Autonomous Self-Assessment of Autocorrections: Exploring Text Message Dialogues

Tyler Baldwin
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48824
baldwi96@cse.msu.edu

Joyce Y. Chai
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48824
jchai@cse.msu.edu

Abstract

Text input aids such as automatic correction systems play an increasingly important role in facilitating fast text entry and efficient communication between text message users. Although these tools are beneficial when they work correctly, they can cause significant communication problems when they fail. To improve its autocorrection performance, it is important for the system to have the capability to assess its own performance and learn from its mistakes. To address this, this paper presents a novel task of self-assessment of autocorrection performance based on interactions between text message users. As part of this investigation, we collected a dataset of autocorrection mistakes from true text message users and experimented with a rich set of features in our self-assessment task. Our experimental results indicate that there are salient cues from the text message discourse that allow systems to assess their own behaviors with high precision.

1 Introduction

The use of SMS text messaging is widespread and growing. Users of text messaging often rely on small mobile devices with limited user interfaces to communicate with each other. To support efficient communication between users, many tools to aid text input such as automatic completion (autocompletion) and automatic correction (autocorrection) have become available. When they work correctly, these tools allow users to maintain clear communication while potentially increasing the rate at which they input their message, improving efficiency in communication. However, when these tools make a mistake, they can cause problematic situations. Consider the following example:

A_1: Euthanasia doing tonight?

B_1: Euthanasia?!

A_2: I typed whatcha and stupid autotype.

In this example, the automatic correction system on person A's phone interpreted his attempt to write the word *whatcha* as an attempt to write *euthanasia* (due to the keyboard adjacency of the *w* and *e* keys, etc.). This completely changed the meaning of the message, which confused person B. Although this instance was eventually discovered and corrected, the natural flow of conversation was interrupted and the participants were forced to make extra effort to clarify this confusion.

This example indicates that the cost of a mistake in autocorrection is potentially high. This is exacerbated by the fact that users will often fail to notice these mistakes in a timely manner, due to their focus being on the keyboard (Paek et al., 2010) and the quick and casual conversation style of text messaging. Because of this, autocorrection systems must have high accuracy to be useful for text messaging. This example also indicates that, when an autocorrection mistake happens (i.e., mistaken correction of *euthanasia*), it often causes confusion which requires dialogue participants to use the follow-up dialogue to clarify the intent. What this suggests is that the discourse between text message users may provide important information for autocorrection sys-

2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 710–719,
Montréal, Canada, June 3-8, 2012.

tems to assess whether an attempted correction is indeed what the user intended to type.

Self-assessment of its correction performance will allow an autocorrection system to detect correction mistakes, learn from such mistakes, and potentially improve its correction performance for future operations. For instance, if a system is able to identify that its current autocorrection policy results in too many mistakes it may choose to adopt a more cautious correction policy in the future. Additionally, if it is able to discover not only that a mistake has taken place but what the ideal action should have been, it will be able to use this data to learn a more refined policy for future attempts.

Motivated by this observation, this paper investigates the novel task of self-assessment of autocorrection performance based on interactions between dialogue participants. In particular, we formulate this task as the automatic identification of correction mistakes and their corresponding intended words based on the discourse. For instance, in the previous example, the system should automatically detect that the attempted correction "euthanasia" is a mistake and the true term (i.e., intended word) should have been "whatcha". To support our investigation, we collected a dataset of autocorrection mistakes from true text message users. We further experimented with a rich set of features in our self-assessment task. Our experimental results indicate that there are salient cues from the text message discourse that potentially allow systems to assess their own behavior with high precision.

In the sections that follow, we first introduce and give an analysis of our dataset. We then highlight the two interrelated problems that must be solved for system self-assessment, and outline and evaluate our approach to each of these problems. Finally, we examine the results of applying the system assessment procedure end-to-end and discuss potential applications of autocorrection self-assessment.

