Using Experience on the Read/Write Web: The GhostWriter System

Derek Bridge and Aidan Waugh

Department of Computer Science,
University College Cork,
Ireland
{d.bridge,a.waugh}@cs.ucc.ie

Abstract. On the Read/Write Web, users are authors as much as they are searchers. We describe one concrete example of this, a waste exchange service where users submit descriptions of items that they have available but wish to get rid of. It is generally to the advantage of subsequent users if authors write comprehensive descriptions. We propose that *successful* descriptions, i.e. ones which did result in the user passing on an item for reuse, be used as cases. We describe the GhostWriter system that we have designed and built: it makes content authoring suggestions using feature-values extracted from the cases. We end with a preliminary, off-line ablation study, which shows promising results.[1]

1 Introduction

Web 2.0 is the era of the Read/Write Web. The current Web makes information-seekers but also information-authors of us all. On the one hand, we search and browse; on the other hand, numerous web pages offer us the opportunity to post our own content.

In his invited talk at the Ninth European Conference on Case-Based Reasoning, Enric Plaza argues that people increasingly use the Web as a repository for recording and sharing experiences [7]. These experiences include reviews of products, such as hotels, electronic goods, books and movies, in which people report their experience of consuming these products; and they include 'how-to' plans and recipes, in which people report their experience of carrying out tasks to achieve certain goals. Users search and browse web pages, blogs and forums, to retrieve, aggregate and reuse these experiences to help them make decisions in the real-world (e.g. which hotel to book) and to help them do things in the real-world (e.g. how to install a piece of software). Reasoning with these unstructured experiences is a new direction in CBR research.

[1] This research was funded by the Environmental Protection Agency of Ireland under Grant Number 2007-S-ET-5. We are grateful to Maeve Bowen and Catherine Costello of Macroom-E and wastematchers.com and to Lisa Cummins of University College Cork for their engagement in our research.

All of this emphasizes how important it is, when exploiting the write-capabilities of the Read/Write Web, that we author high quality content. In this paper, we show how to reuse existing Web content to support the authors of new content.

In Section 2, we present the concrete scenario in which we are conducting our research, namely a waste-exchange service. In Section 3, we present GhostWriter, our case-based approach to making content suggestions to authors. Section 4 contains a discussion, including a review of related work. Section 5 presents preliminary experimental results.

2 Waste Exchange Services: A Case Study

Waste exchange services connect organizations or households who have unwanted items with organizations or households who can use those items. The advantages of such a service include: it provides a way of diverting waste from landfill; it provides a way of saving on storage or disposal costs; and it provides a way of sourcing cheap or even free materials.

Many waste exchange services operate over the Web. Organisations or households use the web site to submit descriptions of the items that they have available. Other organisations or households either search and browse for items that they can reuse, or they submit descriptions of items and the waste exchange service automatically contacts them when such items become available. On-line waste exchange services are a great example of the Read/Write Web: their users engage in both search and authoring.

But, in a waste exchange service there are at least four reasons why an exchange may fail to take place:

- It may, of course, be that an unwanted item that is available through the service is not wanted by anyone else, or that an item that is requested through the service is not available from anyone else using the service.
- Surprisingly, the search facilities of these services are often quite rudimentary. They use simple search engines, which may fail to find matches between descriptions of items available and requested. Although not the focus of this paper, it is part of our work to use ideas from Information Retrieval and case-based retrieval to improve the search engine of the waste exchange service that we are working with.
- The descriptions of items available or wanted that users submit are often quite short, which reduces the likelihood that the search engine will find a match. In the waste exchange service with which we are working, for example, the average length of descriptions of items available is just 8 words or 6 words if we ignore stop-words; and the average length of descriptions of items wanted is just 6 words or 4 words if we ignore stop-words.
- The service may find a match between a pair of descriptions but, when their authors make personal contact, it may turn out that the match found by the service is spurious and does not satisfy at least one of the persons involved. Note how the short descriptions that we mentioned in the previous bullet point increase the likelihood of spurious matches. A transaction may

be abandoned when features that were not included in a description (maybe the colour, the price, the delivery terms, etc.) become known.

