An Information Retrieval Perspective of Filter Selection for Adaptive User Interfaces

Mohamed Reda Bouadjenek
The University of Toronto
Department of Mechanical and Industrial Engineering
Toronto, Ontario M5S 3G8, Canada
mrb@mie.utoronto.ca

Yihao Du
The University of Toronto
Department of Mechanical and Industrial Engineering
Toronto, Ontario M5S 3G8, Canada
duyihao@mie.utoronto.ca

Scott Sanner
The University of Toronto
Department of Mechanical and Industrial Engineering
Toronto, Ontario M5S 3G8, Canada
ssanner@mie.utoronto.ca

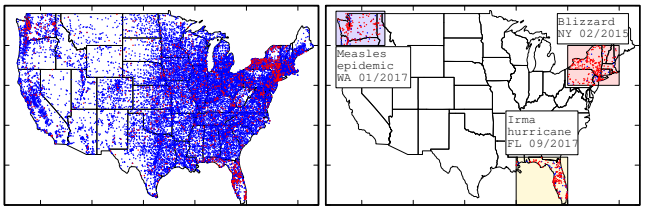
ABSTRACT

Many applications such as real-time monitoring of social network activities or urban traffic congestion reports involve high information and cognitive load tasks. In this context, it is common to have large-scale information visualization interfaces to concurrently display system status, alerts, and events. However, displaying all information elements simultaneously would result in a saturated and unreadable display. Thus, the development of novel adaptive user interfaces (AUIs) are required to help focus the user's attention by filtering information relevant to their current task. In this work, we argue that this information-filtering task is central to a variety of AUIs and well-suited to an Information Retrieval (IR) perspective where we argue that a good overall objective for filter selection is F1-Score; given the absence of known Boolean relevance labels for interface elements, we instead propose optimization of a surrogate metric of expected F1-Score denoted as EF1. We contribute two fast greedy approximate optimization algorithms for EF1 to perform filter selection and we also define an optimal Mixed Integer Linear Programming (MILP) formulation for EF1 intended to benchmark the performance of the greedy approximations on moderately sized datasets. We experiment on three different AUI scenarios and associated datasets under a variety of settings where we vary the amount of data, noise, and relevant vs. irrelevant class imbalance in the ground truth data. We show that EF1 is a good surrogate for optimizing F1-Score and the proposed greedy algorithms are a good approximation of the optimal MILP solution, yet much more scalable. In summary, we hope this work helps better connect IR and AUI research, bringing a general and formal IR perspective to this important research area while opening new research directions for the application of IR methodology to AUIs.

Keywords: Adaptive UIs; Search Algorithms; Optimization for IR.

1 INTRODUCTION

In high information and cognitive load tasks such as real-time monitoring of network security, social media monitoring, or urban traffic congestion, it is common to have large-scale visual interfaces to display current system status, alerts, and events [20, 25, 28].



(a) Global display showing anomalies. (b) Filtered and focused display.

Figure 1: (a) An unfiltered user interface with geolocated tweets appearing more red according to their probability of classification as natural disaster. (b) A filtered version of the interface showing three filtered subsets of data identified to have high coverage of relevant content: a bounding box near New York state limited to data in Feb 2015 and having keyword "Blizzard" and other time and keyword restricted bounding boxes centered on Florida and Washington state. An IR perspective on which objective to evaluate and how to choose these filters and filter settings to optimize surrogates of this objective is the focus of this paper.

However, the massive volume of available information prevents the simultaneous presentation of all information relevant to each alert or event [20]. Thus, adaptive user interfaces (AUIs) are necessary to filter information to provide localized and relevant context for each alert or event. Such AUIs allow the user to efficiently investigate the cause and implications of the alert or event, to identify critical properties of all the elements involved, and to initiate appropriate corrective actions in a timely manner.

In this work, we argue that this information-filtering task is central to a variety of AUIs. Furthermore, we argue that the filtering task shares many common properties of a standard Information Retrieval (IR) setting as summarized in Table 1. Clearly there are important differences between ranking documents in web search and filter selection for AUIs, especially in the way that documents in the AUI can only be selected for a result set through filter settings. However there is also substantial overlap and formal connection with many aspects of IR thus providing new opportunities and challenges of research in this novel IR setting for AUIs.

