Finding Web Pages Relevant To TV News

Monika Henzinger
Google Inc.
2400 Bayshore Parkway
Mountain View, CA 94043
USA
monika@google.com

Bay-Wei Chang
Google Inc.
2400 Bayshore Parkway
Mountain View, CA 94043
USA
bay@google.com

Brian Milch
UC Berkeley
Computer Science Division
Soda Hall
Berkeley, CA 94720
USA
milch@cs.berkeley.edu

ABSTRACT

Many daily activities present information in the form of a stream of text, and often people could benefit from additional information on the topic discussed. In this paper we discuss finding web pages for TV news broadcast.

We evaluate a variety of techniques for this problem. For the best algorithm at least 84% of the pages it found were relevant and at least 64% of the page were on the exact topic of the broadcast.

1. INTRODUCTION

Many daily activities present information using a written or spoken stream of words: television, radio, telephone calls, meetings, simple conversations with others. Oftentimes people could benefit from additional information about the topics that are being discussed. This paper describes our work on how to identify topics in an ongoing stream of TV broadcast news and formulate queries in real time to web information sources in order to find web pages relevant to those topics. The goal is to present relevant web pages that the user can consult to find out more about the information being discussed.

Supplementing television broadcasts is particularly attractive because of the passive nature of TV watching. Interaction is severely constrained, usually limited to just changing the channel; there is no real way to direct what kind of information will be presented.

Indeed, several companies have explored suggesting web pages to viewers as they watch TV.

- The Intercast system, developed by Intel, allows entire HTML pages to be broadcast in unused portions of the TV signal. A user watching TV on a computer with a compatible TV tuner card can then view these pages, even without an Internet connection. NBC transmitted pages via Intercast during their coverage of the 1996 Summer Olympics.
- The Interactive TV Links system, developed by VITAC (a closed captioning company) and WebTV (now a division of Microsoft), broadcasts URLs in an alternative data channel interleaved with closed caption data [16, 2]. When a WebTV box detects one of these URLs, it displays an icon on the screen; if the user chooses to view the page, the WebTV box fetches it over the Internet.

Both of these systems require the producer of a program (or commercial) to choose relevant documents by hand. In fact, the producer often creates new documents specifically to be accessed by TV viewers. To our knowledge, there has been no previous work on automatically selecting web pages that a user might want to see while watching a TV program.

In this paper we study the problem of finding web pages for TV *broadcast news*. We restrict our attention to broadcast news since it is the most popular form of information-oriented TV.

We present a variety of different algorithms. Our approach is to extract a query from the closed caption text, issue the query to a news search engine to retrieve its top results, and then postprocess them. We evaluate 7 algorithms for query extraction and three different ways of postprocessing. The best algorithm achieves a precision of 91% on one data set and 84% on a second data set. It finds a relevant web page on the average about every 16 seconds on the first and every 20 seconds on the second data set.

The algorithm uses a combination of techniques. Our evaluation indicates that the most important features for its success are its "history feature" and a postprocessing step that filters out irrelevant documents. The "history feature" tries to detect when an old topic has ended and a new topic has started. It keeps a data structure representing all the text since the last topic change and uses that data structure together with the latest text to create a new query. The filtering step determines how similar the top two retrieved web pages are with each other and with the caption text. Even though it seems very simple, filtering leads to an increase of precision by 20 percentage points.

There is a large body of research on topic detection and text summarization. Recently, time-based summarization has also been studied [1]. The novelty of our system is that it combines topic detection with keyword extraction in order to issue search queries. Said differently, our goal is time-based keyword extraction, not time-based summarization.

The remainder of this paper is organized as follows: Section 2 describes the different algorithms and postprocessing steps. Section 3 presents the evaluation. Section 4 discusses related work. We conclude in Section 5.

2. OUR APPROACH

Our approach for finding web pages that are related to a stream of text is to create queries based on the text and to issue the queries to a search engine. Then we postprocess the answers returned to find the most relevant ones. In our case the text consists of closed captioning of TV news and thus we will issue the queries to a news search engine.

We will first describe what algorithms we use to create queries and then what technique we use for postprocessing the answers.

2.1 Query Generation

WWW2003, May 20–24, 2003, Budapest, Hungary.
ACM xxx.

We are interested in showing relevant web pages at a regular rate during the news broadcast. Our initial goal was to display a web page long enough for the title to be read and perhaps the first few sentences scanned, and then to have a new web page ready to show. The actual user interface may allow the user to pause and read the current page more thoroughly, or it may show multiple web pages, etc. We'd like the system to have relevant web pages ready to display at any time.

To do this, the query generation algorithm needs to issue a query every s seconds. We empirically chose $s = 15$, returning up to two web pages per query. This gives the user a reasonable chance of viewing a relevant web page at any time. Because postprocessing may eliminate some of the candidate pages, we also tested at $s = 7$, thus allowing up to half of the candidate pages to be discarded while maintaining the same or better coverage as $s = 15$.

The query generation algorithm is given the *text segment* T since the last query generation and it can keep information about the previous stream of text. We consider four different query generation algorithms described in the following sections.

As is common in the IR literature [17] the *inverse document frequency idf* of a term is a function of the frequency f of the term in the collection and the number N of documents in the collection. Specifically, we use the function $\log(N/(f+1))$. Since we do not have a large amount of closed caption available, we used Google's web collection to compute the idf of the terms. This means N was over 1 billion and f was the frequency of a term in this collection. Unfortunately, there is a difference in word use in written web pages and spoken TV broadcasts. As a result we built a set of words that have a low inverse document frequency in the TV captions but a high inverse document frequency in the web data. Examples of such words are *reporter* and *analyst*. All of the algorithms below ignore the words in this list.

