Evaluating query log segmentation for frustration detection

Henry Feild
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003
hfeild@cs.umass.edu

James Allan
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003
allan@cs.umass.edu

ABSTRACT

We explore the problem of identifying and grouping information needs within a stream of queries. Our dataset contains sequences of queries issued by medical specialists to a domain-specific search system. Because the system allows for group logins, a sequence of queries could come from any number of specialists sharing an access point (for example, in an emergency room). We present several approaches for segmenting the stream of queries into tasks. Our ultimate goal is to detect frustrated searchers who are unsuccessful at satisfying their information needs. With that task in mind, we present an alternative evaluation measure and compare it to task-agnostic measures.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Query formulation*

General Terms

Algorithms, Experimentation, Measurement

Keywords

task segmentation, query log mining, user frustration, search goal

1. INTRODUCTION

Search engines serve to help users satisfy some information need (i.e., a *task*). However, when a user has trouble completing his task, he will likely become *frustrated.* We would like to explore methods of detecting when a user is frustrated, which would be helpful in two ways. First, a system that is capable of detecting frustration can modify its behavior to address the user's frustration. Second, the system could log when it believes the user is frustrated for retrospective failure analysis. For example, to examine search sessions where users seem frustrated, but fulfill their information need by clicking a document. Such sessions could uncover parts of the search system or interaction process that cause disruptions to search. More importantly, these sessions contain potential solutions: the users found a way to bypass these problems and ultimately succeed in their tasks. In both situations, the system can be improved to help fulfill more information needs, resulting in better search.

Past research in human-computer interaction, information science, and information retrieval has observed user frustration in Web search [1, 2, 6, 9, 10]. One study found that Web browsing, which presumably includes Web search, is the most frequent source of user frustration, followed by email and word-processing [2].

We know of little work on automatic detection of frustration and none strictly applied to information retrieval (IR). In this paper, we will use a task-based frustration detector, which considers user-system interactions from the beginning of a task to the most recent query for that task. The user interactions we consider are: logging in or out of the system, issuing a query, clicking on or printing a document, clicking on a system suggestion, or paging through results.

A hurdle for such task-based frustration detectors is multi-task sessions. In these sessions, users try to satisfy more than one information need. These may be contiguous or interleaved. Jones and Klinkner found that roughly 17% of tasks are interleaved within search sessions from a three-day sample of Yahoo! query logs [5]. Spink et al. observed that 81% of two-query sessions and 91% of three or more query sessions included multiple topics from samples of AltaVista's query logs [8]. In our dataset, we found that up to 94% of 7–10 query sessions have multiple tasks and as many as 10% of tasks are interleaved. Such sessions are problematic for a task-based frustration detector because the user interactions from one task bleed into the other tasks, hampering the ability of the detector to estimate the users' frustration with respect to the actual tasks.

We can address the problem of detecting frustration across multiple, interleaved tasks by performing task-segmentation on the search session and then using interactions within each task to decide a user's frustration.

We focus here only on detecting general frustration and ignore specific causes. We do not examine how a system could be modified to deal with frustrated users; we instead leave this to future work. It is our opinion, however, that detecting and particularly dealing with a particular user's frustration are very likely dependent on the cause of the frustration.

We use a simple, automatic method to detect frustration and evaluate several task segmentation methods based on the performance of the frustration detector on the segmentations. Our dataset consists of 100 user sessions with between 7 and 10 queries sampled from one month of query logs from a medical publication search system. Each session was manually segmented into tasks to use as a gold standard.

We find that evaluating task segmentation by the accuracy of classifying pairs of queries as belonging to the same task is not sufficient for frustration detection. Rather, techniques that perform well at segmentation do not necessarily produce segmentations on which the frustration detector performs well.

2. BACKGROUND

In this section, we describe previous work from the areas of task segmentation and user frustration and satisfaction. We also provide definitions of the terms used in the remainder of the paper.

2.1 Related Work

There is a large body of research concerning query log mining and how events in the logs should be organized into sessions or tasks.

Jansen, Spink, and Kathuria [4] explored three ways of defining a user session from the events in a query log. The first considered all events that shared a common IP address and cookie. The second method used a combination of IP address, cookie, and time-out between consecutive events. The third method considered IP address, cookie, and a lexical similarity between consecutive queries. Specifically, if two consecutive queries with the same IP and cookie shared at least one word in common, they were considered a part of the same session. They found the third method to be the best with 95% accuracy on sessions examined from Dogpile.com.

