An Evaluation of the GhostWriter System for Case-Based Content Suggestions

Aidan Waugh and Derek Bridge

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

Abstract. The Web has many sites where users can exchange goods and services. Often, the end-users must write free-text descriptions of the goods and services they have available, or the goods and services they are seeking. The quality of these descriptions is often low. In this paper, we describe the GhostWriter system, which encourages users to write descriptions that are more comprehensive. The system makes content suggestions from a case base of successful descriptions. The paper describes a new off-line ablation study that we have carried out to evaluate the system. The results show that GhostWriter has a high success rate in making suggestions that quickly recover ablated content.[1]

1 Introduction

Trade, the exchange of goods and services, has been a feature of human existence from prehistoric times. It has evolved from simple bartering, through the introduction of money, to the development of long-distance commerce. And now the Web is allowing dis-intermediation: potential trading partners can find each other, despite being geographically distributed, and without the intervention of human intermediaries such as sales and distribution agents.

The Web contains many examples of what we will here call *Web-based exchange services*. In these services, users who have items that they wish to trade will insert descriptions into a database of items available, and search a database of items wanted; users who want items will insert descriptions into a database of items wanted, and search a database of items available. (See the left-hand side of Figure 1.) This, in high-level terms, describes the operation of classified ad services such as craigslist; on-line auction and shopping sites such as ebay; waste exchange services such as Macroom-E's Wastematchers, where the emphasis is on diverting waste products from landfill; recruitment services such as Monster; property services such as Daft.ie; and so on.[2] However, it remains a challenge to discover matches through these services.

[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.

[2] www.craigslist.org; www.ebay.com; www.wastematchers.com; www.monster.ie; www.daft.ie

L. Coyle and J. Freyne (Eds.): AICS 2009, LNAI 6206, pp. 262–272, 2010.

Recommender systems, especially *case-based* recommender systems, are already successfully addressing the challenge of finding more and better matches in Web-based exchange services [10]. However, the matches these recommender systems find can only be as good as the descriptions upon which they operate. Matching is easier when descriptions have a regular structure, style or vocabulary. A corporate product catalogue that contains a relatively homogeneous set of items, each described by a domain expert, is a case in point. For example, in the catalogue of operational amplifiers sold by Analogue Devices, engineers describe the op amps using 40 strongly-typed features, e.g. the total supply voltage, which is a real-valued feature [11]. But in the kinds of Web-based exchange services that we have been describing above, these conditions generally do not hold. Items are often not homogeneous; descriptions are typically written by end-users, not by experts; there are few, if any, strongly-typed features; and there is correspondingly high reliance on free text.

In our research, we are investigating how Web-based exchange services can support end-users to write more comprehensive descriptions. We take a case-based approach. There is an obvious source of case base experience that we can exploit: successful descriptions. A successful exchange of an item through a Web-based exchange service implies not just that an item was desirable to someone, but that its description was informative enough to have initiated a transaction. Such a description forms the basis of a case in our case base.

In Section 2, we present GhostWriter, our case-based approach to making content suggestions to authors. Section 3 presents the results of an off-line evaluation of GhostWriter. Section 4 is a concluding discussion.

2 The GhostWriter System

In this section, we summarize our GhostWriter system, which makes content authoring suggestions to the user [2]. Figure 1 shows an overview of a Web-based exchange service that includes a suggestion facility of the kind we have described. The left-hand side of the figure represents a standard Web-based exchange service. But, as the right-hand side of the figure shows, we propose that the service also inserts successful descriptions into a case base. Then the service can use successful descriptions of items that are wanted to make content authoring suggestions to users who are describing wanted items, and use successful descriptions of items that are available to make content authoring suggestions to users who are describing available items.

