

# Creation and Maintenance of Query Expansion Rules

Stefania Castellani, Aaron Kaplan, Frédéric Roulland, Jutta Willamowski, and Antonietta Grasso

Xerox Research Centre Europe  
6 chemin de Maupertuis, 38000 Grenoble, France

**Abstract.** In an information retrieval system, a thesaurus can be used for query expansion, i.e. adding words to queries in order to improve recall. We propose a semi-automatic and interactive approach for the creation and maintenance of domain-specific thesauri for query expansion. Domain-specific thesauri are especially required in highly technical domains where the use of general thesauri for query expansion introduces more noise than useful results. Our semi-automatic approach to thesaurus creation constitutes a good compromise between fully manual approaches, which produce high-quality thesauri but at a prohibitively high cost, and fully automatic approaches, which are cheap but produce thesauri of limited quality. This article describes our approach and the architecture of the system implementing it, named Cannelle. It exploits user query logs and natural language processing to identify valuable synonymy candidates, and allows editors to interactively explore and validate these candidates in the context of a domain-specific searchable knowledge base. We evaluated the system in the domain of online troubleshooting, where the proposed method yielded an improvement in the quality of the search results obtained.

## 1 INTRODUCTION

Domain-specific searchable knowledge bases (KBs) such as online troubleshooting search systems contain technical content describing for instance mechanical parts, operations thereon, configuration settings, etc. Naive users of such a system often have trouble searching it because they are unfamiliar with the domain-specific terminology, whether because it consists of technical words they do not understand, or simply because there are many possible terms for the same thing—for example, the place in a photocopier where the blank paper is loaded might be called a drawer, a tray, or a feeder. To bridge this terminology gap, it can be helpful for an IR system to apply query expansion rules that automatically add additional terms to user queries, so that for example a query containing the word “drawer” will be matched with results containing the word “feeder”.

To be useful, query expansion rules must be chosen carefully. A generic list of synonyms such as a writer’s thesaurus typically is missing many domain-specific synonymy relations, and at the same time includes synonymies that are

inappropriate in specific contexts in the KB. Query expansion rules can be more effective if they are tailored to the particular KB to which they will be applied. However, it can be difficult for KB editors to write such rules by hand. The process involves first identifying candidate rules, e.g. by interviewing KB users about searches they have made that were unsuccessful, or by poring over query logs looking for queries that seem likely to have failed, or by pure intuition. Once a candidate rule has been identified, it must be evaluated to see whether it brings a decrease in precision that outweighs the increase in recall; this involves posing many different queries to the IR system and comparing the results.

In this paper we describe Cannelle, a tool that supports and partly automates the process of identifying and evaluating query expansion rules. We have developed and tested the tool in the context of an online troubleshooting KB for office devices such as printers, but we believe the approach would be applicable to domain-specific text collections of other kinds as well, and particularly ones that use standardized terminology with which users might not be familiar.

## 2 RELATED WORK

We are not aware of any previous work on hand-construction of domain- or corpus-specific query expansion thesauri (We use the term “thesaurus” to mean a lexical resource used for query expansion, even though it may include not only synonyms as in a real thesaurus, but also terms related in other ways). There is, however, work on query expansion using hand-build general-purpose thesauri, and using thesauri constructed automatically from corpora (which can be domain-specific).

Query expansion using a general-purpose thesaurus such as WordNet [Fellbaum, 1998] over a TREC corpus (non-domain-specific) can yield improved results for particular queries, particularly short queries, but can also significantly degrade results by adding contextually inappropriate words to the query [Voorhees, 1994]. In our own tests with a domain-specific KB and a general-purpose thesaurus, we found that relatively few queries were improved, since the most helpful synonymy relations were domain-specific ones that were absent from the thesaurus, but queries with degraded results were still common.

In [Jacquemin et al., 2002], a general-purpose thesaurus is used in an information extraction context. Lexico-syntactic restrictions induced from example sentences in the thesaurus are used to disambiguate words to avoid contextually inappropriate application of synonymy rules. In an early version of our system we experimented with a similarly rich rule formalism, but for the moment we have abandoned this direction primarily because of interface problems: our approach involves machine-assisted development of rules by a person, and it proved difficult to design an interface with which a linguistically-naive user could write lexico-syntactic rule restrictions.

