an unknown word exist across text then the word is likely to be correct*

Considering I_1, distributions of unknown words over multiple documents (whose content texts are generated by different persons) are analyzed and we found that the unknown words having uniform distribution across the text are actually not error words. Figure 2 shows the uniform distribution of such unknown words such as 'facebook', 'Internet', 'google', 'app', 'smartphone' etc. which are correct words and used to enhance dictionary. The distribution is uniform as the y-axis denotes document id, it can be seen that the aggregate values of the occurrence of the words is approximately equal to the number of documents and lesser slopes denote more total number of occurrence. On the other hand, typographical errors generally occur non-uniformly. Figure 3 shows the non-uniform distribution of unknown words which are mostly erroneous, for example, 'evironment', 'seperate', 'Immediatly', 'fullfill' etc.

I_2 has been considered with the insight that in a semi-formal environment lot of work related words and phrases are abbreviated to only upper case letters while communicating with peers or team members within an organization which are domain specific terms and should not be attempted to rectify.

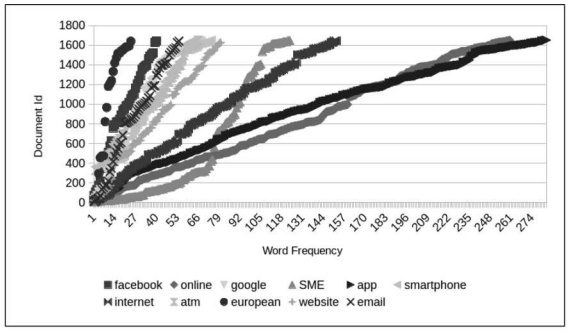

Figure 2: Unknown words having uniform distribution over repository-these are mostly correct words.

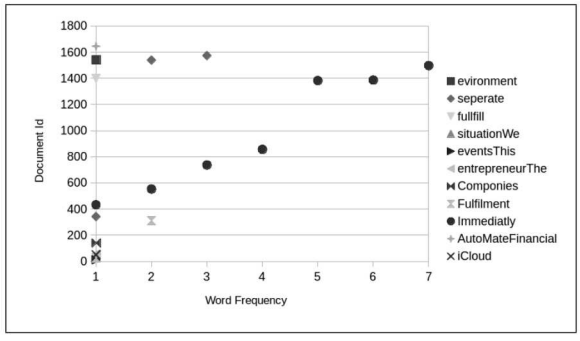

Figure 3: Unknown words having non-uniform distribution over repository-can be observed mostly as erroneous.

For example, 'BLE' is abbreviated for "Bluetooth Low Energy", 'SOS' stands for "Safe Or Secure", 'MAIS' is for "Multi Actor Impact Simulation". On the other hand, few other words in upper case which are not abbreviated but having significance are 'START-UP', 'HOST-TO-HOST', 'EMAIL', 'OFFLINE' etc. Next section presents the evaluation results of considering these insights while measuring performance of the framework.

Apart from the word distribution, existence of the several versions of the unknown words across the tex are also computed considering I_3 to find unknown but non-error words. For example, on-line word can have multiple versions such as 'On-Line', 'Online', 'online' and WI-FI word can occur in several forms such as 'wifi', 'WiFI', 'WIFI', 'WiFi', 'Wifi'. These multiple versions of WI-FI are not confined with a single file but among seven files, similarly the multiple versions of on-line word is distributed over 260 files. These kinds of words having uniform distribution are entered in the dictionary to extend it.

5. Rule based correction and evaluation

This section presents the rule based correction policy and the evaluation of the framework performance in terms of accuracy of identifying proper erroneous words and domain specific words along with the suggestions generated for the error words.