GhostWriter is the system that we have designed and built for making these suggestions. 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. Up to now, our implementation, built using jColibri,[3] is suitable only for running off-line experiments. Ultimately, however, we plan to implement an Ajax client that will proactively send asynchronous requests to the server-side GhostWriter system: as the user's content grows, the client will send the new

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

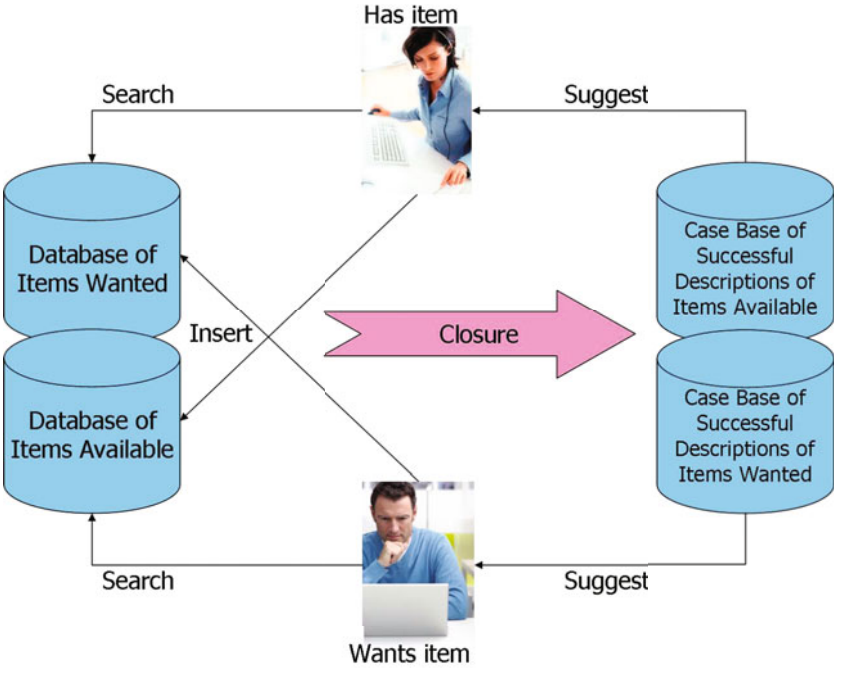


Fig. 1. A Web-based exchange service that includes a suggestion facility

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.

2.1 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 from the text. Hence, the rules augment each case by a set, possibly empty, of feature-value pairs. For example, we have rules for extracting an item's price, condition (e.g. excellent condition, good as new, etc.), colour, manufacturer, and so on. For example, a case might contain a free text description such as "White wooden

cot with mattress...Excellent condition...Will collect". From this, we might extract quite generic feature-value pairs such as $\langle CONDITION, excellent \rangle$ and $\langle DELIVERY, Will\ collect \rangle$ and more specific ones such as $\langle COLOUR, white \rangle$ and $\langle MATERIAL, wooden \rangle$.

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 [3]: 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.[4]

2.2 Conversational Case-Based Suggestions

The GhostWriter approach to making content authoring suggestions to the user is inspired by Conversational Case-Based Reasoning (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:

- 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 cosine similarity between term vectors that represent the free text descriptions, $text(nid)$ and $text(c)$ for each $c \in CB$.
- 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

[4] We avoid the word "query", which is more common in Case-Based Reasoning, 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

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 $c \in C$ is the highest ranked case for which $\langle f_i, v \rangle \in fvs(c)$ and $c' \in C$ is the highest ranked case for which $\langle f_i, v' \rangle \in fvs(c')$ and c has higher rank than 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 exchange service database.

3 Experimental Evaluation

We previously reported the results of a preliminary evaluation of GhostWriter in [2]. However, there we used only one data-set and our experimental methodology had several weaknesses. Here we present the results of a new evaluation using three data-sets and an improved methodology.

One of the data-sets we use comes from the wastematchers.com waste exchange service. We scraped the other two data-sets from craigslist: one describes various computer equipment available; the other describes furniture. We summarize their characteristics in Table 1.

Unfortunately, the descriptions that we took from wastematchers.com and from craigslist, which we use in the case bases in this experiment, are not restricted to successful descriptions. Neither wastematchers.com nor craigslist

Table 1. Data-set characteristics

Data-set name	Num. cases	Avg. num. words (incl. stop-words)	Avg. num. words (excl. stop-words)	Avg. num. feature-values
wastematchers	88	6.67	4.76	0.66
craigslist-computers	68	121.26	80.00	5.00
craigslist-furniture	80	94.25	55.52	7.00

