Proofreading Human Translations with an E-pen

Vicent Alabau and **Luis A. Leiva**
PRHLT Research Center
Universitat Politècnica de València
{valabau,luileito}@prhlt.upv.es

Abstract

Proofreading translated text is a task aimed at checking for correctness, consistency, and appropriate writing style. While this has been typically done with a keyboard and a mouse, pen-based devices set an opportunity for making such corrections in a comfortable way, as if proofreading on physical paper. Arguably, this way of interacting with a computer is very appropriate when a small number of modifications are required to achieve high-quality standards. In this paper, we propose a taxonomy of pen gestures that is tailored to machine translation review tasks, after human translator intervention. In addition, we evaluate the recognition accuracy of these gestures using a couple of popular gesture recognizers. Finally, we comment on open challenges and limitations, and discuss possible avenues for future work.

1 Introduction

Currently, the workflow of many translation agencies include a final reviewing or proofreading process[1] where the translators' work is checked for correctness, consistency and appropriate writing style. If the translation quality is good enough, only a small amount of changes would be necessary to reach a high-quality result. However, the required corrections are often spread sparingly and unequally among the screen, which renders mouse/keyboard interaction both inefficient and unappealing.

As a result of the popularization of touch-screen and pen-based devices, text-editing applications can be operated today in a similar way people interact with pen and paper. This way of reviewing is arguably more natural and efficient than a keyboard or a mouse, since the e-pen can be used both to locate and correct an erroneous word, all at once. Additionally, the expressiveness of e-pen interaction provides an opportunity to integrate useful gestures that are able correct other common mistakes, such as word reordering or capitalization.

2 Related Work

The first attempt that we are aware of to post-edit text with an e-pen interface dates back to the early seventies of the past century (Coleman, 1969). In that work, Coleman proposed a set of unistroke gestures for post-editing. Later on, the same corpus was used by (Rubine, 1991) in his seminal work about gesture recognition with excellent recognition results. However, the gesture set is too simplistic to be used in a real translation task today.

Most of the modern applications to generate and edit textual content using "digital ink" are based on *ad-hoc* interaction protocols[2] and often do not ship handwriting recognition software. To our knowledge, *MyScript Notes Mobile*[3] is the closest system to provide a natural onscreen paper-like interaction style, including some text-editing gestures and a powerful handwriting recognition software. However, this application relies on spatial relations of the ink strokes to perform handwriting recog-

[1]The reviewing process can be seen as a detailed proofreading process where the target sentence is also compared against the source sentence for errors such as mistranslations, etc. However, for the purpose of this paper, we can use the terms reviewing and proofreading indistinguishably.

[2]http://appadvice.com/appguides/show/handwriting-apps-for-ipad

[3]http://www.visionobjects.com

nition. For instance, to insert a new word in the middle of a sentence the user needs to make room for space explicitly (i.e., if the word has N characters, the user needs to perform an *Insert Space* gesture N times). Moreover, the produced text does not flow on the UI, i.e., it is fixed to the position of the ink, which makes it difficult to modify. As a result, this system does not seem suitable for reviewing translations. Other comparable work is MinGestures (Leiva et al., 2013), which proposes a simplified set of gestures for interactive text post-editing. Although MinGestures is very efficient and accurate, it is also very limited in expressiveness. Only basic edition capabilities are allowed (insertion, deletion, and substitution). Thus, advanced e-pen gestures cannot be used to improve the efficiency of the reviewer.

On the other hand, there are applications for post-editing text where user interactions are leveraged to propagate text corrections to the rest of the sentence. CueTIP (Shilman et al., 2006), CATTI (Romero et al., 2009) and IMT (Alabau et al., 2014) are the most advanced representatives of this kind of applications. These systems allow the user to correct text either in the form of unconstrained cursive handwriting or (limited) pen gestures. Then, the corrections are leveraged by the system to provide smart auto-completion capabilities. This way, user interaction is not only taken into account to amend the proposed correction but other mistakes in the surrounding text are automatically amended as well. However, user interaction is limited in these cases. In CueTIP, only one handwritten character can be submitted at a time and only 4 gestures can be performed (join, split, delete, and substitution). In CATTI, the user can handwrite text freely but is still limited to perform 4 gestures as well (substitute, insert, delete, and reject). Finally, IMT does not support gestures other than substitution. Although the auto-completion capability is a very interesting and promising topic, it should not be considered for reviewing: given the locality of the small amount of changes that are probably needed, auto-completion can make more harm than good.

Thus, in light of the current limitations of state-of-the-art approaches, in this work we present an exploratory research of how paper-like interaction should be approached to allow proofreading translated texts.