2.1.1 *The baseline algorithm A1-BASE*

Our baseline algorithm is a simple $tf \cdot idf$ based algorithm. It weights each term by $tf \cdot idf$, where tf is the frequency of the term in the text segment T. The weight is large for unusual the term. This is useful since doing a search with the more distinctive terms of the news story is more likely to find articles related to the story. The baseline algorithm returns the two terms with largest weight as the query.

2.1.2 *The $tf \cdot idf^2$ algorithm A2-IDF2*

This is the same algorithm as the baseline algorithm, but a term is weighted by $tf \cdot idf^2$. The motivation is that rare words, like named entities, are particularly important for issuing focussed queries. Thus, idf is more important than tf.

2.1.3 *The simple stemming algorithm A3-STEM*

In the previous two algorithms each term is assigned a weight. Algorithm A3-STEM assigns instead a weight to each *stem*. The *stem* of a word is approximated by taking the first 5 letters of the word. For example, *congress* and *congressional* would share the same stem, *congr*. The intention is to aggregate the weight of terms that describe the same thing.

For each stem we store all the terms that generated the stem and their weight. The weight of a term is $c \cdot tf \cdot idf^2$, where $c = 1$ if the term was a noun and $c = 0.5$ otherwise. We use this weighting scheme since nouns are often more useful for formulating focussed queries. The weight of a stem is the sum of the weights of its terms.

To issue a query the algorithm determines the two top-weighted stems and finds the top-weighted term for each of these stems. These two terms form the query.

2.1.4 *The stemming algorithm with compounds, algorithm A4-COMP*

Algorithm A4-COMP consists of algorithm A3-STEM extended by 2-word compounds. Specifically, we build stems not only for one-word terms, but also for 2-word compounds. For this we use a list of allowed compounds compiled from Google's corpus of web data. Stems are computed by stemming both words in the compound, i.e., the stem for the compound *veterans administration* is *veter-admin*. Compounds are treated like terms, except that $c = 1.2$, i.e. compounds are boosted in the weighting, since they are more likely to distinctly identify the topic. Since a compound consists of two words, a query can now consist of two, three, or four words.

2.1.5 *The history algorithm A5-HIST*

Algorithm A5-HIST is algorithm A4-COMP with a "history feature". All previous algorithms generated the query terms solely on the basis of the text segment T that was read since the last query generation. Algorithm A5-HIST uses terms from previous text segments to aid in generating a query for the current text segment, the notion being that the context leading up to the current text may contain terms that are still valuable in generating the query.

It does this by keeping a *stem vector* which represents some or all of the text seen before the last query generation. It combines this information with the information produced by algorithm A4-COMP for the text segment T and finds the top weighted stems from it.

To be precise, a *stem vector* keeps for each stem a weight and additionally a list of terms that generated the stem, each with its individual weight. The stem vector keeps the stems of all words that were seen before the last query generation and after the last reset or the beginning of the text stream. A *reset* simply sets the stem vector to be the empty vector. We describe below under which conditions a reset happens.

When algorithm A5-HIST receives text segment T it builds a second stem vector for it using algorithm A4-COMP. Then it checks how similar T is to the text represented in the old stem vector by computing a similarity score s (described below). If it is similar, the old stem vector is *aged* by multiplying every weight by 0.9 and then the two vectors are added. To add the two vectors, both vectors are expanded to have the same stems by suitably adding stems of weight 0. Also the set of terms stored for each stem is expanded to consist of the same set by adding terms of weight 0. Then the two vectors are added by adding the corresponding weights of the stems and of the terms.

If text segment T is very dissimilar from the earlier text, then the old stem vector is reset and the two vectors are added, i.e., the old stem vector is replaced by the new stem vector. To put it another way, when the current text is very different than the previous text, it means that the topic has changed, so previous history should be discarded in deciding what query to issue.

If text segment T is somewhat similar to the earlier text, then the stem vector is not reset, but the weights in the old stem vector are decreased by multiplying them with a weight that decreases with the similarity score s. Afterwards the old stem vector and the new stem vector are added. So even though the topic has not completely changed, previous terms are given less weight to allow for topic drift.

To compute the similarity score we keep a stem vector for each of the last three text segments. (Each text segment consists of the text between two query generations.) We add these vectors and compute the dot-product of this sum with the vector for T, only considering the weights of the terms and ignoring the weights of

the stems. If the similarity score is above a threshold a_1, then T is *similar* to the earlier text. If the similarity score is above a_1 but below a_2, then T is *somewhat similar* to the earlier text. Otherwise T is *dissimilar* from the earlier text.

We used a test data set (different from the evaluation data sets) to choose values for a_1 and a_2. In our implementation, $a_1 = 0.001$ and $a_2 = 0.0003$. When T is somewhat similar, we use the weight multiplier $a = 0.9^{2-1000 \cdot s}$, which was chosen so that $a \leq 0.9$, i.e., the weights are more decreased than in the case that T is similar to the early text.

In the resulting stem vector the top two terms are found in the same way as in algorithm A4-COMP.

2.1.6 *The query shortening algorithm A6-THREE*

All of the previous algorithms generate queries with two terms, with each term possibly a two word compound. One term queries are not specific enough, returning a wide variety results that are mostly not relevant. Three (and greater) term queries run the risk of being so specific as to return no results.

To verify our intuition, we experimented with a query shortening algorithm, which issues a multiple term query, and shortens the query until results are returned from the news search engine. It was clear that reducing the query to one term hurt precision. For example, a query about a possible arson fire *dawson retaliation arson* found no results; the two word query *dawson retaliation* also found no results. However, the single word query *dawson* succeeded in finding many results, none of which were relevant to the arson fire story: the top two results were an article mentioning an IMF spokesman named Dawson and an article about the television show Dawson's Creek.