2 Related Work

Spelling autocorrection systems grew naturally out of the well studied field of spell checking. Most spell checking systems are based on a noisy channel formulation (Kernighan et al., 1990). Later refinements allowed for string edit operations of arbitrary length (Brill and Moore, 2000) and pronunciation modeling (Toutanova and Moore, 2002). More recent work has examined the use of the web as a corpus to build a spell checking and autocorrection system without the need for labeled training data (Whitelaw et al., 2009).

Traditional spell checking systems generally assume that misspellings are unintentional. However, much of the spelling variation that appears in text messages may be produced intentionally. For instance, text message authors make frequent use of acronyms and abbreviations. This motivates the task of text message normalization (Aw et al., 2006; Kobus et al., 2008), which attempts to transform all non-standard spellings in a text message into their standard form. The style of misspelling in text messages is often quite different from that of standard prose. For instance, Whitelaw et. al. (2009) applied the Aspell spell checker[1] on a corpus of mistakes in English prose and achieved an error rate of under 5%. Conversely, the same spell checker was found to have an error rate of over 75% on text message data (Choudhury et al., 2007).

Autocorrection in text messaging is similar to predictive texting and word completion technologies (Dunlop and Crossan, 2000). These technologies attempt to reduce the number of keystrokes a user must type (MacKenzie, 2002), potentially speeding up text entry. There are 2 primary sources of literature on text prediction. In one (often called *autocompletion*), systems attempt to predict the intended term before the user has finished typing it (Darragh et al., 1990; Chaudhuri and Kaushik, 2009). In the second, the system attempts to interpret ambiguous user input typed on a keyboard with a small number of keys, such as the 12 key keyboards found on many mobile phones (MacKenzie and Tanaka-Ishii, 2007). Few studies have looked at the effects SMS writing style has on predictive text performance. How and Kan (2005) analyze a corpus of 10,000 text messages and conclude that changing the standard mapping of letters to keys on 12 key keyboards could improve input performance on SMS data.

Although never examined in the context of autocorrection systems, system self-assessment has been studied in other domains. One of the most com-

[1] http://aspell.net/

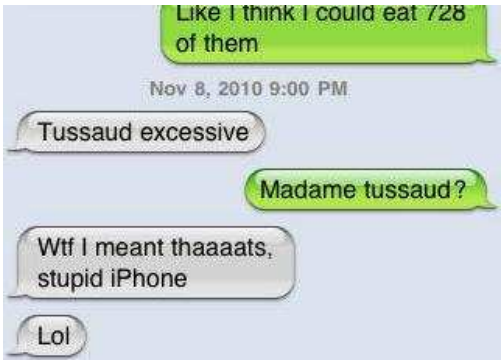

Figure 1: Example text message dialogue from our corpus with an automatic correction mistake

mon application domains is spoken dialogue systems (Levow, 1998; Hirschberg et al., 2001; Litman et al., 2006), where detecting problematic situations can help the system better adapt to user behavior. These systems often make use of prosody and task specific dialogue acts, two feature sources unavailable in general text message dialogues.

In summary, while a large body of work addresses similar problems, to our knowledge no previous work has looked into the aspect of self-assessment of autocorrection based on dialogues between text message users. The work presented in this paper represents a first step in this direction.

3 Data Set

To support our investigation, we collected a corpus of data containing true experiences with autocorrection provided by text message users. The website "Damn You Auto Correct"[2] (DYAC) posts screenshots of text message conversations that contain mistakes caused by phone automatic correction systems, as sent in by cellphone users. An example screenshot is shown is Figure 1.

Speech bubbles originating from the left of the image in Figure 1 are messages sent by one dialogue participant while those originating from the right of the image are sent by the other. In this example, the automatic correction system incorrectly decides that the user's attempt to write the non-standard word form *thaaaats* was an attempt to write the word *Tussaud*. This confuses the reader, and several dialogue turns are used to resolve the confusion. The author explicitly corrects her mistake by writing "I meant thaaaats".