Spink et al. [8] conducted a qualitative investigation of multi-query sessions from AltaVista's query logs. They found that 81% of two-query sessions and 91% of three or more query sessions included multiple topics. They also observed that the variety of topics covered in multi-tasking sessions was broad. For example, one user entered queries related to fashion alongside queries of a medical nature. Finally, they found frequent topic changes were present in sessions with three or more queries.

Jones and Klinkner [5] defined a user *search session* to consist of all user activity within a fixed time window. They defined a *search goal* as an atomic information need, resulting in one or more queries and a *search mission* as a related set of information needs, resulting in one or more goals. They then annotated a sample of sessions from Yahoo!'s query logs, assigning each query within a session a goal and mission identifier. These definitions allow goals and missions to be interleaved. Indeed, Jones and Klinkner found that 16% of goals and 17% of missions interleaved in their 312 user sessions sample.

They then investigated logistic regression classifiers to address two tasks. The first was to automatically find goal or mission boundaries between pairs of consecutive queries within a session. They were able to do this with 90.8% accuracy for mission boundaries and 93.0% accuracy for goal boundaries. The second task was to classify pairs of queries from within a session as belonging to the same goal or mission. This was accomplished with an accuracy of 88.8% for missions and 97.2% for goals.

Our work builds on that of Jones and Klinkner, but with the goal of identifying frustration within goals or missions. We also explore the questions on a particular vertical search system where conclusions from the general Web may not apply.

Huffman and Hochster [3] examined how a relevance measure based on the top three documents returned by the first query in a user sessions compares to session-level user satisfaction. They found that "this relationship is surprisingly strong". Because of the nature of their experimental setup, the sessions they examined probably did not contain multiple tasks. Thus, if a similar study was to be performed on full sessions from a query log, task segmentation would be important.

Bilal and Kirby [1] examined the search behaviors of middle school and graduate students on the Yahooligans! search system. They found that over 50% of the graduate student were frustrated at some point during their search session. However, 89% of the graduate students felt satisfied with their search by the end. This demonstrates that frustration does not necessarily lead to dissatisfaction.

Wang, Hawk, and Tenopir [10] conducted a study to explore the interaction of graduate students with a Web search interface. They found that "negative feelings can result in a decision to give up on the right strategy". Thus, addressing frustrated users could avoid such situations. Ceaparu et al. [2] examined user frustration in the context of a variety of computer applications. They found that Web browsing caused the most frustration, in part because of its popularity.

2.2 Definitions

We will use the Jones and Klinkner's definitions for *goal* and *mission* [5]. We define a *session* to be a set of query log events that share the same session cookie identifier. We will also define a *task* to be either a search goal or mission. Thus, goals and missions differ only in their granularities of a task.

To obtain a working definition of frustration, we look to the human-computer interaction literature. Ceaparu et al state that "people are frustrated if they are prevented from achieving expected satisfying results" and that "satisfaction is also defined as the completion of a goal or task" [2]. The authors go on to say that "users can still achieve satisfaction in their tasks despite the presence of frustration in the path of task achievement". In this paper, we use the following definitions with regard to *frustration* and *satisfaction*:

Frustration is the affective state of the user at any given point in a search and is based on the interactions between the user and the search system up until that point. A user is frustrated if the events leading up to this moment in time are preventing the achievement of satisfaction with respect to the task. Thus, frustration is defined on the event-level. We will specifically consider frustration at query-events.

Satisfaction is the affective state of the user at the end of their search task. A user is satisfied if the task has been successfully completed. Thus, satisfaction is defined on the task-level.

	Goals	Missions
Min	1.0	1.0
Median	1.0	2.0
Mode	1.0	1.0
Max	10.0	10.0
Avg	1.9	3.1

Table 1: General statistics about the number of queries per goal and mission in the sample sessions used.

We will assume that users' state of frustration at the time a query is entered is the same as their satisfaction if the task were to have ended with the previous event. Therefore, frustration is a function of success at different intervals during a task.