A number of techniques have been proposed that exploit statistics of term distribution in the text being searched in order to retrieve relevant documents that do not contain the original query words. This class of techniques includes

latent semantic indexing [Deerwester et al., 1990], as well as query expansion using automatically-constructed thesauri such as presented in [Qiu and Frei, 1993]. These techniques are based on the idea that if two terms tend to occur in similar contexts within the corpus being searched, then they are likely to be similar in meaning, and thus that a user who poses a query including one of them is likely to be interested in documents that contain the other. This kind of technique can only identify similarities between two words both of which appear in the documents being searched. In the case of a curated technical KB, editors generally try to ensure that a given concept is referred to consistently using a single standard term, and thus the frequencies of non-standard synonyms of a standard term tend to be near zero. In other words, the terms for which we need synonyms are precisely terms that do not occur in the KB, so KB statistics are not helpful for finding synonyms. Applying similar techniques to a corpus other than the one to be searched, as in [Turney, 2001], might yield some useful synonym pairs, but then the non-corpus-specific nature of the resulting thesaurus would result in a loss of precision, as discussed above for general-purpose hand-built resources. For this reason, we use logs of past user sessions, rather than documents, as a source of statistics for identifying term similarities. The idea of using query logs as a source for query expansion has precedents in e.g. [Amitay et al., 2005, Baroni and Bisi, 2004, Cucerzan and Brill, 2005, Jones et al., 2006] and [Cui et al., 2003]. The details of the way in which we calculate term similarities from query log statistics differ from those of other systems, but we have not performed detailed comparisons and do not claim that our method is superior. The novelty of our system is in the way candidates are subsequently contextualized and evaluated using an interactive, semi-automatic tool.

Another approach to query expansion is to involve the user explicitly in selecting additional query terms, e.g. [Fonseca et al., 2005] and [Roulland et al., 2007]. The studies reported in [Shiri et al., 2002] examine how KB end users interactively exploit structured domain specific thesauri for query term selection and query reformulation. Our approach puts some of the burden of evaluating candidate expansion terms on the editor of the KB, rather than on the end user. This is feasible in the case of a domain-specific KB, where a small number of synonyms can have a positive effect on a relatively large number of searches. Note that the two approaches are complementary—synonymy rules defined by an editor can be applied automatically without user intervention, while other query refinement choices are left to the user.

### 3 SEMI-AUTOMATIC THESAURUS CREATION AND MAINTENANCE

In general creating a thesaurus involves first identifying a set of promising synonymy candidates, and then evaluating each candidate to decide if it should be used as-is, used with contextual restrictions, or discarded. In our context promising synonymy candidates are candidates mapping frequently used query terms

to corresponding technical terms (i.e. terms actually present and indexed in the KB). Such candidates can be automatically identified through the analysis of query session logs and of the searchable KB. Nevertheless, automatically identified candidates might not be appropriate or be too general when used as such for query expansion and in consequence introduce noise into the search results.

Therefore, it might be necessary to restrict their application. To restrict the application of a candidate, both terms, the query term and the technical term, can be contextualized: the query term with respect to the query context in which it was used, and the technical term with respect to context in which it appears in the KB. Our system allows the KB editor to evaluate the impact that a (possibly contextualized) candidate would have on the search results.

Accordingly, the Cannelle system proposes the following interaction areas to its user (see Figure 1):

- The list of synonymy candidates, i.e. pairs (query term, technical term), with the scores assigned to them by the selection heuristic (detailed below).
- The contexts in which the query / technical term appears in the query session logs / KB content respectively.
- The summary of the impact a synonymy might have on user queries.

The synonymy candidates are displayed in the top left-hand area of the interface. The editor can select a candidate and evaluate it with respect to the impact it will have on typical user searches in the KB, in particular in terms of new results brought back through the introduction of the corresponding synonymy rule.

Figure 1 shows an example in the troubleshooting context, where the editor has selected the candidate (‘error’, ‘fault’) from the list of synonymy candidates. The editor can estimate if it is useful to create immediately a corresponding synonymy rule or not, or if a corresponding rule could be potentially useful but would introduce too much noise if introduced without restrictions. In the latter case, the editor can specify possible application contexts for the synonymy. These contexts are derived on one hand from the usage context of the query term in the session logs and on the other hand from the sentence contexts in the KB documents where the KB term appears. In our example (Figure 1), for the candidate (‘error’, ‘fault’) the editor may observe that ‘error’ is synonymous with ‘fault’ but only within sentences in the KB where ‘fault’ is part of the phrase “fault code”.

