The GhostWriter-2.0 System: Creating a Virtuous Circle in Web 2.0 Product Reviewing

Paul Healy and Derek Bridge

Department of Computer Science,
University College Cork,
Ireland
pjh1@student.cs.ucc.ie,d.bridge@cs.ucc.ie

Abstract. In this paper, we present GhostWriter-2.0, a Case-Based Reasoning system that supports a user who is writing a product review. GhostWriter-2.0's case base is populated with relevant and helpful reviews from Amazon. It extracts phrases from these reviews and offers these as suggestions to the user. In a trial, users made considerable use of the suggested phrases.

1 Introduction

A lot of user-generated content on the Web takes the form of records of *personal experiences* [5, 7, 6]. Epitomizing these records of personal experience are the reviews and ratings that users contribute on everything from movies to books to music to hotels to consumer goods to professional services and to on-line content, whether user-generated or otherwise. Interestingly, Web sites that solicit user reviews often solicit meta-experiences too: users can review or rate other reviews or reviewers. For example, Amazon users vote on reviews, so the site can report review helpfulness; Epinions users additionally vote on reviewers, so the site can report reviewer trustworthiness.[1] Meta-experience like this provides a partial remedy for the problem of how to ignore noisy reviews.

Once shared on the Web, one user's experiences can be reused by another user to support her in some task that she is performing. Case-Based Reasoning (CBR), often defined as reasoning from experiences [4], offers one way to support such a user. Examples of using CBR in this way include the Poolcasting system, which uses CBR to select music to play in social situations [6]; the work of [1], which uses CBR to recommend data visualizations; and CommunityCook, which extracts recipe adaptation knowledge from Web-based recipe corpora [2].

Our position is that there are in fact at least two ways in which records of experience on the Web can be reused:

– to support the user in her real-world task, e.g. booking a hotel, selecting music to play, installing software, and visualizing data; and
– to support the user when she authors new content, e.g. writing a new review.

[1] www.amazon.com, epinions.com

In our own work, we have been developing the GhostWriter family of systems, which uses CBR in the second of these two ways. In GhostWriter-1.0, we showed how CBR could help the user to write more comprehensive descriptions in Web-based exchange systems [8]. GhostWriter-2.0, described for the first time here, is our system which uses CBR to help the user to write reviews.

We developed GhostWriter-2.0 in response to the WebCBR Challenge associated with this workshop.[2] The Workshop Organizers challenged CBR researchers to apply CBR ideas to the Experience Web, the only constraint being that the researchers should "showcase their research in the form of a demonstration system using Amazon's API". The GhostWriter-2.0 case base is populated with existing relevant and high quality Amazon product reviews; and the system uses the case content to assist the user to author a new review.

GhostWriter-2.0 has the potential to create a virtuous circle: if its assistance results in the user writing and submitting a better review than she otherwise would, then a new higher quality record of experience becomes available both to users making purchasing decisions but also to GhostWriter-2.0 next time it assists someone to write a review of a similar item.

Section 2 presents GhostWriter-2.0 from the user point of view; section 3 describes how GhostWriter-2.0 converts Amazon reviews into cases; section 4 explains how GhostWriter-2.0 populates its case base, and how it uses the cases to make suggestions; section 5 reports the results of a real user trial.

2 An End-User View of GhostWriter-2.0

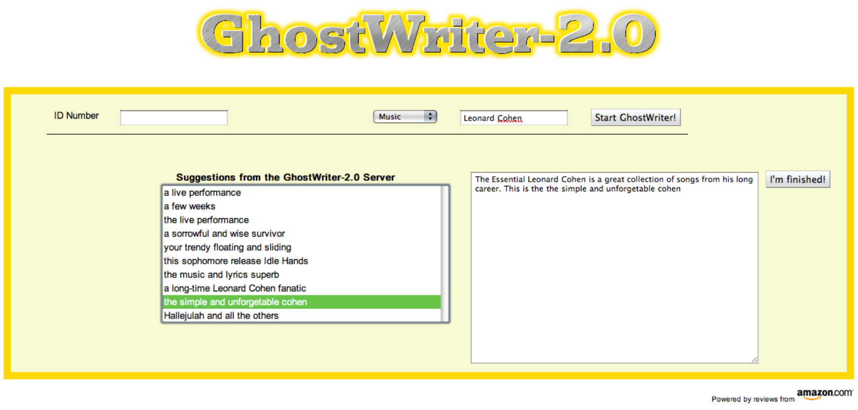
A user who wishes to review an Amazon product with support from GhostWriter-2.0 accesses GhostWriter-2.0 through her browser. We give a screenshot in Figure 1. The user starts at the top of the screen by using the drop-down select list to choose the category of product that she will review; in the example, the user has chosen Music. In the adjacent text-field, she enters keywords to describe the product. This might be the author of a book, the artist for a music CD, or the manufacturer of a digital camera, for example; in the screenshot, the user has entered "Leonard Cohen".[3]