Therefore we kept two terms as the minimum query length. The query shortening algorithm A6-THREE is identical to A5-HIST, but begins with three-term queries, reissuing the query with the two top-weighted terms if there are no results.

2.1.7 *Algorithm A7-IDF*

Algorithm A7-IDF is identical to algorithm A5-HIST with idf^2 replaced by idf.

(Note that each increasing algorithm A1-A6 adds one additional feature to the previous. A7-IDF does not fit this pattern; we created it in order to test the specific contribution of idf^2 to A5-HIST's performance.)

2.2 Postprocessing

After generating the search queries we issue them to a news search engine and retrieve the top at most 15 results. We applied several ways of improving upon these search results, described in the sections below, and then selected the top two results to show to the user as news articles related to the broadcast news story.

Since several queries will be issued on the same topic, they may yield similar result sets and many identical or near identical pages may end up being shown to the user. In fact, we found that, across 99 algorithm/data combinations, an average of 34% of urls returned would be near-duplicates. Such a large number of duplicates would lead to a poor user experience, so we employed a near-duplicate backoff strategy across all the algorithms. If a url is deemed a near-duplicate of one that has already been presented, the next url in the ranking is selected. If all urls in the result set are exhausted in this manner, the first url is returned after all.

To detect duplicates without spending time fetching each article, we looked at the titles and summaries of the articles returned by the search engine. We compared these titles and summaries to a cache of article titles and summaries that have already been displayed during the broadcast. A similarity metric of ¿20% word overlap in the title, or ¿30% word overlap in the snippet, was successful in identifying exact matches (e.g., the same article returned in the results for a different query) and slight variants of the same article, as is common for news wires to issue as the story develops over time.

The postprocessing steps we used were boosting, similarity ranking, and filtering.

2.2.1 *Boosting*

The news search engine gets a two-word query and does not know anything else about the stream of text. The idea behind boosting is to use additional high-weighted terms to select from the search results the most relevant pages. To implement this idea the query generation algorithm returns along with the query associated *boost terms* and *boost values*. The boost terms are simply the top five terms found in the same way as the query terms. The boost values are the idf values of these terms.

The boosting algorithm then reranks the results returned from the search by computing a weight for each result using the boost terms. For a boost term which has IDF idf and occurs f times in the text snippet returned with the result, the weight is incremented by the value $(idf * 4f)/(f + 3)$. The weight is further incremented for boost terms in the title, according to the formula $(idf * 8f)/(f + 3)$. The results are then reordered according to their weight; non-boosted results or ties are kept in their original order.

[These don't seem to be the right formulas. Also, the date is of the article is taken into account as well – recency is preferred.]

2.2.2 *Similarity reranking*

A second way of reranking is to compute for each of the results returned by the search engine its similarity to the text segment T and to rerank the search results according to the similarity score. To implement this idea we built a $tf \cdot idf$-weighted term vector for both the text segment T and the text of the web page and compute the dot-product similarity score. (The first 500 characters of the article are used.) This filtering step requires first fetching the web pages, which can be time-expensive.

2.2.3 *Filtering*

The idea behing filtering is to discard web pages that are very dissimilar to the caption. Additionally, when the issued query is too vague, then the top two search results often are very dissimilar. (Indeed, all the results returned by vague queries are often very different from one another.) So whenever we find two candidate web pages and they are dissimilar, we suspect a vague query and irrelevant results. So we discard each of the web pages unless it is itself highly similar to the caption.

To implement this idea we used the same $tf \cdot idf$-weighted term vector for the text segment T and the text of the web page and computed the dot-product similarity score as in the similarity reranking, above. Whenever the page-T similarity score is below a threshold b the page is discarded. If there are two search results we compute their similarity score and discard the pages if the score is below a threshold p – but allowing each page to be retained if its page-T similarity score is above a threshold g.

We analyzed a test data set (different from the evaluation data sets) to determine appropriate thresholds. In our implementation, $b = 0.1$, $g = 0.3$, and $p = 0.35$.

3. EVALUTION

To evaluate different algorithms on the same data set the evaluators worked off-line. They were supplied with two browser win-

dows. One browser window contained the web page to be evaluated. The web page was annotated with a javascript input box so that the score for the page could simply be input into the box. The other browser window contained the part of the closed caption text for which the web page was generated. The evaluators were instructed as follows:

You will be reading a transcript of a television news broadcast. What you will be evaluating will be the relevance of web pages that we provide periodically during the broadcast. For each displayed web page consider whether the web page is relevant to at least one of the topics being discussed in the newscast for this page. Use the following scoring system to decide when a page is relevant to a topic:

- *0 - if the page is not the topic*
- *1 - if the page is about the topic in general, but not the exact story*
- *2 - if the page is about the exact news story that is being discussed*

For example, if the news story is about the results of the presidential election, then a web page about a tax bill in congress would score a 0; a web page about the candidates' stands on the environment would score a 1; a web page about the winner's victory speech would score a 2.

Don't worry if two web pages seem very similar, or if you've seen the web page previously. Just score them normally. The "current topic" of the newscast can be any topic discussed since the last webpage was seen. So if the web page is relevant to any of those topics, score it as relevant. If the web page is not relevant to those recent topics, but is relevant to a previous segment of the transcript, it is considered not relevant; give it a 0.

We count a document as "relevant" (R) if it was given a score of 1 or 2 by the human evaluator. We count it as "very relevant" (R+) if it was given a score of 2.

3.1 Data sets

We evaluated all these approaches using the following two data sets:

- **HN:** three 30-minute sessions of CNN Headline News, each taken from a different day , and
- **CNN:** one hour of Wolf Blitzer's Report on CNN from one day and 30 mins from another day.

The Headline News sessions ("HN") consists of many, relatively short, news stories. The Wolf Blitzer Report ("CNN") consists of fewer news stories discussed for longer and in greater depth.

