Robustly Anchoring Annotations Using Keywords

A.J. Bernheim Brush and David Bargeron

November 16, 2001

Technical Report

MSR-TR-2001-107

Microsoft Research

Microsoft Corporation

One Microsoft Way

Redmond, WA 98052

Robustly Anchoring Annotations using Keywords

A.J. Bernheim Brush and David Bargeron
Microsoft Research
One Microsoft Way
Redmond, WA 98052
ajb@cs.washington.edu, davemb@microsoft.com

Abstract

As the number of systems which offer annotations on digital documents increases, robust anchoring of annotations is a key challenge. It is especially important if the annotator lacks write-access to the document being annotated. For example, if someone makes an annotation on a web page today and the web page changes tomorrow, what happens to the annotation? One approach is simply to "orphan" the annotation (separate it from its context), while another is to attempt to re-position the annotation in the modified document. This paper introduces Keyword Anchoring, a method for robustly anchoring annotations in digital documents so they can be appropriately re-positioned in a modified version of the document. Unlike existing robust anchoring algorithms, the design of Keyword Anchoring is motivated by an evaluation of user expectations. This evaluation revealed that unique words in the vicinity of an annotation are distinguishing anchor characteristics, which can be tracked among successive versions of a document. By focusing on document content rather than on the underlying document structure, Keyword Anchoring requires no cooperation from the document, and can be used with a variety of digital document formats. We present our algorithm, describe several user studies we conducted to gauge user expectations, and discuss implications of our work for applications.

Keywords: Annotation, robust, anchor, location, annotation system design.
Approximate Word Count: 7,000 words

1 Introduction

The number and variety of systems offering annotation functionality for digital documents increases every year. Many of these systems store annotations outside of the documents on which they were made. When a user wishes to view the annotations on a document, a client application loads the annotations from a store and displays them in their appropriate locations throughout the document.

But what happens if the document gets modified after the annotations are created? Digital documents often change over time. A recent study of the use of an electronic annotation system by roughly 450 users over 10 months found that the biggest user complaint was "orphaning" of annotations [3]. That is, when the annotated document changed, annotations lost their proper positions and were shown at the bottom of the document, outside of any useful context.

Thus robust anchors that allow annotations to remain associated with the correct portion of the document even if the document is modified are crucial. A number of existing systems have methods for locating positions within a document when the document changes [5][6][7][9][10][11][13]. While these algorithms are robust to varying degrees, none of them take users' expectations into account. The primary contribution of this paper is to introduce Keyword Anchoring, a robust anchoring method we designed based on what users expect to happen to annotations when the document changes [2]. The algorithm primarily uses unique words from the annotated document to anchor and re-position annotations, and it ignores any specific internal document structure. This allows it to reflect user assumptions about the document, and to be used with many different digital document representations. Keyword anchoring is complimentary to existing robust anchoring methods including Robust Locations [9] and WebVise [5], and could be used in conjunction with them to provide additional robustness that may better meet user expectations.

In addition to presenting our Keyword Anchoring algorithm, we focus on issues that arise when positioning annotations in a modified document. In particular, results from an initial user assessment of the keyword algorithm suggest the value of showing annotations that have been re-positioned in a modified document with moderate confidence as guesses, rather than representing them as either correctly positioned or orphaned.

In the next section we define an annotation's anchor and outline the assumptions we make about when annotations require robust anchoring. Section 3 describes the user expectations gathered in an earlier study that motivated the design of keyword anchoring. In Sections 4 and 5 we describe the keyword anchoring algorithm and results from a user study of the algorithm. Section 6 outlines the implications of our work for robust anchoring algorithms and Section 7 compares keyword anchoring to existing methods for robust anchoring.

2 Annotations and Anchors

The anchor specifies the location of an annotation in a digital document. In this paper we focus on situations where the annotations are not stored in the document, allowing users to annotate documents they do not have permission to modify. We also assume the document is unaware of the annotation and the document author has not provided identifiers or other markers to support annotation. Finally, we do not assume any particular document structure or document object model, only that the document contains words.

Currently the algorithm focuses on anchors for annotations made on a range of text in a document. As shown in Figure 1, there are two types of information that can be saved for the text selected by the user for the location of the annotation:

- *Anchor text:* Information about the exact text selected by the user (e.g. content, length).
- *Surrounding context:* Information about the text in the document near the annotation, but not explicitly selected by the user. Also, information about location of annotation in the document. (e.g. in the third section, or character offset from beginning of document)

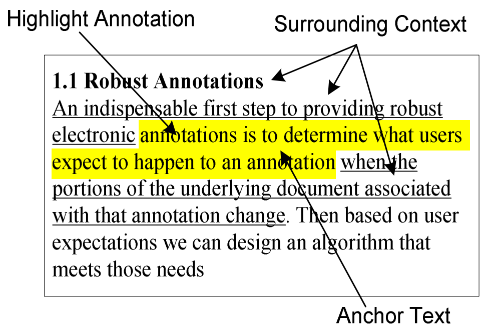

Highlight Annotation
Surrounding Context
1.1 Robust Annotations
An indispensable first step to providing robust electronic annotations is to determine what users expect to happen to an annotation when the portions of the underlying document associated with that annotation change. Then based on user expectations we can design an algorithm that meets those needs
Anchor Text

Figure 1: Robust anchoring algorithms choose what type of information to save about the anchor text selected by the user and its surrounding context.

In our keyword anchoring algorithm we extract and use keywords from the anchor text for our robust anchors along with some information about the surrounding context. Before we describe the algorithm, we first explain why we decided to focus on keywords.

3 Understanding User Expectations

In a previous study [2] we looked at what users expected to happen when they made annotations on documents that were subsequently modified. We wanted to determine what information should be saved in an annotation's anchor to meet user expectations. In addition we were interested in determining the relative value of anchor text and surrounding context information when trying to position anchors in modified documents.