From this analysis, it seems useful to consider how a waste exchange service can support people in authoring better descriptions. There is an obvious source of experience that we can exploit: successful descriptions.

Many waste exchange services have transaction closure facilities. Consider a user who had submitted a description of an unwanted item, for example. When she deletes the description, the service shows a form that requires her to explain why the description is being deleted. She may have sold or given away the item through the waste exchange service; she may have sold or given away the item but not through the service; she may have disposed of the item (e.g. by sending it to landfill); or she may have failed to dispose of the item. We propose that descriptions of items that have been sold or given away through the waste exchange service should be retained in a case base. These are successful descriptions: ones that work.

We can use these successful descriptions to make suggestions. When a user is authoring a new description, we can prompt her to think about including certain kinds of content: content that we find in successful descriptions. We explain the details of the way we do this in the next section.

Figure 1 shows an overview of a waste exchange service that includes a suggestion facility of the kind we have described. The left-hand side of the figure represents a standard waste exchange service. Users who have items insert them into a database of items available, and search a database of items wanted; users who want items insert them into a database of items wanted, and search a database of items available. Transaction closure results in the update of statistics. But, as the right-hand side of the figure shows, we propose the service also inserts successful descriptions into a case base. Then the service can use successful descriptions of items wanted to make content authoring suggestions to users who are describing wanted items, and use successful descriptions of items available to make content authoring suggestions to users who are describing available items.

3 GhostWriter: Case-Based Content Authoring Suggestions

GhostWriter is the system that we have designed and built. It relies on feature extraction, which we apply in advance to successful descriptions (cases) and incrementally to the author's own description as she writes it. The mechanism we have designed and built for making the suggestions is novel but is inspired by Conversational CBR techniques.

Up to now, our implementation, built using jColibri,[2], is suitable only for running off-line experiments, preliminary results for which are described in Section 5. Ultimately, however, we plan to implement an Ajax client that will proactively

[2] `http://gaia.fdi.ucm.es/projects/jcolibri/`

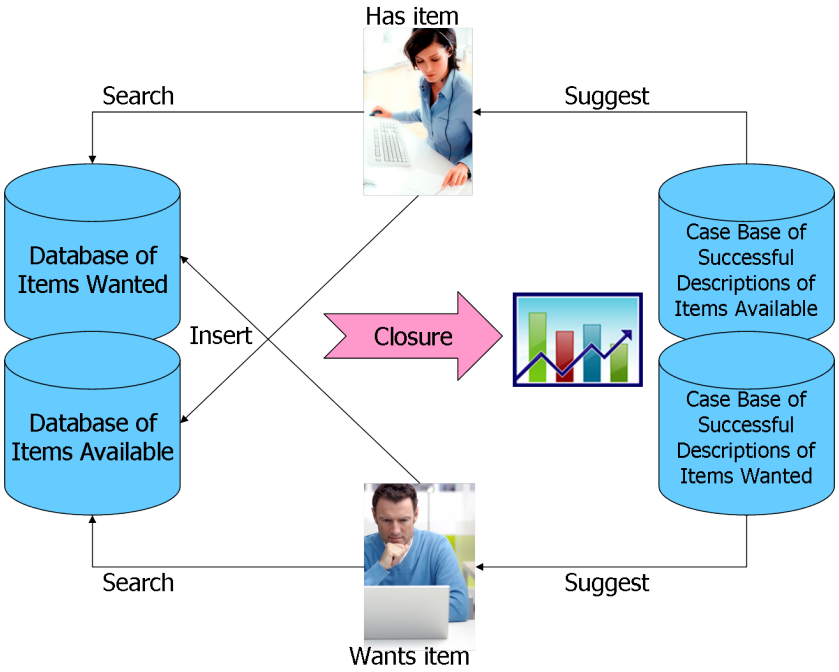

Fig. 1. A waste exchange service that includes a suggestion facility

send asynchronous requests to the server-side GhostWriter system: as the user's content grows, the client will send the new content in an asynchronous HTTP request to the server; the client will update a suggestions pane with GhostWriter's responses. In this way, the user is not interrupted from her normal work, either to invoke GhostWriter or to receive its results. The user can click on a suggestion in the suggestion pane if it is close enough to her current intentions and it will be incorporated into her content, where she can edit it. More likely, the suggestions will not be close enough to what is wanted but will prompt the user to include content she hadn't thought of including. For example, if one of the suggestions is "Will deliver within a 10 mile radius", this might prompt the user to include her own delivery terms, even if these are very different from the suggested ones. Hence, even if the use of suggestions means that descriptions more often have the same *features*, they may still be novel descriptions by virtue of not having the same *feature-values*.

