Topic Distillation with Knowledge Agents

Einat Amitay[1], David Carmel[1], Adam Darlow[1], Ronny Lempel[2], Aya Soffer[1]
[1]IBM Research Labs, Haifa 31905, Israel
[2]Computer Science Department, Technion, Haifa, Israel

1 Introduction

This is the second year that our group participates in TREC's Web track. Our experiments focused on the Topic distillation task. Our main goal was to experiment with the Knowledge Agent (KA) technology [1], previously developed at our Lab, for this particular task. The knowledge agent approach was designed to enhance Web search results by utilizing domain knowledge. We first describe the generic KA approach and then articulate on the use of this technology in the context of the topic distillation task. We focus mainly on the Knowledge Agent features that were used in this task. The rest of this paper is organized as follows: Section 2 describes KA in general. In Section 3 we describe how KA was used for the topic distillation experiment. Section 4 describes the obtained results. Section 5 concludes.

2 Knowledge Agents

Knowledge Agents (KA) provide domain-specific Web search in the context of dynamic domains. Knowledge agents' domains are defined by users and can thus be of any granularity and specialty. The key to the knowledge agent approach is that an agent specializes in a domain by extracting relevant information every time it performs a search and using this knowledge to improve the precision of subsequent search efforts. To this end, the KA maintains a knowledge base (KAB) that stores this information persistently. The KAB consists of a set of prominent pages in its domain, a search index of the content of these pages, and a repository of frequent terms in these pages. Each term is associated with a list of lexical affinities -- closely related terms frequently found in its proximity [2]. The agent updates the KAB continuously during search. New highly relevant pages found by the agent are inserted into the KAB; possibly replacing the place of old pages with lower utility. Pages are assigned a fitness score reflecting their relevance to the agent's domain. The KAB is used to enhance search results by automatically performing several tasks that are normally performed by users as a post-processing step after an initial unsuccessful search.

The first role the KA can perform on behalf of the user is query refinement. The KA expands the user's query by adding to each of the terms its most notable lexical affinities as found in the KAB. The advantage of this approach is that the agent's local thesaurus characterizes its domain-specific ontology and thus relations between terms are domain dependent.

The second role that the KA performs on behalf of the user is domain-specific Web search followed by shallow Web crawling. The agent applies a topological search mechanism similar to the one applied by Clever [3]. It first compiles a list of candidate result pages (root set) by sending the refined query to one or several search engines. This basic set is extended to include pages that are pointed to by pages in this set (satellite pages), pages that point to pages in the root set, and pages saved in the KAB (these are assumed to be the most authoritative information sources for the given domain). The KA ranks the retrieved pages such that the most relevant pages will be listed first. Ranking is performed based on both textual and topological aspects, utilizing information stored in its KAB. The textual score is computed by measuring the similarity of the pages to the specific query as well as to the agent's domain. The link topology score is computed using a combination of Kleinberg's mutual reinforcement algorithm [4] and stochastic link analysis [5].

The third role of the KA is local search. The KAB pages, as well as the anchor text associated with the links to the satellite pages, are indexed to form the KAB search index. The satellite pages significantly increase the number of resources accessible from the KAB. Given a query, the inverted index is used to locate the KAB pages and satellite pages that best answer this query. The results are ranked based on the similarity to the query as well as their fitness score. Since more qualitative pages in the KAB have higher fitness scores, the ranking of the results will reflect the similarity to the particular query as well as the page's authority in the KAB's domain.

The combination of a broad search of the entire Web, using general purpose search engines, with domain-specific textual and topological scoring of results, enables knowledge agents to find the most relevant documents at search time for a given query within the realm of the domain of interest. The following subsections describe the KA architecture in more detail.

2.1 The Agent's Knowledge Base

The knowledge base contains a bounded collection of ranked pages, a search index of these pages for supporting local search, and an aggregate profile of the textual content of these pages. The textual profile contains all the words that appear in the pages, after deletion of stop words and a mild stemming process, along with their number of appearances. Each word in the profile is associated with a list of its lexical affinities. The flow of pages into and out of the KAB is regulated using an evolutionary adaptation mechanism. Pages fight for the right to be included in the agent's KAB. Each page is assigned a history score reflecting its relevance to the domain through the life of the agent. A combination of the history score and the relevance score for a specific query determines which pages are inserted and removed from the KAB.