Both data sets contain *news stories* and *meta-text*. Meta-text consists of the text between news stories, like "and now to you Tom" or "thank you very much for this report". We manually decomposed the news stories into *topics*, ignoring all the meta-text. Each topic consists of at least 3 sentences on the same theme; we do not count 1-2 sentence long "teasers" for upcoming stories as topics. The shortest topic in our data sets in 10 seconds long, the longest is 426 seconds long. The average length of a topic in the HN data set is 51 seconds and the median is 27 seconds. The topics comprise a total of 4181 seconds (70 mins) out of the 90 mins long caption. In the CNN data set the average topic length is 107 seconds and the median is 49 seconds. The topics comprise a total of 3854 seconds (64 mins).

Table 1: HN data set: Precision, i.e., the percentage of relevant (R) documents and in parenthesis the number of such documents

Technique	s	None	Boost'N' Filter
A1-BASE	7	58% (315)	86% (264)
A2-IDF2	7	58% (316)	87% (266)
A3-STEM	7	64% (273)	88% (246)
A4-COMP	7	64% (271)	88% (239)
A5-HIST	7	64% (303)	91% (257)
A6-THREE	7	72% (279)	89% (234)
A7-IDF	7	63% (228)	89% (262)
A1-BASE	15	63% (165)	91% (147)
A2-IDF2	15	62% (165)	91% (152)
A3-STEM	15	69% (210)	88% (200)
A4-COMP	15	70% (221)	90% (210)
A5-HIST	15	67% (219)	89% (201)
A6-THREE	15	75% (201)	91% (187)
A7-IDF	15	59% (223)	91% (207)

Table 2: CNN data set: Precision, i.e., the percentage of relevant (R) documents and in parenthesis the number of such documents

Technique	s	None	Boost'N' Filter
A1-BASE	7	43% (221)	77% (171)
A2-IDF2	7	46% (220)	75% (150)
A3-STEM	7	43% (185)	76% (143)
A4-COMP	7	44% (186)	76% (140)
A5-HIST	7	55% (259)	84% (190)
A6-THREE	7	60% (246)	86% (190)
A7-IDF	7	50% (254)	82% (186)
A1-BASE	15	48% (140)	83% (116)
A2-IDF2	15	60% (134)	85% (108)
A3-STEM	15	54% (139)	76% (113)
A4-COMP	15	59% (144)	82% (120)
A5-HIST	15	61% (200)	88% (164)
A6-THREE	15	71% (189)	83% (168)
A7-IDF	15	56% (202)	82% (172)

3.2 Evaluation of the Query Generation Algorithms

We first evaluated all the baseline algorithms with two different ways of postprocessing, namely no postprocessing and postprocessing by both boosting and filtering. The CNN data set consists of 3854 seconds, and thus an algorithm that issues a query every 15 seconds issues 257 queries. We return the top two web pages for each query so that a maximum of 514 relevant web pages could be returned for this data set when $s = 15$. For the HN data set the corresponding number is 557.

Table 1 presents the achieved precision for different algorithms, i.e., the percentage of relevant documents together with the actual number of relevant documents found by each algorithm for the HN data set. Table 2 shows the corresponding numbers for the CNN data set. It leads to a few observations:

- All algorithms perform statistically significantly better with boosting and filtering postprocessing than without postprocessing. Depending on the algorithm the postprocessing seems to increases the precision by 20-30 percentage points.

- For both data sets the highest precision numbers are achieved with postprocessing and $s = 15$. However, the largest number of relevant pages is returned without postprocessing and $s = 7$. This is no surprise: Postprocessing reduces not only the number of non-relevant documents that are returned, but also the number of relevant ones. The impact on the number of relevant pages that are returned varies greatly between algorithms. The maximum change is 70 pages (A2-IDF2 with $s = 15$ on CNN), the minimum change is 8 pages (A7-IDF with $s = 15$ on HN). Also reducing s increases the number of queries issued and thus one would expect the number of returned pages to increase, both the relevant ones as well as the non-relevant ones.

- Precision on the CNN data set is lower than precision on the HN data set. This is somewhat surprising as longer topics might be expected to lead to higher precision.

- Algorithm A5-HIST with $s = 7$ and with postprocessing performs well in both precision and number of returned relevant pages. For the HN data set, it achieves a precision of 91% with 257 relevant pages returned, for the CNN data set it achies a precision of 84% with 190 relevant pages returned. This means it returns a relevant page every 16 seconds, respectively, every 20 seconds in the average. None of the other algorithms performs as well for both criteria and on both data sets. For example, algorithms A1-BASE and A2-IDF2 with $s = 15$ have precision 91% on the HN data set but return a much smaller number of pages and on the CNN data set their precision is lower.

  Also, for $s = 7$ there is a statistically significant difference in precision between A5-HIST and A1-BASE and between A5-HIST and A2-IDF2. Note that precision for A1-BASE and A2-IDF2 for $s = 15$ matches that of A5-HIST for $s = 7$, but the number of returned relevant pages is roughly 100 pages smaller, since queries are only issued every 15 seconds.

We also discuss the contribution of different techniques.

- *idf versus idf^2*: The baseline algorithm A1-BASE and algorithm A2-IDF2 differ only in the use of *idf^2* versus *idf*. Their performance is very similar. However, also algorithm A5-HIST and A7-IDF differ only in the use of *idf^2* versus *idf*. On the HN data set A5-HIST outperforms A7-IDF when no postprocessing is used, but with postprocessing their performance is stimilar. On the CNN data set A5-HIST outperforms A7-IDF in all settings; for $s = 15$ and no postprocessing the difference is statistically significant. So altogether *idf^2* seems to work slightly better than *idf*.