The user next presses the *Start GhostWriter!* button. The GhostWriter-2.0 server takes the user's category and keywords and uses them to query the Amazon information servers. The search returns products and their reviews. These are not shown to the user. Instead, we use the best of these reviews to populate the case base (see Section 4.1).

The user may now type her review into the right-hand text-area. After approximately every 60 characters that the user types, GhostWriter-2.0 will make a set of suggestions, which the browser displays in the selection form to the left of the text-area. GhostWriter-2.0 selects these suggestions from the case base (see Section 4.2).

[2] `www.comp.dit.ie/aigroup/?page\_id=549`

[3] The text-field in the top-left of the screenshot allows us to identify users when we are running user trials (Section 5) and is otherwise not used.



Leonard Cohen

Suggestions from the GhostWriter-2.0 Server

- a live performance
- a few weeks
- the live performance
- a sorrowful and wise survivor
- your trendy floating and sliding
- this sophomore release Idle Hands
- the music and lyrics superb
- a long-time Leonard Cohen fanatic
- the simple and unforgetable cohen
- Hallejulah and all the others

The Essential Leonard Cohen is a great collection of songs from his long career. This is the the simple and unforgetable cohen

Fig. 1. A screenshot from GhostWriter-2.0

A user, glancing at the suggestions, may take one of three actions:

- She might decide to include a suggestion exactly 'as is' in her review. To do this, she double-clicks on the suggestion.
- She might decide that none of the suggestions can be included verbatim, but one or more of them may inspire her to write on a related topic.
- She may find none of the suggestions helpful. In this case, she continues to type her review.

When she has finished her review, she presses the *I'm finished!* button to submit her review to the GhostWriter-2.0 server, where statistics about the review are logged. Unfortunately, the Amazon API appears not to allow the GhostWriter-2.0 server to directly upload the review to Amazon.

3 Cases in GhostWriter-2.0

Fundamentally, cases in GhostWriter-2.0 are Amazon product reviews. We convert an Amazon product review r into a three-part GhostWriter-2.0 case $c = \langle W, S, h\rangle$ as follows. The first part, W, which can be thought of as the problem description part of the case, is the set of words that occur in the text of review r. Information about which product is being reviewed is not needed in the problem description part of the case because this contextual focus is already provided by the case base as a whole (see Section 4.1). The second part, S, which can be thought of as the problem solution part of the case, is a set of suggestions, i.e. a set of phrases that GhostWriter-2.0 might show to the author of a new review. We explain how we extract these from r in the paragraphs below. The third part of a case, h, is a non-negative integer. Recall that Amazon users can vote to signal that they found a review to be helpful. h is r's helpfulness rating. It can be thought of as the outcome part of the case [4]. It gives an indication of c's quality and, indirectly, an indication of the quality of the suggestions in S.

We have yet to explain how we extract S from r. In GhostWriter-1.0, suggestions were feature-value pairs [8]. There, we were working in the domain of Web-based exchange systems, such as classified ads systems, and so each case described an item that someone wanted to dispose of. We used regular expressions to extract the value of features such as Price (e.g. "€25 or nearest offer"), Condition (e.g. "in excellent condition"), and so on. This was adequately effective because classified ads tend to be written in a domain-specific sub-language [3] that is quite restrictive and uses a lot of recurring 'stock' phrases and phrasing.

Product reviews are written in a much less restrictive way than classified ads. Reviews of novels, for example, rarely mention the price but often give a synopsis of the plot, make comparisons with other novels by the same author or similar authors, describe the passions aroused when reading the book, and offer opinions about the quality of the writing. The author of a new review might welcome suggestions that span all of these. Information Extraction techniques, whether based on regular expressions or something more powerful, could extract simple features such as the price, if present, or the technical attributes of digital electronics goods (e.g. their battery life). Sentiment Analysis techniques could extract the polarity of the opinions expressed, with the risk that it would mine suggestions that would bias new reviews towards the majority opinions of past reviews in the case base. But, even if we were to use both of these techniques together, we would extract only a subset of the potentially useful suggestions.