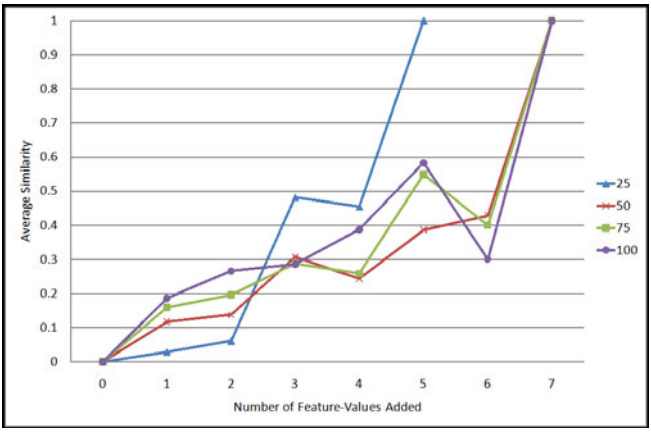
stores information about successful exchanges. In wastematchers.com, when a user deletes a description, the service shows a form that requires her to explain why the description is being deleted. She may have sold or given away the item through wastematchers.com; she may have sold or given away the item 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. At present, responses to this form allow wastematchers.com to report summary statistics. It would be a small step to retain in a case base (perhaps after expert review) descriptions of those items that were sold or given away through the service. But this has not been a characteristic of the system's existing operation. Our inability to obtain only successful descriptions may make the experimental results different from what they could be in practice.

We use a leave-one-out methodology. We temporarily remove a case from the case base, we determine what feature-values it contains, and we delete a random proportion of the feature-values. We treat this as the user's *nid*; the ablation simulates an incomplete description. We supply this *nid* to GhostWriter. GhostWriter returns an ordered set of suggestions. We randomly select one of the suggestions, where the probability of selection is greater for those suggestions that GhostWriter ranks higher. We add the selected feature-value to both the text and feature-values of the *nid*, and move on to making the next suggestion. 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.

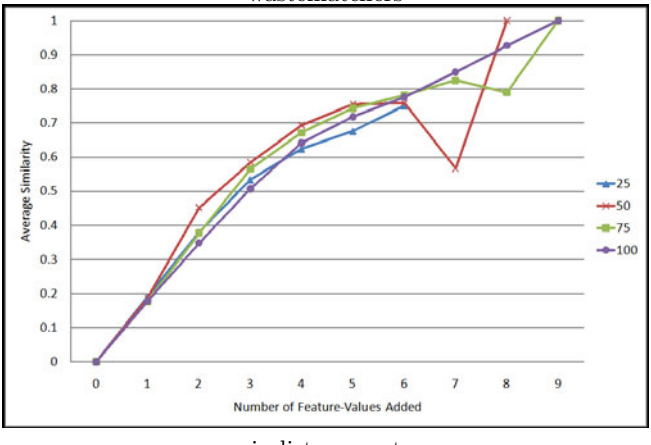
After we add a suggested feature-value to the *nid*, we measure the similarity between the current state of the *nid* and the original case from which we created the *nid*. We measure similarity as the proportion of the ablated *features* that have been restored to the *nid*, irrespective of their *values*. In other words, we reward GhostWriter for making the right kind of suggestion (e.g. price, condition, delivery terms, etc.), even if the text of that suggestion is not the same as what was ablated. Formally, let *Ablated* be the set of ablated feature-values and let *Selected* be the set of feature-values that have been suggested, selected, and added to the *nid*, then the similarity is measured as follows:

$$\frac{|\{f : \langle f, v\rangle \in Ablated \wedge \langle f, v'\rangle \in Selected\rangle\}|}{|Ablated|} \tag{1}$$

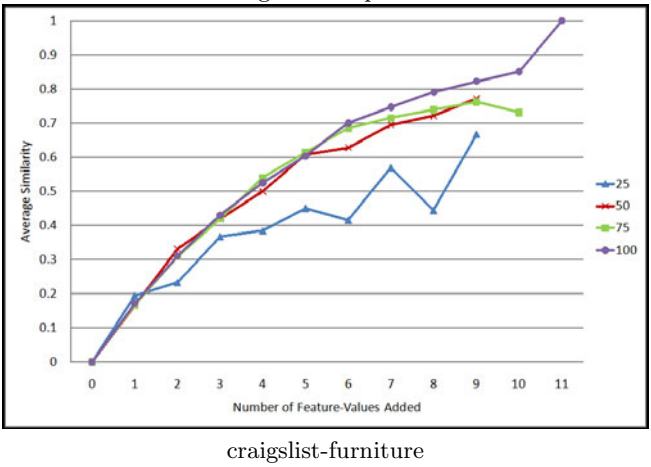
The results are shown in Figure 2. In each graph, on the y-axis is average similarity; on the x-axis is the number of feature-values that we have added to the *nid*.