- *Stemming:* Adding stemming to algorithm A2-IDF2 gives algorithm A3-STEM. On the HN data set stemming gives an improvement without postprocessing, it is statistically significant for $s = 7$ and $s = 15$, but with postprocessing and $s = 15$ stemming gives slightly worse performance. On the CNN data set stemming hurts precision, with the difference being statistically significant for $s = 15$ and postprocessing on. In conclusion, stemming seems to hurt performance when postprocessing with both boosting and filtering is used.

- *Compounds:* Algorithm A4-COMP consists of algorithm A3-STEM with 2-word compounding added. Thus, we only evaluated compounding for algorithms that use stemming. Their performance is very similar. There is an improvement for $s = 15$ on the CNN data set but it is not statistically significant.

Table 3: HN data set: Precision and in parenthesis the number of such documents

	s	None	Boost	Filter	Boost Filter	Re-rank	Rerank Filter
A2-IDF2	7	58% (316)	58% (313)	88% (269)	87% (266)	60% (323)	84% (289)
A4-COMP	7	64% (271)	66% (277)	86% (230)	88% (239)	68% (289)	86% (269)
A5-HIST	7	64% (303)	64% (305)	91% (249)	91% (257)	64% (301)	88% (263)
A2-IDF2	15	62% (165)	64% (170)	89% (143)	91% (152)	66% (174)	92% (166)
A4-COMP	15	70% (221)	72% (229)	93% (192)	90% (210)	74% (229)	91% (212)
A5-HIST	15	67% (219)	69% (224)	92% (188)	89% (201)	71% (223)	92% (209)

- *History:* Adding a "history-feature" to algorithm A4-COMP gives algorithm A5-HIST. The history gives a small improvement for $s = 7$ on the HN data set, while it seems to slightly hurt for $s = 15$. On the CNN data set, A5-HIST clearly outperforms A4-COMP; the difference is statistically significant for both variants of $s = 7$. This is not surprising. For longer topics it becomes valuable to have a history feature, especially if queries are issued every 7 seconds. Each text segment may not on its own contain highly relevant text that can be used as a query in finding similar stories. Shorter text segments suffer even more from this problem. The history rectifies this by effectively extending the length of the text segment in a time-aged manner.

  For example for one of the data sets three shootings (one in Arizona, one in Oklahoma, and one in Jordan) were in the news. The algorithms without history sometimes returned a non-relevant pages since it was for a different shooting. Algorithm A5-HIST never made this mistake. Altogether, we recommend adding a history-feature.

- *Query shortening:* Algorithm A6-THREE first issues a three-word query and "backs off" to a two-word query if no results were found. Its performance is very similar to algorithm A5-HIST. Thus, trying out three-word queries does not lead to an improvement.

Table A and table 10 in the appendix give the percentage of documents exactly on topic (R+: given a score of 2 by the evaluator) together with the actual number of such documents found by each algorithm. They confirm the above observations with one difference: Stemming clearly hurts the precentage of documents exactly on topic for $s = 15$ with postprocessing. On the CNN data set the difference is statistically significant with a p-value of 0.0033.

In conclusion, filtering and the "history feature" give the largest improvement in search precision.

3.3 Postprocessing

As we saw in the previous section postprocessing using boosting and filtering gives a big improvement in precision without decreasing the number of relevant documents that are returned by much. The obvious question is what contributed most the improvement, boosting or filtering. A second question is whether postprocessing by similarity reranking performs better than postprocessing by boosting.

Table 4: HN data set with $s = 7$: Percentage of queries that are identical when sorted lexicographically. Note that this table is not symmetric.

	A1-BASE	A2-IDF2	A3-STEM	A4-COMP	A5-HIST	A6-THREE	A7-IDF
A1-BASE	7	1.00	0.94	0.27	0.25	0.10	0.08
A2-IDF2	7	0.94	1.00	0.30	0.27	0.12	0.09
A3-STEM	7	0.59	0.62	1.00	0.87	0.42	0.34
A4-COMP	7	0.57	0.59	0.87	1.00	0.48	0.36
A5-HIST	7	0.36	0.37	0.33	0.40	1.00	0.84
A6-THREE	7	0.30	0.31	0.19	0.19	0.40	1.00
A7-IDF	7	0.38	0.38	0.32	0.38	0.67	0.10

Table 5: HN data set with $s = 7$: Percentage of URLs of algorithm Alg1 that are also returned by algorithm Alg2. Since different algorithms return a different number of URLs the table is not symmetric.

	A1-BASE	A2-IDF2	A3-STEM	A4-COMP	A5-HIST	A6-THREE	A7-IDF	
A1-BASE	7	1.00	0.93	0.36	0.33	0.15	0.11	0.
A2-IDF2	7	0.96	1.00	0.37	0.36	0.17	0.13	0.
A3-STEM	7	0.41	0.41	1.00	0.83	0.36	0.23	0.
A4-COMP	7	0.36	0.38	0.80	1.00	0.42	0.24	0.
A5-HIST	7	0.16	0.18	0.35	0.42	1.00	0.39	0.
A6-THREE	7	0.13	0.15	0.23	0.26	0.43	1.00	0.
A7-IDF	7	0.15	0.17	0.22	0.30	0.43	0.38	1.

Since the improvement was unanimous among algorithms and data sets, we evaluated only the HN data set for 3 algorithms. Table 3 shows the details. In all six cases the improvement is clearly achieved by the filtering step, the boosting step only giving a small improvement. In some cases filtering alone gives even higher precision than filtering and boosting together.

Similarity reranking seems to give a slightly higher gain in precision than boosting. However, combined with filtering it does not perform better than boosting and filtering combined. None of the differences between boosting and reranking and between boosting with filtering and reranking with filtering are statistically significant.