To make the task of AUI filtering more concrete and to link it clearly to information retrieval, let's introduce an example filtering scenario for AUIs. Consider the case of monitoring events that are discussed on Twitter for the purpose of identifying natural

WOODSTOCK'97, July 1997, El Paso, Texas USA

ACM ISBN 123-4567-24-567/08/06…$15.00
DOI: https://doi.org/10.475/123_4

Table 1: Comparison between conventional Web Search and Filtering for AUIs.

	Web Search	Filtering for AUIs
Information Need	Realized by query	Selection of an event or alert
Form of Results	Ranked list of documents	Filter settings (bounding box in visual display, time ranges, property filters) that select a subset of information elements to display
Relevance Scoring	Comparison of query and document content (and other relevant information) to generate a relevance score, e.g., via TF-IDF or BM25	Third party application-specific tool (e.g., a machine learning approach) that predicts the probability that each information element is relevant to the selected alert
Test Data for Benchmarking	A set of queries; human-labeled judgments of document relevance to each query for a test corpus (i.e., ground truth relevance)	A set of alerts or events; human-labeled judgments of information element relevance to a selected alert (i.e., ground truth relevance)
Evaluation	Ranking metrics (P@k, AP, etc.) over documents given ground truth relevance	Boolean metrics (e.g., F1-score) and Ranking metrics (P@k, AP, etc.) over information elements selected by the filter given ground truth relevance

disasters. Typically, as shown in Figure 1(a), there would be some visual display of all raw information in the network along with relevant Tweets on diverse natural disasters (red nodes in Figure 1(a)). We denote all displayed content (in this case individual tweets) as an information element that could be optionally shown or not. Displaying all information elements simultaneously would result in a saturated and an unreadable display for any reasonably sized problem. To ease the investigation task, users could restrict the information displayed through a variety of filter settings to get a clear overview of individual events as shown in Figure 1(b) – by panning and zooming in the graph display, by restricting upper and lower bounds on a time filter, and/or by selecting properties (e.g., in a drop-down selection or fielded keyword search in fields such as Tweet content).

While the user would typically find it hard to manually set these filters to optimally reveal information about an anomaly they are currently investigating, an AUI that is aware of the alert of interest to the user and aware of the probability that each information element is relevant to the alert could automatically suggest filter settings that might display the relevant information element set with the least amount of irrelevant clutter. Thus, the user could use these filter settings to carry out his investigation by identifying common properties of the elements involved in the alert, find the origin of the deficiency, and eventually, take a set of actions to address the problem. We assume the prediction of information element relevance to an alert to be provided by a third-party since this prediction is highly specific to each particular application setting and not the focus of this work (see Related Work in Section 6 for further discussion).

Since "retrieved" information elements are not individually selected, but rather chosen through a filter setting (that the user can further modify), the problem is clearly an optimization problem of how to restrict filter settings to show the user the most relevant information to the alert or event they have to investigate. While this diverges from the standard information retrieval setting where ranked documents are chosen individually [3], these additional constraints do not change the overall information retrieval objective to select relevant information given the user's information need.

To the best of our knowledge, this work is the first to address the AUI problem as an information retrieval problem of optimizing filter settings for relevance coverage. The contributions of this paper are summarized as follows:

(1) We propose a new theory of IR for adaptive visual user interfaces.
(2) We propose new expected metrics for optimization based on conventional IR metrics, i.e., precision, recall, and F1-Score. We argue that F1-Score and hence expected F1-Score is a good metric for this domain.
(3) We propose new filter setting search algorithms based on greedy strategies and optimal solutions. These algorithms are based on the optimization of new proposed metrics.
(4) We present a thorough evaluation on three different datasets to show that the greedy algorithms we propose are a good approximation of the optimal method, and perform well in the presence of noise (i.e., corruption of the ground truth labels). We experimentally demonstrate that the expected F1-Score metric is a good surrogate of F1-Score.

2 FRAMEWORK AND BACKGROUND

In this section, we first define the problem we address, then the mathematical notation we use, and finally our evaluation metrics.