Note that, in this example, the word *Tussaud* could be an autocompletion or an autocorrection by the system. However, there may be no significant distinction between these two operations from a user's point of view. These two operations could also take place at the same time. For instance, a system may both suggest possible completions after the user has only typed a small number of characters and perform autocorrection once the user presses the space bar to go on to the next word. Therefore, for the purposes of our discussion here, we use autocorrection to refer to any changes made by the system (either by autocompletion or autocorrection) without the user explicitly selecting the correction themselves.

Throughout the paper, we use the term **attempted correction** to refer to any autocorrection made by the system; for example, *Tussaud* is an attempted correction in Figure 1. Some attempted corrections could correct to the word that the user intended, which will be referred to as **unproblematic corrections** or **non-erroneous corrections**. Other attempted corrections may mistakenly choose a word that the user did not intend to write, which will be referred to as **correction mistakes** or **erroneous corrections**. For example, *Tussaud* is an erroneous correction. We use the term **intended word** to refer to the term that the user was attempting to type when the autocorrection system intervened. For instance, in the erroneous correction in Figure 1, the intended term was *thaaaats*.

To build our dataset, screenshots were extracted from the site and transcribed, and correction mistakes were annotated with their intended words, if the intended word appeared in the dialogue. Because the website presents autocorrection mistakes that submitters find to be humorous or egregious, there may be an incentive for users to submit falsified instances. To combat this, we performed an initial filtering phase to remove instances that were unlikely to have been produced by a typical autocorrection system (e.g., instances that substituted letters that were far from each other on the keyboard and not phonetically similar) or that were otherwise believed to be falsified. Using this methodology we compiled a development set of 300 dialogues and an

[2] www.damnyouautocorrect.com

Figure 2: Text message dialogue with several correction mistakes for the same intended term.

additional 635 dialogues for evaluation.

Some dialogues contained several correction mistakes. It was common for multiple correction mistakes to be produced in an attempt at typing a single word; an example is shown in Figure 2, in which the intended term *cookies* is erroneously corrected at first as *commies* and then as *cockles*.

We will use the term *message* to refer to one SMS text message sent in the course of the conversation, while a *turn* encompasses all messages sent by a user between messages from the other participant. For instance, the first 3 speech bubbles in Figure 2 all represent separate messages, but they are all part of the same turn.

While this dataset provides us with instances of autocorrection mistakes, in order to differentiate between problematic and unproblematic correction attempts we will need a dataset of unproblematic attempts as well. It should be noted that, from the perspective of the reader, a successful autocorrection attempt is equivalent to the user typing correctly without any intervention from the system at all. To build a dataset of unproblematic instances, we collected text message conversations from pairs of users without the aid of autocorrection. Users were then asked to correct any mistakes they produced. Snippets of these conversations that did not contain mistakes were then extracted to act as a set of unproblematic autocorrection instances. In total 554 snippets were extracted. These snippets were combined with the problematic instances from the DYAC data to make the final dataset used for training and evaluation.

4 Autocorrection Self-Assessment

It is desirable for an autocorrection system to have the capability to assess its own performance. For each correction attempt it makes, if the system can evaluate its performance based on the dialogue it can acquire valuable information to learn from its own mistakes and thus improve its performance for future operations. Next we describe how we formulate the task of self-assessment and what features can be used for this task.

Because each correction attempt is system generated, an autocorrection system should have knowledge of all correction attempts it has made. Let C be the set of all correction attempts performed by an autocorrection system over the course of a dialogue and let W be the set of all words in this dialogue which occur after the correction attempt. We model this problem as two distinct subtasks: 1) identify attempted corrections $c_i \in C$ which are erroneous (if there are any), and 2) for each erroneous correction c_i, identify a word $w_j \in W$ which is the intended word for c_i (i.e., $Intended(c_i) = w_j$).

4.1 Identifying Erroneous Corrections

The first task involves a simple binary decision; given an arbitrarily sized dialogue snippet containing an automatic correction attempt, we must decide whether or not the system acted erroneously when making the correction. We thus model the task as a binary classification problem in which we classify every correction attempt $c \in C$ as either erroneous or non-erroneous.