3. DATASET ANALYSIS

This study focuses on task segmentation in query logs from the *UpToDate* medical publication database search system. The database serves hundreds of thousands of clinicians across 140 countries[1]. The search system receives about 250,000 queries across 115,000 sessions per day. Sessions are tracked using a session cookie. Averaged over ten months of query logs, there are 2.2 queries per session.

It is interesting to investigate searcher frustration on this data set, since the primary users are medical professionals. When information needs are related to someone's health, it is important to help address those needs as quickly as possible—and keep the user as non-frustrated as possible along the way.

To explore methods of evaluating task-segmentation, we first created a gold standard for a sample of user sessions. We randomly selected 100 sessions with between 7 and 10 queries from February 2008. These sessions were used with the expectation that they would be more likely to contain multiple tasks, and at the same time, were not so long as to be overly burdensome for annotators. The 100 sessions contained 819 queries.

We asked annotators to re-enact each session as though they were the searcher, much as the annotators in Jones and Klinkner's study did [5]. They were to label each query with a goal and mission identifier, such that all queries that they felt belonged to the same goal shared a goal identifier, and likewise for the missions. In addition, any time a new goal or mission identifier was used, the annotators were asked to enter a description of the goal or mission. Annotators were permitted to look up information in other resources, such as the Internet, to better understand the user intent. Our annotators were computer science graduate students, one of which was one of the authors, and not always familiar with the medical vocabulary of queries (e.g., "bun gi bleed", which refers to blood urea nitrogen gastrointestinal bleeding).

Across the 100 sessions, annotators found 424 goals and 267 missions. Ninety-four percent of sessions have more than one goal while 67% have more than one mission. There are 1.9 queries per goal and 3.1 queries per mission (see Table 1). Missions consist of roughly 1.6 goals. Each session has an average of 4.2 goals and 2.7 missions. Forty-seven goals or 9.5% were revisited, while only 19 missions or 4.5% were revisited. This is lower than in previous work [5] that dealt with Web search. This could be a reflection of the population of users in this study. Since our system is used by medical professionals, it may be that the searches are geared toward very specific information needs (e.g., "what is the recommended treatment of this condition in a diabetic?").

4. TASK SEGMENTATION

In this section, we describe the task-segmentation methods we examine. The results of these techniques are presented in Section 6. We chose a variety of simple techniques as baselines, such as assigning each query its own task or assigning one task to all queries within a session. We also explore a variety of timeouts, such that all queries occurring between session boundaries or timeouts are considered a part of one task, and no tasks span those boundaries. Typical boundaries are 15 and 30 minutes, meaning that as soon as there has been no logged user interaction with the system for at least 15 or 30 minutes, a new task boundary is declared. A 15-minute timeout is used by our medical search organization to do session analysis. A 30-minute timeout was found to be effective in task segmentation by Randlinski and Joachims [7], though Jones and Klinkner found this was not the case with their sample of Yahoo! query logs [5]. In addition to looking at just the typical timeouts, we evaluate task-segmentation using timeouts of 0 to 120 minutes at one minute increments. This way, we can find the optimal timeout for the medical users of this system.

We also trained a maximum entropy classifier using feature sets that past research has shown to work well [5], with a few exceptions. The features used are listed in Table 2. The keyword-based features used in this study depend on a set of keywords that are associated with each document in the our corpus. These keywords are assigned by human editors at the time of publication. The keyword-based features are the cosine similarity between the top 5, 10, 25, and 50 documents returned by a pair of queries. We use th $pEOS\_q2$ feature to specify the probability of the second query in a pair occurring at the end of a search session as opposed to the probability of its occurring as a user's last query before midnight, as it was used by Jones and Klinkner [5]. The query reformulations used for the log-based features were aggregated over the query logs from November 2007 through January 2008.

Two classifiers were trained and tested using 10-fold cross-validation; one for goals and the other for missions. Below we use 'task' to mean either goals or missions. To train the classifier, we took all pairs of queries such that 1) the queries were from the same session and 2) the second query in the pair occurred after the first. Each pair was labeled as belonging to class 'SAME' if the two queries were a part of the same task according to the gold standard and 'DIFFERENT' otherwise.