3 A Taxonomy of Proofreading Gestures

Indicating text modifications on a sheet of paper can be made in many different ways. However, the lack of a consensus may lead to misinterpretations. Fortunately, a series of authoritative proofreading and copy-editing symbols have been proposed (AMA, 2007; CMO, 2010), even leading to an eventual standardization (BS, 2005; ISO, 1983).

We have studied the aforementioned authoritative sources and have found that there is a huge overlap in the proposed symbols, with only minor variations. Moreover, such symbols are meant to ease human-human communication and therefore we need to adapt them to ease human-computer communication. This way, we will focus on those symbols that could be used to review using stroke-based gestures. As such, we will study gestures that allow to change the *content* and not the *formatting* of the text. We can define the following high-level operations; see Figure 1:

Word change: change text's written form.
Letter case: change word/character casing.
Punctuation: insert punctuation symbols.
Word combination: separate or join words.
Selection: select words or characters.
Text displacement: move text around.

It is worth noting that punctuation symbols are represented explicitly in the literature, probably because of their importance in copy-editing tasks. In addition, dot and hyphen symbols are represented differently from other insertion symbols. The purpose of this convention is to reduce visual ambiguity in human recognition. Finally, the selection operation is often devoted to spell out numbers or abbreviations.

4 Preliminary Evaluation

The initial taxonomy (Figure 1) aims to be a complete set of symbols for proofreading and copy-editing onscreen. Nonetheless, the success of these gestures will depend on the accu-

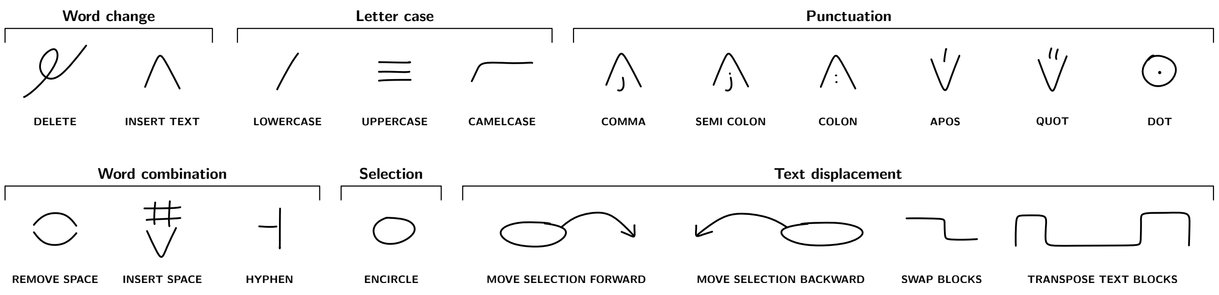
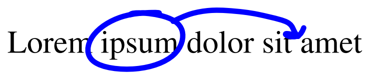


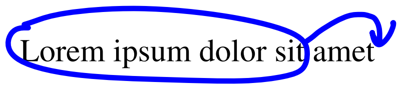
Figure 1: Initial taxonomy, based on *de facto* proofreading symbols.

racy of gesture recognizers, to correctly translate gestures into commands.

As a first approach, we wanted to evaluate these symbols with state-of-the-art gesture recognizers. The initial taxonomy differs significantly from other gesture sets in the literature (Anthony and Wobbrock, 2012; Vatavu et al., 2012), in the sense that the symbols we are researching are not expected to be drawn in isolation. Instead, reviewers will issue a gesture in a very specific context, and so a proofreading symbol may change its meaning. This is specially true for symbols involving multiple spans of text or block displacements: depending of the size of the span or the length of the displacement, the aspect ratio and proportions among the different parts of the gesture strokes may vary. Thus, the final shape of the gesture can be significantly different. An example is given in Figure 2.



(a) Move forward with 1 selected word and 2 word displacement.



(b) Move forward with 4 selected words and 1 word displacement.

Figure 2: Examples of the same gesture executed with different proportions. As a result, the shapes of both gestures significantly diverge from each other.

4.1 Gesture Samples Acquisition

We carried out a controlled study in a real-world setup. We developed an application that requested a set of random challenges to the users (Figure 3). Then, we asked the users if they would prefer to do the acquisition on a digitizer tablet or on a tablet computer. On a 1 to 5 point scale, with 1 meaning 'I prefer writing with a digitizer pen' and 5 'I prefer writing with a pen-capable tablet', users indicated that they would prefer a tablet computer (M=4.6, SD=0.8). Consequently, we deployed the application into a Lenovo ThinkPad tablet, which had to be operated with an e-pen. To make the paper-like experience more realistic, the touchscreen functionality was disabled, so that users could rest their hands on the screen. Eventually, 12 users aged 24–36 submitted 5 times each gesture following the aforementioned random challenges.