Hence, we decided to take an approach that was less specific than either Information Extraction or Sentiment Analysis. We took the view that most of the descriptive content of a review lies in its noun phrases. They can cover: technical attributes of consumer goods; characters, places and events in books and films; authors and artists; and some of the opinion.

Noun phrases are easy to extract from text, using just quite shallow natural language processing techniques. We use OpenNLP's maximum-entropy-based 'chunking'.[4] We retain only those phrases that it labels as noun phrases. In fact, we decided to retain only noun phrases that were longer than two words (because shorter noun phrases tend to lack enough descriptive content to make useful suggestions) but shorter than six words (because longer noun phrases tend to make overly specific suggestions). Note that there is an additional scoring process that will determine which of a case's suggestions are actually shown to the user (see Section 4.2).

In summary then, in GhostWriter-2.0 an Amazon review r becomes a case c and comprises: the set of words in r; the set of r's noun phrases that contain three to five words; and r's helpfulness rating.

4 How Ghostwriter-2.0 Works

The GhostWriter-2.0 server sits between the user's browser and the Amazon servers. There are two phases to its processing: populating the case base, and making suggestions.

[4] `opennlp.sourceforge.net`

4.1 Populating the case base

As explained in Section 2, a user who wishes to write a review with support from GhostWriter-2.0 begins by entering data about the product she wishes to review: she enters its category and some keywords into her browser and submits them to the GhostWriter-2.0 server. The GhostWriter-2.0 server forwards this data to the Amazon servers. The operation it requests is an Amazon `ItemSearch` for the given category and keywords, which returns up to 4000 products in pages of up to ten products each. We additionally request that, for each product, the `ItemSearch` returns reviews, ordered by decreasing helpfulness. It returns up to five reviews per product, and it these we use to populate the case base.

We aim to populate the case base with 250 cases. As a way of trying to avoid duplicated products and reviews, we check customer ids and only take the first review for each customer. If the `ItemSearch` fails to provide us with 250 cases, we take each product in turn and use an Amazon `ItemLookup` to obtain a further 10 reviews, again checking customer ids before inserting them into the case base.

Note that this means that the case base is populated afresh each time a user starts a new review. This ensures that its contents are relevant to the current product being reviewed, and that the case base takes into account the very latest data (products, reviews and helpfulness ratings) on the Amazon site. The downside is that there is an appreciable delay (up to 2 minutes) in inserting reviews into the case base. This is caused by the time it takes to launch OpenNLP and to use it to extract noun phrases from the 250 reviews. A production version of Ghostwriter-2.0 would need to be better integrated with Amazon to bring this time down.

4.2 Making suggestions

After GhostWriter-2.0 has populated its case base, the user starts typing her review into her browser. As explained in Section 2, after approximately every 60 characters that the user types, the browser uses AJAX to request a set of suggestions from the GhostWriter-2.0 server. The browser supplies the server with the current contents of the user's review. The GhostWriter-2.0 server retrieves $k_1 = 50$ cases from the case base. From the suggestions contained in these cases, it selects $k_2 = 10$ suggestions, which it returns to the user. We explain both the retrieval and the selection in more detail below.

Retrieval. Let the current contents of the user's review be called the *new product review* and designated *npr*.[5] This is the set of words that the user has typed. GhostWriter-2.0 retrieves from the case base the k_1 cases that are most similar to the *npr*. We measure similarity using the Jaccard similarity coefficient:

$$sim(npr, c = \langle W, S, h \rangle) = \frac{|npr \cap W|}{|npr \cup W|}$$

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

Suggestion selection. At this point, GhostWriter-2.0 has retrieved k_1 cases C from the case base, and in each case $c = \langle W, S, h \rangle$ there is a set of suggestions S; each suggestion is a noun phrase. GhostWriter-2.0 must select the k_2 suggestions that it will send back to the browser for display. When considering a unique candidate suggestion, $s \in \bigcup_{\langle W,S,h\rangle \in C} S$, several criteria seem relevant, discussed over the next few paragraphs.

Consider the number of retrieved cases in which a suggestion s appears:

$$\mathit{freq}(s) = |\{\langle W, S, h \rangle \in C : s \in S\}|$$

To some extent, the more frequent s is, the more its content is something that different reviewers want to talk about, and this makes it a better suggestion. However, frequency favours short suggestions as these are the ones that are more likely to recur. The downside of this is that short, recurring noun phrases are more likely to be vague or generic.