This pair-wise classification is convenient and easy to understand; however, it does not necessarily segment a session into tasks. Problems arise when the output of a classifier forms a contradiction. For example, consider the queries q_1, q_2, and q_3. Lets suppose that our classifier assigns the following labels:

$$\begin{aligned} \langle q_1, q_2 \rangle &\leftarrow \text{SAME} \\ \langle q_1, q_3 \rangle &\leftarrow \text{SAME} \\ \langle q_2, q_3 \rangle &\leftarrow \text{DIFFERENT} \end{aligned}$$

[1] http://www.uptodate.com/home/about/about.html

Feature	Description
Query log based	
llr	LLR(q_1, q_2); log-likelihood ratio of the co-occurrence of $q1$ and $q2$.
pEOS_q2	Probability that $q2$ is the last query in a session.
pq12	$\frac{p(q_1 \rightarrow q_2)}{\max_{q_j} p(q_1 \rightarrow q_2)}$
entropy_x_q1	$\sum_i p(q_1 \vert q_i) \lg p(q_1 \vert q_i)$
entropy_q1_x	$\sum_i p(q_i \vert q_1) \lg p(q_i \vert q_1)$
nsubst_x_q1	$count(X : \exists p(X \rightarrow q_1))$
nsubst_x_q2	$count(X : \exists p(X \rightarrow q_2))$
usubst_q2_x	$count(X : \exists p(q_2 \rightarrow X))$
seen_in_logs_qp	True if LLR$(q_1, q_2) > n$ for some n.
p_change	$\sum_i p(q_1 \rightarrow q_i) : q_1 \neq q_i$
Keyword based	
keywordSim_5	The cosine similarity between human-editor assigned keywords for the top 5 documents retrieved for q_1 and q_2.
keywordSim_10	Uses the top 10 documents.
keywordSim_25	Uses the top 25 documents.
keywordSim_50	Uses the top 50 documents.
Temporal	
inter_query_5	True if more than five minutes have passed between q_1 and q_2.
inter_query_30	True if 30+ minutes have passed.
inter_query_60	True if 60+ minutes have passed.
inter_query_120	True if 120+ minutes have passed.
time_diff	The time (in seconds) that has passed between q_1 and q_2.
seq_queries	True if q_2 immediately follows q_1 in the session log.
Word and character similarity	
lev	Normalized Levenshtein edit difference between q_1 and q_2
edlevGT2	True if the Levenshtein difference is greater than 2.
char_pov	# of common prefix characters.
char_suf	# of common suffix characters.
word_pov	# of common prefix words.
word_suf	# of common suffix words.
commonw	# of common words.
wordr	Jaccard distance between sets of words in q_1 and q_2.

Table 2: The features used to train the same-goal and same-mission classifiers.

How is this supposed to be segmented? q_1 should go with q_2 and q_3, but we should not put q_2 and q_3 together. If we suppose that the gold standard has all three of the queries belonging to the same task, then we can see that the above classification is 67% accurate. However, in light of the contradiction, we have to ignore at least one of our classified labels, which could reduce our accuracy to 33% or, if the final label were ignored, raise our accuracy to 100%.

A clustering method would be helpful, since it could decide which query pairs should be kept together and which ones should be kept separate. The problem is, we need some sort of distance measure to use for clustering.

Fortunately, maximum entropy classifiers output probabilities in addition to class labels. The particular classifier used in this study uses 0.5 as a threshold, so any query pair having a probability of being from the same task greater than 0.5 is assigned the class 'SAME' and 'DIFFERENT' otherwise. We can use the pair-wise probability as a distance measure, and then stop clustering when no two clusters have a distance greater than 0.5.

As a distance function, we can use average-link (the average distance between the queries in two clusters), minimum-link (the smallest distance between any two queries from two clusters), and maximum link (the maximum distance between any two queries from two clusters).

In addition to the distance measure, we also explore two inherently different methods of clustering with regard to the amount of information available at each stage of the clustering. The first method, on-line clustering, is meant for use in live systems for dynamic task segmentation. The second method, retrospective clustering, is useful for failure analysis and other off-line activities that make use of all the information that was generated from a user session.

4.1 On-line task clustering

In on-line clustering, we are given the logged events from a session one at a time in chronological order. Each time we are given a query, we add it to an existing task or create a new task.

This is useful for a live search system that tries to keep track of what task a user is currently working on. If a user has searched for information about the—we will assume disjoint—tasks of fishing and restaurants and their most recent query is about pike, the system might want to pull information from the events that are specific to the 'fishing' task to help aid the search.