3.1 Case and new item representation

As mentioned, a case is a successful description. It therefore primarily consists of free text. But, as it enters the case base, we apply Feature Extraction rules. For now in our work we produce these rules manually. Each is in essence a regular expression that aims to find and extract a particular feature-value pair

```
[ItemCondition]{0}(\w+\s+([Cc]ondition|[Qq]uality))
[ItemCondition]{0}(VGC|[Vv]gc|[Ww]orn|[Tt]or[en]|[Bb]roke|[Bb]roken)
```

Fig. 2. Example Feature Extraction rules in the format `[FeatureName]{FeaturePosition}RegularExpression`

from the text. Hence, the rules augment each case by a set, possibly empty, of feature-value pairs. Two example rules are shown in Figure 2. Both these rules extract the ItemCondition feature. The zero indicates that the rule extracts the entire expression. In the first rule, the regular expression matches phrases such as "excellent quality" and "very good condition"; the regular expression in the second rule matches 'stock' phrases and words for describing an item's condition such as "vgc" (very good condition), "worn", "torn", etc. (In GhostWriter, this rule has more disjuncts than are shown here.)

More formally then, a case c comprises some free text, $text(c)$, and a set of feature-value pairs, $fvs(c)$. We will denote a feature-value pair by $\langle f, v \rangle \in fvs(c)$. Note that cases do not comprise problem descriptions and solutions. There is no solution part to the cases. This is because making content authoring suggestions is in some sense a form of case completion [2]: we use cases to suggest content that the author might add to her description.

New items that the user is authoring have exactly the same representation as cases: free text and feature-value pairs. The only difference is that they grow in size, as the author adds to her content. We will denote a new item description as nid.[3]

3.2 Conversational case-based suggestions

The GhostWriter approach to making content authoring suggestions to the user is novel, but it is inspired by Conversational CBR (CCBR) [1]. In CCBR, a typical case has a problem description, comprising of a free text description and a set of question-answer pairs, and a problem solution, comprising a sequence of actions. This is very similar to our case representation, described above, except, as already mentioned, our cases have no solution component.

Aha et al's generic CCBR algorithm [1] starts with the user entering a free text query. Then the following repeats until the user selects a case or no further cases or questions can be suggested to the user: the system retrieves and displays a set of cases that are similar to the user's query; from these cases, the system ranks and displays a set of important but currently unanswered questions; then the user inputs more free text or answers one of the questions.

Figure 1 shows the GhostWriter approach to making content authoring suggestions. Recall that we invoke this algorithm repeatedly as the user's content grows. Each time we invoke it, it does the following:

[3] We avoid the word "query", which is more common in CBR, since we have found it leads to confusion.

Algorithm 1 GhostWriter's content authoring suggestion algorithm

Inputs: CB: case base
nid: new item description
k_1, k_2, k_3: number of cases, features and values, resp.

$R \leftarrow [\,]$
$C \leftarrow rank\_cases(nid, CB, k_1)$
$F \leftarrow rank\_features(C, nid, k_2)$
for each $f_i \in F$, taken in decreasing order **do**
  $V_i \leftarrow rank\_values(f_i, C, k_3)$
  insert $\langle f_i, v \rangle$ onto the end of R for each $v \in V_i$ taken in decreasing order
end for
return R