The system can be used by the KB editors at various stages. At the initial deployment of the KB, the system will provide support to the KB editors to evaluate opportunities for the generation of new synonymy rules from other KBs system usage logs. When new content is created and added to the KB, the system can help the KB editor check if the new content also calls for modifications of the existing thesaurus, e.g. listing all the rules that apply to the new text. Then, periodically the system can be used to check if user terminology is still adequately supported by the KB. If not, or if a problem is detected, the system can help determine if adding new synonymy rules or modifying existing ones would help to better link unsupported user terminology with technical terms represented in the KB.

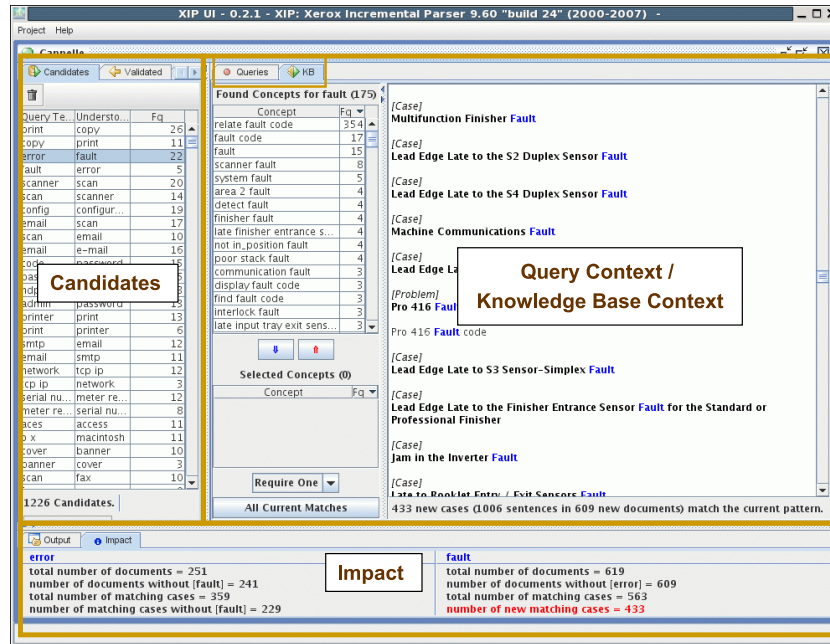


Fig. 1. Main interaction areas in Cannelle.

In the rest of this section we describe in more detail the various components and functionalities of the system starting with a description of its architecture.

### 3.1 Architecture

The main modules composing the system and the dependencies with the modules of the targeted search engine are shown in Figure 2.

A first module (“Candidate detection”) identifies a list of candidates based on a heuristic which will be discussed below. These candidates are stored internally in the system and can be accessed by the other modules in further steps or updated with a subsequent run of the candidate detection module on new user session logs. The second module (“Interactive definition of synonymy rules”) provides a graphical user interface in order to define some synonymy rules from the collected candidates. It collects from the KB sentences and syntactic contexts where a candidate will potentially impact a search. This information is presented to the editor who can then generate a contextual synonymy rule from a candidate. This rule is stored internally in the system. Finally, the module (“Synonymy export”) is required to export the generated rules to the targeted search engine thesaurus, i.e. to translate them into the corresponding format.