Pages can enter the KAB in two ways:

- Through explicit insertion by the user. The user may supply a set of seed pages when a new KA is created. Such seeds may come from the user's bookmarks file, or from any other collection known to the user. The user may add relevant pages to an existing KAB at any point in time. Once the KAB page limit is reached, the page with the lowest history score becomes stale and is removed from the KAB. Pages that are entered into the KAB explicitly by the user receive a high initial history score.
- Through automatic insertion by the agent. Upon completion of the search process, the *t*-generation history score $h_t(s)$ of each KAB page *s*, is updated in the following manner :

  $$h_t(s) = (1 - \beta_t)h_{t-1}(s) + \beta_t S(s)$$

  $S(s)$ is the score of *s* for the *t*'th search (scoring is described in the next section); $h_{t-1}(s)$ is the history score of *s* prior to the *t*'th search; β_t is a learning coefficient which controls the adaptation rate of the KAB. The learning coefficient balances the two factors which set the value of the new history score of the KAB's pages, namely the prior history score of the page, and its current specific score. The relative importance of the two components changes with the agent's age. As the number of queries performed by the agent grows, the weight of the history is increased. This reflects our confidence in KAB pages of mature agents, which have processed many queries and are therefore more likely to be highly relevant to the domain in question. Thus, we set

  $$\beta_t \leftarrow \beta_0 \delta^t .$$

  β_0 is an initial coefficient value; δ is decay factor, which controls the rate of decay of the learning coefficient. The new history scores of the KAB pages are compared against the scores of new pages returned by the search. High scoring new pages may replace low scoring KAB page. Their initial history score is set to their score for the current query.

2.2 The Search Process

The search process (Figure 1) starts with the user entering a query and ends with the agent returning a ranked set of (hopefully highly relevant) pages.

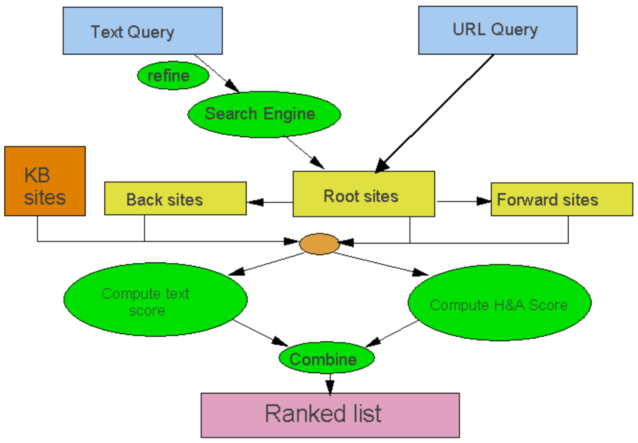

Figure 1: The search process.

2.2.1 Collecting the root set

The KA system supports two types of queries, text queries and sample-url queries. A text query is a keyword-based query such as those typically submitted to general-purpose Web search engines. The user's query can be automatically refined in the context of the agent's domain by adding to each of the query's terms its most notable lexical affinities as found in the profile of the KAB. The refined query is submitted to the user's choice of one or more search engines. The results returned by the search engine(s) to the refined query are called the root set of pages.

A sample-url query specifies a few (typically 1-5) seed urls. In sample-url queries, the user-supplied seed pages assume the role of the root set of pages, as if a search engine returned them in response to some textual query. The seed pages are read, and their combined content serves as a pseudo query for the purpose of evaluating the textual content of other pages in the search process. For the Topic distillation task, we used text queries with no query refinement.

2.2.2 Expanding the root set

The collection of pages, which at the beginning contains just the root pages, is expanded by following the hyperlinks surrounding the root pages, and by adding the pages which are stored in the KAB. The exact expansion model depends on the type of query that is being processed :

- When processing a text query, the expansion follows the scheme presented in [4] and adds two sets of pages:
  - The Backward set *B*, which contains pages that point to one or more root pages.
  - The Forward set *F*, which contains pages that are pointed to by one or more root pages.
- When processing a sample-url query, the expansion phase is more exhaustive and adds the following sets of pages to the root set:
  - The previously mentioned sets, *B* and *F*.