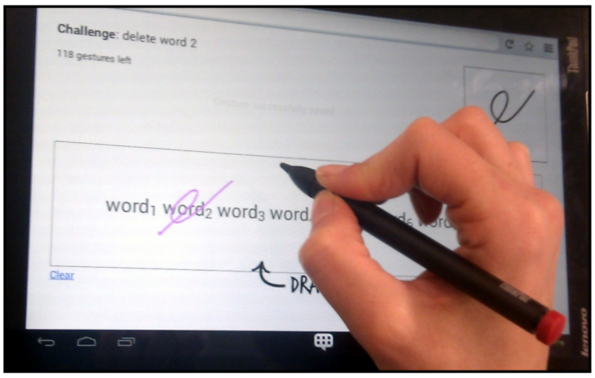


Figure 3: Acquisition application.

4.2 The Family of $ Recognizers

In HCI, there is a popular "dollar series" of template-matching gesture recognizers, using a nearest-neighbor classifier with scoring functions based on Euclidean distance. The $ recognizers present several advantages over other classifiers based on more complex pattern recognition algorithms. First, $ recognizers are easily understandable and fast to integrate or re-implement in different programming languages. Second, they do not depend on large amounts of training data to achieve high accuracy, just on a small number of predefined templates.

In particular, \$N (Anthony and Wobbrock, 2012) and \$P (Vatavu et al., 2012) can be used to recognize multi-stroke gestures, so they were the only suitable candidates to recognize our initial gesture taxonomy. On the one hand, \$N deals with multiple strokes by recombining in every possible way the strokes of the templates in order to generate new instances of unistroke templates, and then apply either the \$1 recognizer (Wobbrock et al., 2007) or Protractor (Li, 2010). On the other hand, \$P considers gesture strokes as a cloud of points, removing thus information about stroke sequentiality. Then, the best match is found using an approximation of the Hungarian algorithm, which pairs points from the template with points of the query gesture.

4.3 Results

We evaluated three fundamental aspects of the recognition process: accuracy, recognition time and memory requirements to store the whole set of templates. Aiming for a portable recognizer that could work on most everyday devices, we decided to use a JavaScript (rotation invariant) version of the \$ family recognizers. Experiments were executed as a `nodejs` program on a Ubuntu Linux computer with a 2.83 GHz Intel QuadCore^{TM}and 4 GB of RAM. We followed a leaving-one-out (LOO) setup, i.e., each user's set of gestures was used as templates and tested against the rest of the user's gestures. All the values show the average of the different LOO runs.

Table 1 summarizes the experimental results. For the \$N recognizer we found that, by resampling to 32 points and 5 templates, we can achieve very good recognition times (0.7ms in average) but high recognition error rate (23.6%). On the other hand, the \$P recognizer behaves even worse, with 27.1% error rate. Memory requirements are marginal but recognition times increase more than one order of magnitude.

It must be noted that the space needed by \$N to store just one template of n strokes is $n! \times 2^n$ times the space for the original template (Vatavu et al., 2012). This is actually a huge waste of resources. For instance, one template of the INSERT SPACE gesture requires 3840 times the original size, assuming that the user has introduced the minimum strokes required. With a resampling 8 points, \$N needs almost 33MB of RAM to store 5 templates per gesture.

Recognizer	**Error**	**Time**	**Mem. usage**
\$N	23.6%	0.7 ms	102 MB
\$P	27.1%	45 ms	1.8 MB

Table 1: Results for \$N and \$P recognizers, with gestures resampled to 32 points and using 5 templates per gesture.

4.4 Error analysis

Surprised by the high error rates we decided to delve into the results of the most accurate setup so we could find the source of errors. We observed that the most difficult gesture to recognize was REMOVE SPACE, which represented 12% of the total number of errors; being confused with COMMA and SEMI COLON more than 50% of the time, probably because they are formed by two arcs. It was also confused, though less frequently, with MOVE SELECTION FORWARD/BACKWARD. These gestures, excepting the circle part, are also composed by two arcs.

On the other hand, punctuation symbols accounted for 37% of the errors, being mostly confused with each other, as they have very similar shapes. Finally, some errors are harder to dissect. For instance, UPPERCASE was confused mainly with both MOVE SELECTION (4.4% of the errors), and punctuation and displacement operations were also confused with each other at some time, despite their very different visual shapes and sizes. We suspect it is because of the internal normalization procedures of the \$ recognizers.