Suppose instead we consider the length in words of the suggestions, $\mathit{length}(s)$. To some extent, the longer s is, the more descriptive it is, and this makes it a better suggestion. But length alone may favour overly specific suggestions.

We found that the product of frequency and length offered a good balance between the two criteria:

$$\mathit{score}(s) = \mathit{freq}(s) \times \mathit{length}(s)$$

The short, three-word phrase "a fantastic album" needs to appear in two separate cases ($2 \times 3 = 6$) if it is to have a score that betters that of the five-word phrase "an excellent, lively rock classic", assuming this appears in only one case ($1 \times 5 = 5$).

Many suggestions share the same score. To break ties, we sum and compare the helpfulness of the reviews that contain the two suggestions. Formally, if $\mathit{score}(s) = \mathit{score}(s')$, then s will be ranked higher than s' if

$$\sum_{h_s \in \{h : \langle W,S,h\rangle \in C \wedge s \in S\}} h_s \quad > \sum_{h_{s'} \in \{h' : \langle W',S',h'\rangle \in C \wedge s' \in S'\}} h_{s'}$$

(If s and s' have the same total helpfulness, then tie-breaking is arbitrary.)

In addition to this scoring, we also want to apply other criteria.

We do not want to make a suggestion s if it is already a noun phrase in the user's *npr*. This is quite simple-minded. In the future, it may be worth considering ways of measuring the similarity of semantic content: we would discard s if the *npr*'s content was similar enough to cover the content of s.

We also do not want to make a suggestion s if it is one that we have made several times already. This criterion is not one that we built into an early version of GhostWriter-2.0. But in a small-scale pilot study that we conducted prior to the user trial reported in Section 5, we found that users preferred versions of the system that more often made fresh suggestions over versions that allowed suggestions to linger. We subsequently decided to limit the number of times a suggestion could be made. Specifically, if s has already been suggested $\theta = 4$ times, then it cannot be suggested again, allowing another suggestion to take its place. This increases the number of different suggestions that get made, but the suggestion turnover is not so great as to be distracting to the user.

5 Experimental Evaluation

Here, we report the results of a user trial. For comparison purposes, we developed a version of GhostWriter-2.0 that made less intelligent use of the cases. It populates its case base in the same way as GhostWriter-2.0 (Section 4.1). So we know it has a set of relevant and helpful cases. But instead of retrieving the $k_1 = 50$ cases that are most similar to the *npr*, it retrieves $k_1 = 50$ cases at random from the case base. Like GhostWriter-2.0, it will not make suggestions that are already contained in the *npr*, nor will it make a suggestion more than $\theta = 4$ times. But otherwise, the $k_2 = 10$ suggestions that it makes are drawn at random from the retrieved cases: it does not use the *score* function, nor the helpfulness ratings. In this section we will refer to these two systems as GW-2.0 (for GhostWriter-2.0) and GW-R (for the more random version).

The two systems were evaluated by twenty users. Each user reviewed two products with which they were familiar, either two music CDs or two books. As they wrote their reviews, they received suggestions from one of the GhostWriter systems. Ten users had help from GW-2.0 for their first review, and help from GW-R for their second review. The other ten users had help from GW-R first, and then GW-2.0. They were unaware of the difference between the systems.

During the experiment, GW-2.0 and GW-R recorded: the review the user typed; the time taken to write the review; and the suggestions that were explicitly incorporated, i.e. when the user double-clicked on a suggestion. We also administered a questionnaire to each participant.

At one level, reviews written with support from GW-2.0 and with support from GW-R were similar. Their average length was a little over 150 words in both cases; users created their reviews at an average of 13–14 words per minute in both cases; and the average number of descriptive noun phrases in the reviews (which we took to be noun phrases of more than two words) was also 13–14 in both cases. But what is interesting is the breakdown of those noun phrases.

Figure 2 shows, for each user (designated A to T), how many suggestions they directly used (by double-clicking on them). In total, 116 GW-2.0 suggestions were directly incorporated, an average of 5.8 per user, compared with only 83 GW-R suggestions, an average of 4.15. Fourteen of the twenty users used more GW-2.0 suggestions than GW-R ones; one used the same number of suggestions from both; and one used none from either system. We take this as an indication that GW-2.0's greater reuse of Web-based experience data (case similarity, frequency across reviews, and helpfulness) promotes more useful suggestions.