- The set *BF*, which contains pages which point to one or more *F*-pages.
- The set *FB*, which contains pages pointed to by one or more *B*-pages.

The breadth of the expansion is a parameter that can be set by the user. This expansion factor, which is a natural number, specifies how many pointed/pointing pages will be added to the collection for each page in each expansion stage.

We denote the entire collection of pages by *C*. Each page in *C* is assigned a textual and topological score, which are then combined as described below.

2.2.3 Computing the textual score of a Web Page

We first create a profile consisting of each word in the text query (not including stop words) along with its lexical affinities, and compute a textual similarity score for each page with respect to the query profile. For each term in the query profile, both keyword and lexical affinity, we compute a weight using a tf-idf formula. The textual profiles of the pages saved in the KAB serve as the set of documents from which the terms' document frequencies are taken.

The term weights are used to score the textual content of each page *s* with respect to a query using the following procedure. Text extracted from *s* is separated into three parts: strong, medium, regular. Strong text includes the words that appear in the title or in large font headers, medium text includes words that are either highlighted (bold, italics, etc.) or in small font headers, and regular text includes the rest. The textual score of page *s* is a weighted combination of the textual score for each text type. The textual score for each text type *t* is set to the sum of the weights of each query term appearing in the page in type *t,* normalized by the total number of terms of type *t* in the page.

The agent computes an additional textual score reflecting the similarity of each page to the domain. This score is set to the dot product of the vectors of lexical affinities representing *s* and the domain. Term weights are assigned in relation to their frequency in the domain. The rationale for this is that pages that have many lexical affinities in common with the domain are most related to the domain. The two textual scores are normalized and combined to yield the overall textual similarity score of a page.

2.2.4 Computing the link topology score of a Web page

The agent builds a Web sub graph, induced by the collection of pages *C*, on which connectivity analysis is performed in order to find authoritative Web pages. The link topology score is computed by first assigning weights to the edges of the Web sub graph. Every link receives a positive weight that is set according to the anchor text associated with the link, and the "type" associated with the pages on both sides of the link (the source page and the target page of the directed hyperlink):

- Anchor Text contribution: the anchor text is the method by which the pointing page describes the destination page to surfers, and is often a good source of information regarding the contents of the destination page. Thus, anchor text that resembles the query adds weight to the link that it describes.
- Anchor Links: links that connect a KAB page with a non-KAB page (in either direction) are considered more important, since they connect a page, which is presumed to be central to the domain (the KAB page), with a page that presumably answers the specific query. Such cross-links are called anchor links, and their weight is increased by a constant.

This weighted Web sub graph is used to assign the hub and authority scores to each page. Each page receives two hub and two authority scores from which a link topology score is derived.

- Mutual Reinforcement hub & authority scores. These are the hub & authority scores that the page receives when applying Kleinberg's Mutual Reinforcement algorithm [4] to the weighted graph at hand.
- Stochastic hub & authority scores. These are the hub & authority scores that the page receives by applying SALSA, the Stochastic Approach for Link Structure Analysis [5] to the weighted graph at hand.

These scores are normalized and combined to form the overall link topology score.

2.2.5 Computing the overall score of a Web page

The textual score *T(s)* and link topology score *L(s)* are combined to yield the overall score of each page in *C*:

$$S(s) = \alpha_C T(s) + (1-\alpha_C)L(s).$$

Link topology scores are reliable only for collections in which many neighboring pages have been added around the direct search results. We therefore set the value of α_C according to the ratio between the size of the compiled collection *C* and the size of the root set. The larger that ratio, the more confidence we have in the link-based score, and the lower we set α_C. When the ratio is low, meaning that the link expansion phase did not add many pages, we increase the influence of the text-based scores by increasing α_C.