In the study twelve participants made annotations on a document so that it "could be skimmed quickly by a busy executive." Next, they rated where their annotations had been placed in a modified version of the document. We used a very simple anchoring algorithm that saved only the anchor text selected by the user and then used partial string matching to find a position for the annotation in the modified document. No information about surrounding context of the annotation was saved. We expected this algorithm to fail often, alerting us to scenarios where participants were most unhappy with the algorithm's performance.

Participants' ratings of the annotation's new positions in the modified document along with their comments highlighted when they expected the simple anchoring algorithm to do a better job, and what types of information should be saved to meet these expectations. In the cases where participants gave the algorithm low ratings, we found they often expected the algorithm to do a better job locating keywords or phrases. Participant comments included: "*the key words are there, it should have been able to somehow connect that sentence [in the modified version] with the original",* and *"Should have at least highlighted the name."* These comments suggest that paying special attention to unique or keywords in the anchor text may be valuable when designing a robust anchoring algorithm.

Our study also indicated that users might not consider the surrounding context very important for annotations made on ranges of text. Participants gave high ratings to annotations whose anchor text had significantly moved and did not seem to expect a more sophisticated search when orphans were caused by deletion of the anchor text. However, this does not necessarily mean that surrounding context should not be saved. Surrounding context could be used to quickly find an annotation if the document has not changed or to determine the appropriate position if the anchor text selected by the user occurs multiple times in the document. But, in general users may not consider surrounding context very important and algorithms may want place less emphasis on it.

Finally, participant ratings indicated that for some annotations where the simple anchoring algorithm thought it had found a likely spot for the annotation (because some of the exact anchor text was present) participants might have preferred the annotation orphaned. They appeared to feel the position found by the algorithm was incorrect. In cases where the algorithm has lower confidence in the position found in the modified document, it might be reasonable to orphan the annotation and provide the found position as the best guess. While some of the algorithm's best guesses might not be completely correct from the user's standpoint, they could provide a starting point for the user to easily reattach the orphaned annotation.

A complete description of the initial study and findings can be found in [2]. While the user expectations gathered by the study were for a particular task, we believe that they are reasonable for other uses of annotations.

4 Keyword Robust Anchoring

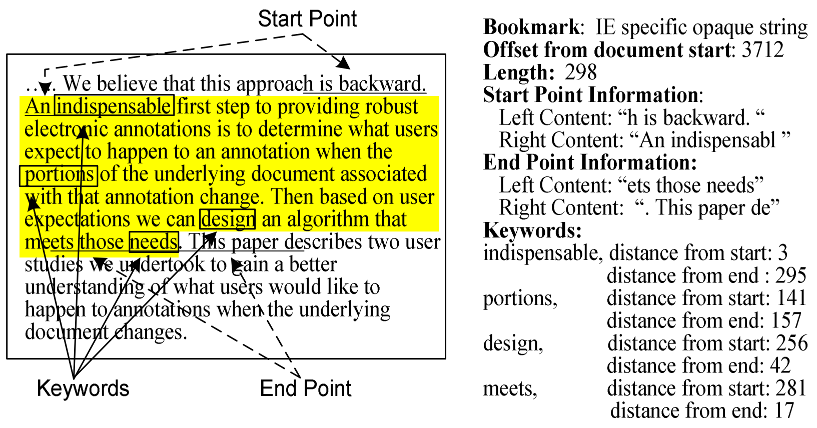
Robustly anchoring annotations requires two steps: deciding what anchoring information to save for an annotation, and then using the anchor to find the annotation in the document. Based on user feedback from the initial study, when designing keyword robust anchoring we focused on unique words in the anchor text, put less emphasis on the surrounding context of the annotation, and experimented with different thresholds for when to display the anchor positions found by the algorithm in the modified document.

4.1 Creating Keyword Annotation Anchors

User expectations, particularly the focus of the users on unique words in the anchor text, greatly influenced the information saved in keyword anchors. The anchors are specialized for ranges of text rather than a particular point in the document, since we are interested in anchoring annotations.

As shown in Figure 2, keyword annotation anchors contain:

- **HTML bookmark for the selection**: An Internet Explorer-specific string used to quickly anchor annotations in documents that have not changed.
- **Offset from start of document**: The number of characters from the beginning of the document to the anchor text**.**
- **Length of the anchor text**: The length, in characters, of the text selected by the user.
- **Information start and end points of the anchor text**: A small of amount of text from the document surrounding the start and end of the anchor text.
- **Information about keywords in the anchor text**: A list of unique words from the anchor text and their locations within the anchor text.



…. We believe that this approach is backward. An indispensable first step to providing robust electronic annotations is to determine what users expect to happen to an annotation when the portions of the underlying document associated with that annotation change. Then based on user expectations we can design an algorithm that meets those needs. This paper describes two user studies we undertook to gain a better understanding of what users would like to happen to annotations when the underlying document changes.

Bookmark: IE specific opaque string
Offset from document start: 3712
Length: 298
Start Point Information:
Left Content: "h is backward. "
Right Content: "An indispensabl "
End Point Information:
Left Content: "ets those needs"
Right Content: ". This paper de"
Keywords:

indispensable,	distance from start: 3
	distance from end : 295
portions,	distance from start: 141
	distance from end: 157
design,	distance from start: 256
	distance from end: 42
meets,	distance from start: 281
	distance from end: 17

Figure 2: A highlight annotation and its keyword anchor information

The HTML bookmark is used in our current implementation to find an annotation's anchor when the document has not changed. While specific to Microsoft Internet Explorer, any browser is likely to have some method of uniquely marking a section of text (e.g ID, bookmark, or character offset) and the appropriate data for a particular implementation could be substituted in the keyword anchor. Alternatively this information could be ignored to ensure the anchor is completely independent of the document format.