We got the same results when analyzing the percentage of documents with score R+. Table 13 in the appendix presents the data for the CNN data sets. Specifically, it shows that reranking alone performs very similar to using no postprocessing and that reranking and boosting performs very slightly worse than reranking and filtering. However, the difference is not statistically significant.

To summarize, filtering gives a large precision improvement, about 20-30 percentage points with only a mild decrease in the number of relevant documets that are returned.

3.4 Query Overlap and URL Overlap

Given a postprocessing step the performance of the different query selection algorithms is very similar. An obvious question to ask is whether the reason for this similarity is that the algorithms issue very similar queries. To answer this question we compute the similarity between the queries issued by the different query selection algorithms, i.e., we compare the ith query issued by one algorithm with the ith query isseud by another algorithm. Table 4 gives the percentage of queries that are exactly identical for $s = 7$ and the HN data set. Note that we are looking at all generated queries, i.e., the queries *before* the postprocessing step. Table ?? gives the percentage of queries that overlap in at least one word. As one can see the queries are identical or overlapping for related algorithms like A1 and A2. However, for algorithms A1 and A5 for example, less than 10% of the queries are identical and only 45% of the queries overlap. Table 14 give the coorresponding data for $s = 15$. They can be found in the appendix.

Even if the queries are quite different, there could still be a large overlap in the URLs returned at a given point in the stream of text. However, that is also not the case as table 5 show for the HN data set and $s = 7$. The results for $s = 15$ are similar.

To summarize the overlap both in queries and in web pages is high between A1-BASE and A2-IDF2 and is high between A3-STEM and A4-COMP but is low otherwise.

3.5 Topic Coverage

Table 6: HN data set: Percentage of topics with at least one relevant document.

Technique	s	None	Boost'N' Filter
A1-BASE	7	0.78	0.73
A2-IDF2	7	0.79	0.76
A3-STEM	7	0.74	0.70
A4-COMP	7	0.76	0.72
A5-HIST	7	0.77	0.70
A6-THREE	7	0.73	0.70
A7-IDF	7	0.73	0.73
A1-BASE	15	0.63	0.59
A2-IDF2	15	0.63	0.61
A3-STEM	15	0.72	0.67
A4-COMP	15	0.76	0.72
A5-HIST	15	0.72	0.65
A6-THREE	7	0.71	0.66
A7-IDF	15	0.71	0.69

Another question to ask is how many of the topics receive at least one relevant document. In the HN data set there were a total of 82 topics. In Table 3.5 we show the percentage of topics with at least one relevant document for the HN data set. Not surprisingly, these percentages are strongly correlated with the number of relevant documents. They are the highest for $s = 7$ with no postprocessing and the lowest for $s = 15$ with postprocessing. Table 3.5 presents the percentage of topics with at least one document rated R+ for the HN data set. It is interesting to note that the numbers are not much lower than for Table 3.5: If a topic has a relevant document it most likely also has a topic rated R+.

Table 11 and table 12 in the appendix give the corresponding percentages for the CNN data sets. The values are higher as we would expect since the topics are longer. However, there is also more variation in these numbers as there are only 36 topics in the CNN data set.

In summary, roughly 70% of the topics have at least one document rated relevant, and almost as many have at least one document rated very relevant (R+).

3.6 Filtering Effectiveness

The filtering technique is very powerful in improving precision. Recall that there can be two reasons why a page is filtered out: (F1) Its similarity with text segment T is below threshold b. (F2) Its similarity with text segment T is below threshold g *and* there are two search results and their similarity score is below a third threshold. Note that it is possible that both rules apply. We analyzed which

Table 7: HN data set: Percentage of topics with at least one document rated R+.

Technique	s	None	Boost'N' Filter
A1-BASE	7	0.76	0.70
A2-IDF2	7	0.76	0.72
A3-STEM	7	0.70	0.67
A4-COMP	7	0.70	0.68
A5-HIST	7	0.73	0.67
A6-THREE	7	0.70	0.68
A7-IDF	7	0.72	0.70
A1-BASE	15	0.60	0.56
A2-IDF2	15	0.60	0.69
A3-STEM	15	0.70	0.67
A4-COMP	15	0.73	0.71
A5-HIST	15	0.68	0.65
A6-THREE	15	0.66	0.63
A7-IDF	15	0.70	0.63

Table 8: HN data set: For each filtering technique the percentage of filtered pages that are filtered by the technique. The percentages for a given algorithm can add up to over 100% since both filtering techniques can apply.

	Number of filtered pages	(F1)	(F2)
7.A1	218	0.39	0.97
7.A2	202	0.38	0.96
7.A3	139	0.27	0.98
7.A4	127	0.30	0.98
7.A5	175	0.54	0.86
7.A6	209	0.48	0.86
7.A7	130	0.54	0.78
15.A1	126	0.24	0.98
15.A2	85	0.29	0.96
15.A3	76	0.24	0.93
15.A4	76	0.22	0.93
15.A5	95	0.32	0.92
15.A6	130	0.26	0.97
15.A7	36	0.33	0.81

of the two rules filters out more web pages. Table 3.6 below shows for either filtering technique the percentage of pages it filtered on the HN data set. The percentage can add up to over 100% since both rules can apply. It clearly shows that (F2) filters out most of the pages.

4. RELATED WORK

4.1 Query-free search

To our knowledge, there has been no previous work on automatically selecting documents that a user might want to see while watching a TV program. However, there is a significant literature on the broader problem of query-free information retrieval: finding documents that are relevant to a user's current activity, without requiring an explicit query. The different systems differ in what stream of text they consider as input and what genre of related documents they return. We will use the "Input—Output" notation below.