2.1 Problem definition

The problem we address in this paper is related to the need of an AUI to ease the task of monitoring alerts in large-scale displayed networks. The new interface is expected to filter irrelevant information to provide localized context for each event or alert (defined loosely as a relevant related content localized in time, space, and/or keyword usage). Hence, we define the problem we are addressing in this paper as a problem of filter optimization. We assume that selection of information elements to display in visual interface is

obtained via settings of three filters that can jointly express a global filter. These three sub-filters are the following:

- Time: expresses the time window of a retrieved set.
- Keyword: expresses a set of keyword to include or exclude from a retrieved set.
- Location: expresses a bounding box of a retrieved set. The location can be expressed as longitude/latitude values, or pure coordinates obtained using a given display layout.

The problem we study is how to optimize one or multiple filter selection settings to maximize retrieval of relevant content.

2.2 Mathematical Notation

Throughout this paper, we present all algorithms for Greedy and Optimal search using the following mathematical notation:

- An element j, for which three types of metadata are associated: (i) a textual content, which is composed of a set of terms of size n, (ii) a timestamp t_e, which may represent the creation date of j, and (iii) a position coordinates (x_e, y_e).
- Three variables $I(j)$, $B(j)$ and $S(j)$ are associated to each element j. $I(j)$ refers to whether element j is in the retrieved set (E). $B(j)$ is the boolean variable about relevance of element j. $S(j)$ is a probabilistic score of element j being relevant. Note that $I(j)$ is correlated with the UI system. $B(j)$ and $S(j)$ are independent of the system. Also, $B(j)$ follows a *Bernoulli* distribution with parameter $S(j)$, and hence, the expectation of $B(j)$ is $S(j)$, i.e., $\mathbb{E}_{\mathbb{S}}[B(j)] = S(i)$.
- A global collection of elements GC with size m.
- A retrieved set E with $E \subseteq GC$, for which we use E^* to refer to any subset, i.e., $E^* \subseteq E$. This variable depends on our user-interface filter system. Note that size of E can be represented the sum of $I(j)$ among the global collection. Therefore, we have $|E| = \sum_{j=1}^{m} I(j)$.
- A relevant set RS of $|RS|$ elements, which contains all relevant elements. This is independent of the UI-filter system. Note that size of RS is equal to the sum of $B(j)$ among the global collection. Therefore, we have $|RS| = \sum_{j=1}^{m} B(j)$.
- A keyword query $Q_k = \{\neg t_1^*, \dots \neg t_k^*\}$, which expresses query term exclusion, i.e., find elements that don't contain the terms $t_1^*, \dots t_k^*$.
- A time query $Q_t = [t_{start}, t_{end}]$, which expresses the search for elements in the time window $[t_{start}, t_{end}]$.
- A position query $Q_p = [(x_{min}, y_{min}), (x_{max}, y_{max})]$, which expresses the search of elements falling in the bounding box represented by the lower and upper bound coordinates – respectively (x_{min}, y_{min}) and (x_{max}, y_{max}).
- A query filter Q, which combines the three sub-filters Q_k, Q_t, and Q_p in a conjunction, i.e., $Q = [Q_k \wedge Q_t \wedge Q_p]$.

2.3 Evaluation metrics

We adopt the Boolean relevance framework of IR and thus assume that all information elements j have a ground truth relevance assessment $B(j)$ available at evaluation time. Because filters are equivalent to Boolean retrieval (they either select or do not select content as relevant) and we have a probabilistic estimate of relevance $S(j)$, we propose to evaluate expected variants of standard precision, recall, and F1-score. Specifically because the filter that selects all information elements would maximize expected recall, the filter that selects the highest probability singleton information element would maximize expected precision, we focus on expected F1-score as a trade-off to balance expected precision and recall.