We implemented this clustering technique with average-, minimum-, and maximum-link distance functions. In on-line clustering, the incoming query is a singleton and it is compared with each of the existing clustered tasks. Thus, merges only occur between a singleton and a cluster with one or more queries.

4.2 Retrospective task clustering

Retrospective task clustering allows for all of the logged events to be examined at the time of clustering. This technique can be used as soon as a search session is complete. Rather than merging the most recent query with an existing task or creating a new one, this method takes an agglomerative approach: initialize every query as a singleton cluster and then merge the closest two. This is performed until the stopping criterion is met. We also implemented this clustering method with the three distance functions above.

Retrospective clustering should perform better than on-line clustering since it has more information available, ensuring that only the best clusters are merged. This would make it more reliable for retrospective analysis of search sessions, such as providing statistics.

Ultimately, however, the best method to use for retrospective analysis can be decided empirically, as reported in Section 6.

5. FRUSTRATION DETECTION

Assume we are given a task consisting of some number of user interactions and a new query. Our goal is to predict if

	TP	FP	TN	FN
Goals				
Naïve Detector	119	109	545	46
Always Frustrated	165	654	0	0
Missions				
Naïve Detector	143	151	479	46
Always Frustrated	189	630	0	0

Table 3: Lists the true positive, false positive, true negative, and false negative counts between two objective frustration classifiers and the frustration labels provided by human annotators. The FRUSTRATED label is considered a positive observation.

the user is frustrated or not when the new query is issued. The frustration label is not saying that the user is frustrated with the query or its results, which we have yet to see, but rather that the user is frustrated at the point in time when he enters the query.

We adopt a very simple method that says a user is frustrated if he did not click on any results for the previous query in the task. Even though this method does not necessarily use all of the preceding user interactions in the task to make a prediction, its reliance on the most recent events in the task make it sensitive to the task segmentation. If we do a poor job segmenting tasks, we should do poorly here, as well.

Table 4 shows an example of an actual session from the medical query logs with three goals, one of which is interleaved. The task assignments and frustration labels are shown for the gold standard, for a segmentation where each query is assigned to its own task, and for a segmentation where all queries are considered part of the same task. The tasks in the example are goal-based, so two queries are a part of the same task if they are part of the same atomic information need. The example demonstrates how the accuracy of task segmentation can affect frustration detection. Table 5 shows the same session, but with same-mission labels.

To evaluate this objective frustration detector, we would need to have feedback from the original searchers about their level of frustration for each query they entered. We do not have this information, however. Instead, we asked our annotators to indicate their *intuition* as to whether or not the searcher was frustrated on the goal- and mission-level for each query. They were told to mark the searcher as frustrated if they felt they themselves would have been frustrated at that point in the search. Because this work focuses on task segmentation and not frustration detection, we present the following agreement data to demonstrate where the naïve frustration detector lies with regard to the annotations and a baseline that assumes the searcher is always frustrated.

Table 3 shows the contingency table values for the naïve frustration detector and the always-frustrated baseline versus the annotators' frustration labels. For frustration detection on goals, the naïve detector has 81% accuracy, 52% precision, and 72% recall. Always classify the user as frustrated achieves 20% accuracy, 20% precision, and 100% recall. In other words, always classifying the user as frustrated will annoy more users than the naïve approach.

Technique	Goal	Mission
One task per query	72.10%	43.03%
One task per session	27.90%	56.97%
Timeouts		
15-minute timeout	53.06%	73.98%
30-minute timeout	49.42%	74.64%
Best performing timeouts		
0-minute timeout	72.10%	43.03%
11-minute timeout	56.27%	76.00%
Using classifier results		
No clustering	74.93%	75.56%
on-line, average-link	74.78%	75.70%
on-line, minimum-link	75.04%*	74.38%
on-line, maximum-link	74.08%	76.99%*
retrospective, average-link	74.78%	74.54%
retrospective, minimum-link	74.84%	75.31%
retrospective, maximum-link	74.08%	76.40%

Table 6: The accuracy of a number of same-task classification techniques. The accuracy is measured on pair-wise classification of queries as belonging to the same task within a session. The timeouts refer to maximum amount of time that can pass between two consecutive events in a sessions before a new task is begun. All events between such boundaries, as well as the session start and end, are considered to be a part of the same task. The best results are denoted with an asterisks.