5 Discussion

Our results suggest that the \$ family of gesture recognizers, although popular, are not appropriate for proofreading translated texts. Our assumption is that the normalization procedures of these recognizers—mainly scaling and resampling— are not appropriate to gestures for which the proportions of its constituent parts may vary according to the context. For

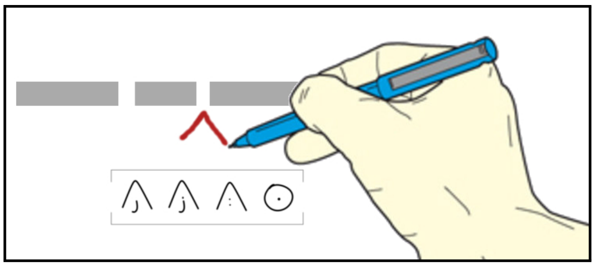


Figure 4: One proposal for gesture set simplification. A Pop-up menu could assist the user to disambiguate among perceptually similar gestures.

example, after resizing a MOVE SELECTION FORWARD that selects a small word and has a long arrow, the final shape would be primarily that of the arrow (Figure 2).

In the light of this analysis, several actions can be taken for future work. Firstly, other gesture recognizers should be explored that can deal with stroke sequences without resampling (Myers and Rabiner, 1981; Sezgin and Davis, 2005; Álvaro et al., 2013). However, it must be remarked that response time is crucial to ensure an adequate user experience. Therefore, the underlying algorithms should be implementable on thin clients, such as mobile devices, with reasonable recognition times.

Secondly, it would be also necessary to reduce the set of gestures, but not at the expense of reducing also expressiveness as Leiva et al. (2013) did. For instance, taking advantage of the interaction that computers can provide, we can group punctuation operations, SPACE, and INSERT HYPHEN all into INSERT ABOVE and BELOW gestures. Both gestures would pop-up a menu where the user could select deterministically the symbol to insert; see Figure 4. In the same manner, letter casing operations could be grouped into a single SELECTION category, which would also provide a contextual menu to trigger the right command. The resulting set of gestures should be, in principle, much easier to recognize.

Additionally, the current set of proofreading gestures present further challenges. For instance, we would need to identify the *semantics* of the gestures, i.e., which elements in the text are affected by the gesture and how the system should proceed to accomplish the task.

6 Conclusions

In this work we have defined a set of gestures that is suitable for the reviewing process of human-translated text. We have performed an evaluation on gestures generated by real users that show that popular recognizers are not able to achieve a satisfactory accuracy. In consequence, we have identified a series of areas for improvement that could make e-pen devices realizable in the near future.

7 Acknowledgments

This work is supported by the 7th Framework Program of European Commission under grant agreements 287576 (CasMaCat) and 600707 (tranScriptorium).

References

V. Alabau, A. Sanchis, and F. Casacuberta. 2014. Improving on-line handwritten recognition in interactive machine translation. *Pattern Recognition*, 47(3):1217–1228.

2007. AMA manual of style: A guide for authors and editors. 10th ed. Oxford University Press.

L. Anthony and J. O. Wobbrock. 2012. $N-protractor: a fast and accurate multistroke recognizer. In *Proc. GI*, pages 117–120.

2005. BS 5261-2:2005. Copy preparation and proof correction.

2010. The Chicago manual of style. 16th ed. University Of Chicago Press.

M. L. Coleman. 1969. Text editing on a graphic display device using hand-drawn proofreader's symbols. In *Pertinent Concepts in Computer Graphics, Proc. 2nd Univ. Illinois Conf. on Computer Graphics*, pages 283–290.

1983. ISO 5776:1983. Symbols for text correction.

L. A. Leiva, V. Alabau, and E. Vidal. 2013. Error-proof, high-performance, and context-aware gestures for interactive text edition. In *Proc. CHI EA*, pages 1227–1232.

Y. Li. 2010. Protractor: a fast and accurate gesture recognizer. In *Proc. CHI*, pages 2169–2172.

C. S. Myers and L. R. Rabiner. 1981. A comparative study of several dynamic time-warping algorithms for connected-word. *Bell System Technical Journal*.

V. Romero, L. A. Leiva, A. H. Toselli, and E. Vidal. 2009. Interactive multimodal transcription of text images using a web-based demo system. In *Proc. IUI*, pages 477–478.

D. Rubine. 1991. Specifying gestures by example. In *Proc. SIGGRAPH*, pages 329–337.

T. M. Sezgin and R. Davis. 2005. HMM-based efficient sketch recognition. In *Proc. IUI*, pages 281–283.

M. Shilman, D. S. Tan, and P. Simard. 2006. CueTIP: a mixed-initiative interface for correcting handwriting errors. In *Proc. UIST*, pages 323–332.

R. D. Vatavu, L. Anthony, and J. O. Wobbrock. 2012. Gestures as point clouds: A $P recognizer for user interface prototypes. In *Proc. ICMI*, pages 273–280.

J. O. Wobbrock, A. D. Wilson, and Y. Li. 2007. Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes. In *Proc. UIST*, pages 159–168.

F. Álvaro, J.-A. Sánchez, and J.-M. Benedí. 2013. Classification of on-line mathematical symbols with hybrid features and recurrent neural networks. In *Proc. ICDAR*, pages 1012–1016.