Given a retrieved information element set E, a relevant element set RS selected by a filter setting, the global information element collection GC with size m, the precision of a retrieved set $P(E)$ is defined as follows:

$$P(E) = \frac{\sum_{j \in RS} B(j)}{|E|} = \frac{\sum_{j=1}^{m} B(j)I(j)}{\sum_{j=1}^{m} I(j)} \tag{1}$$

The relaxation of the Boolean label $B(j)$ to probabilistic score $S(j)$ gives the following definition of *expected precision*:

$$EP(E) = \mathbb{E}_{\mathbb{S}}\left[\frac{\sum_{j=1}^{m} B(j)I(j)}{\sum_{j=1}^{m} I(j)}\right] = \frac{\sum_{j=1}^{m} \mathbb{E}_{\mathbb{S}}[B(j)]I(j)}{\sum_{j=1}^{m} I(j)} = \frac{\sum_{j=1}^{m} S(j)I(j)}{\sum_{j=1}^{m} I(j)} \tag{2}$$

Similarly the recall of a retrieved set $R(E)$ is defined as:

$$R(E) = \frac{\sum_{j \in RS} B(j)}{|RS|} = \frac{\sum_{j=1}^{m} B(j)I(j)}{|RS|} \tag{3}$$

Taking a 1st order Taylor expansion, we have the following expectation approximation $\mathbb{E}(X/Y) \approx \mathbb{E}(X)/\mathbb{E}(Y)$ for two dependent random variables X and Y [27]. Hence, we can now define an *approximated expected recall*:

$$ER(E) = \mathbb{E}_{\mathbb{S}}\left[\frac{\sum_{j=1}^{m} B(j)I(j)}{|RS|}\right] \approx \frac{\sum_{j=1}^{m} \mathbb{E}_{\mathbb{S}}[B(j)]I(j)}{\mathbb{E}_{\mathbb{S}}[|RS|]} = \frac{\sum_{j=1}^{m} S(j)I(j)}{\sum_{j=1}^{m} S(j)} \tag{4}$$

Finally, we define the *approximated expected F1-Score* using the *expected precision* and the *approximated expected recall* as follows:

$$EF1(E) \approx \frac{2 \times EP \times ER}{EP + ER} = \frac{2 \times \sum_{j=1}^{m} S(j)I(j)}{\sum_{j=1}^{m} I(j) + \sum_{j=1}^{m} S(j)} \tag{5}$$

In Figure 2, we experimentally show that while EF1 is only an approximation of the true expectation, EF1 serves as an excellent surrogate for F1, which is our main concern when considering filter optimization. That is, maximizing EF1 score with respect to a noisy classifier (noise decreases to 0 as $\lambda \rightarrow 1$) is strongly correlated with maximizing F1 score evaluated on the ground truth. Specifically, while the EF1 and F1 scores are not perfectly calibrated along the diagonal, there is a linear correlation in that as the EF1 score increases for a scenario, the F1-score proportionally increases on average (as shown by the best fit linear regression in the plots).

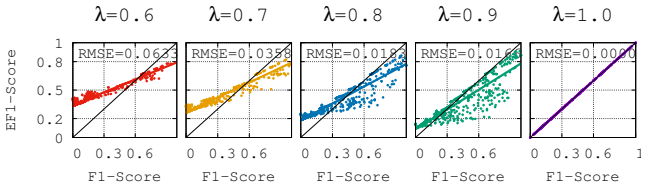


Figure 2: Scatter plot showing EF1-Score vs. F1-Score.

3 FILTER SETTING BUILDING ALGORITHMS

This section first describes the greedy algorithms used to build filters setting for querying large data graphs in order to retrieve relevant UI elements that are of interest to the users. Following this, we describe an optimization-based approach based on Mixed Integer Linear Programming (MILP) to benchmark the performance of the proposed greedy algorithms on moderate-sized problems.

3.1 Greedy search

As is discussed previously, three types of sub-filters are used to select a subset of relevant information in a data visualization interface: keywords sub-filter, time sub-filter, and space (location) sub-filter. A global filter that is generated by *conjoining* these three sub-filters, is used to select the subset *RS* of relevant UI elements. While considering a greedy top-down approach for building these filter settings, we describe in the following how we propose to build each of these sub-filters individually, and then how we propose to combine them in order to select a global filter.

3.1.1 Keywords greedy algorithm. The idea behind the keywords-based greedy top-down algorithm is the following: given a set of elements and an evaluation measure, the algorithm aims to select a set of keywords to form a negation query in order to exclude a subset of elements containing these keywords for the purpose of maximizing the evaluation measure.