Saving the keywords from the anchor text is the crucial part of our algorithm. The keywords are determined by selecting the words in the anchor text that are most unique with respect to the rest of the document. For a particular document, we initially calculate a map of word frequencies. For example, in a document "the" may occur 500 times, while "purple" occurs only twice. When the user creates an annotation we select as keywords all words in the anchor text that only occur once in the document. If there are fewer than three words in the anchor text that occur only once in the document, we select words with increasing frequency (those that occur twice, those that occur three times, etc.) until at least three keywords have been found. All words in the anchor text are selected if it is shorter than three words.

As Figure 2 illustrates, for each keyword selected, we save the word and also its distance from the start and end points of the anchor text. Saving keywords, rather than the entire anchor text as some systems do, reduces storage requirements. It also makes the anchors more robust in the face of wording changes.

4.2 Finding Keyword Annotation Anchors

After the user creates an annotation, the keyword positioning algorithm uses the keyword anchor to find the appropriate position for the annotation any time annotations are viewed on the document. Similar to other methods, the keyword positioning algorithm initially assumes the document has not changed, and tries to find the anchor using only bookmark and offset information. While finding the annotation quickly when the document has not changed is important, in this section we focus on finding an appropriate location when the document has been modified.

First, we discuss how the algorithm finds and considers candidate anchors for the annotation in the modified document, and then how the presentation of the new anchor to the user changes based on the anchor’s confidence score. The keyword positioning algorithm handles a variety of document changes including movement of anchor text, reordering of keywords, deletion of some of the anchor text (even keywords), and rewording of the anchor text to a certain extent.

4.2.1 Candidate Anchors

When the positioning algorithm does not find an annotation’s anchor using the bookmark, the algorithm looks for the keywords from the anchor in the modified document. For each keyword it finds, the algorithm builds a “candidate anchor” using the other keywords plus the start and end points from the original anchor. The positioning algorithm gives each candidate anchor a confidence score based on how well it matches the original anchor. The candidate anchor with the highest confidence score becomes the annotation’s new anchor. The basic keyword positioning algorithm is conservative, and in Section 6.2 we outline performance tradeoffs that all robust anchoring algorithm may want to consider.

To create candidate anchors, the positioning algorithm loops through each keyword in the original anchor and examines every time it occurs in the modified document. When an instance of a keyword is found, the positioning algorithm creates a candidate anchor containing this *seed* keyword and then extends and scores the candidate anchor by:

1. Looking for other keywords

The algorithm searches in turn for each of the other keywords in the original anchor after the current seed keyword. If the positioning algorithm finds the keyword, it extends the candidate anchor to include the new keyword unless adding the new keyword would make the candidate anchor unreasonably long or the candidate anchor already includes the new keyword. The current implementation requires that the candidate anchor does not grow larger than twice the length of the original anchor.

If the candidate anchor already includes the new keyword or the positioning algorithm extends the candidate anchor to include it, the algorithm increases the candidate anchor's confidence score based on the relative change in distance between the seed keyword and the newly added keyword from the original anchor to the candidate anchor. For example, if the two keywords were 20 characters apart in the original anchor and are still 20 characters apart in the modified version then the relative change in distance is 0, and the confidence score is increased by the maximum value, 100 points in the current implementation. Otherwise as the change in distance between the two keywords increases, the number of points added to the confidence score for that keyword decreases. Figure 3 shows two of the candidate anchors the algorithm finds for the annotation in Figure 2 in a modified version of the document.

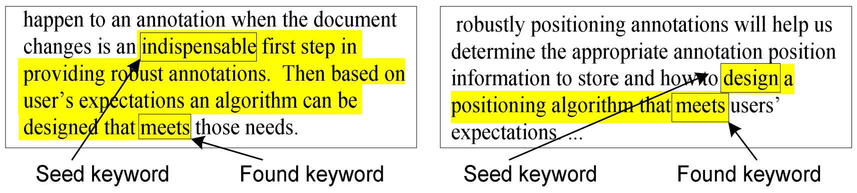
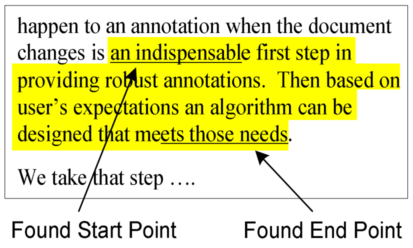

happen to an annotation when the document changes is an indispensable first step in providing robust annotations. Then based on user's expectations an algorithm can be designed that meets those needs.
Seed keyword
Found keyword
robustly positioning annotations will help us determine the appropriate annotation position information to store and how to design a positioning algorithm that meets users' expectations ...
Seed keyword
Found keyword

Figure 3: Two candidate anchors in a modified document after the keyword expansion step for the annotation in Figure 2.

2. Looking for the start and end points

After including as many keywords as possible in the candidate anchor, the algorithm uses information about the start and end points of the original anchor. Initially, the algorithm only looks at the content from the start and end points that was explicitly part of the anchor text selected by the user.

The algorithm looks in front of the current candidate anchor for the right content of the start point. If it finds part or all of the content the algorithm extends the candidate anchor to include the found text if that does not make the candidate anchor too long. Again, the confidence score increases based on how closely the new distance from the seed keyword to the start point matches the distance in the original annotation, with a maximum of 50 points in the current implementation. The algorithm performs a similar check looking for the left content from the end point of the original anchor after the current candidate anchor. Figure 4 shows a candidate anchor that has been extended to include content from the start and end points.



happen to an annotation when the document changes is an indispensable first step in providing robust annotations. Then based on user's expectations an algorithm can be designed that meets those needs.

We take that step ….

Figure 4: The complete candidate anchor for an anchor from Figure 3 including context found from the start and end points.