This shows that the simple frustration detector, while it still mislabels many instances of frustration, largely agrees with the annotators' labels. It should be noted that using an always-not-frustrated baseline is not appropriate here. While the accuracy and precision will be high, the recall is always zero, therefore making it no different from today's search interfaces, which do not detect frustration. A recall of zero also causes all F-measures to be undefined, rendering them useless as an evaluation metric.

6. EVALUATION

In this section, we describe how the task segmentations were evaluated directly and with respect to their effect on frustration detection.

6.1 Same-task evaluation

In previous studies, accuracy is reported directly for task segmentation [5, 7]. We do this by comparing the same-task labels for each of the query pairs from segmentation techniques with the gold standard for both goals and missions. In the sample of sessions, there are a total of 3,021 query pairs. According to the annotators, 843 pairs (56.97%) have a SAME-GOAL label and 1,721 pairs (27.92%) have a SAME-MISSION label.

Table 6 shows the accuracies for each of the segmentation techniques. Considering just segmenting tasks by timeouts, 0-minutes, which is the same as assigning one task per query, performed best for goals at 72%. An 11-minute timeout worked best for segmenting missions at 76%. Overall, using on-line clustering with the minimum-link distance function was most accurate for goal segmentation (75%), and on-line clustering with the maximum-link distance function was best

Event No.	Query/Event	Gold std. ID	Gold std. Frust.	One Goal per Query ID	One Goal per Query Frust.	One Goal per Query Err.	One Goal per Session ID	One Goal per Session Frust.	One Goal per Session Err.
1	renal transplant	1	NF	1	NF		1	NF	
2	renal transplant and hypertension	2	NF	2	NF		1	F	X
3	norvasc	3	NF	3	NF		1	F	X
4	⟨*click*⟩	3		3			1		
5	renal transplant	2	F	4	NF	X	1	NF	

Table 4: Three examples of how a session might be segmented into *goals* and the corresponding frustration labels applied. The first baseline assigns every query to its own goal whereas the second baseline assigns every query to the same goal. The frustrations labels (abbreviated 'Frust.') are 'F' for frustrated and 'NF' for not frustrated. The gold-standard task labels are with respect to the assumed user goals. Two of the queries seek information about kidney transplants, one is about kidney transplants for patients with hypertension, and Norvasc is a drug used to treat hypertension. Errors are marked when an 'X'.

Event No.	Query/Event	Gold std. ID	Gold std. Frust.	One Mission per Query ID	One Mission per Query Frust.	One Mission per Query Err.	One Mission per Session ID	One Mission per Session Frust.	One Mission per Session Err.
1	renal transplant	1	NF	1	NF		1	NF	
2	renal transplant and hypertension	1	F	2	NF	X	1	F	
3	norvasc	1	F	3	NF	X	1	F	
4	⟨*click*⟩	1		3			1		
5	renal transplant	1	NF	4	NF		1	NF	

Table 5: Three examples of how a session might be segmented into *missions* and the corresponding frustration labels applied. The first baseline assigns every query to its own mission whereas the second baseline assigns every query to the same mission. The frustrations labels (abbreviated 'Frust.') are 'F' for frustrated and 'NF' for not frustrated. The gold-standard task labels are with respect to the assumed user missions. All of the above queries have to do with information about renal transplants and drugs used to treat hypertension. Errors are marked when an 'X'.

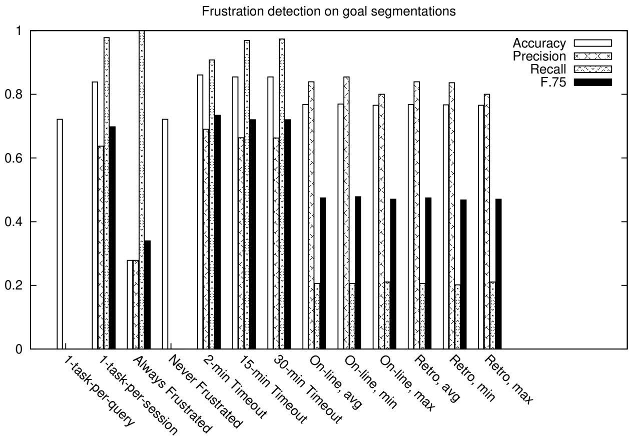

Figure 1: A comparison of frustration classification for several goal-segmentation techniques using four evaluation metrics (accuracy, precision, recall, and $F_{0.75}$).