3 Topic Distillation using Knowledge Agents

For the topic distillation task we trained a knowledge agent for each topic using a query extracted from the topic title for training. We experimented with two types of training queries: 1) free text queries consisting of all title terms, and 2) an AND of all title terms (only pages that contain all title terms are retrieved). In both cases we first eliminate stop-words and stem the query terms. Note that since each topic was defined by only one query, we did not utilize the Knowledge Agent's capacity to learn a domain from a collection of queries.

While knowledge agents were originally designed to submit their training queries to several Web search engines, in this experiment agents submitted their queries to a search engine over the ".gov" collection. We used the Juru search engine [6] to index and search the ".gov" domain. The index was built similarly to the WT10G index created for the 2001 Web Track ad-hock task; each page was indexed based on its content as well as its anchor descriptions – the anchors associated with its incoming links [6].

After retrieving the root set, the set of pages retrieved by Juru for the training query, the agent retrieves the forward set, the set of pages that are pointed by the root set, and the backward set, the set of pages that link to the root set. The forward set and the backward set are retrieved using the special files attached to the ".gov" data, that provide for each page its in-links and out-links within the ".gov" domain.

Each page in the retrieved-page set is ranked by a linear combination of its textual score as returned by Juru, and its link topology score computed as described in Section 2.2.4. The combined scores are used to rank the set of pages. The top 100 pages are inserted into the agent's KAB. This set is re-ranked using a sequence of filters designed to guarantee a mixture of good sources in the top-10 list returned by the system. In the following we describe the filters used by our system. We experimented with different combinations of these filters.

We further experimented with a static (query-independent) link-based scoring mechanism that measures the quality of a page a priory at indexing time. Static scoring was used as an alternative to Hub & Authority scoring which is query dependent. Each page *p* is associated with a static score based on its number of in-links *n*.

$$St(p) = \begin{cases} 1.0 & n \geq 20 \\ \sqrt{n/20} & otherwise \end{cases}$$

At query time, the textual score of the page returned by Juru is linearly combined with the page's static score to yield its final score.

3.1 Site Compression

During our experiments with the ".gov" collection, we found that for several queries, the top-10 results returned by our system were populated mainly by pages from one or two sites. Furthermore, very short link paths usually connected the pages of each such dominant site. From a user's point of view, and according to the spirit of the WebTrack guidelines, retrieving such groups of pages is redundant – the average user will easily be able to reach all those pages by navigating from one or two good starting points in the site. When considering that groups of same-site pages usually represent the same aspect of the topic being searched, we conclude that the top-10 results should contain quality pages from a *diverse* set of sites, hopefully covering several aspects of the topic in question.

The purpose of the site compression (SC) filter is to ensure that the top-10 results of each query will indeed be diverse. Note that SC is applied in most of the popular search engines. Specifically, in our system, we did not allow the top-10 results to contain more than 3 results from any *logical site*. The notion of logical sites is used in many link analysis computations, to identify intra-domain links. Here, the logical site was derived from the name of the site as follows: the site name was stripped of the possible leading "www." and trailing ".gov". The remaining string was divided into dot-separated tokens, and the last two tokens were taken as the logical site name.

3.1.1 Technical details

Site Compression is only applied if the top-10 results contain more than 3 results from some logical site. For the purposes of the explanation below, we need the following definitions:

- The *path* to a page p is the URL of p, where p's filename (if explicitly part of the URL) is removed.
- A link from page p to page q is called a *retreating* link if the path of q is a prefix of the path of p.
- The *neighborhood* of a page p, $N(p)$, is the set of pages in p's logical site that are reachable from p by following one or two *non-retreating* links.

The three steps of Site Compression are as follows:

1. The score of each page p is altered to reflect not only its own score, but also the scores of the pages of $N(p)$, in the following depth-two BFS-like process.

   a. Initially, all pages are considered to be unmarked.
   b. Let $q_1,\ldots,q_k$ be the pages reachable from p by following non-retreating out links, ordered by non-decreasing scores. Mark pages p and $q_1,\ldots,q_k$ and initialize the altered score of p, $S'(p)$, to $S(p)+0.33\sum_{j=1}^{k}2^{-(j-1)}S(q_i)$.
   c. For $j=1,\ldots,k$: Let $w_0,\ldots,w_{k(j)}$ be the yet unmarked pages reachable from q_j by following non-retreating outlinks, ordered by non-decreasing scores. Mark those pages, and increase $S'(p)$ by $\sum_{t=1}^{k(j)}2^{-(t+j-2)}S(w_t)$