3. Comparing the length and offset of the found candidate anchor to original anchor

The keywords and start and end point information determine the complete candidate anchor for the current seed keyword. Next, the positioning algorithm scores how well the candidate anchors' length and location in the document match the original anchor. In the current implement the algorithm adds 50 points to the confidence score if the candidate and original anchors have the same length, and decreases the amount of points added as the different in lengths increases. Similarly, the algorithm adds 20 points if the candidate anchor is at the same place in the document as the original anchor and decreases the number of points added as the difference in positions increases.

4. Look for surrounding text from end points

Finally, in order to handle instances where the anchor text repeats throughout the document, the positioning algorithm looks in front and behind the candidate anchor for the left content of the start point, and right content of the end point, which were outside the anchor text selected by the user. If the algorithm finds the complete left content of the start point in front of the candidate anchor, it increases the confidence score by 10 points in the current implementation. The algorithm does a similar check for the right content of the end point after the candidate anchor.

After considering all possible candidate anchors the algorithm uses the one with the highest confidence score to replace the original anchor. The positioning algorithm normalizes the confidence score to between 0 and 100 by dividing by the highest possible score for the original anchor. If the positioning algorithm finds no candidate anchor for an annotation, all the keywords from the original anchor were deleted from the modified version of the document and the confidence score is set to 0. The weights described above for calculating the confidence score of a candidate anchor are influenced by user expectations and our own implementation experience. They could benefit from additional empirical testing and tuning for particular applications.

4.2.2 Presenting the Candidate Anchor

The score of the candidate anchor denotes how closely the new anchor matches the original and determines how to present the anchor to the user. As shown in Figure 5, if the confidence score exceeds the guess threshold then the candidate anchor replaces the original anchor and the positioning algorithm places the annotation in the document at the new anchor. When the confidence score exceeds the guess threshold, we assume the algorithm has found a place for the annotation that will meet user expectations. Allowing users to easily modify the found anchor would let them make adjustments if any of the anchors do not meet their expectations.

When the confidence score of the final candidate anchor falls below the guess threshold, the positioning algorithm orphans the annotation and does not place it in the document. The algorithm instead makes the candidate anchor available to the user as the algorithm's best guess. The user can then specify the new position of the annotation choosing to accept the guess, modify the guess, select a different anchor text or delete the annotation. If the confidence score of the anchor is below the guess threshold, even though the algorithm found a possible location, user interaction is necessary to ensure the new anchor meets user expectations. One of the primary goals of the user study described in the next section was to determine an appropriate value for the guess threshold.

After the user study we added a complete orphan threshold to the keyword positioning algorithm as shown in Figure 5. When the confidence score falls below this value, even if a candidate anchor was identified by the positioning algorithm it is not shown as a guess. Scores below the complete orphan threshold mean that the vast

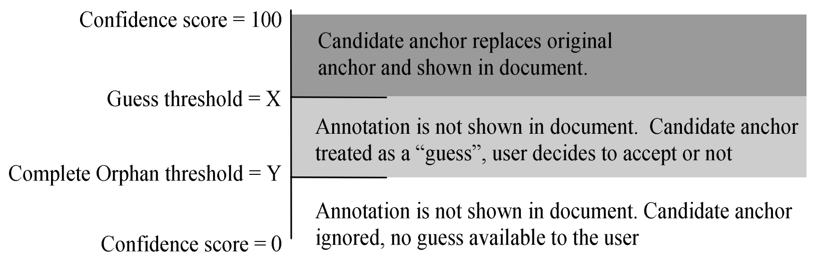

Confidence score = 100
Candidate anchor replaces original anchor and shown in document.
Guess threshold = X
Annotation is not shown in document. Candidate anchor treated as a "guess", user decides to accept or not
Complete Orphan threshold = Y
Annotation is not shown in document. Candidate anchor ignored, no guess available to the user
Confidence score = 0

Figure 5: Thresholds used to interpret the confidence score of the candidate anchor. One of the goals of the user study described in Section 5 was to determine appropriate values for the Guess and Complete Orphan thresholds.

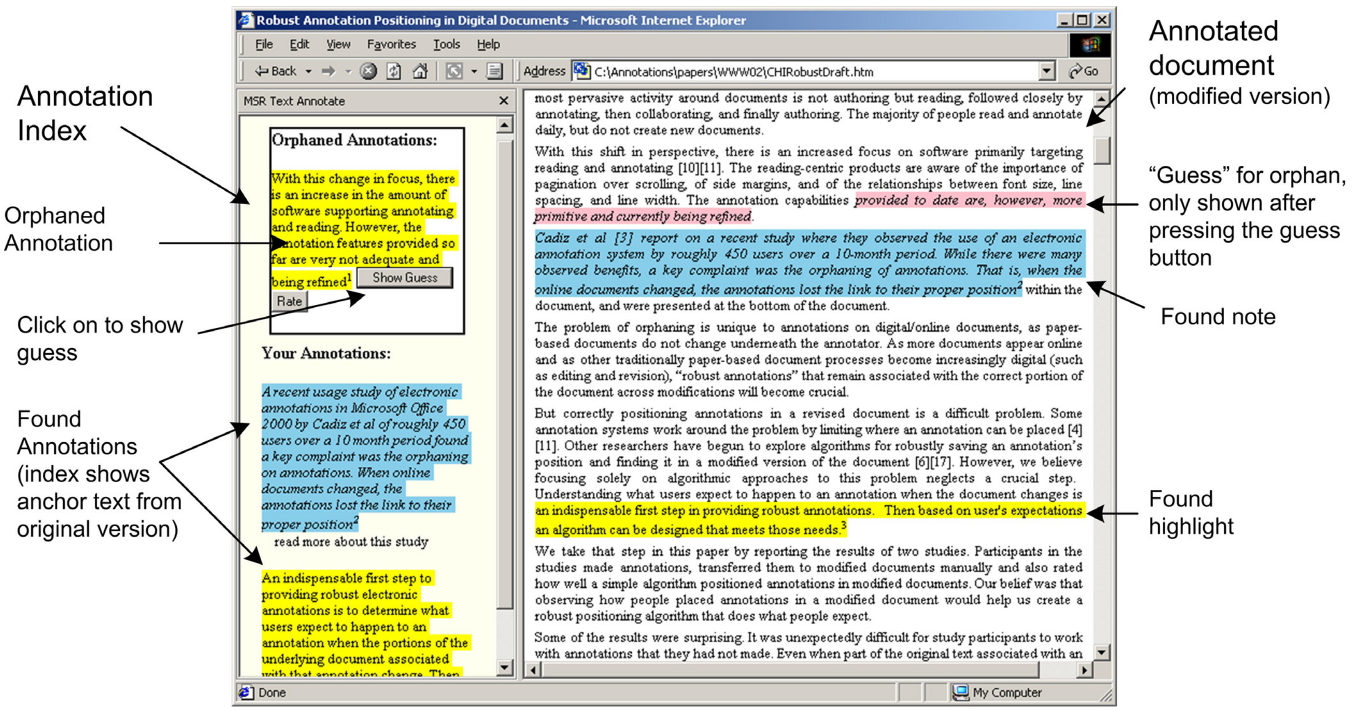
majority of the anchor text has been deleted from the document and the candidate anchor is unlikely to meet user expectations.