The proposed method follows a standard procedure for supervised binary classification. First we must build a set of labeled training data in which each instance is represented as a vector of features and a ground truth class label. Given this, we can train a classifier to differentiate between the two classes. For the purposes of this work we use a support vector machine (SVM) classifier.

4.1.1 Feature Set

In order to detect problematic corrections, we must identify dialogue behaviors that signify an error has occurred. We examined the dialogues in our development set to understand which dialogue behaviors are indicative of autocorrection mistakes. While in unproblematic dialogues users are able to converse freely, in problematic dialogues users must spend dialogue turns reestablishing common ground (Clark, 1996). Our feature set will focus on two common ways these attempts to establish common

ground manifest themselves: as confusion and as attempts to correct the mistake.

Confusion Detection Features. Because autocorrection mistakes often result in misleading or semantically vacuous utterances, they are apt to confuse the reader, who will often express this confusion in the dialogue in order to gain clarification. These features examine the dialogue of the uncorrected user (the dialogue participant that reads the automatic correction mistake, not the one that was automatically corrected). One sign of confusion is the use of the question mark, so one feature captured the presence of question marks in the messages sent by the uncorrected user. Similarly, users may often use a block or repeated punctuation of show suprise or confusion, so another feature detected instances of repeated question marks and exclamation points (*???*, *!?!!*, etc.). When confused, readers will often retype the confusing word as a request for clarification (e.g., *Tussaud?*), or simply type "what?". We therefore include features that detect whether or not the corrected term appears in the first message sent by the uncorrected user after the correction mistake has occurred, and whether or not this message contains the word "what" as its own clause.

Clarification Detection Features. In contrast to utterances of confusion which are generally produced by the reader of the autocorrection mistake, clarification attempts are usually initiated by the user that was corrected. Several methods are used to indicate that the term shown by the system was incorrect. One convention is to use an asterisk (*) either before or after the corrected term:

A_1: Indeed Sid

A_2: Sir*

Another common method is to explicitly state what was intended using phrases such as "I meant to type", "that was supposed to say", etc. We included several features to capture these word patterns. Another method is to simply quickly reply with the word that was intended, so we included a feature to record whether the next message after the correction attempt contains only a single word. As users often feel the need to explain why the mistake occurred, we included a feature that recorded any mention or autocompletion, autocorrection or spell checking. One additional feature recorded whether or not the corrected user's dialogue contained words written in all capital letters.

Features	Precision	Recall	F-Measure
All Features	.861	.751	.803
-Confusion	.857	.725	.786
-Clarification	.848	.676	.752
-Dialogue	.896	.546	.679
Baseline	.568	1	.724

Table 1: Feature ablation results for identifying autocorrection mistakes

Dialogue Features. A few features captured information more closely tied to the flow of the dialogue than to confusion or clarification. In our development set, we observed a few common dialogue formats. In one, a correction mistake is immediately followed by confusion, which is then immediately followed by clarification. The dialogue in Figure 1 gives an example of this. To capture this form, we included a feature that recorded whether a confusion feature was present in the message immediately following the correction attempt and whether a clarification feature was present in the message immediate following the confusion message. Similarly, clarification attempts are often tried immediately after the mistake even if no confusion was present, so an additional feature captured whether the first message after the mistake by the corrected user was a clarification attempt. Additionally, we observed that autocorrection mistakes frequently appeared in the last word in a message, which was recorded by another binary feature. Finally, we recorded a count of how often the corrected term appeared in the dialogue.

4.1.2 Evaluation

To build our classifier we used the SVMLight[3] implementation of a support vector machine classifier with an RBF kernel. To ensure validity and account for the relatively small size of our dataset, evaluation was done via leave-one-out cross validation.

Results are shown in Table 1. A majority class baseline is given for comparison. As shown, using the entire feature set, the classifier achieves above baseline precision of 0.861, while still producing recall of 0.751.