5.1. Rule based correction policy

Different correction policies have been adopted for different kinds of erroneous words. For type-1 error i.e. *All_Lower*, we have computed its nearest neighbor word in the dictionary using minimum Levenshtein distance (Levenshtein, 1966). The corrections for the error words 'simillar', 'demopgraphic',

Table 1: Unknown words of several types and their corrections suggested using the rule based corrections

Type: *All_Lower*	Type: *Single_Upper_in_Middle*
Unknown word: Suggested correction	Unknown word: Suggested correction
evironment: environment	eventsThis: events This
seperate: separate	servicesEmployees: services Employees
uncomplicadted: uncomplicated	viewLess: view Less
regsitration: registration	transactionWhat: transaction What
simillar: similar	developerAnd: developer And
continuus: continuous	consumersImprove: consumers Improve
fullfill: fulfill	forecastLeverage: forecast Leverage
remeber: remember	bugsSuch: bugs Such
produtcts: products	informationTransmit: information Transmit
recuitment: recruitment	entrepreneurThe: entrepreneur The
assistent: assistant	itemSellers: item Sellers
Type: *Multi_Caps* Unknown word	[Suggested corrections]
TibcoVendor	[Tibco, TibcoVendor, Vendor]
crowdfundingsInterim	[crowdfundings, crowdfundingsInterim, Interim]
AutoMateFinancial	[Auto, Mate, AutoMate, Financial, AutoMateFinancial]
iCloud	[i, iCloud, Cloud]
ProvisionsPOS	[Provisions, P, PO, POS, ProvisionsPOS]
BigData	[Big, Data, BigData]
FarmVille	[Farm, Ville, FarmVille]
SMEOrange	[S, SM, SME, SMEOrange, Orange]
OnLine	[On, Line, OnLine]
DropBox	[Drop, Box, DropBox]
TripAdvisor	[Trip, Advisor, TripAdvisor]

'featue' are given as 'similar', 'demographic', 'feature' respectively. As per the intuitions considered, type-2 unknown words i.e. *All_Upper*, no correction is done for these types of error words as they are marked as domain words.

In order to correct *Single_Upper_in_Middle* types of errors which is type-3, the error word is split at the upper case letter. After splitting two separate words are obtained which are then matched if they are standard English word (through matching with the dictionary). If both are found to be English words then both the words with a space introduced between them are output as the correction. For example, 'bothExample', 'situationWe', 'clientsManage' are corrected as 'both Example', 'situation We', 'clients Manage' respectively. Type-4 errors are not corrected because most of them consist of nouns which need not be corrected.

The correction scheme of type-5 errors consists of splitting the error word at all the upper case letters to obtain substrings and then generating a set of strings as suggestion which consists of the individual substrings, combination of the substrings in an ordered manner and the error word itself. The error word itself is included in the suggestion because there may be a case that the word is correct and although splitting up would cause different correct dictionary words but that would change the meaning. For example, the word 'DropBox' which is non-dictionary but has a technical meaning (data storage space in a cloud platform), however, if it is split into 'Drop' and 'Box' where both the words exists in dictionary with different meanings, the entire meaning of the word will be changed and that would be a wrong suggestion. Other examples to illustrate the correction scheme include, 'PayPal', 'BankOnLine' for which suggestions are [Pay, PayPal, Pal], [Bank, BankOn, BankOnLine, On, OnLine, Line] respectively. The overall correction policy is outlined in Algorithm 2 and Table 1 depicts examples of different types of errors and their correction/suggestions generated by the framework.

5.2. Evaluation

Experimental data and execution time:
The framework has been implemented in java associating a standard English dictionary available over Internet. However, the framework can be used with any other dictionary in the same language. We have run our framework on two data sets in a machine having configuration as follows. Intel® Core™ i7-4600U CPU @ 2.10GHz × 4 with 15.6 GiB memory having 64-bit ubuntu 14.04 LTS operating system. As the correction policy is rule based and edit distance is not calculated for all error words for its correction, instead for a particular type of error (type-1) edit distance is computed, this saves computational overhead based on the amount of the particular error type present in the data set. One of the data sets is collected by a financial institution where the huge amount of text is generated by its customers and employees. In this data set total 7,72,020 words are analyzed and CPU execution time was measured as 6 minutes and 16 seconds having 40.3% type-1 error of all the unknown words.

Another data set comprises of the gathered emails generated within an enterprise. These mails are work related and generated by the employees to communicate with team members within the enterprise. From this data set total 5,782 words are analyzed and the framework implementation took 13.7 seconds as CPU execution time having 11.74% type-1 error of all unknown words.