5 User Assessment

We have implemented keyword robust anchors in a prototype annotation system that plugs into Microsoft Internet Explorer and allow users to make highlights and notes on any web page. Annotations and anchors are stored on a server using the Common Annotation Framework (CAF) [1].

A user makes an annotation by selecting text on a web page, and then left-clicking the selection. A menu pops up from which the user can choose to highlight or attach a note to the selected text. Highlighted text is displayed with a yellow background and text with a note attached is displayed with a blue one. Annotations can be deleted by left-clicking on an existing annotation and selecting "delete" from the menu.

Figure 6 shows the interface from our user study after a user's annotations have been positioned in the modified version of a document. The annotation index window on the left displays a list of all annotations for the web page and shows the anchor text selected by the user in the original document so participants could easily rate the position found in the modified document. Any orphaned annotations are listed at the top of the index.   Clicking


Robust Annotation Positioning in Digital Documents - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Address C:\Annotations\papers\WWW02\CHIRobustDraft.htm
Go
MSR Text Annotate
Show Guess
Rate
Done
My Computer
Annotation Index
Orphaned Annotation
Click on to show guess
Found Annotations (index shows anchor text from original version)
Annotated document (modified version)
"Guess" for orphan, only shown after pressing the guess button
Found note
Found highlight

Orphaned Annotations:

With this change in focus, there is an increase in the amount of software supporting annotating and reading. However, the annotation features provided so far are very not adequate and being refined[1]

Your Annotations:

A recent usage study of electronic annotations in Microsoft Office 2000 by Cadiz et al of roughly 450 users over a 10 month period found a key complaint was the orphaning on annotations. When online documents changed, the annotations lost the link to their proper position[2]

read more about this study

An indispensable first step to providing robust electronic annotations is to determine what users expect to happen to an annotation when the portions of the underlying document associated with that annotation change. Then

most pervasive activity around documents is not authoring but reading, followed closely by annotating, then collaborating, and finally authoring. The majority of people read and annotate daily, but do not create new documents.

With this shift in perspective, there is an increased focus on software primarily targeting reading and annotating [10][11]. The reading-centric products are aware of the importance of pagination over scrolling, of side margins, and of the relationships between font size, line spacing, and line width. The annotation capabilities *provided to date are, however, more primitive and currently being refined*.

Cadiz et al [3] report on a recent study where they observed the use of an electronic annotation system by roughly 450 users over a 10-month period. While there were many observed benefits, a key complaint was the orphaning of annotations. That is, when the online documents changed, the annotations lost the link to their proper position[2] within the document, and were presented at the bottom of the document.

The problem of orphaning is unique to annotations on digital/online documents, as paper-based documents do not change underneath the annotator. As more documents appear online and as other traditionally paper-based document processes become increasingly digital (such as editing and revision), "robust annotations" that remain associated with the correct portion of the document across modifications will become crucial.

But correctly positioning annotations in a revised document is a difficult problem. Some annotation systems work around the problem by limiting where an annotation can be placed [4] [11]. Other researchers have begun to explore algorithms for robustly saving an annotation's position and finding it in a modified version of the document [6][17]. However, we believe focusing solely on algorithmic approaches to this problem neglects a crucial step. Understanding what users expect to happen to an annotation when the document changes is an indispensable first step in providing robust annotations.   Then based on user's expectations an algorithm can be designed that meets those needs.[3]

We take that step in this paper by reporting the results of two studies. Participants in the studies made annotations, transferred them to modified documents manually and also rated how well a simple algorithm positioned annotations in modified documents. Our belief was that observing how people placed annotations in a modified document would help us create a robust positioning algorithm that does what people expect.

Some of the results were surprising. It was unexpectedly difficult for study participants to work with annotations that they had not made. Even when part of the original text associated with an

Figure 6: Text annotation software used by participants in the user study. Annotations were positioned in the modified document by the keyword anchoring algorithm with a guess threshold of 50. Clicking on the guess button for an orphan (if available) shows the guess in the document. The index shows the text selected by the user in the original document so participants could easily rate the position found by the anchoring algorithm.

on the “guess” button for an orphan (if available) annotation highlights the algorithm’s best guess in the document.

5.1 Experimental Method

Using the prototype we performed a lab study to verify that the keyword approach met user expectations and develop some intuition about where to set the guess threshold. We recruited 8 participants from the general public, that were either college educated or college students, and read at least 30 minutes on average every day. Participants received a gratuity.