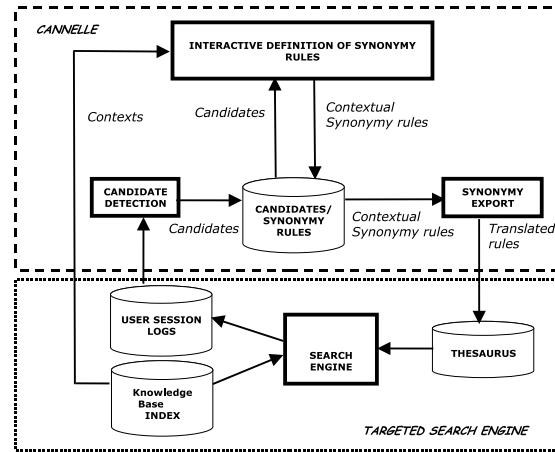


Fig. 2. Architecture of Cannelle

### 3.2 Detection of Candidates for Synonymy

One can imagine different heuristics for identifying candidate synonymy pairs. Each heuristic detects candidate synonymy pairs and assigns a quality score to each candidate; if multiple heuristics are used, then the candidates can be ranked according to a weighted sum of the scores assigned by the individual heuristics. When the candidates are presented to the editor, those falling below a threshold score are automatically eliminated, and the score is used to rank those above the threshold.

For the current prototype of Cannelle we have defined and implemented a heuristic based on logs of past users' interactions with the KB. It requires that the logs contain information on the interactions of the user with the search engine during a session, i.e. that queries issued by the same user within a short period of time are grouped together. We assumed as elsewhere in the literature [Amitay et al., 2005] that all queries in a session refer to the same problem and that such sessions have a first query formulation that has not brought satisfactory results. This first formulation is then followed by reformulations of the same problem description that are made by users in the attempt of finding what they are looking for.

Using this information, we count reformulation frequencies: for each pair of terms  $(X, Y)$ , we count how often a query containing  $X$  is followed in the same session by another query that is identical except that  $X$  is replaced by  $Y$ . Each term  $X$  and  $Y$  can be composed of one or more words. For example, if a user issues the query 'error code' and subsequently the query 'fault code,' we count this as one occurrence of the reformulation  $\text{error} \rightarrow \text{fault}$ . If another user makes the query 'scanner error' and subsequently 'scanner fault,' it is counted as another occurrence of the same reformulation. The reformulation pairs are taken as synonymy candidates, and their reformulation frequencies are used as scores. We filter out reformulation pairs whose replacement term does not occur

in the KB, since using such pairs as query expansion synonyms would have no effect on search results.

In our experiments, many of the reformulations ranked highly by this heuristic are corrections of spelling errors, e.g. ('configuration', 'configuration'). If the query interface already includes a spelling corrector, then adding misspellings to the thesaurus would be redundant. We thus filter respellings as follows: we apply our spell checker to the problematic term in each candidate pair, and if it proposes the replacement term as a respelling, then we drop the pair from the candidate list.

### 3.3 Interactive Definition of Synonymy Rule

During the processing of a synonymy candidate (T1, T2) selected from the list of candidates, the editor can estimate that:

- it is not useful to create a corresponding synonymy rule, for example because it would introduce too much noise, or that
- it is useful to create a simple synonymy rule stating that any query containing T1 should be matched with all results containing T2, or that
- it is useful to create a synonymy rule but with contextual restrictions; then the work consists of exploring the contexts in which T2 appears in the KB, and possibly results in a synonymy rule specifying in which contexts the rule should apply.

In the first case, the editor can move the synonymy candidate (T1, T2) to a list of rejected candidates. In the second case, the editor can ask the system to directly create the synonymy rule for (T1, T2). In the last case, the evaluation allows the editor to explore the possible contexts of application of the synonymy. These contexts are automatically derived from the sentences in the KB documents. For example, for the candidate ('error', 'fault') the editor may observe that 'error' is synonymous with 'fault' but only within sentences where 'fault' is associated with 'code'. Choosing some contexts corresponds to constraining the synonymy rule so that the rule will be applied only within those contexts. Constraints can be applied to the context of the problematic term, the replacing term, or both.

**Analysing the Candidates for Synonymy** The synonym candidates can be analysed with respect to the impact of the corresponding synonymy rules when searching the KB both from a quantitative and a qualitative point of view.

A first element of consideration for a candidate for synonymy (T1, T2) is the frequency of the replacement  $T1 \rightarrow T2$  in the queries. Also, for a given candidate for synonymy (T1, T2) the reverse pair (T2, T1), in particular if detected in the reformulations, can be taken into consideration at the same time. The editor can also see (1) the queries where a reformulation  $T1 \rightarrow T2$  has taken place, (2) the queries that contain T1 but which have not been reformulated, and (3) the set of sentences that would be retrieved from the KB for queries containing T1 if the synonymy rule for  $T1 \rightarrow T2$  were activated. Occurrences of the replacing term are highlighted in the sentences.