- It initializes the result R to the empty list.
- It retrieves k_1 cases C from the case base CB, ranking them on their similarity to the user's new item description nid. In fact, we compute similarity between the free text descriptions, $text(nid)$ and $text(c)$ for each $c \in CB$. It may be worthwhile to use a diversity-enhancing algorithm, e.g. [9], for this retrieval.
- From the cases retrieved in the previous step C, we obtain up to k_2 features F. Candidates for inclusion in F are all features in each $c \in C$, after removing duplicates and any feature that is already among the features of the user's item description $fvs(nid)$, irrespective of that feature's value in $fvs(nid)$. There are many ways of ranking these candidates. At the moment we use the simplest approach: frequency of occurrence across the cases in C. We place in F the k_2 features that have the highest frequency of occurrence.
- For each of the features obtained in the previous step $f_i \in F$, we obtain up to k_3 values for that feature V_i. Candidates for inclusion in V_i are all values for that feature in each of the cases $c \in C$, after removing duplicates. Again there are many ways to rank these candidates. At the moment, we use the original ranking of the cases C. In other words, if $\langle f_i, v \rangle \in fvs(c)$ and $\langle f_i, v' \rangle \in fvs(c')$ and $c \in C$ has higher rank than $c' \in C$, then v has higher rank than v'.
- We return the ranked list of up to k_2 features, each with their ranked list of up to k_3 values, for display to the user in the suggestion pane.

When the user makes sufficient change to nid, possibly by incorporating suggestions from the suggestion pane, we run GhostWriter again to make fresh suggestions. This continues until the user is satisfied with her description and submits it to the waste exchange service database.

4 Discussion

Our first goal in this section is to discuss related work. Several researchers have investigated ways of proposing completions for incomplete phrases and sentences,

representative of which are [4, 6]. This kind of work tends to focus on data structures for representing phrases and sentences in a way that supports fast matching with phrase and sentence prefixes. Lamontagne and Lapalme use CBR for the more challenging task of generating email replies [5]. But their work, and the work on phrase and sentence completion, is concerned with making suggestions in situations where there are 'stock responses'. We would argue that GhostWriter's task is different in nature. Its goal is to prompt the user to write a more comprehensive description. On occasion, 'stock responses' may be relevant, and the user may click on a suggestion to include it directly in her content. But just as likely, she will not accept any of the phrases (*feature-values*) that we suggest to her. Nevertheless, we hope that she will be prompted to include a phrase of her own, inspired by the *features* that we suggest.

In its goal, Recio-García et al's Challenger 1.0 system is much more similar to our own work [8]. Their system supports the author of air incident reports. However, their texts are longer and their techniques are quite different from ours. They have no feature-value pairs and do not draw ideas from CCBR. Instead, they use standard text retrieval coupled with clustering of the results.

Our second goal in this section is to discuss objections to what we have done. It might be objected that a simpler approach is the use of forms. These forms could include fields inviting the author to provide much more information than normal (e.g. the colour, the price, the delivery terms, etc.). But we think there are problems with a form-based approach: forms can become quite long; there is the difficulty of anticipating what fields to include on the form, although this could be solved by using Feature Extraction from successful descriptions; a form-based approach assumes greater regularity in the descriptions than our approach assumes; and a form-based approach may result in less distinctive descriptions, when in fact the author's real goal is to make her descriptions stand out. Our approach, by contrast, prompts the user dynamically, based on the current description and the content of related successful descriptions (cases).

Our final goal in this section is to discuss the generality of our approach. We have made our presentation more concrete by giving a context, namely waste exchange services. But we believe that the same approach can be used in any 'classified ads' service, where cars, jobs, housing, dates, and many other things are advertised. In fact, we intend doing experiments with data that we have scraped from craigslist.[4] We are also interested to apply our approach to support the authors of reviews of products such as hotels and electronic goods. The content of these reviews is even less predictable than that of classified ads, so our approach may then be even more promising than an approach based on form-filling. In the domain of product reviews, other users can often indicate whether they found a review to be useful or not. This is what we would use as a measure of whether a description is successful. It implies that the case base becomes a fuzzy set, where descriptions have different degrees of membership depending on how useful people have found them to be.

[4] `http://www.craigslist.org`

5 Experimental Evaluation

Here, we report the results of a preliminary, off-line ablation study. The results are promising, but they do show that we need to use a different dataset and we do need to make some changes to our experimental methodology.