Participants performed the same task as in our first user study. After reading and annotating an article they rated where their annotations were placed in a modified version of the document. The ratings were done on a 7 point Likert scale where 7 was “perfect”, 4 was “ok” and 1 was “terrible.” The modified version was created before the study by a colleague unfamiliar with the annotation positioning algorithm.

For comparison purposes we asked participants to rate the positions of their annotations for both the keyword algorithm (Key) and a simple text search algorithm (Simple). The simple text search algorithm saves only the anchor text string selected by the participant and uses text matching to find a new anchor position. If all the original text is not found, the algorithm looks for shorter versions of the anchor text in three ways: alternating cutting words off the front and back of the string, only cutting words off the front, and only cutting words off the back, until the longest partial match is found or the length of the string falls below 15 characters. The anchor’s confidence scores was the percentage of the original anchor found in the modified document. The simple text search algorithm is similar to robust positioning algorithms used in a number of systems, in particular the context method reported in [9].

Participants rated the positions found for their annotations in the modified document by two versions of each algorithm: one where all annotation anchors the algorithm found were shown in the document regardless of their confidence score (Key0, Simple0), and another where annotation anchors that scored below 50 points were shown as guesses (Key50, Simple50). For a particular algorithm the position found for the annotation in the modified document was the same for both versions (e.g. Key0, Key50). However, if the confidence score was less 50 then one version (Key50), would orphan the annotation and the other (Key0) would place it in the document.

Unfortunately, asking participants to repeatedly rate their annotations had an unintended side effect. Participants appeared to become fatigued and less concerned about their ratings by the end of the study. While participant’s ratings support their comments made during the study, we have placed less emphasis on numerical analysis of the ratings because of this fatigue effect.

5.2 Study Findings

The 8 participants in the study made 115 annotations, averaging 14.4 annotations per person. The annotations were mostly highlights (76%) with some notes (24%). Based on comments and ratings the keyword robust anchoring algorithm appears to meet user expectations. Participant comments made while rating new positions found by the keyword algorithm included: "*sentence changed, but computer did a good job of finding whole thing including main point,*" "*good job to pick that out, very similar, a few minor changes with words more concise,*" "*found words that were pretty much the same, so ended up getting the jist of it even though it was very different,"* and *"they did a lot of editing on this one but the computer's guess is exactly what I would have highlighted..*

The keyword algorithm is more complicated that the simple text search algorithm so we examined feedback from participants about the positions found simple text algorithm to determine whether the keyword algorithm met expectations better than simple text search. Participant comments highlight some issues with the positions found by the simple text search including: "*there's a lot extra that it missed, word rearrangement stuff,*" "*it should have found more,"* "*it seemed like it would have picked out some of the keywords that exist farther on,*" and "*lost crucial part of the highlight, did not find key words.*"

Comparing participant's ratings of the new anchor positions found by the algorithms also shows the preference for the keyword algorithm. A Friedman test to evaluate the difference in median ratings was significant, χ^2 (3, N =8) = 9.813, p = .02, with a Kendall coefficient of concordance equal to 0.409. The median ratings and some of the significance values from a follow-up pairwise comparison using the Wilcoxon test are shown in Table 1. The median rating for positions found by Key0 was significantly higher than both of the simple text algorithm's median ratings at the p=0.05 level, while the median rating for Key50 was significantly higher than the median ratings for the simple text algorithm at the p=0.1 level. While we hesitate to put too much emphasis on the numerical ratings due to a small number of participants and some issues with rating fatigue, the numerical ratings support participant comments and reinforce that participants preferred the keyword algorithm.

Algorithms	**Simple50**, Median = 5.5	**Simple0**, Median = 5.0
Key0, Median = 6.5	$p < 0.04^*$	$p < 0.05^*$
Key50, Median = 6.5	$p < 0.07^+$	$p < 0.08^+$

*Table 1: Significance values from the Wilcoxon follow-up test to compare median ratings for annotation positions found by the algorithms. *The median rating for the keyword algorithm with no guess threshold is significantly higher than the two simple text match versions at the p=0.05 level. $^+$The median rating for the keyword algorithm with the guess threshold at 50 is significantly higher than the simple text match versions at the p=0.1 level.*

5.2.1 Experience with Keyword Algorithm

In this section we focus on the participant's experience with the keyword algorithm. From repeated participant comments one simple modifications, extending anchors to sentence or phrase boundaries, would improve satisfaction with the keyword algorithm. Some example comments include: "*not selecting the whole sentence, it would be nice if it did that,*" "*why didn't it highlight the rest of the sentence,*" and "*should have grabbed stuff just before.*" Some participants also commented on the importance of names in the anchor text they selected, saying "*should have found author's name,*" and "*the computer missed the name.*" Treating proper nouns as keywords by default even if they occur multiple times would be a reasonable extension of the keyword algorithm.

One of our initial goals for the study was to develop an intuition for the appropriate guess threshold value. We were looking for a general guideline that did not depend on our particular scoring algorithm. For example, if finding 50% of the anchor's keywords in the modified version meant the new position was likely to meet user expectations. Instead, the user study made apparent the need to add an additional lower threshold in the keyword algorithm and reinforced the importance of user interaction.

When the annotation's anchor text dramatically changed in the modified version, occasionally the keyword algorithm still found a candidate anchor with a very low confidence score. It quickly became apparent that the low scoring candidate anchors were completely unrelated to the original annotation, particularly when the candidate anchor contained only one keyword. Two participants who experienced this situation commented: "*found only one word, guess was really bad*" and "*only found one word, bad guess.*" Not surprisingly the median rating was 1.0 ("terrible") for the six cases where the keyword algorithm's new anchor contained only one keyword. As discussed in Section 4.2.2, we have added a lower complete orphan threshold to the keyword positioning algorithm and are experimenting with appropriate values for this threshold.