[3]Version 6.02, http://svmlight.joachims.org/

Although F-measure is reported, it is unlikely that precision and recall should be weighted equally. Because one of the primary reasons we may wish to detect problematic situations is to automatically collect data to improve future performance by the autocorrection system, it is imperative that the data collected have high precision in order to reduce the amount of noise present in the collected dataset. Conversely, because problematic situation detection can monitor a user's input continuously for an indefinite period of time in order to collect more data, recall is less of a concern.

To study the effect of each feature source, we performed a feature ablation study, the results of which are included in Table 1. For each run, one feature type was removed and the model was retrained and reassessed. As shown, removing any feature source has a relatively small effect on the precision but a more substantial effect on the recall. Confusion detection features seem to be the least essential, causing a comparatively small drop in precision and recall values when removed. Removing the dialogue features results in the greatest drop in recall, returning only slightly above half of the problematic instances. However, as a result, the precision of the classifier is higher than when all features are used.

4.2 Identifying The Intended Term

Note that one purpose of the proposed self-assessment is to collect information online and thus make it possible to build better models. In order to do so, we need to know not only whether the system acted erroneously, but also what it should have done.Therefore, once we have extracted a set of problematic instances (and their corresponding dialogues), we must identify the term which the user was attempting to type when the system intervened. First, assume that via the classification task described in Section 4.1 we have identified a set of erroneous correction attempts, EC. Now the problem becomes, for every erroneous correction $c \in EC$, identify $w \in W$ such that $w = Intended(c)$. We model this as a ranking task, in which all $w \in W$ are ranked by their likelihood of being the intended term for c. We then predict that the top ranked word is the true intended term.

4.2.1 Feature Set

To support the above processing, we explored a diverse feature set, consisting of five different feature sources: contextual, punctuation, word form, similarity, and pattern features, crafted from an examination of our development data. Several of the features are related to those used in the initial classification phase. However, unlike our classification features, these feature focus on the relationship between the erroneous correction c and a candidate intended term w.

Contextual Features. Contextual features capture relevant phenomena at the discourse level. After an error is discovered by a user, they may type an intended term several times or type it in a message by itself in order to draw attention to it. These phenomena are captured in the *word repetition* and *only word* features. Another common discourse related correction technique is to retype some of the original context, which is captured by the *word overlap* feature. The *same author* feature indicates whether c and w are written by the same author. The author of the original mistake is likely the one to correct it, as they know their true intent.

Punctuation Features. Punctuation is occasionally used by text message writers to signal a correction of an earlier mistake, as noted previously. We included features to capture the presence of several different punctuation marks occurring before or after a candidate word such as *,?,!, etc. Each punctuation mark is represented by a separate feature.

Word Form Features. Word form features capture variations in how a word is written. One word form feature captures whether a word was typed in all capital letters, a technique used by text message writers to add emphasis. Two word form features were designed to capture words that were potentially unknown to the system, out-of-vocabulary words and words with letter repetition (e.g., "yaaay"). Because the system does not know these words, it will consider them misspellings and may attempt to change them to an in-vocabulary term.

Similarity Features. Our similarity feature captured the character level distance between a word changed by the system and a candidate intended word. We calculated the normalized levenshtein edit distance between the two words as a measure of sim-

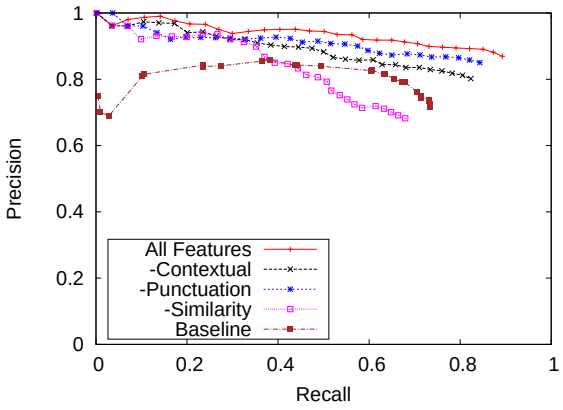

Figure 3: Precision-recall curve for intended term selection, including feature ablation results

ilarity.