From an operational waste exchange service, we took a set of 73 descriptions of items available. Most waste exchange services, including the one we work with, do not retain successful descriptions. Therefore, unfortunately, these 73 descriptions, which we use as a case base in this experiment, are not restricted to successful descriptions. As mentioned already, they are rather short: 8 words on average, and 6 if we exclude stop-words.

We use a leave-one-out methodology. We temporarily remove a case from the case base and delete a random proportion of its words. We treat this as the user's *nid*; the ablation simulates an incomplete description. We supply this *nid* to GhostWriter. GhostWriter is run with $k_1 = 10$ (the number of cases it retrieves), $k_2 = 2$ (the number of features it suggests), and $k_3 = 2$ (the number of values it suggests for each feature). Hence it returns up to four suggestions (two values for two features). We randomly select one of the suggestions and add the suggested feature-value to the *nid*. We keep doing this until GhostWriter is unable to make further suggestions. We repeat this for each case in the case base, and we repeat the whole procedure five times to average out differences that result from random ablation.

After we add a suggested feature-value to the *nid*, we measure the similarity between the current state of the *nid* and the original case from which we created the *nid*. We compute similarity using the standard cosine measure. The results are shown in Figure 3.

On the y-axis is similarity; on the x-axis is the number of feature-values that we have added to the *nid*. There are different lines according to the starting amount of ablation. For example, one line records what happens when we form the *nid* by ablating 20% of the original case; another plot measures what happens when there is 40% ablation; and so on. The x-axis goes up to 6. But GhostWriter will not make 6 suggestions for every *nid*. For some *nid*s, GhostWriter may run out of suggestions much earlier: if the features of the retrieved cases C are all already present in the *nid*, then GhostWriter can make no fresh suggestions. The percentages alongside each data point record this information. For example, on the line for 0% ablation, we were able to add one feature-value pair to 85% of *nid*s; we were able to add two feature-value pairs to 63% of *nid*s; three to 38%; four to 25%; and so on.

In interpreting these results, the question is: when we add suggested feature-values to a *nid*, are we restoring some of the original content that we ablated earlier? If this is so, then the suggestions are useful ones. In judging this, we must compare with the line for 0% ablation. When there is 0% ablation, any feature-values we add inevitably reduce the similarity between the *nid* and the original case: we are adding content that was not originally there. Provided the gradient in the other lines is not as steep as the gradient in the 0% ablation line, then we know that the content that we are adding is at least partly restoring

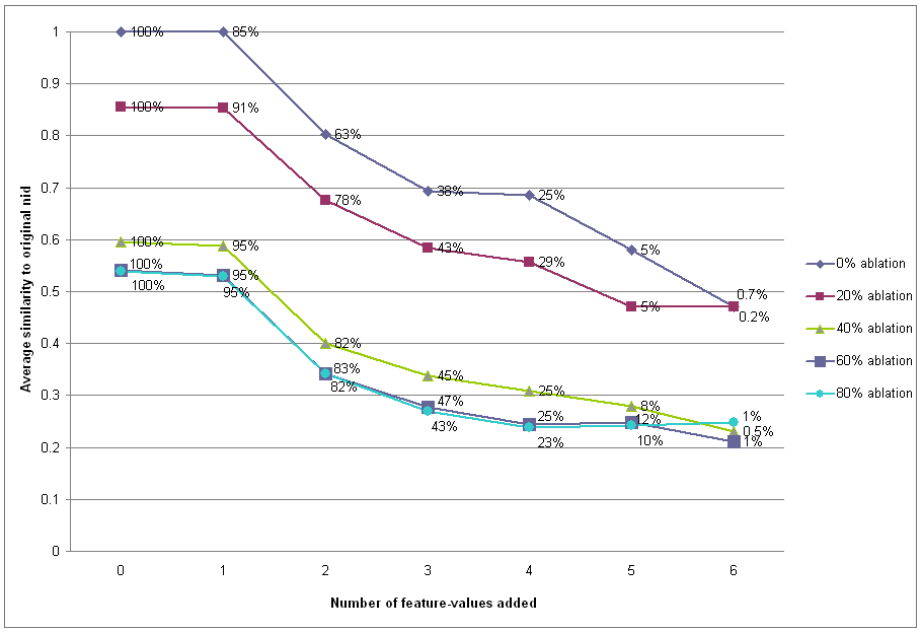

Fig. 3. Average similarity between the *nid* as we add suggested feature-values and the case from which it was created

ablated content. This does seem to be so: with 0% ablation similarity falls from from 1.0 to 0.47, i.e. by 0.53; for 20% ablation it falls from 0.86 to 0.47, i.e. by 0.39; for 40% ablation it falls by 0.37; for 60% it falls by 0.33; and for 80% it falls by 0.29. We believe this shows that the GhostWriter algorithm is promising.