The importance of user interaction with the algorithm was reinforced when we looked at potential values for the guess threshold. Individual preference for when to place a found anchor in the modified document instead of showing it as an orphan with a guess varied dramatically. Comments included: *"[about the guess option] quite accurate, sooner have them left them in the document,*" and *"[algorithms] should have been picky, last one [Key0] wasn't picky at all and it should have been.*" Other participants expressed a desire to have annotations with lower confidence scores placed directly in the document at the guessed candidate anchors, but shown in a different color.

We believe that future studies should initially start with a high guess threshold so more annotations are shown orphaned with guesses and then collect data by allowing users to manually lower the threshold. The algorithm might also automatically adjust the guess threshold based on the guesses the user accepts without modification.

We favor starting conservatively with a high guess threshold because our participants worked with an unfamiliar document. In a real situation the user may be more familiar with the document and directly placing annotations in the document at candidate anchors with low confidence scores runs a higher risk of failing to meet user expectations.

With a high guess threshold providing the found candidate anchor as a guess is even more important. In the user study we observed that providing a guess dramatically sped up rating the annotation even when the participants did not necessarily agree with the guess. In a real situation this could be analogous to users taking advantage of the guess to quickly determine how to handle orphaned annotations. Ideally the guess would point them to a reasonable starting point for reattaching the annotation or deciding it should be deleted.

6 Implications for Robust Anchoring Algorithms

Our experience with the keyword algorithm and user study highlights a number of general issues that robust anchoring methods should address.

6.1 User Interface Refinements

Determining the appropriate user experience remains a challenge for robust anchoring algorithms. Most current systems, including [6][9][10], alert the user to fact that an annotation has been orphaned, but do not provide much additional assistance. Our user study demonstrated the value of presenting potential anchors as guesses, but more studies are necessary, particularly field studies, to fully understand the best way to help users cope with orphaned annotations. We have extended our prototype annotation interface to support threaded discussions and multiple authors. In the future it could be deployed in a field study to gather more data.

6.2 Performance Tuning

Robust anchoring algorithms may need to make tradeoffs between finding their ideal position for an annotation and keeping loading time of documents with annotations within user tolerance. Although we have not specifically evaluated the speed of the keyword algorithm, like other robust positioning algorithms it could potentially be quite slow if there are many annotations that are affected by modifications to the document. Robust positioning algorithms could easily improve their speed in two ways: by limiting the search for a new anchor position in the modified document to a range around the location of the original annotation, and using the first acceptable anchor found rather than looking at all possible candidates. Both changes would require user testing to ensure user expectations were still being met by the faster algorithms.

Looking specifically at improving the speed of the keyword algorithm, one reasonable option is limiting the number of keywords saved in the annotation anchor. This would reduce the number of candidate anchors that need to be considered when repositioning the annotation. The appropriate number of keywords saved for an annotation should probably depend on the length of the anchor text selected by the user. The longer the annotation, the more keywords are necessary to describe that annotation and robustly locate it.

Tuning the speed of a robust positioning algorithm for a particular application makes the most sense, since the application designer will have the best idea of how many anchors will typically need to be repositioned and how sensitive users will be to delay. A particular application could also choose how much to use document structure to potentially increase performance based on the types of documents it wanted to support.

6.3 Complex Anchoring Situations

Our current prototype and most annotation systems only allow users to annotate continuous ranges of text. In our user studies, participants have asked for the ability to anchor one comment to multiple pieces of non-contiguous text in the document. This raises numerous user experience questions. How do users specify they would like a comment anchored to multiple places in the document? How should robust anchoring methods handle this? Are there multiple anchors that are found independently and only linked together to display the user? Or does the robust positioning algorithm use knowledge about the relationship of the two anchors in the original document? Users might also like to have annotations with more implicit anchors robustly anchored. For example, annotations that are drawings or marks made in the margin of the document.

Robust anchoring algorithms may also want to handle instances where document modifications have split the original anchor text selected by the user into multiple pieces. This is potentially challenging for the anchoring algorithm, which needs to realize when two possible locations for an annotation contain different parts of the anchor, and the user interface. What is the best way to show that one annotation anchor in the original document has now been separated in the modified version?

6.4 Other Media Types

In the longer term, robust anchors could be valuable for other media types including images and videos. The keyword algorithm approach may extend to these media types very naturally. For example, while a simple image anchoring algorithm might use the x,y position of the user's annotation in the image, a "keyword" approach could save anchor information about particular features or objects at the location in the image annotated by the user. The positioning algorithm could then attempt to locate the feature or object in the modified image and would be robust to common image modifications such as cropping, and scaling. For video the keyword approach translates even more directly with the notion of key frames. Selecting the key frames from the section of video annotated by the user could help robustly anchoring an annotation to the video.

7 Related Work

Existing systems address robustly positioning annotations in a variety of ways. Some of them use information about both the document structure and content of the anchor text selected by the user, while others focus on information about the content or just the annotation's position relative to the underlying document structure.

7.1 Structural and Content Information

Robust Locations [9] and WebVise [5], both save structural and content information and have the most detailed robust anchoring strategies. Robust Locations, part of the Multivalent document system [8], describes a minimal location descriptor and a recommended positioning algorithm. In Robust Locations, points in the document are identified redundantly using unique identifiers, document tree walk information, and context. Span anchors that contain two robust points and the entire anchor text are used to anchor annotations. To find an annotation's position in the document Robust Locations first tries the unique identifier, then the tree walk information and finally uses the context. They describe one method of presenting orphans to the user and some initial testing.

