Annoflow - Handwritten Annotation and Proof-reading on Dynamic Digital Documents

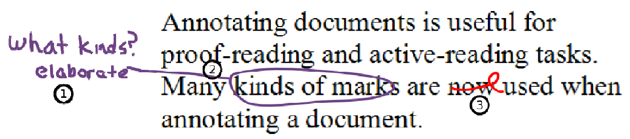
Phil Crosby *Michael Quinn* *François Guimbretière*
Department of Computer Science
Human-Computer Interaction Lab
University of Maryland,
College Park, MD, 20742
philc@umd.edu mikejquinn@gmail.com francois@cs.umd.edu

ABSTRACT

Proof-reading digital documents is a difficult task, because the ink annotations made on documents do not maintain their relevance as the document changes. In addition, applying changes indicated by proof-reading marks to the document is tedious and error-prone. We propose Annoflow, a system for managing document annotations. Annoflow manages both free-form margin annotations and proof-reading marks, intelligently reflowing them as the document changes to maintain their relevance. It also interprets and applies the changes indicated by ANSI proof-reading marks made on the document. In this paper, we describe Annoflow as a platform for measuring the best strategies for anchoring, reflowing and applying annotations.

INTRODUCTION

Annotating digital documents with ink is an important function for proof-reading tasks. Figure 1 illustrates common forms of ink annotations. Existing systems like Microsoft Word [10] and XLibris [6] enable digital document annotation with ink and integrate the annotations directly into the document. However, authors often want the ink annotations to stay meaningful and relevant as the document changes. For example, comments written next to a paragraph should stay with the paragraph as it moves around the document. Additionally, when authors review proof-reading marks made by themselves or others on their digital documents, they have to manually interpret all of the marks and make the changes to the document themselves. This is a tedious process; authors need a faster way to apply proof reading suggestions to their documents.



Annotating documents is useful for
proof-reading and active-reading tasks.
Many kinds of marks are ~~now~~ used when
annotating a document.

Figure 1: Margin comments (1) can be linked to marks made in the document via “callout marks” (2). Proof-reading marks (3) are made in the body of the document and are colored red by our system.

Our contribution addresses these needs. We propose Annoflow, a complete implementation for annotating digital documents that manages annotations penned in Microsoft Word, preserves their meaningfulness as the document changes, and automatically applies standard ANSI proof-reading marks [2].

Several solutions exist that address sub-problems in the area of managing document annotations. Grouping pen strokes into meaningful units is important for distinguishing between different annotations. Dynomite [9] is a note-taking system that clusters pen strokes into groups based strictly on time; XLibris [6] relies on temporality, proximity, context in the document, and the shape of the strokes. Our implementation uses a mix of two techniques for grouping: explicit cues from the user, and the Tablet PC SDK grouping implementation, which uses proximity and temporality criteria to form groups of strokes.

"Reflow" is positioning the annotations intelligently as the document structure changes. Microsoft Word supports ink annotation to documents, but does not reflow marks made in the body of the document. Word reflows annotations made in the margin of the document, but annotations can only be made in the right margin. These annotations can be anchored only to individual words, and cannot be "linked" to words or strokes made in the body of the document via hand-written call-out marks. ProofRite [5] demonstrates accurate reflow of proof-reading marks made in the body of the document, but does not address margin annotations. Several tactics for annotation reflow have been proposed in the Callisto system [3], but Callisto's reflow of margin annotations is limited to moving them vertically up and down as the paragraphs move. Our system robustly reflows both document proof-reading marks and margin annotations, handling annotation collision and re-rendering of anchoring marks.

In addition to managing margin annotations, we have also developed a solution for interpreting and applying standard ANSI [7] proof-reading marks. There are few solutions in this area that apply to digital documents. MATE [7] can interpret a few custom editing marks and apply them to the document, but it does not reflow the marks when the document content changes. ProofRite is able to recognize and anchor only a few single-stroke proof-reading marks, and it is left up to the user to manually apply them to the text. Our system can recognize a much greater subset of ANSI proof-reading marks, reflow them as the document changes, and can apply the desired changes to the document.

MANAGING PROOF-READING ANNOTATIONS

Proof-reading annotations are made in the body of the document. To stay relevant as the document changes, they must anchor accurately, reflow sensibly, and be recognized and interpreted correctly.

Recognizing Proof-Reading Annotations

Annoflow uses the Tablet PC SDK gesture recognition functionality, complemented with a modified version of the Siger recognition engine [8]. We chose Siger for ease of implementation; it recognizes gestures based on regular expressions of directional vectors. For greater accuracy, we use many additional pen stroke characteristics, such as the percentage of the stroke traveling in a specific direction and the distance between the stroke's start and end points.