Figure 2 gives no indication of how many other descriptive noun phrases there are in user reviews. Figure 3 shows this in the case of GW-2.0. It enables us to see that, while users vary, directly incorporated suggestions account on

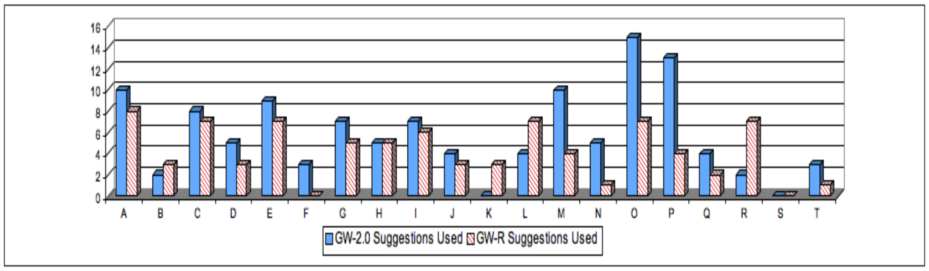


Fig. 2. The number of suggestions that users directly incorporated into reviews

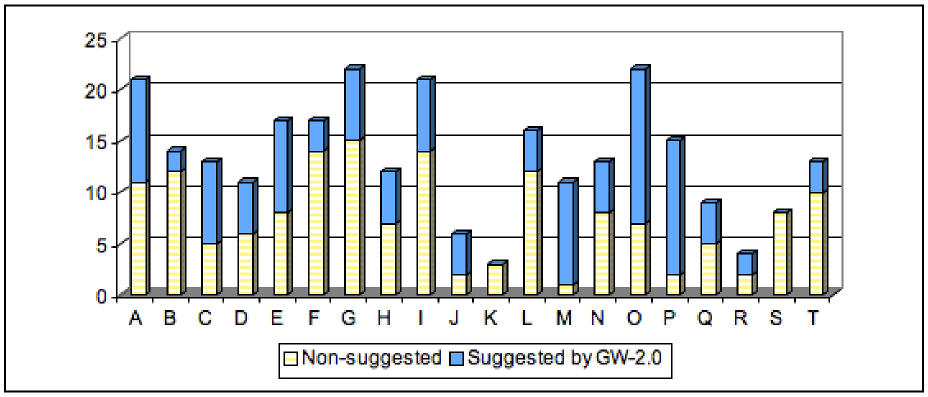


Fig. 3. Noun phrases in reviews written with GW-2.0

average for about 43% of descriptive noun phrases (116 out of the 268 noun phrases of more than two words). We do not have space to show the graph for GW-R but on average only about 30% of descriptive noun phrases were ones GW-R suggested (83 out of 275 noun phrases of more than two words).

Figures 4, 5 and 6 summarize results from the questionnaire. Nineteen of the twenty people agreed or strongly agreed that GW-2.0 helped them to write a comprehensive review; for GW-R, sixteen people agreed or strongly agreed (Figure 4). Eighteen people agreed or strongly agreed that GW-2.0's suggestions were helpful, more than half of them strongly agreeing; for GW-R, sixteen people agreed or strongly agreed, only a minority strongly agreeing (Figure 5). We also asked participants to estimate for each review that they wrote how many times they were inspired by a suggestion but did not actually double-click on it and so did not incorporate it directly. This was obviously very subjective and not likely to be reliable. But it at least gives an impression of the extent to which the numbers reported in Figures 2 and 3 understate the helpfulness of the systems. According to the responses to this (Figure 6) GW-2.0 and GW-R were fairly evenly-matched in their ability to inspire, and inspired their users one or more times in all but three reviews.

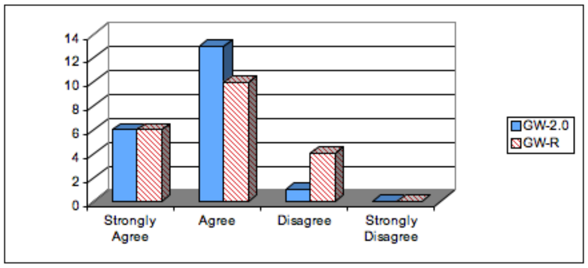


Fig. 4. "The system helped me to write a comprehensive review"