Pattern Features. Pattern features attempt to capture phrases that are used to explicitly state a correction. These include phrases such as "(I) meant to write w", "(that was) supposed to say w", "(that) should have read w", "(I) wrote w", etc.

4.2.2 Evaluation

To find the most likely intended term for a correction mistake, we rank every candidate word in W and predict that the top ranked word is the intended term. We used the ranking mode of SVMlight to train our ranker. By thresholding our results to only trust predictions in which the ranker reported a high ranking value for the top term, we were able to examine the precision at different recall levels. That is, if the top ranked term does not meet the threshold, we simply do not predict an intended term for that instance, hurting recall but hopefully improving precision by removing instances that we are not confident about. This thresholding process may also allow the ranker to exclude instances in which the intended term does not appear in the dialogue, which are hopefully ranked lower than other cases. As before, evaluation was done via leave-one-out cross validation.

Results are shown in Figure 3. As a method of comparison we report a baseline that selects the word with the smallest edit distance as the intended term. As shown, using the entire feature set results in consistently above baseline performance.

As before, we are more concerned with the precision of our predictions than the recall. It is difficult to assess the appropriate precision-recall tradeoff without an in-depth study of autocorrection usage by text messagers. However, a few observations can be made from the precision-recall curve. Most critically, we can observe that the model is able to predict the intended term for an erroneous correction with high precision. Additionally, the precision stays relatively stable as recall increases, suffering a comparatively small drop in precision for an increase in recall. At its highest achieved recall values of 0.892, it maintains high precision at 0.869.

Feature ablation results are also reported in Figure 3. The most critical feature source was word similarity; without the similarity feature the performance is consistently worse than all other runs, even falling below baseline performance at high recall levels. This is not suprising, as the system's incorrect guess must be at least reasonably similar to the intended term, or the system would be unlikely to make this mistake. Although not as substantial as the similarity feature, the contextual and punctuation features were also shown to have a significant effect on overall performance. Conversely, removing word form or pattern features did not cause a significant change in performance (not shown in Figure 3 to enhance readability).

5 An End-To-End System

In order to see the actual effect of the full system, we ran it end-to-end, with the output of the initial erroneous correction identification phase used as input when identifying the intended term. Results are shown in Figure 4. The results of the intended term classification task on gold standard data from Figure 3 are shown as an upper bound.

As expected, the full end-to-end system produced lower overall performance than running the tasks in isolation. The end-to-end system can reach a recall level of 0.674, significantly lower than the recall of the ground truth system. However, the system still peaks at precision of 1, and was able to produce precision values that were competitive with the ground truth system at lower recall levels, maintaining a precision of above 0.90 until recall reached 0.396.

It is worth mentioning that the current evalua-

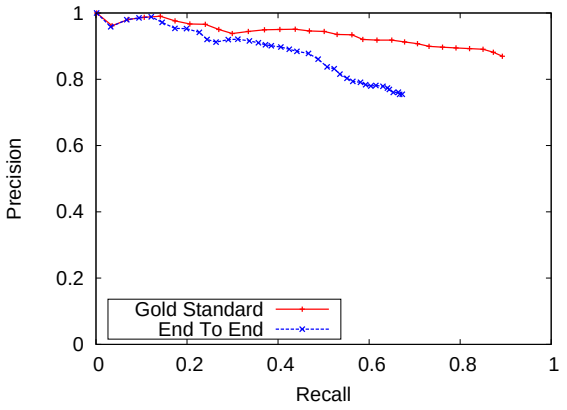

Figure 4: Precision-recall curve for the end-to-end system

tion is based on a balanced dataset with roughly even numbers of problematic and unproblematic instances. It is likely that in a realistic setting an autocorrection system will get many more instances correct than wrong, leading to a data distribution skewed in favor of unproblematic instances. This suggests that the evaluation given here may overestimate the performance of a self-assessment system in a real scenario. Although the size of our dataset is insufficient to do a full analysis on skewed data, we can get a rough estimate of the performance by simply counting false positives and false negatives unevenly. For instance, if the cost of mispredicting a unproblematic case as problematic is nine times more severe than the cost of missing a problematic case, this can give us an estimate of the performance of the system on a dataset with a 90-10 skew.