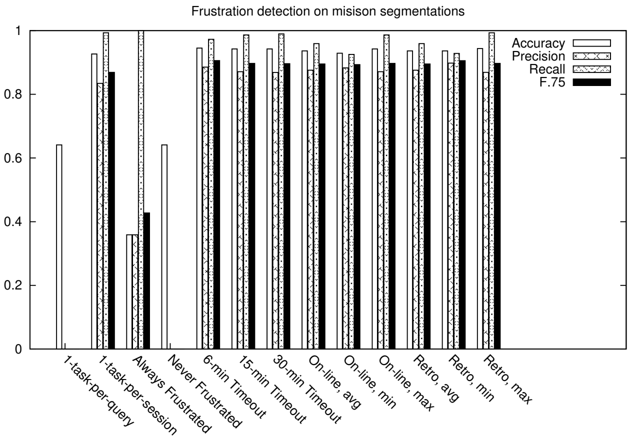

Figure 2: A comparison of frustration classification for several mission-segmentation techniques using four evaluation metrics (accuracy, precision, recall, and $F_{0.75}$).

Technique	**Accuracy**	$F_{0.75}$
Simple Baselines		
One task per query	72.16%	—
One task per session	83.88%	69.80%
Always frustrated	27.84%	33.97%
Never frustration	72.16%	—
Timeouts		
15-minute timeout	85.47%	72.05%
30-minute timeout	85.47%	72.02%
Best performing timeouts		
2-minute timeout	86.08%*	73.40%*
Using classifier results		
on-line, average-link	76.80%	47.47%
on-line, minimum-link	76.92%	47.84%
on-line, maximum-link	76.56%	47.06%
retrospective, average-link	76.80%	47.47%
retrospective, minimum-link	76.68%	46.82%
retrospective, maximum-link	76.56%	47.06%

Table 7: The accuracy and $F_{0.75}$ scores for detecting frustration on a number of goal-segmentations. The values are measured on a per-query basis. The best results are denoted with an asterisks.

for mission segmentation (77%). For both same-goal and same-mission classification, the difference between the best clustering technique's accuracy and that for the baselines and timeouts are statistically significant at the $p = 0.05$ level.

In general, we find that with accuracy as the optimization measure, all categorization techniques we tried were comparable and typically beat simpler approaches. It seems clear that classifiers are successful, a result consistent with Jones and Klinkner [5].

However, we are interested in detecting frustration.

6.2 Frustration detection evaluation

To evaluate frustration detection, we report both the accuracy of the frustration labels for each issued query in a session as well as $F_{0.75}$. Recall that the F-measure formula is as follows:

$$F_\alpha = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}},$$

where P is the precision and R is the recall. We chose $\alpha = 0.75$ because in the case of detecting frustrated users, it seems more important to avoid mislabeling non-frustrated searchers than frustrated ones. That is, in an on-line setting we do not want to pester satisfied users, and in a retrospective setting, we do not want to suggest someone does failure analysis on successful tasks. This α value gives more weight to precision (mislabeling non-frustrated searchers) than recall (mislabeling frustrated searchers).

Table 7 and Figure 1 show several evaluation metrics of the frustration detection for each goal-segmentation method. In addition, two simple baselines are shown; the first assumes the user is always frustrated and the second that the user is never frustrated. For goal segmentation, timeouts worked best. Timeouts of 15 and 30 minutes outperformed both the simple baselines and the classifier results, both with an accuracy of 85% and an $F_{0.75}$ score of 72%. Overall, a 2-minute timeout has the optimal $F_{0.75}$ score of 73% and an accuracy of 86%. The differences between the accuracy and $F_{0.75}$ score for the 2-minute timeout and that of the clustering methods are all statistically significant at the $p = 0.05$ level. All of the clustering methods performed roughly the same, but rarely very well, in contrast to the results of Section 6.1.