Clustering and Anchoring Proof-Reading Annotations

Some annotations consist of more than one stroke. To recognize multi-stroke proof-reading marks, we must cluster strokes that might be related, and then analyze them as a whole unit. We do this by keeping track of all "incomplete" annotations. When a new stroke is added, it is first determined whether or not the new stroke is recognized as a part of a multi-stroke annotation. If it is, the new stroke is checked against all known incomplete annotations. An incomplete annotation can then "claim" the new stroke as its own if it is determined that the strokes are associated with one another, which is determined by the new stroke's proximity and its shape.

After an annotation is interpreted, it must be anchored to a point in the document to allow for proper reflow. For each supported proof-reading mark, we have identified a specific point within its strokes that is used for identifying the text most relevant to the annotation. Annotations are anchored to the word or character in the document which lies directly below the stroke's anchor point.

The anchor points are annotation dependent (Figure 2). Many annotations, such as the "italic" proof-reading mark, simply use the midpoint of the stroke. One of the more interesting anchor points is that of the transpose proof-reading mark, which uses the point of inflection between the concave-down and concave-up halves of the mark (Figure 3). We find the point of inflection by associating a vector with each point of the stroke. The vector with its angle closest to 90 degrees (pointing down, if written left-to-right) is determined to be the point of inflection.

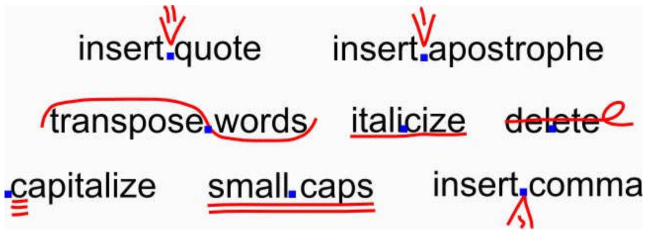


Figure 2: Common proof-reading marks and their anchor points. Anchor points are determined by heuristics specific to each mark.

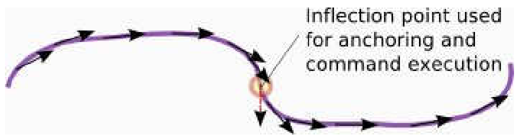


Figure 3: The transpose mark's anchoring point is determined by the inflection point of the curve.

Some annotations require contextual information from the document to be anchored accurately. For instance, the "insert period" and "insert comma" annotations are always meant to be placed directly behind a word. We look at the words surrounding the anchor point and use this information to fine tune the accuracy of the anchor point.

Strokes which are not recognized as ANSI proofreading marks are simply anchored to the page at the stroke's midpoint.

Reflowing Proof-Reading Annotations

Proof-reading annotations within the document must properly reflow as the document changes in order to remain relevant. Our system anchors proof-reading marks to words or characters, which then follow the text as it reflows.

The problem of reflowing annotations that extend across multiple words has been previously addressed by both XLibris and ProofRite. As such, our system focuses on proof-reading annotations that apply to a single word or character within a word, with the exception of the "transpose" proof-reading mark.

If a word is deleted that contains a proof-reading mark, the mark is deleted along with the word. We avoid leaving "floating," unanchored marks on the document.

Applying Proof-Reading Annotations

Each proof-reading annotation contains an associated action which it performs when the proof-reading mark is applied to the document. The annotations that only apply to a specific point in the document (such as the "insert period" annotation) execute directly at their anchor point. Other annotations may extend across an entire word (such as the "italic" annotation), and so we must use the context surrounding the anchor point to accurately apply the proof-reading mark.

When to apply the proof-reading marks is a difficult question. Our system affords two options: applying them as soon as they're recognized, or applying them later and all at once. Each of these strategies is addressed in the discussion section.

MANAGING MARGIN ANNOTATIONS

When writing annotations in the margin, we encourage users of our system to use an explicit interface for grouping and anchoring their margin annotations; when this interface is neglected, we fall back to an implicit grouping and anchoring system.

Explicit Marking Interface

Annotations made in the margins of documents can have grouping and anchoring ambiguity (Figure 4), even for humans [3]. Users recognize these ambiguities and can become frustrated in trying to "guess" what the system will do, and evidence suggests that users are very unforgiving if their annotations are grouped and anchored incorrectly [4]. To solve this problem, we've designed an explicit marking interface that eliminates the ambiguity and guesswork.