We examined the 90-10 skew case to see if the procedure outlined here was still viable. Results of an end-to-end system with this data skew are consistently lower than the balanced data case. The skewed data system can keep performance of 90% or better until it reaches 13% recall, and 85% or better until it reaches 22%. These results suggest that the system could still potentially be utilized. However, its performance drops off steadily, to the point where it would be unlikely to be useful at higher recall levels. We leave the full exploration of this to future work, which can utilize larger data sets to get a more accurate understanding of the performance.

6 Discussion

When an autocorrection system attempts a correction, it has perfect knowledge of the behavior of both itself and the user. It knows the button presses the user used to enter the term. It knows the term it chose as a correction. It knows the surrounding context; it has access to both the messages sent and received by the user. It has a large amount of the information it could use to improve its own performance, if only it were able to know when it made a mistake. The techniques described here attempt to address this critical system assessment step. Users may vary in the speed and accuracy at which they type, and input on small or virtual keyboards may vary between users based on the size and shape of their fingers. The self-assessment task described here can potentially facilitate the development of autocorrection models that are tailored to specific user behaviors.

Here is a brief outline of how our self-assessment module might potentially be used in building user-specific correction models. As a user types input, the system performs autocorrection by starting with a general model (e.g., for all text message users). Each time a correction is performed, the system examines the surrounding context to determine whether the correction it chose was actually what the user had intended to type. Over the course of several dialogues, the system builds a corpus of erroneous and non-erroneous correction attempts. This corpus is then used to train a user-specific correction model that is targeted toward system mistakes that are most frequent with this user's input behavior. The user-specific model is then applied on future correction attempts to improve overall performance. This monitoring process can be continued for months or even longer. The results from self-assessment will allow the system to continuously and autonomously improve itself for a given user (Baldwin and Chai, 2012).

In order to learn a user-specific model that is capable of improving performance, it is important that the self-assessment system provides it with training data without a large amount of noise. This suggests that the self-assessment system must be able to identify erroneous instances with high precision. Conversely, because the system can monitor user behav-

ior indefinitely to collect more data, the overall recall may not be as critical. It might then be reasonable for a self-assessment system to be built to focus on collecting high accuracy pairs, even if it misses many system mistakes. Although a full examination of this tradeoff is left for future work which may more closely examine user input behavior, we feel that the results presented here show promise for collecting accurate data in a timely manner.

7 Conclusions and Future Work

This paper describes a novel problem of assessing its own correction performance for an autocorrection system based on dialogue between two text messaging users. Our evaluation results indicate that given a problematic situation caused by an autocorrection system, the discourse between users provides important cues for the system to automatically assess its own correction performance. By exploring a rich set of features from the discourse, our proposed approach is able to both differentiate between problematic and unproblematic instances and identify the term the user intended to type with high precision, achieving significantly above baseline performance. As discussed in Section 6, this self-assessment task can potentially be important for building user-specific autocorrection models to improve auto-correction performance.

The results presented in this paper represent a first look at autocorrection self-assessment. There are several areas of future work. There is certainly a need to examine additional feature sources. Because automatic correction mistakes can potentially create semantically vacuous utterances, a computational semantics based approach, similar to those used in semantic autocompletion systems (Hyvnen and Mkel, 2006), may prove fruitful. Additionally, although this work focused solely on dialogue-related features, future work may wish to take a closer look at the autocorrection mistakes themselves (e.g., which words are most likely to be mistakenly corrected, etc.). Lastly, although our current work demonstrated some potential, more thorough evaluation in realistic settings will allow a more full understanding of the impact and limitations of the proposed self-assessment approach.

Acknowledgments

This work was supported in part by Award #0957039 from the National Science Foundation and Award #N00014-11-1-0410 from the Office of Naval Research. The authors would like to thank the reviewers for their valuable comments and suggestions.