Formally, given a keyword query representation Q_k, the algorithm aims to select an optimal subset of k terms $T_k^* \subset E$ (where $|T_k^*| = k$ and E is the initial set of elements) to build a negation keyword query Q_k, i.e. $Q_k = \{\neg t_1^*, \dots \neg t_k^*\}$, in order to optimize a given evaluation measure, e.g. expected precision, expected recall, expected F1-score. This is achieved by building T_k^* in a greedy manner by choosing the next optimal term t_k^* given the previous set of optimal term selections $T_{k-1}^* = \{t_1^*, \dots, t_{k-1}^*\}$ (assuming $T_0^* = \emptyset$) using the following selection criterion:

$$t_k^* = \underset{t_k \notin T_{k-1}^*}{\arg\max}[EF1(E^* \text{ that satisfies } Q_k = \{\neg t_1^*, \dots \neg t_k^*\})] \quad (6)$$

assuming we use expected F1-score to assess E^*, which is a subset of the initial element set E that satisfy the negation keyword query Q_k. In order to reduce the keyword search space, we propose to use the top 100 terms ranked using mutual information. Note that the best indexing strategy to use here is obviously the inverted index data structure [31].

3.1.2 Time greedy algorithm. The idea behind the time-based greedy top-down algorithm is as simple as finding a query time window range $Q_t = [t_{start}, t_{end}]$, which allows to select a subset of elements $E^* \subseteq E$ falling in that time window, with E^* having a higher evaluation metric value.

Formally, given an evaluation measure (say EF1) and a list of elements $E = \{j_{t_1} \leq \dots \leq j_{t_n}\}$, where "$\leq$" specifies the timestamp order, we propose the following two time greedy algorithms:

Naive greedy algorithm: First, at each iteration of this algorithm, an early ranked element j_{t_i} is removed, and then, the remaining set is assessed using expected F1-Score. If the remaining set has a lower expected F1-score value than the set of the previous iteration, the algorithm assigns t_i to the lower time bound of the time query, i.e., $t_{start} = t_i$. Next, the algorithm does the same set of operations, by removing at each iteration a lastly ranked element j_{t_i}, and by stopping once the removal of j_{t_i} causes a decrease in the expected F1-Score value. Then, the algorithm assigns t_i to the upper time bound of the time query, i.e., $t_{end} = t_i$. Lastly, the algorithm returns the time query $Q_t = [t_{start}, t_{end}]$, with obviously $EF1(E^* \text{ that satisfies } Q_t = [t_{start}, t_{end}]) \geq EF1(E)$.

Algorithm 1: Dichotomy Algorithm

input : A set of ordered elements $E = \{j_{v_1} \dots j_{v_n}\}$
output: A timestamp t;

```
v_min = v_1; v_max = v_n; v_mid = (v_1 + v_n)/2;
while v_min != v_mid != v_max do
    if [EF1({j_{v_min} ... j_{v_n}}) ≥ EF1({j_{v_mid} ... j_{v_n}})] then
        v_max = v_mid; v_mid = (v_min + v_mid)/2;
    else
        v_min = v_mid; v_mid = (v_min + v_max)/2;
    end
end
return v_mid;
```

Dichotomy-based greedy algorithm: Having a large dataset with a low number of positive data (e.g., 0.5% of alerts in a security graph) will cause the previous algorithm to take a large number of iterations to terminate since it greedily adjusts filter settings in a minimal way at each step. A way to address this problem is to use binary partitioning search, which we call Dichotomy search. Hence, instead of removing a single element j at each iteration, this algorithm operates by selecting between two distinct alternatives (dichotomies) at each iteration.

As an example of the Dichotomy approach for the time sub-filter, the algorithm first sorts E in increasing order of time stamp. Then, it applies the procedure described by Algorithm 1. This procedure will return the lower time bound of the time query, i.e., $t_{start} = t_{mid}$. Next, the algorithm sorts E in decreasing order of time stamp, and then, it applies again the procedure described by Algorithm 1 to get the upper time bound of the time query, i.e., $t_{end} = t_i$. Lastly, the algorithm returns the time query $Q_t = [t_{start}, t_{end}]$, such that $EF1(E^* \text{ that satisfies } Q_t = [t_{start}, t_{end}]) \geq EF1(E)$. Note that this algorithm proceeds in a total of $log(n)$ iterations in the best case, and $2 \times log(n)$ iterations in the worst case.