Results: To evaluate the correction policy we used manual intervention. Manually we validated total 2,554 suggestions given by the framework for identifying domain words and correction(s) suggested for different kinds of erroneous words. Table 2 shows the evaluation results along with the accuracy. The suggestion consisting of single word is evaluated positively if the suggested word is actually correct.

As *All_Upper* types of unknown words are considered as domain words, 400 such words were evaluated with human supervision whether the word is actually a domain word or not. We got 68.25% accuracy in this case i.e 68.25% such words were actually domain words.

Total 1506 error words of type *Single_Upper_in_Middle* are evaluated. The suggestion is evaluated positively when two words merged (last word of a sentence and first word of next sentence) and in suggestion those are corrected properly by splitting those words and adding a space between them. In this case, 79.61% accuracy has been obtained i.e. 79.61% suggestions were actually correct.

Suggestion for *All_Lower* type unknown word consist of single word. During evaluation, it was checked whether the suggested word was appropriate correction for that unknown word or not. In this category, 200 unknown words were evaluated with 76.5% accuracy achieved i.e. 76.5% suggestions were actually correct.

As the correction scheme considers words having two or more versions of spelling as non-erroneous words, in order to evaluate, we considered 217 words having two or more versions of spelling. These words are validated whether they are actually non-dictionary but correct words. In this case, 82.03% were found actually correct words.

For the case of *Multi_Caps* type unknown words, framework generated suggestion includes more than one word, so, we evaluated the suggestion positively if it contains the correct word. 231 words from this category were evaluated and 90.04% accuracy was obtained i.e. 90.04% suggestions contained the correct word in the list of words suggested for a *Multi_Caps* type unknown word.

6. Conclusion

In this paper we have presented a framework which cleans consumer generated semi-formal text and automatically identi-

Table 2: Accuracy of rule based correction scheme with manual evaluation

Type	Suggestions evaluated (total: 2554)	Accuracy
All_Upper	400	68.25%
Single_Upper _in_Middle	1506	79.61%
All_Lower	200	76.5%
Multiple versions	217	82.03%
Multi_Caps	231	90.04%

Algorithm 1: FindType(unknownWord)

Input : Letters of unknown word
Output: Returns the type based on different patterns
foreach $letter \in unknownWord$ is lower case **do**
  $type \leftarrow All\_Lower$;
end
foreach $letter \in unknownWord$ is upper case **do**
  $type \leftarrow All\_Upper$;
end
if $(beginLetter \bigwedge endLetter) \in unknownWord$ is lower case **then**
  foreach $letter \in (unknownWord \setminus \{beginLetter, endLetter\})$ **do**
    if *letter is upper case* **then**
      increment $UpperCaseCount$;
    end
  end
  if $UpperCaseCount = 1$ **then**
    $type \leftarrow Single\_Upper\_in\_Middle$;
  else if $UpperCaseCount > 1$ **then**
    $type \leftarrow Multi\_Caps$;
  end
end
else if $beginLetter \in unknownWord$ is upper case **then**
  foreach $letter \in (unknownWord \setminus beginLetter)$ **do**
    if *letter is lower case* **then**
      $type \leftarrow Initial\_Upper$;
    else
      $type \leftarrow Multi\_Caps$;
    end
  end
end
else
  $type \leftarrow Multi\_Caps$;
end

Algorithm 2: RuleBasedCorrection(errorWord)