wastematchers



craigslist-computers



craigslist-furniture

Fig. 2. Average similarity at various levels of ablation as we add suggested feature-values

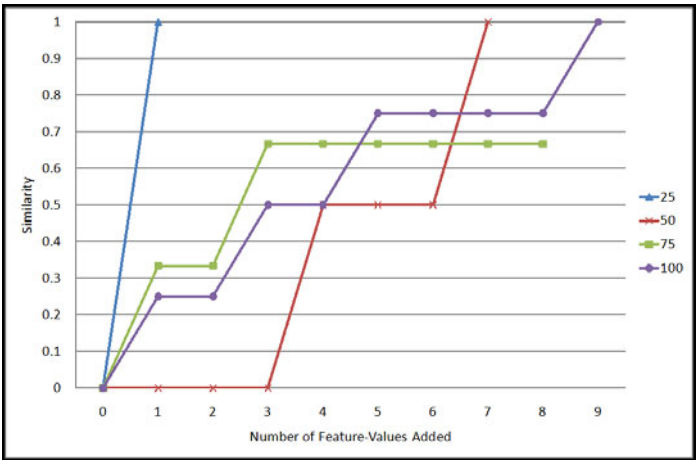


Fig. 3. Similarity at various levels of ablation as we add suggested feature-values for a single craigslist-furniture *nid*

GhostWriter was 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). The graphs contain different plots according to the starting amount of ablation. For example, one plot records what happens when we form the *nid* by ablating 25% of the original feature-values in the case; another plot measures what happens when there is 50% ablation; 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 features that we ablated earlier? If this is so, then the suggestions are useful ones. We also want to see that content is restored as early as possible.

The first point to explain is why the lines can fall as well as rise. For any *individual nid*, each feature-value that we add to the ablated *nid* cannot reduce similarity to the un-ablated *nid* (Equation 1). This is illustrated in Figure 3, which shows the similarity in the case of a *single nid*. The reason then that the lines in Figure 2 may fall is that each point is an average over a *different* set of *nid*s. For some *nid*s, GhostWriter may run out of suggestions much earlier than others: if the features of the retrieved cases C are all already present in the *nid*, then GhostWriter can make no fresh suggestions. So as we look at points in Figure 2 from left-to-right, each average is computed from a possibly smaller and smaller set. This can be seen in Figure 4. The percentages alongside each data point in this figure record this information. For example, on the line for 75% ablation, we were able to add two feature-value pairs to 98% of *nid*s; three to only 93%; and four to 80%.

Returning to Figure 2, we think the results are good. On average, selected suggestions do quickly recover ablated features. The lines generally climb steeply (showing that ablated features are being restored), only tailing off as the averages

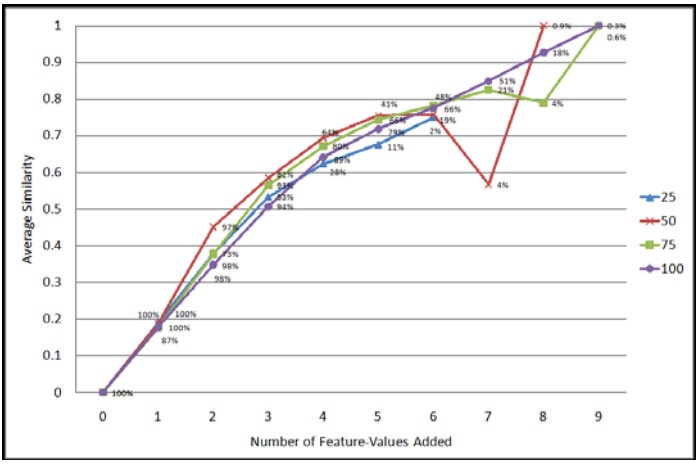


Fig. 4. Average similarity at various levels of ablation as we add suggested feature-values for craigslist-computers

are computed over smaller sets. The wastematchers plot looks much worse, but this is due to the data: as we see from Table 1, wastematchers cases have very few feature-values to begin with.