Some quantitative measures on the occurrences of the problematic term and replacing terms in the KB provide further support to evaluating the impact of introducing a synonymy rule:

- Number of documents where the problematic term occurs
- Number of documents where the replacing term occurs
- Number of sentences containing each of the terms
- Number of sentences containing the problematic term but not the replacing term and vice versa

### **Specifying Contextual Constraints on the Problematic Term in Queries**

From the comparison of the two lists of queries where the problematic term of a candidate appears, i.e. the list of queries that have been reformulated, and the list of non-reformulated queries, the editor can decide that the problematic term should be made more specific. For example, with the candidate ('code', 'password'), the queries in which 'code' was not reformulated as 'password' may contain 'area code' or 'fault code' whereas the reformulated queries may contain 'user code' or 'admin code'. A better candidate to consider for query expansion would be in this case ('user code', 'password') or ('admin code', 'password').

The editor can select a new problematic term from the list of reformulated queries or manually enter a new term. This will result in the creation of a new candidate in the system that can be processed in the further steps described below similarly than the ones that were automatically detected.

### **Specifying Contextual Constraints on the Replacement Term in the Knowledge Base**

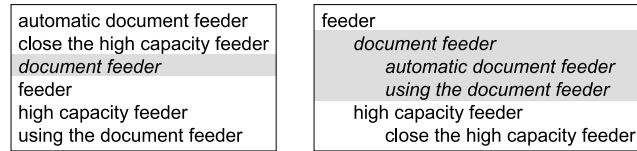
Ideally, if editors had unlimited time and patience, they would specify the list of KB documents that are relevant to each possible query term. Since it is typically not feasible for the editors to consider each document individually, we provide an interface that groups documents according to the context of occurrence of a replacement term. Editors can then choose contexts in which the rule should apply, and this has the effect of specifying entire groups of documents as relevant to the problematic term.

We use as contexts the syntactically coherent expressions identified by the method described in [Roulland et al., 2007]. This method uses a parser to segment a text into a sequence of expressions. The granularity of the segmentation is defined such that expressions are typically quite short (a few words), so that there is a high probability of the same expression being found in multiple documents, yet long enough that each expression makes sense as a choice in a query refinement process. For example, the sentence fragment “image area partially blank when printing and copying” is segmented as follows: image area/partially blank/when printing and copying. Some normalization (stop word removal and lemmatization) is applied to increase the frequency with which equivalent expressions are found in multiple documents.

When the editor wishes to specify that a rule applies only when the replacement term is found in certain contexts in the KB, we propose as contexts all expressions (as defined above) that occur in the KB and contain the replacement term. For example, consider the pair ('handler', 'feeder'). The replacement term

‘feeder’ may be found in the expressions “automatic document feeder” (the part of the copier that handles stacks of originals) and “high capacity feeder” (a tray that holds blank paper). The editor can specify that queries for ‘handler’ should be matched with occurrences of “automatic document feeder”, but not “high capacity feeder”.

In the current version of Cannelle, the contexts are presented as a flat list, but a future version of the system will use a collapsible tree organized by subsumption. For example, Figure 3 shows the two alternatives to represent the contexts found for the term ‘feeder’ in our KB.



**Fig. 3.** Examples of context presentations (flat list and tree).

In the tree representation, selecting ‘document feeder’ would have the effect of selecting all documents containing either “document feeder” as a self-contained expression or “automatic document feeder”; the context ‘automatic document feeder’ would still be available as a choice, but as a refinement of ‘document feeder’ rather than as a separate choice.

**Considering Symmetric and Transitive Rules** When evaluating a candidate for synonymy (T1, T2), the editor may be interested in considering the reverse pair (T2, T1) as well, for example because he knows that the two expressions are truly equivalent. In the current version of the system, for each candidate for synonymy Cannelle just displays the reverse pair if the heuristics find evidence for it. In the next version of the system, after a rule has been created from a candidate, Cannelle will generate the list of additional candidates that can exist by symmetry. These are, for a rule where a term T1 will match T2 in the contexts C1T2 and C2T2, the new candidate pairs (T2, T1), (C1T2, T1), and (C2T2, T1). For example, if for the candidate (‘smtp’, ‘email’) the editor generates the rule ‘smtp’ → ‘email server’ — ‘email setup’ then the system will propose the following additional candidates: (‘email’, ‘smtp’), (‘email server’, ‘smtp’) and (‘email setup’, ‘smtp’).