References

AiTi Aw, Min Zhang, Juan Xiao, and Jian Su. 2006. A phrase-based statistical model for sms text normalization. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 33–40, Morristown, NJ, USA. Association for Computational Linguistics.

Tyler Baldwin and Joyce Chai. 2012. Towards online adaptation and personalization of key-target resizing for mobile devices. In *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*, IUI '12, pages 11–20, New York, NY, USA. ACM.

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

Surajit Chaudhuri and Raghav Kaushik. 2009. Extending autocompletion to tolerate errors. In *Proceedings of the 35th SIGMOD international conference on Management of data*, SIGMOD '09, pages 707–718, New York, NY, USA. ACM.

Monojit Choudhury, Rahul Saraf, Vijit Jain, Animesh Mukherjee, Sudeshna Sarkar, and Anupam Basu. 2007. Investigation and modeling of the structure of texting language. *Int. J. Doc. Anal. Recognit.*, 10(3):157–174.

Herbert H. Clark. 1996. *Using Language*. Cambridge University Press.

J.J. Darragh, I.H. Witten, and M.L. James. 1990. The reactive keyboard: a predictive typing aid. *Computer*, 23(11):41 –49, November.

Mark Dunlop and Andrew Crossan. 2000. Predictive text entry methods for mobile phones. *Personal and Ubiquitous Computing*, 4:134–143. 10.1007/BF01324120.

Julia Hirschberg, Diane J. Litman, and Marc Swerts. 2001. Identifying user corrections automatically in spoken dialogue systems. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics*.

Yijue How and Min yen Kan. 2005. Optimizing predictive text entry for short message service on mobile

phones. In *in Human Computer Interfaces International (HCII 05). 2005: Las Vegas*.

Eero Hyvnen and Eetu Mkel. 2006. Semantic autocompletion. In *Proceedings of the first Asia Semantic Web Conference (ASWC 2006*, pages 4–9. Springer-Verlag.

Mark D. Kernighan, Kenneth W. Church, and William A. Gale. 1990. A spelling correction program based on a noisy channel model. In *Proceedings of the 13th conference on Computational linguistics*, pages 205–210, Morristown, NJ, USA. Association for Computational Linguistics.

Catherine Kobus, François Yvon, and Géraldine Damnati. 2008. Normalizing SMS: are two metaphors better than one ? In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 441–448, Manchester, UK, August. Coling 2008 Organizing Committee.

Gina-Anne Levow. 1998. Characterizing and recognizing spoken corrections in human-computer dialogue. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 736–742, Montreal, Quebec, Canada, August. Association for Computational Linguistics.

Diane Litman, Julia Hirschberg, and Marc Swerts. 2006. Characterizing and predicting corrections in spoken dialogue systems. *Comput. Linguist.*, 32:417–438, September.

I. Scott MacKenzie and Kumiko Tanaka-Ishii. 2007. *Text Entry Systems: Mobility, Accessibility, Universality (Morgan Kaufmann Series in Interactive Technologies)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

I. Scott MacKenzie. 2002. Kspc (keystrokes per character) as a characteristic of text entry techniques. In *Proceedings of the 4th International Symposium on Mobile Human-Computer Interaction*, Mobile HCI '02, pages 195–210, London, UK. Springer-Verlag.

Tim Paek, Kenghao Chang, Itai Almog, Eric Badger, and Tirthankar Sengupta. 2010. A practical examination of multimodal feedback and guidance signals for mobile touchscreen keyboards. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, MobileHCI '10, pages 365–368, New York, NY, USA. ACM.

Kristina Toutanova and Robert Moore. 2002. Pronunciation modeling for improved spelling correction. In *40th Annual Meeting of the Association for Computational Linguistics(ACL 2002)*.

Casey Whitelaw, Ben Hutchinson, Grace Y Chung, and Ged Ellis. 2009. Using the Web for language independent spellchecking and autocorrection. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 890–899, Singapore, August. Association for Computational Linguistics.