The main lesson from this preliminary experiment, however, is that, going forward, we need to change the experimental set-up. We need a case base with more comprehensive descriptions, more akin to what we hope to find in a case base of successful descriptions. We also probably need to take item category into account so that when GhostWriter suggests content to someone describing a desk, it should only use feature-values that come from other descriptions of furniture, and not from descriptions of electrical appliances, for example. In the experiment at present, this restriction is not in place. At the moment also the ablation deletes a proportion of a case's words at random. It may be a fairer experiment to delete a proportion of a case's phrases (i.e. its existing feature-values) and to use a similarity measure that rewards GhostWriter the earlier it suggests the right kinds of features, even if the suggested feature-values do not match those in the original case.

6 Conclusions and Future Work

In this paper, we have argued that the Web contains experience in the form of successful descriptions, which we can treat as cases in making suggestions to the authors of new content. We have presented a concrete scenario, that of a waste exchange service, where this perspective can be useful. We have presented a novel algorithm, implemented in the GhostWriter system, for making these suggestions, inspired by work in Conversational CBR. And we have reported some promising preliminary results.

This is early-stage research, with many lines of future inquiry. In particular, we want to apply the idea in other domains, especially classified ads and product reviews. We want to try some of the many ways of learning the Feature Extraction rules, see, e.g., [3]. We want to investigate variants of the algorithm, where we use different ways of ranking the cases, features and feature-values. We mentioned, for example, the use of diversity-enhanced methods for retrieving the cases. We want to use a different dataset and make some changes to the methodology in our off-line ablation study. Finally, we want to carry out evaluations with real users.

References

1. David W. Aha, Leonard A. Breslow, and Héctor Muñoz-Avila. Conversational case-based reasoning. *Applied Intelligence*, 14:9–32, 2001.
2. Hans-Dieter Burkhard. Extending some concepts of CBR — Foundations of case retrieval nets. In M. Lens et al, editor, *Case-Based Reasoning Technology: From Foundations to Applications*, pages 17–50. Springer, 1998.
3. Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74, 2008.
4. Korinna Grabski and Tobias Scheffer. Sentence completion. In M. Sanderson et al, editor, *Procs. of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 433–439. ACM Press, 2004.
5. Luc Lamontagne and Guy Lapalme. Textual reuse for email response. In P. Funk and P. A. González Calero, editors, *Procs. of the 7th European Conference on Case-Based Reasoning*, LNCS 3155, pages 234–246. Springer-Verlag, 2004.
6. Anranb Nandi and H. V. Jagadish. Effective phrase prediction. In C. Koch et al, editor, *Procs. of the 33rd International Conference on Very Large Data Bases*, pages 219–230. ACM Press, 2007.
7. Enric Plaza. Semantics and experience in the future web. In K.-D. Althoff, R. Bergmann, M. Minor, and A. Hanft, editors, *Procs. of the 9th European Conference on Case-Based Reasoning*, LNCS 5239, pages 44–58. Springer Verlag, 2008.
8. Juan A. Recio-García, Belén Díaz-Agudo, and Pedro A. González-Calero. Textual CBR in jCOLIBRI: From retrieval to reuse. In D. C. Wilson and D. Khemani, editors, *Procs. of the Workshop on Textual Case-Based Reasoning, 7th International Conference on Case-Based Reasoning*, pages 217–226, 2007.
9. B. Smyth and P. McClave. Similarity vs. diversity. In D. W. Aha and I. Watson, editors, *Procs. of the 4th International Conference on Case-Based Reasoning*, LNCS 2080, pages 347–361. Springer, 2001.