Another capability we are developing in Cannelle is the ability to generate additional candidate pairs considering the potential transitivity between synonymy rules. Following the same example used for the symmetry, if the rule ‘smtp’ → ‘email server’ | ‘email setup’ already exists and the editor generates the additional rule ‘email’ → ‘e-mail’ the system will propose by transitivity the following candidates: (‘smtp’, ‘e-mail server’) and (‘smtp’, ‘e-mail setup’).

## 4 EXPERIMENT

We conducted an experiment in order to estimate the capability of Cannelle to generate synonymy rules that improve the quality of the documents retrieved by a search engine. The experiment was designed in collaboration with the editors of the troubleshooting KB we have considered, i.e. the intended users of the tool. This experiment consisted in an evaluation of Cannelle using query logs from the KB collected over a year’s time.

The evaluation consisted of the following steps:

1. Automatic identification of synonymy candidates from the user session logs;
2. Creation of a set of synonymy rules from the evaluation of the highest ranked candidates (including the reverse candidates);
3. Estimation of the impact of these synonymy rules in terms of number of sessions where they would be activated;
4. Estimation of the impact of these synonymy rules in terms of quality of results retrieved when rules are activated.

The list of the synonymy candidates was obtained by applying the heuristics on the user sessions covering the first 10 months (roughly 24 000 sessions, of which roughly 6 600 contained at least one query and roughly 2 000 contained reformulations). We then evaluated 80 candidates, consisting of the most frequent candidates and their reverses. The evaluation consisted in determining if these candidates constitute desirable query expansion rules or not, and if the rules needed to be restricted to certain contexts. Half of the synonymy candidates were evaluated by two different evaluators, the other half by only one evaluator. 57 of the candidates were approved. In addition, during the process the evaluators identified 3 additional synonymies that were not in the candidate list. Moreover, the evaluation of 3 candidates became redundant with respect to rules generated for previous candidates. In total this process lead to the specification of 60 synonymy rules, and more precisely of:

- 37 rules without context
- 23 rules with context
  - 3 only with query context
  - 13 only with KB contexts
  - 7 with both query and KB contexts

Table 1 shows some examples of synonymy candidates evaluated during the tests together with the query contexts and/or KB contexts, if any, in the corresponding synonymy rules generated.

In the final evaluation, we analyzed the impact of the generated synonymy rules on the last two months of logged user sessions. We counted in how many sessions the synonymy rules would have applied, and evaluated the quality of results returned with and without the rules. To evaluate the quality of the results we considered only the first query to which rules would apply within each session (later queries are less interesting, since the results returned for the first query would probably have influenced the user’s subsequent queries). From these

**Table 1.** Examples of synonymy candidates and corresponding generated rules.

Problematic term → replacing term	Query context	KB context
email→e-mail		
error→fault		‘fault’ not followed by ‘inter- rupted’
code→password	‘admin code’ or ‘invalid code’ or ‘access code’	
sheet→page		‘banner page’ or ‘cover page’
toner→cartridge		‘cartridge’ not preceded by ‘sta- ple’
printer→print	‘printer device’ or ‘printer ser- vice’ or ‘printer driver’	‘print device’ or ‘print service’ or ‘print driver’

queries, 100 unique queries were randomly selected, and for each of them two different evaluators rated the relevance of the 20 top-ranked documents retrieved by the search engine with and without the synonymy rules. For each query, evaluators gave a score from 1 to 5, with 1 indicating that results with the synonyms were much less relevant than results without the synonyms, and 5 indicating the reverse. We then averaged the scores of the two evaluators for each query.

In our evaluation the specified synonymy rules applied in 38% of the sessions, which gives an indication of the significance of the method’s scope. In 16% of queries the quality of the retrieved documents was improved by the application of the synonyms (score of 4 or above). We observed a decrease in result quality (score of 2 or below) for none of the tested queries, and the KB editors considered this an important outcome. Given the positive outcome of our evaluation, the editors have agreed to test the tool in their working environment. The tests are ongoing and will complement the results of our experiment, and they will also help in refining the next version of the tool, which we are currently developing.