   The new score of each page reflects the (query specific) quality of its own content, as well as the quality of pages that are easily accessible from it by navigation within the logical site. This heuristic resembles the ideas of Marchiori in [7].
2. In this intermediate stage, the pages are first resorted according to the altered scores. Then, some pages are eliminated from contention by the following process: starting from the top ranking page, each non-eliminated page p eliminates from contention all the pages of $N(p)$. This ensures that the pages that remain as candidates, even if from the same logical site, are separated by browsing paths of length at least 3.
3. The pages that were not eliminated are added to the top-10 list one by one, according to the altered scores, as long as they do not violate the restriction of having no more than three pages per logical site. Violating pages are skipped over.

3.2 Title Filtering

During our experiments we additionally observed that the title of the most relevant pages contains at least one query term. The opposite was also true – the title of most irrelevant pages did not contain any query term. Based on this observation, we use the similarity of a page title to the topic title as another relevance filtering mechanism.

The title filter (TF) was designed to filter out pages from the top-10 results with a title that is not similar to the query. This filter receives as input a similarity threshold and a parameter *k* that determines how many pages to filter out. For each page in the KAB, it analyses the similarity of the page's title to the topic title (i.e., the query). The page is marked as *frail* if the similarity is lower than the given threshold. The title filter replaces the lowest *k* ranked *frail* pages in the top-10 set with the highest *k* ranked *non-frail* pages in the rest of the KAB set.

3.3 Duplicate Elimination

The ".gov" domain is rich with duplicate pages as is usually the case with Web collections. As a result, the agent returns many duplicate pages for the same topic. The duplicate elimination (DE) filter was invoked over the set of the top-10 pages to filter out duplicate results. For each page in the top-10 set, we computed its textual similarity to all other pages in the set. If the textual similarity is higher than a given threshold, the lower ranked page is filtered out from the top-10 set, replaced by the highest ranked page in the complement set. The process is terminated only after the mutual similarity of all top-10 results is lower than the given threshold.

The DE filter was invoked after the SC and the TF filters. The pages filtered out by SC and TE were assigned a very low score thus pushing them to the end of the KAB set and ensuring they will not become contenders for the top-10 list when applying subsequent filters.

4 Experimental Results

Our experiments evaluated the utility of the Knowledge Agent technology, followed by a sequence of filters for the topic distillation task, applied over the given set of 50 topics in the ".gov" domain. For each topic we trained an agent based on the topic's title and then invoked a combination of the SC, TF, and the DE filters. We submitted five runs:

1. BASE – KA search followed by SC and TF filters. The TF filter was invoked with parameter *k = 3* (meaning replacing up to 3 pages with non-relevant titles in the top-10 results).
2. T10 – the same as BASE except the TF filter was invoked with parameter *k=10*.
3. T10D – the same as T10 followed by the DE filter.
4. AP – the same as BASE except that all topic title terms were marked as "must appear" terms in the training query. As a result, the root set returned by Juru included only pages that contain all the terms of the topic title.
5. PR – in this experiment we replaced the Hub & Authority ranking applied by the agent with a static in-link based ranking. All three filters were invoked on the top-10 results.

Figure 2 shows results of our five runs for the 50 topics. The graph presents for each topic, the difference between P@10 of our runs and the median P@10 of all participants.