4 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 our GhostWriter system, for making these suggestions, inspired by work in Conversational CBR. And we have reported results from an off-line ablation study, showing that GhostWriter can quickly restore ablated features.

We believe that GhostWriter provides a new kind of support to content authors, different from existing research. Several researchers have investigated ways of proposing completions for incomplete phrases and sentences, representative of which are [5,7]. 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 [6]. 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. 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*.

In its goal, Recio-García et al's Challenger 1.0 system is 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.

We envisage four main lines of future inquiry. The first, of course, is more evaluation, including comparisons with 'benchmark' systems such as the use of random suggestions, but also evaluations with real users. The second is to try some of the many ways of learning the Feature Extraction rules, see, e.g., [4]. The third is to investigate variants of the algorithm, where we would use different ways of ranking the cases, features and feature-values. In particular, we intend to replace similarity-based case retrieval by a diversity-enhanced method for retrieving the cases [9], and we might try to incorporate negative cases (unsuccessful descriptions) into the suggestion process in some way. The fourth line of inquiry, described in the next paragraph, is perhaps the most interesting.

We want to apply our approach to support the authors of *reviews* of products such as hotels and electronic goods. This introduces several new dimensions to the work. (a) The content of these reviews is probably even less predictable than the content of the descriptions in Web-based exchange services. (b) In the domain of product reviews, other users can often indicate whether they found a review to be useful or not, or whether the reviewer is trust-worthy 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 or trust-worthy people have found them to be. (c) In Web-based exchange services, an item typically has just one description. But on review sites, an item may have several reviews, and these may complement or contradict each other.

References

1. Aha, D.W., Breslow, L.A., Muñoz-Avila, H.: Conversational case-based reasoning. Applied Intelligence 14, 9–32 (2001)
2. Bridge, D., Waugh, A.: Using experience on the read/write web: The Ghostwriter system. In: Bridge, D., et al. (eds.) Procs. of WebCBR, Reasoning from Experiences on the Web, Workshop Programme at the 8th International Conference on Case Based Reasoning (2009)
3. Burkhard, H.-D.: Extending some concepts of CBR — Foundations of case retrieval nets. In: Lens, M., et al. (eds.) Case-Based Reasoning Technology: From Foundations to Applications, pp. 17–50. Springer, Heidelberg (1998)
4. Etzioni, O., Banko, M., Soderland, S., Weld, D.S.: Open information extraction from the web. Communications of the ACM 51(12), 68–74 (2008)
5. Grabski, K., Scheffer, T.: Sentence completion. In: Sanderson, M., et al. (eds.) Procs. of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 433–439. ACM Press, New York (2004)

6. Lamontagne, L., Lapalme, G.: Textual reuse for email response. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 234–246. Springer, Heidelberg (2004)
7. Nandi, A., Jagadish, H.V.: Effective phrase prediction. In: Koch, C., et al. (eds.) Procs. of the 33rd International Conference on Very Large Data Bases, pp. 219–230. ACM Press, New York (2007)
8. Recio-García, J.A., Díaz-Agudo, B., González-Calero, P.A.: Textual CBR in jCOLIBRI: From retrieval to reuse. In: Wilson, D.C., Khemani, D. (eds.) Procs. of the Workshop on Textual Case-Based Reasoning, 7th International Conference on Case-Based Reasoning, pp. 217–226 (2007)
9. Smyth, B., McClave, P.: Similarity vs. diversity. In: Aha, D.W., Watson, I. (eds.) ICCBR 2001. LNCS (LNAI), vol. 2080, pp. 347–361. Springer, Heidelberg (2001)
10. Smyth, B.: Case-based recommendation. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) Adaptive Web 2007. LNCS, vol. 4321, pp. 342–376. Springer, Heidelberg (2007)
11. Wilke, W., Lenz, M., Wess, S.: Intelligent sales support with cbr. In: Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D., Wess, S., et al. (eds.) Case-Based Reasoning Technology. LNCS (LNAI), vol. 1400, pp. 91–113. Springer, Heidelberg (1998)