## 5 CONCLUSION

We have developed a tool for interactive development and testing of query expansion rules. The editor of a domain-specific KB can use the tool to find contextualized rules that are likely to improve the recall of many user queries while minimally affecting precision. The tool automatically proposes candidate rules using a heuristic based on logs of past query sessions. When the editor wishes to contextualize a rule, candidate contexts are suggested automatically based on query logs and on the application of natural language processing techniques to the KB text. Initial tests indicate that rules discovered using this tool significantly improve information retrieval results for a device troubleshooting KB, and the tool is currently being tested in a production environment.

We are developing a new version of the tool that will use a tree representation for presenting the contexts to the editor (see Section 3.3) and allow the evalua-

tion of additional candidates for synonymy by symmetry and/or by transitivity (Section 3.3). This new version will also integrate the feedback we will receive from the tests in the production environment.

## References

- [Amitay et al., 2005] Amitay, E., Darlow, A., Konopnicki, D., and Weiss, U. (2005). Queries as anchors: selection by association. In *HYPertext '05: Sixteenth ACM Conference on Hypertext and Hypermedia*, pages 193–201, New York, NY, USA. ACM Press.
- [Baroni and Bisi, 2004] Baroni, M. and Bisi, S. (2004). Using cooccurrence statistics and the web to discover synonyms in a technical language. In *LREC '04: 4th International Conference on Language Resources and Evaluation*.
- [Cucerzan and Brill, 2005] Cucerzan, S. and Brill, E. (2005). Extracting semantically related queries by exploiting user session information. Technical report, Microsoft Research.
- [Cui et al., 2003] Cui, H., Wen, J.-R., and Nie, J.-Y. (2003). Query expansion by mining user logs. *IEEE Trans. on Knowl. and Data Eng.*, 15(4):829–839. Member-Wei-Ying Ma.
- [Deerwester et al., 1990] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407.
- [Fellbaum, 1998] Fellbaum, C., editor (1998). *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press.
- [Fonseca et al., 2005] Fonseca, B. M., Golgher, P., Póssas, B., Ribeiro-Neto, B., and Ziviani, N. (2005). Concept-based interactive query expansion. In *CIKM '05: 14th ACM International Conference on Information and Knowledge Management*, pages 696–703, New York, NY, USA. ACM.
- [Jacquemin et al., 2002] Jacquemin, B., Brun, C., and Roux, C. (2002). Enriching a text by semantic disambiguation for information extraction. In *LREC '02: 3rd International Conference on Language Resources and Evaluation*, pages 45–51.
- [Jones et al., 2006] Jones, R., Rey, B., Madani, O., and Greiner, W. (2006). Generating query substitutions. In *WWW '06: 15th International Conference on World Wide Web*, pages 387–396, New York, NY, USA. ACM.
- [Qiu and Frei, 1993] Qiu, Y. and Frei, H.-P. (1993). Concept based query expansion. In *SIGIR '93: 16th ACM International Conference on Research and Development in Information Retrieval*, pages 160–169, New York, NY, USA. ACM.
- [Roulland et al., 2007] Roulland, F., Kaplan, A., Castellani, S., Grasso, A., Roux, C., O’Neill, J., and Pettersson, K. (2007). Query reformulation and refinement using nlp-based sentence clustering. In *ECIR '07: European Conference on Information Retrieval*, pages 210–221. Springer. LNCS 4425.
- [Shiri et al., 2002] Shiri, A. A., Revie, C., and Chowdhury, G. (2002). Thesaurus-assisted search term selection and query expansion: a review of user-centred studies. *Knowledge organization*, 29(1):1–19.
- [Turney, 2001] Turney, P. D. (2001). Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *EMCL '01: 12th European Conference on Machine Learning*, pages 491–502, London, UK. Springer-Verlag.
- [Voorhees, 1994] Voorhees, E. M. (1994). Query expansion using lexical-semantic relations. In *SIGIR '94: 17th ACM International Conference on Research and Development in Information Retrieval*, pages 61–69, New York, NY, USA. Springer-Verlag New York, Inc.