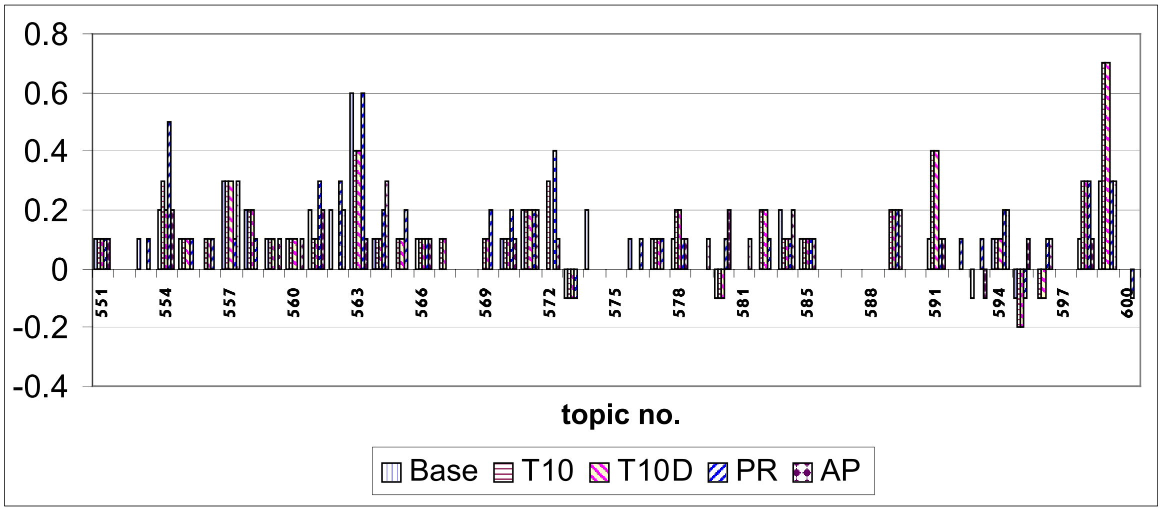

Figure 2: The difference between P@10 of our runs and the median P@10 of all participants.

As we can see, our runs achieved better results than the median for almost all topics. The following table shows the average P@10 over the 50 topics for the different runs, and the average best and median P@10 results of all participants:

BASE	T10	T10D	PR	AP	Best	Median
0.190	0.212	0.204	0.229	0.194	0.455	0.11

Table 1: Average P@10 of our runs and average-best and median P@10 of all participants.

The PR run that uses static ranking instead of link analysis achieved the best average result. This was quite surprising and contradicted our own expectations and evaluations. This is most likely due to the difference between our assessment and the official assessment, which seems to have been very strict and marked only highly *overall* authoritative pages as relevant. The T10 run achieved a better result than the BASE run, demonstrating the expected benefit of title filtering. Surprisingly however, it also outperformed T10D, which invoked the duplicate elimination filter (DE). In fact it turned out that the DE filter impaired the results. The reason is that the *trec-eval* program does not penalize duplicate pages in the top-10 set. As an example consider topic 572 –"selecting a nursing home". The T10 run returned the following three duplicate pages in its top 10: "G06-27-2524213", "G01-62-3424657", and "G01-76-0926870". All three were considered relevant by the assessor and marked as such. The DE filter filtered out two of them and left only one representative in the top-10. As a result, the number of relevant results in the top-10 was smaller after duplicate elimination and the P@10 score for this particular query was lower. This negative effect of the DE filter contradicts the guidelines for the topic distillation task that specifically stated that several pages should be returned from the same site only if they are significantly different in terms of the information they provide.

After receiving the qrel files for the 50 topics we conducted some experiments in order to better test the specific contribution of the various filters our results. Specifically, we re-created the agents, using different combinations of filters. For the Basic run, no filter was invoked. For the +SC run, the SC filter was invoked on the results of the Basic run. For The +TF(*k*) runs, the TF filter was invoked on the results of the +SC run, varying parameter *k* - the number of titles to filter. Finally, for the +DE run, the DE filter was invoked on the results of the TF(10) run. We performed the experiments using both the original KA algorithm (H&A agents) and the static ranking algorithm (PR agents). Figure 3 presents the average P@10 for each run. We can clearly see the contribution of the SC and TF filters. From these results, it is clear that the PR agents with maximal TF filtering (*k*= 10) indeed achieved the best results. These results demonstrate once more the negative effect of the DE filter. In fact the PR run with SC and TF(10) filtering achieves an average P@10 of 2.4, which surpasses all of our submitted runs.