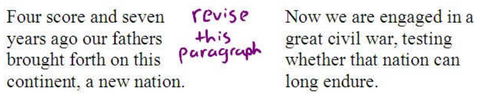

Four score and seven years ago our fathers brought forth on this continent, a new nation.
revise this paragraph
Now we are engaged in a great civil war, testing whether that nation can long endure.

Figure 4: Does the comment apply to the right or to the left paragraph? This annotation is ambiguous.

After writing an annotation, the user can draw "ticks" around the corners of the annotation which explicitly groups the strokes together. Furthermore, to anchor the annotation to a paragraph, the user can draw a straight line from the annotation to the desired paragraph it's referring to. Margin annotations can also be linked to inline annotations by connecting the two with a "call-out" line (Figure 5). This explicit marking interface eliminates ambiguity and error in grouping and anchoring, and raises the confidence in the system's accuracy.

Implicit Marking Interface

It’s unreasonable to expect the user to perfectly adhere to the explicit marking interface all of the time. In fact, having to continually draw tick marks around every annotation can become burdensome, and is unnecessary in cases where it’s fairly obvious how the annotation should be interpreted.

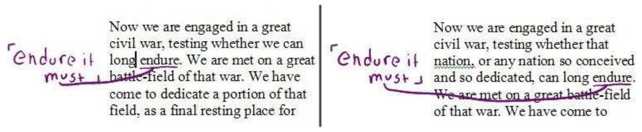
When no explicit grouping marks are present, we anchor and group according to some reasonable defaults. For grouping, we run the strokes through the Tablet PC recognizer, which gives fairly accurate groups according to words, sentences and paragraphs. Any strokes that are classified as being in the same paragraph are grouped together. Grouping doesn’t discriminate based on temporality; if a mark is written near another annotation later, it is grouped together with that existing annotation based on position alone.

In the absence of explicit anchoring marks, the annotation is anchored to the paragraph located nearest to the top of the bounding box of the annotation; if it’s between two paragraphs, like in Figure 4, it anchors to the one on the right.

Reflowing Margin Annotations

Like proof-reading marks, margin annotations must adapt to changes in the document to remain relevant. Annotations made in the margin are anchored to a paragraph either implicitly or explicitly, and then move vertically in the margin as that paragraph moves up and down in the document. If a paragraph is deleted, we interpret this as meaning that any annotations anchored to that paragraph are no longer relevant, and we remove those annotations from the document.

When annotations written in the margins are linked via a “call-out” line to an anchoring mark made in the body of the document, the margin annotation must be sensitive to how the anchoring mark moves. In our design, when the anchoring mark moves around in the paragraph, the margin annotation stays fixed; only the anchoring mark is updated (Figure 5). We avoid “cleaning up” the anchor mark and redrawing it as a smooth line; instead, we favor preserving the look of the user’s hand-drawn anchor mark.



Now we are engaged in a great civil war, testing whether we can long endure. We are met on a great battle-field of that war. We have come to dedicate a portion of that field, as a final resting place for

Now we are engaged in a great civil war, testing whether that nation, or any nation so conceived and so dedicated, can long endure. We are met on a great battle-field of that war. We have come to

Figure 5: A margin annotation is linked to an anchoring mark in the body of the document via a call-out mark. Anchor marks are transformed to remain connected as the document annotation is moved.

When the anchoring mark changes paragraphs, the margin annotation is moved along with it. This is to avoid very long anchor marks stretching across the page.

As the document is changed, margin annotations may run into each other and overlap as they're reflowed around the document. To prevent this, we've designed an algorithm to optimize the position of annotations in the margin so that they avoid overlap and yet still remain meaningful. Briefly, our overall strategy is to make as little disturbance to the annotations as necessary. Our algorithm shifts margin comments up and down to make room for a newly reflowed annotation. It performs a multi-pass location optimization to fill all available white space. It reflows an annotation above or below the bounds of the paragraph it's anchored to only as a last resort, and never moves an annotation to the opposite margin.

IMPLEMENTATION

Our implementation is built as a Microsoft Word add-in, using the Tablet PC SDK and its recognizer. Some annotations are recognized using a modified version of the Siger recognizer, while others use the Tablet PC SDK's built-in gesture recognizers. We used the Siger recognizer in some cases because, while we found it to be less accurate, it is more easily programmable than the Tablet PC SDK recognizers.