- *Web pages—web pages*: The Letizia system [9] observes a user browsing the web, and suggests other web pages the user may find interesting. Rather than searching an index of web pages, it "surfs ahead" of the user, following hyperlinks from the page the user is currently viewing. Similarly, commercial browser assistants such as Autonomy Kenjin and PurpleYogi (both no longer available) suggested related web pages based on the content of web pages the user has been viewing.

- *Problem report—repair manual*: Another early query-free IR system is FIXIT [7], which helps technicians as they use an expert system to diagnose and repair copiers. FIXIT identifies the currently reported symptoms and the faults it considers likely, then maps these symptoms and faults to keywords, and retrieves sections of the copier documentation that match these words.

- *User behavior—personal files*: The just-in-time IR project at MIT [14, 13] has focused on retrieving personal files – such as notes and archived email messages – that a user would currently find useful. This project first produced the Remembrance Agent, which looks at a document the user is editing in Emacs and matches fragments of this document (such as the last 50 words) against a corpus of personal files. The followup Margin Notes system performs a similar task, but observes the web pages that a user views in a web browser. Finally, the Jimminy system runs on a wearable computer. Jimminy bases its suggestions on what the user is reading or writing on the heads-up display, as well as on Global Positioning System data and active badge data indicating what other people are nearby. All these systems use a common information retrieval backend based on the Okapi similarity metric [15].

  The XLibris pen-based document reader [12] allowed users to markup documents as they are reading. The system would derive queries from the passages of text that were marked, and search over a local corpus for relevant documents to present to the user.

- *User behavior—News and stock quotes*: The SUITOR system [10] tracks user behavior like what applications are running and what text the user currently writes to build a model of the users current interest. It uses this model to find information that is interesting to the user like news headlines and stock quotes.

- *Open documents in editor or browser—web pages*: The system most similar in purpose to our own is Watson [4], which suggests web pages to a computer user based on the documents currently open in a word processor or web browser. Watson uses a variety of heuristics to construct queries from the text of the documents, then sends these queries to the AltaVista search engine.

- *Email—web pages*: Our work is also related to a small prototype system that constructed queries from email messages and sent them to an early version of the Google search engine [3].

4.2 Text Summarization and Keyword Extraction

In the Information Retrieval literature there has been a plethora of work on topic detection and text summarization. The more recent research on time-based summarization is the work most closely related to ours. See [1] for an excellent overview of the area. Our work is different in two ways:

1. It doesn't need to identify topics; it only needs to detect whether the current topic is different from previous topics. If a later topic is very similar to a topic discussed much earlier, the system does not need to recognize this.

2. The system does not need to construct a summary; it extracts keyphrases that can be used to formulate a search query.

The research on keyphrase extraction, see, e.g., [8, 11, 18, 6], and specifically the algorithm by [19], is the most related to our work. In comparison, the novelty in our work is the addition of a time component, combining keyphrase extraction with topic aging and topic change detection.

5. CONCLUSION

This paper evaluated seven algorithms for finding web pages relevant to news broadcasts and three postprocessing techniques. For this genre of television show, the best algorithm finds a relevant page every 16-20 seconds in the average and achieves a precision of 84-91%. Our main findings are: All query generation algorithms that we studied work roughly as well, adding a history feature seems to help somewhat on longer topics. It would be interesting to experiment with different ways of using the history for query generation. Postprocessing by filtering out web pages that are very vague gives a large improvement in precision without decreasing the number of returned relevant documents by much.

The framework of the system is not limited to news, however; we have considered simple methods of detecting other genres (such as sports, weather, and "general" topics) and sending such queries to appropriate web information sources. The genres could be identified by using machine learning on a labelled corpus of television captions; but perhaps an even simpler way would be to use television schedules and their associated metadata to categorize the current show into a genre.

Finally, as voice recognition systems improve, the same kind of topic finding and query generation algorithms described in this paper could be applied to conversations, providing relevant information immediately upon demand.

5.1 Acknowledgments

We would like to thank Shahid Choudhry for providing us with closed caption transcripts for our experiments.

6. ADDITIONAL AUTHORS

Additional authors: Sergey Brin (Google Incorporated).

7. REFERENCES

[1] J. Allan, R. Gupta, and V. Khandelwal. Temporal summaries of news topics. In *Research and Development in Information Retrieval*, pages 10–18, 2001.

[2] E. I. Alliance. Eia-746-a: Transport of internet uniform resource locator (url) information using text-2 (t-2) service. Technical report, 1998.

[3] S. Brin, R. Motwani, L. Page, and T. Winograd. What can you do with a web in your pocket? *Data Engineering Bulletin*, 21(2):37–47, 1998.

[4] J. Budzik, K. Hammond, and L. Birnbaum. Information access in context. *Knowledge based systems*, 14(1-2):37–53, 2001.

[5] J. Davis. Intercast dying of neglext. CNET News, January 29, 1997.

[6] E. Frank, G. W. Paynter, I. H. Witten, C. Gutwin, and C. G. Nevill-Manning. Domain-specific keyphrase extraction. In *IJCAI*, pages 668–673, 1999.

[7] P. Hart and J. Graham. Query-free information retrieval. *IEEE Expert*, 12(5):32–37, 1997.

[8] B. Krulwich and C. Burkey. Learning user information interests through the extraction of semantically significant phrases. In *AAAI 1996 Spring Symposium on Machine Learning in Information Access*, 1996.

[9] H. Lieberman. Letizia: An agent that assists web browsing. In C. S. Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 924–929, Montreal, Quebec, Canada, 1995. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.

[10] P. Maglio, R. Barrett, C. Campbell, and T. Selker. Suitor: An attentive information system. In *International Conference on Intelligent User Interfaces*, 2000.

[11] A. Munoz. Compound key word generation from document databases using a hierarchical clustering art model. *Intelligent Data Analysis*, 1(1), 1997.