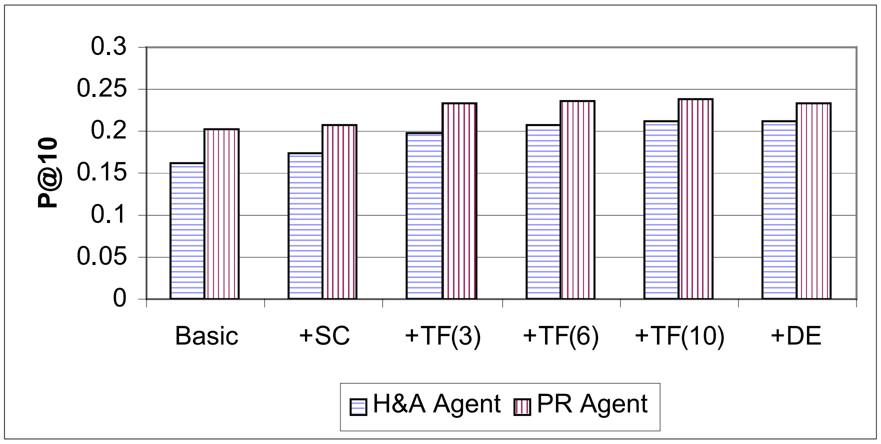

Figure 3: Filter contribution to P@10.

5. Summary

The WebTrack guidelines page [8] describes Topic Distillation as follows:

> *"Topic distillation involves finding a list of key resources for a particular topic. A key resource is a page which, if someone built me a (short) list of key URLs in a topic area, I would like to see included. ... The question is, what evidence + algorithms can be used to find such key resources? For Web searches where queries are short and users only look at the top ten results, effective topic distillation is potentially very useful".*

This report describes how Knowledge Agents can be used for a topic distillation task. We have shown how to train an agent for a specific topic. The combination of a broad search of the entire ".gov" domain with textual and topological scoring of results, enables the knowledge agent to find relevant documents for a given topic. The experiments we conducted showed that while the agent's KAB pages can be used as a basis for the result set, extra filters are required to further distill this set and surface its most valuable results.

While the SC and the TF filters improved precision significantly, the DE filter deteriorated the overall precision. This contradicts the WebTrack guidelines, which clearly state:

> "*Penalizing duplication: If your top ten consists of ten pages from the same site, judges might only reward the home page."*

We would like to suggest that for next year's task, *trec-eval* be modified in order to account for this irregularity. For example, duplicates could be grouped and only counted as one relevant result per top-10 list.

We did not make use of many of the more advanced features of the Knowledge Agent system for the topic distillation task this year. For future work, we would like to experiment with the system's domain specific query expansion and learning capabilities. The results could perhaps be further improved if, for example, agents were trained using more that one training query or by expanding the query.

References

[1] Y. Aridor, D. Carmel, R. Lempel, Y. Maarek and A. Soffer. *Knowledge Agents on the Web*. In Proceedings of the 4th International Workshop, CIA 2000, Boston, MA, July 2000. LNAI 1860, pages 15--26, Springer.

[2] Y. Maarek and F. Smadja. *Full text indexing based on lexical relations: An application: Software libraries*. In Proceedings of the 12th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 198--206, 1989.

[3] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan. *Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text*. In Proceedings of the 7th World-Wide Web conference, 1998.

[4] J. M. Kleinberg. *Authoritative Sources in a Hyperlinked Environment*. In Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms, volume 25-27, pages 668--677, January 1998.

[5] R. Lempel and S. Moran. *The stochastic approach for link-structure analysis (SALSA) and the TKC effect*. WWW9 /Computer Networks, June, 2000, Vol. 33, No. 1-6, pages 387--401.

[6] D. Carmel, E. Amitay, M. Herscovici, Y. Maarek, Y. Petruschka, and A. Soffer. *Juru at TREC 10 - Experiments with Index Pruning*. In Proceedings of the Tenth Text REtrieval Conference (TREC 2001), National Institute of Standards and Technology (NIST).

[7] M. Marchiori, *The Quest for Correct Information on the Web: Hyper Search Engines*. WWW6/Computer Networks and ISDN Systems, 29(1997) 1225-1235.

[8] TREC-2002 Web Track Guidelines. *http://www.ted.cmis.csiro.au/TRECWeb/guidelines_2002.html*