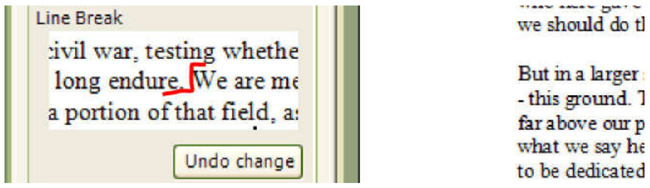
While working with the internals of Word, we've found that our task could have been made considerably easier if it supported the concept of anchors that can be attached to a word or paragraph, which inform listeners as they're moved in the document. Currently, we do this in a circuitous fashion by attaching marks to ranges of text in a document, and manually polling those ranges for changes in their position.

Another useful addition to Word's SDK would be access to its drawing layer. Currently we attach an ink-collecting overlay to Word and use that as our drawing surface. Since Word doesn't report when it needs to redraw, it can periodically overwrite what's displayed on the overlay, causing flicker.

DISCUSSION AND FUTURE WORK

Currently, we're conducting a user study to test which is more effective: instant-apply of proof-reading marks as they're written, or applying them later and all at once. Instant apply offers immediate feedback and gratification. If a mistake is made, it can be corrected right away. However, having your document change as you write on it can be distracting, and applying annotations immediately introduces ambiguity in some multi-stroke commands.

Delaying the execution of proof-reading marks is more optimal in peer review scenarios, where only the author wants to apply changes to the document. This model decreases confidence, because you cannot be sure if the system interpreted and anchored the stroke correctly. To alleviate some of these problems, we've created an undo sidebar where the user can undo the application of any proof-reading mark (Figure 7).



civil war, testing whethe
long endure. We are me
a portion of that field, a:

we should do tl

But in a larger
- this ground. T
far above our p
what we say he
to be dedicated

Figure 7: All marks are shown in their original context and can be undone after they've been applied.

Our hypothesis is that instant-application of proof-reading marks is better suited for managing simple proof-reading annotations. This method has two advantages that make it preferred for proof-reading digital documents on a tablet PC. First, seeing immediately the effect of your mark gives you instant feedback and eliminates uncertainty in the state of the system. Secondly, the immediate feedback allows you to detect failures as soon as they happen (if they happen), and undo any mistakes made by the system.

While general freeform annotations and comments are important to preserve on the document in group collaboration scenarios, preserving specific proof-reading marks in their written form is not necessarily important.

In future work, we hope to present the findings of this study. Additionally, we will measure the accuracy of our system for margin-reflow tasks. We also wish to get user feedback on how well our explicit and implicit margin annotation interfaces work. It is less obvious how margin annotations should be anchored and reflowed, and there are many different reflow strategies. Using Annoflow, we wish to compare different reflow strategies and see which is most effective for document annotation.

There are many deficiencies in the system that we'd like to address. Ink marks made on the Tablet PC are larger and sloppier than those made on paper. We'd like to add a zoomable input panel like DIZI [1] to make annotations made in Annoflow mimic the look of annotations made on paper. We would also like to implement an interface for fixing anchoring and reflow mistakes made by the system, as suggested by [3].

ACKNOWLEDGMENTS

I’d like to thank Chunyuan Liao, Catherine Plaisant and Jerry Fails for their helpful comments on drafts of this paper.

REFERENCES

1. Agrawala, M. and Shilman, M. DIZI: A Digital Ink Zooming Interface for Document Annotation. *INTERACT*, 2005.
2. ANSI, *American national standard proof corrections*. 1981: American National Standards Institute.
3. Bargeron, D. and Moscovich, T. 2003. Reflowing digital ink annotations. *Proceedings of CHI '03*, pp. 385-393.
4. Brush, A. J., Bargeron, D., Gupta, A., and Cadiz, J. J. 2001. Robust annotation positioning in digital documents. *Proceedings of CHI '01*, pp. 285-292
5. Conroy, K. Levin, D. Guimbretière, F. 2004. ProofRite: A Paper-Augmented Word Processor. *Proceedings of UIST ’04*.
6. Golovchinsky, G. and Denoue, L. 2002. Moving Markup: Repositioning Freeform Annotations. *Proceedings of UIST '02*, pp. 21-30.
7. Hardock, G., Kurtenbach, G., and Buxton, W. 1993. A marking based interface for collaborative writing. *Proceedings of UIST '93,* pp 259-266.
8. Siger – Simple Gesture Recognizer. http://sourceforge.net/projects/siger/
9. Wilcox, L. D., Schilit, B. N., and Sawhney, N. 1997. Dynomite: A Dynamically Organized Ink and Audio Notebook. *Proceedings of CHI '97*, pp. 186-193.
10. Microsoft Word 2003 – http://office.microsoft.com/word/