Input : Letters of unknown word
Output: Returns the rule based correction for different patterns
Notations: FindNearestWord: returns word from dictionary with minimum edit distance w.r.t errorWord
$type \leftarrow$FindErrorType($errorWord$);
if $type$ *is* All_Lower **then**
  $out \leftarrow$ FindNearestWord($errorWord, Dictionary$);
else if $type$ *is* $(All\_Upper \bigvee Initial\_Upper)$ **then**
  No correction;
else if $type$ *is* $Single\_Upper\_in\_Middle$ **then**
  $\{str_1, str_2\} \leftarrow$ get substrings by splitting $errorWord$ at upper case;
  if $\{str_1, str_2\} \in Dictionary$ **then**
    $out \leftarrow$ concatenation of str_1 followed by space and then str_2;
  else
    No correction;
  end
else
  $SubstringList \leftarrow$ get substrings by splitting $errorWord$ at upper cases; /* $Multi\_Caps$ */
  foreach $str \in SubstringList$ **do**
    $Next_{str} \leftarrow$ next string of $str \in SubstringList$;
    if $Next_{str}$ *is* $All\_Upper \bigwedge$ *Length of* $Next_{str}$ *is* 1 **then**
      $Appended_{str} \leftarrow \{str, Next_{str}\}$;
      $out \leftarrow out \bigcup \{str, Appended_{str}\}$;
    else
      $out \leftarrow out \bigcup \{str, Next_{str}\}$;
    end
  end
  if $errorWord \notin out$ **then**
    $out \leftarrow out \bigcup \{errorWord\}$;
  end
end

fies unknown words for correction along with domain specific words, where semi-formal environment is the text generated in the work context within an enterprise/organization. Among the unknown words, the framework recognizes the set of actual error words which need to be corrected and the set of domain specific words which should not be corrected using patterns and word distributions. A rule based correction policy has been developed for suggesting correction to the error words. Our framework has been executed on large data sets collected from the text generated within enterprise and organization in work context. We also have manual evaluation of the performance of the framework which is quite satisfactory. Future work includes enhancing the correction policy to handle error words in lower case where multiple dictionary words are merged without space forming a compound word, the dictionary being enhanced by the recognized domain specific words. Currently the compound word correction is done for other types of error mentioned in the paper.

7. References

Choudhury, Monojit, Rahul Saraf, Vijit Jain, Animesh Mukherjee, Sudeshna Sarkar, and Anupam Basu, 2007. Investigation and modeling of the structure of texting language. *International Journal of Document Analysis and Recognition (IJDAR)*, 10(3-4):157–174.

Clark, Alexander, 2003. Pre-processing very noisy text. *Proc. of Workshop on Shallow Processing of Large Corpora*:12–22.

Dey, Lipika and SK Mirajul Haque, 2009. Opinion mining from noisy text data. *International Journal on Document Analysis and Recognition (IJDAR)*, 12(3):205–226.

Elmi, Mohammad Ali and Martha Evens, 1998. Spelling correction using context. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*. Association for Computational Linguistics.

Hodge, Victoria J and Jim Austin, 2001. A novel binary spell checker. In *Artificial Neural Networks—ICANN 2001*. Springer, pages 1199–1204.

Kukich, Karen, 1992. Techniques for automatically correcting words in text. *ACM Computing Surveys (CSUR)*, 24(4):377–439.

Levenshtein, VI, 1966. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet Physics Doklady*, volume 10.

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

Nartker, Thomas A, Kazem Taghva, Ron Young, Julie Borsack, and Allen Condit, 2003. Ocr correction based on document level knowledge. In *Electronic Imaging 2003*. International Society for Optics and Photonics.

O'Connor, Brendan, Michel Krieger, and David Ahn, 2010. Tweetmotif: Exploratory search and topic summarization for twitter. In *ICWSM*.

Owoputi, Olutobi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith, 2013. Improved part-of-speech tagging for online conversational text with word clusters.

Owoputi, Olutobi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, and Nathan Schneider, 2012. Part-of-speech tagging for twitter: Word clusters and other advances. *School of Computer Science, Carnegie Mellon University, Tech. Rep*.

Sarma, Arup and David D Palmer, 2004. Context-based speech recognition error detection and correction. In *Proceedings of HLT-NAACL 2004: Short Papers*. Association for Computational Linguistics.

Wong, Wilson, Wei Liu, and Mohammed Bennamoun, 2006. Integrated scoring for spelling error correction, abbreviation expansion and case restoration in dirty text. In *Proceedings of the fifth Australasian conference on Data mining and analystics-Volume 61*. Australian Computer Society, Inc.