WebVise is an open hypermedia service that supports collaborative annotation and allows creation of links to parts of web pages. WebVise describes a LocSpec that saves anchor information for a variety of media types. For a span of text the LocSpec contains: a reference (ID or HTML target name), the anchor text, some surrounding text, the length of the anchor text and its start position in the document. Unfortunately their positioning algorithm is not described in detail and primarily consists of alerting the user when exact match of the anchor text is not found.

Keyword robust anchoring uses some of the same information as Robust Locations and WebVise, but focuses on using keywords instead of document structure, meeting user expectations, and how to present annotations to users when their anchors have been affected by document modifications. Keyword robust anchoring algorithm could easily be used in conjunction with these methods to provide additional robustness that may better meet user expectations.

7.2 Content Information

Using only information about the content of the anchor text selected by the user makes the robust anchors independent of document format. ComMentor [11], Annotator [7] and CritLink [13] all support annotation on web documents and use content information for robust anchoring. ComMentor and Annotator use a unique substring from the anchor text to find a new position for the annotation in the document. While this will be robust when the anchor text moves in the document, it is unclear how easily a new position will be found if the unique substring in the document has been modified. CritLink saves the entire anchor text and some amount of text surrounding the anchor text. To locate an anchor, CritLink searches for the entire anchor text in the modified document and uses the surrounding context to distinguish duplicates. Similar to ComMentor and Annotator, this method also finds the anchor text if it moves, but appears not to handle modifications to the anchor text.

In contrast, keyword anchoring focuses on finding appropriate positions that meet user expectations precisely when the annotation's anchor text has been modified. The keyword approach is more flexible than the use of substrings and keywords can be selected faster than a unique substring.

7.3 Structural Information

Another approach for anchoring annotations is to use information about the annotation location in the document's underlying structure. Annotea [6], a shared annotation system based on a general purpose open RDF infrastructure, uses XPointers [12] to specify the position of an annotation. XPointers save structural information about the location of an annotation and the specification does not discuss robustness aside from stressing that unique identifiers are most likely to survive document modifications. However, as the authors of the Annotea system observe, XPointer positions can be orphaned or incorrectly positioned if the document changes. They describe one user interface for presenting annotations that have unresolved XPointers. Using identifiers as the primary anchoring method requires cooperation from document authors and reduces the number of documents that can be annotated compared to using keyword anchoring or other methods that save content information.

8 Concluding Remarks

Online documents are frequently modified and this can cause annotations to lose the link to their proper position in the document. The primary contribution of this paper has been to introduce Keyword Anchoring, a robust anchoring algorithm for annotations designed based on what users expect to happen when they make annotations on a digital document that is subsequently modified.

Keyword anchoring uses unique words from the text selected by the user to anchor an annotation and does not assume cooperation from the document or knowledge of the underlying document structure. Saving keywords makes the anchoring very flexible and extremely robust to document modifications, while ignoring document structure allows the algorithm to be used with any document format. Keyword anchoring could easily be used in conjunction with other robust anchoring methods to provide increased robustness that may better meet user expectations.

Our user study of keyword anchoring suggests that the algorithm meets user expectations better than a simple text search algorithm and highlights some improvements that could enhance keyword anchoring. The study also provided insight on how systems can present annotations to the user based on the positioning algorithm's confidence in the location found in the modified version. While keyword anchoring is a step toward addressing the robust anchoring problem in a way that meets user expectations, additional work remains. Future studies, particularly field studies, are needed to refine robust positioning algorithms to ensure they meet user needs and determine the appropriate user experience.

Acknowledgements

We thank Anoop Gupta for valuable feedback on the algorithm. We also thank the participants in the user study.

References

[1] Bargeron, D. and Gupta, A. A Common Annotation Framework, Microsoft Tech. Report MSR-TR-2001-108.

[2] Brush, A., Bargeron, D., Gupta, A., and Cadiz, J. Robust Annotation Positioning in Digital Documents. *Proc. CHI 2001*, 285-292.

[3] Cadiz, J., Gupta, A., and Grudin, J. Using Web Annotations for Asynchronous Collaboration Around Documents. Proc. of CSCW '00, 309-318.

[4] E-Quill. http://www.e-quill.com/

[5] Grønbæk,, K., Sloth, L., and Ørbæk, P. Webvise: Browser and Proxy Support for Open Hypermedia Structuring Mechanisms on The WWW, Proc. of the 5th International WWW Conference. http://www8.org/w8-papers/3a-search-query/webvise/webvise.html

[6] Kahan, J. Koivunen, M., Prud'Hommeaux, E., and Swi, R. Annotea: An Open RDF Infrastruture for Shared Web Annotations, Proc. of the 10th WWW Conference. http://www.w3.org/2001/Annotea/Papers/www10/annotea-www10.html

[7] Ovsiannikov, I., Arbib, M., and McNeill, T. Annotation Technology, Int. J. Human Computer Studies, (1999) 50, 329-362.

[8] Phelps, T., and Wilensky R. Multivalent Annotations, Proc. of the First European Conference on Research and Advanced Technology for Digital Libraries. http://www.cs.berkeley.edu/~phelps/Multivalent/papers/edl97-abstract.html

[9] Phelps, T., and Wilensky R. Robust Intra-document Locations, Proc. of the 9th WWW Conference, http://www9.org/w9cdrom/312/312.html

[10] Microsoft Office 2000 Web Discussions, http://office.microsoft.com/assistance/2000/wWebDiscussions.aspx

[11] Röscheisen, M., Mogensen, C., and Winograd, T. Shared Web Annotations as a Platform for Third-Party Value-Added, Information Providers: Architecture, Protocols, and Usage Examples, Technical Report CSDTR/DLTR (1997), Stanford University.

[12] XML Pointer Language, http://www.w3.org/TR/xptr/

[13] Yee, K. Draft RFC for text-search fragment identifiers used in CritLink, http://crit.org/http://crit.org/draft-yee-url-textsearch-00.txt