Figure 2 shows the same metrics for the mission-segmentation methods. This graph is less interesting, as the clustering methods perform comparably to the timeout methods. A 6-minute timeout performed best with 91% $F_{0.75}$ score and 95% accuracy. However, if we use a different stopping criteria for on-line clustering using the maximum-link distance function, we can achieve a better $F_{0.75}$ score here. Specifically, raising the stopping criterion from 0.5 to 0.64, we get an $F_{0.75}$ score of 93%. A similar increase occurs with higher stopping criterion for the other clustering methods. The differences in accuracy and the $F_{0.75}$ scores between the timeouts and clustering methods are not statistically different at the $p = 0.05$ level.

7. DISCUSSION OF RESULTS

In this section, we discuss several interesting observations about the results from Section 6. We see that the same-goal classification accuracy for the clustering methods corresponds nicely to the frustration classification accuracy. However, for all the other segmentation techniques, accuracy rose dramatically. So while the on-line clustering method using minimum-link distance performed best at the actual task-segmentation, using a 2-minute timeout substantially outperformed all clustering techniques in frustration detection. This tells us that same-goal classification is not sufficient for frustration detection on the goal level in our sample.

In contrast, same-mission classification transferred well to frustration detection.

Task segmentation itself still important for our task. For example, in our sample of sessions there are an average of 4.61 goals per mission that has at least one instance of frustration. This compares to 2.44 goals per mission with no instances of frustration. While we did not utilize this information for mission segmentation, it seems like a reasonable feature to add.

It should be noted that the classifiers used to produce the distance measures used in task clustering were not optimized. Performing best-subsets regression to find the best set of features for goal and mission segmentation may have produced a more accurate same-task classifier. We also did not try other classifiers, such as decision trees or support vector machines. The focus of this study was to examine the effects of evaluating task segmentation with frustration detection.

8. SUMMARY

We showed several clustering methods that can be used to remove contradictions introduced when classifying pairs of queries as belonging to the same task. We also showed that segmentation techniques that perform well at same-task classification do not necessarily perform well for other task-based applications, particularly frustration detection. Future research should be conducted to evaluate task-segmentation classifiers constructed to optimize frustration detection.

In this paper, we considered a simple heuristic for detecting user frustration. We did not explore how a search system might be modified in order to address certain types of frustration. Future work includes an investigation into the types of user frustration, examining machine learning approaches to detecting varying types of frustration, and exploring the effect of various system interventions on user frustration.

9. ACKNOWLEDGMENTS

We thank Niranjan Balasubramanian and Sam Huston for their feedback regarding this research. This work was supported in part by the Center for Intelligent Information Retrieval and in part by *UpToDate*. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

10. REFERENCES

[1] D. Bilal and J. Kirby. Differences and similarities in information seeking: children and adults as Web users. *Information Processing and Management*, 38(5):649–670, 2002.

[2] I. Ceaparu, J. Lazar, K. Bessiere, J. Robinson, and B. Shneiderman. Determining Causes and Severity of End-User Frustration. *International Journal of Human-Computer Interaction*, 17(3):333–356, 2004.

[3] S.B. Huffman and M. Hochster. How well does result relevance predict session satisfaction? In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 567–574. ACM Press New York, NY, USA, 2007.

[4] B.J. Jansen, A. Spink, V. Kathuria, and S. Koshman. How to Define Searching Sessions on Web Search Engines. *Lecture Notes in Computer Science*, 4811:92–109, 2007.

[5] Rosie Jones and Kristina Lisa Klinkner. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. *Proceedings of CIKM 2008*, 2008.

[6] J. Lazar, J. Feng, and A. Allen. Determining the impact of computer frustration on the mood of blind users browsing the web. In *Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility*, pages 149–156. ACM New York, NY, USA, 2006.

[7] F. Radlinski and T. Joachims. Query chains: learning to rank from implicit feedback. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 239–248. ACM New York, NY, USA, 2005.

[8] A. Spink, M. Park, B.J. Jansen, and J. Pedersen. Multitasking during Web search sessions. *Information Processing and Management*, 42(1):264–275, 2006.

[9] Y. Tonta. Analysis of Search Failures in Document Retrieval Systems: a Review. *The Public-Access Computer Systems Review*, 3(1):4–53, 1992.

[10] P. Wang, W.B. Hawk, and C. Tenopir. Users' interaction with World Wide Web resources: an exploratory study using a holistic approach. *Information Processing and Management*, 36(2):229–251, 2000.