[12] M. N. Price, G. Golovchinsky, and B. N. Schilit. Linking by inking: Trailblazing in a paper-like hypertext. In *Hypertext '98*, pages 30–39, 1998.

[13] B. Rhodes and P. Maes. Just-in-time information retrieval agents. *IBM Systems Journal*, 39(3-4), 2000.

[14] B. J. Rhodes. *Just-In-Time Information Retrieval*. PhD thesis, MIT Media Laboratory, Cambridge, MA, May 2000.

[15] S. Robertson, S. Walker, and M. Beaulieu. Okapi at trec-7: automatic ad hoc, filtering, vlc and interactive track. In *Proceedings of the 7th International Text Retrieval Conference (TREC)*, pages 253–264, 1999.

[16] G. D. Robson. Closed captions, v-chip, and other vbi data. *Nuts and Volts*, 2000.

[17] G. Salton. *The SMART System – Experiments in Automatic Document Processing*. Prentice Hall, 1971.

[18] A. M. Steier and R. K. Belew. Exporting phrases: A statistical analysis of topical language. In *Second Symposium on Document Analysis and Information Retrieval*, pages 179–190, 1993.

[19] P. D. Turney. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2(4):303–336, 2000.

APPENDIX

A. MORE EVALUATION DATA

Table 9: HN data set: Percentage of documents with score R+ and in parenthesis the number of such documents

Technique	Sampling Freq	None	Boost'N' Filter
A1-BASE	7	44% (240)	69% (211)
A2-IDF2	7	45% (245)	70% (214)
A3-STEM	7	49% (209)	73% (204)
A4-COMP	7	50% (209)	72% (196)
A5-HIST	7	47% (222)	76% (214)
A6-THREE	7	56% (216)	75% (197)
A7-IDF	7	47% (171)	74% (219)
A1-BASE	15	53% (139)	81% (131)
A2-IDF2	15	51% (136)	78% (131)
A3-STEM	15	54% (165)	75% (169)
A4-COMP	15	51% (162)	72% (169)
A5-HIST	15	52% (168)	71% (159)
A6-THREE	15	59% (158)	75% (155)
A7-IDF	15	46% (173)	75% (171)

Table 10: CNN data set: Percentage of documents with score R+ and in parenthesis the number of such documents

Technique	s	None	Boost'N' Filter
A1-BASE	7	30% (155)	61% (134)
A2-IDF2	7	31% (148)	59% (117)
A3-STEM	7	31% (133)	59% (112)
A4-COMP	7	31% (130)	59% (109)
A5-HIST	7	36% (169)	64% (145)
A6-THREE	7	40% (161)	61% (136)
A7-IDF	7	33% (169)	65% (148)
A1-BASE	15	35% (101)	66% (92)
A2-IDF2	15	43% (96)	67% (85)
A3-STEM	15	37% (95)	51% (76)
A4-COMP	15	39% (95)	58% (85)
A5-HIST	15	40% (129)	60% (112)
A6-THREE	15	49% (130)	59% (120)
A7-IDF	15	36% (129)	56% (118)

Table 11: CNN data set: Percentage of topics with at least one relevant document.

Technique	s	None	Boost'N' Filter
A1-BASE	7	0.86	0.81
A2-IDF2	7	0.83	0.81
A3-STEM	7	0.83	0.72
A4-COMP	7	0.83	0.75
A5-HIST	7	0.89	0.72
A7-IDF	7	0.92	0.69
A1-BASE	15	0.81	0.78
A2-IDF2	15	0.75	0.72
A3-STEM	15	0.69	0.64
A4-COMP	15	0.72	0.67
A5-HIST	15	0.78	0.75
A7-IDF	15	0.78	0.75

Table 12: CNN data set: Percentage of topics with at least one document rated R+.

Technique	s	None	Boost'N' Filter
A1-BASE	7	0.83	0.81
A2-IDF2	7	0.81	0.75
A3-STEM	7	0.72	0.69
A4-COMP	7	0.78	0.69
A5-HIST	7	0.81	0.72
A7-IDF	7	0.78	0.67
A1-BASE	15	0.72	0.72
A2-IDF2	15	0.67	0.61
A3-STEM	15	0.64	0.61
A4-COMP	15	0.64	0.64
A5-HIST	15	0.69	0.67
A7-IDF	15	0.64	0.69

Table 13: HN data set: Precision and in parenthesis the number of such documents

	s	None	Boost'N' Filter	Rerank	Rerank'N' Filter
A2-IDF2	7	46% (220)	75% (150)	48% (233)	75% (188)
A4-COMP	7	44% (186)	76% (140)	47% (200)	77% (164)
A5-HIST	7	55% (259)	84% (190)	56% (251)	85% (165)
A2-IDF2	15	60% (134)	85% (108)	59% (138)	78% (116)
A4-COMP	15	59% (144)	82% (120)	59% (145)	76% (126)
A5-HIST	15	61% (200)	88% (164)	65% (191)	81% (166)

Table 14: HN data set with $s = 15$: Percentage of queries that are identical when sorted lexicographically.

	A1-BASE	A2-IDF2	A3-STEM	A4-COMP	A5-HIST	A6-THREE	A7-IDF	
A1-BASE	15	1.00	0.75	0.34	0.27	0.11	0.09	0.
A2-IDF2	15	0.75	1.00	0.40	0.32	0.13	0.11	0.
A3-STEM	15	0.36	0.42	1.00	0.82	0.34	0.26	0.
A4-COMP	15	0.30	0.35	0.82	1.00	0.45	0.29	0.
A5-HIST	15	0.14	0.16	0.34	0.46	1.00	0.51	0.
A6-THREE	15	0.09	0.11	0.21	0.21	0.38	1.00	0.
A7-IDF	15	0.19	0.17	0.28	0.35	0.60	0.38	1.