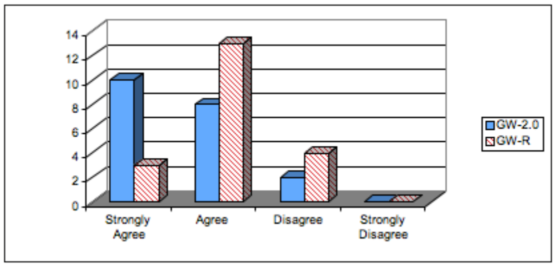


Fig. 5. "I found the suggestions helpful"

We also asked participants for their view of the quality of the reviews they found in on-line stores: nineteen out of twenty thought they were Good or Fair; no-one thought they were Excellent; no-one thought they were Bad; but one person thought they were Terrible. Finally, asked whether they would write more comprehensive reviews if a GhostWriter-style system were available on a store's Web site, everyone agreed or strongly agreed.

6 Conclusions and Future Work

We have presented GhostWriter-2.0, which reuses the experience captured in Amazon product reviews and their meta-review data to make suggestions to a user who is writing a new product review. Our user trial has very promising results, showing a high level of use of the suggestions, both directly and indirectly. In the trial, GhostWriter-2.0 was more helpful to users than a less sophisticated version with an element of randomness to its selections. The differences, however, were small. This may be because, irrespective of which system was being used, users in the trial relied on the suggestions to a degree greater than they would in reality, where they would be writing reviews for endogenous reasons, rather than at our behest.

There are many lines of future research. We would like to measure and, if necessary, explicitly enhance the diversity of the suggestions that GhostWriter-2.0 makes, for example. We would also like to see GhostWriter-2.0 more closely

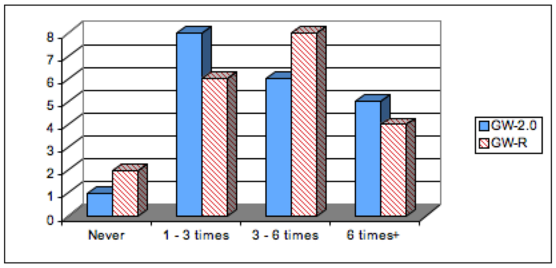


Fig. 6. “The number of times I was inspired by suggestions but I didn’t click on them”

integrated with the on-line store that it supports, as this can both reduce the time it takes to populate the case base and make the system less cumbersome to use (e.g. by automatically taking the category and keywords from the page the user is visiting). Finally, we are keen to extend the GhostWriter family into new domains where authors can benefit from the kind of support that GhostWriter systems can offer.

References

1. Jill Freyne and Barry Smyth. Many Cases Make Light Work for Visualization in Many Eyes. In D. Bridge, E. Plaza, and N. Wiratunga, editors, *Procs. of WebCBR: The Workshop on Reasoning from Experiences on the Web (at the 8th ICCBR)*, pages 25–44, 2009.
2. Norman Ihle, Alexandre Hant, and Klaus-Dieter Althoff. Extraction of Adaptation Knowledge from Internet Communities. In D. Bridge, E. Plaza, and N. Wiratunga, editors, *Procs. of WebCBR: The Workshop on Reasoning from Experiences on the Web (at the 8th ICCBR)*, pages 35–44, 2009.
3. Richard Kittredge and John Lehrberger. *Sublanguage: studies of language in restricted semantic domains*. de Gruyter, 1982.
4. Janet L. Kolodner. *Case-Based Reasoning*. Morgan Kaufmann, 1993.
5. 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.
6. Enric Plaza and Claudio Baccigalupo. Principle and Praxis in the Experience Web: A Case Study in Social Music. In D. Bridge, E. Plaza, and N. Wiratunga, editors, *Procs. of WebCBR: The Workshop on Reasoning from Experiences on the Web (at the 8th ICCBR)*, pages 55–63, 2009.
7. Barry Smyth and Pierre-Antoine Champin. The Experience Web: A Case-Based Reasoning Perspective. In S. Craw, D. W. Aha, S. Singh, and B. Smyth, editors, *Procs. of the Workshop on Grand Challenges for Reasoning From Experiences (at the 21st IJCAI)*, pages 53–61, 2009.
8. Aidan Waugh and Derek Bridge. An Evaluation of the GhostWriter System for Case-Based Content Suggestions. In L. Coyle, J. Dunnion, and J. Freyne, editors, *Procs. of the 20th Irish Conference on Artificial Intelligence and Cognitive Science*, pages 264–273, 2009.