For both the naive and time-based greedy algorithms, we use the red-black tree as the indexing data structure [11].

3.1.3 Position greedy algorithm. The aim of this algorithm is to return a query $Q_p = [(x_{min}, y_{min}), (x_{max}, y_{max})]$, which expresses the best set (assessed using one of the expected metrics described) of elements falling in the bounding box represented by the lower and upper bound coordinates – respectively (x_{min}, y_{min}) and (x_{max}, y_{max}). This two dimensional problem is similar to the previous one dimensional problem of finding the best time window. Therefore, the two greedy algorithms described above can be adapted for this problem by first applying each algorithm on the x-axis to determine (x_{min}, x_{max}), then on the y-axis to determine

(y_{min}, y_{max}). We omit the description of these two algorithms for lack of space, but a detailed description can be found in [1].

We use the R-tree as a data structure for indexing multi-dimensional continuous data [13].

3.1.4 Multi-filter algorithm. To get a global query combining all filters, we propose a greedy algorithm, which basically at each iteration selects the best filter to apply, then checks if that filter improves the evaluation metric used. If so, the algorithm makes another iteration using the sub-set obtained in the previous iteration, otherwise, it stops. Note that here, we use $k = 1$ for the keywords greedy algorithm. The algorithm will then determine a sequence of filters to apply on the initial set, and the final query is then built by combining these filters by types.

For example, let's suppose the algorithm determines the following filter sequence: $\{Q_{k_1} = \{\neg natural\} \rightarrow Q_{t_1} = [50, 2030] \rightarrow Q_{p_1} = [(10, 60), (50, 100)] \rightarrow Q_{t_2} = [60, 1230] \rightarrow Q_{k_2} = \{\neg fictive\} \rightarrow Q_{p_2} = [(10, 60), (30, 85)] \rightarrow Q_{t_3} = [60, 800]\}$. The final query is built by combining these filters by types as follows: $Q_k = Q_{k_1} \cup Q_{k_2} = \{\neg natural, \neg fictive\}$, $Q_t = Q_{t_1} \cap Q_{t_2} \cap Q_{t_3} = [60, 800]$, and $Q_p = Q_{p_1} \cap Q_{p_2} = [(10, 60), (30, 85)]$, which gives $Q = [Q_k = \{\neg natural, \neg fictive\} \wedge Q_t = [60, 800] \wedge Q_p = [(10, 60), (30, 85)]]$.

Finally, note that this algorithm can use the keywords greedy algorithm with the time and position naive greedy algorithms to which we refer as Greedy Algorithm, or the keywords greedy algorithm with the time and position dichotomy-based algorithms to which we refer as Dichotomy Algorithm.

3.2 Optimal Solutions for Benchmarking

Next we propose an optimization-based algorithm to maximize EF1 and provide a benchmark for evaluation of the previous Greedy algorithms. Given the trivial solution of optimizing the expected precision (singleton) and expected recall (the whole collection), we consider in the following only the optimization of EF1.

3.2.1 Original intuitive formulation.
Let's first reformulate the objective EF1 to prepare further optimization step. Note that the global sum of scores of all information elements is a constant C given the current system.

$$EF1 = \frac{2 \times \sum_{j=1}^{m} S(j)I(j)}{\sum_{j=1}^{m} I(j) + \sum_{j=1}^{m} S(j)} = \frac{2 \times \sum_{j=1}^{m} S(j)I(j)}{\sum_{j=1}^{m} I(j) + C} \tag{7}$$

In order to obtain the optimal element set, we intend to directly optimize the EF1 metric in terms of decision indication variables $I_{filter} \in [0, 1]$ for filter setting as follows:

$$\begin{aligned} \underset{I_{filter}(j)}{\text{maximize}} \quad & \frac{\sum_{j=1}^{m} S(j)I(j)}{\sum_{j=1}^{m} I(j) + C} \\ s.t \quad & I(j) = \bigwedge I_{filter}(j) \end{aligned} \tag{8}$$

Note that our goal is to optimize the element set in terms of *filters settings*. This means elements in the interface sharing the same property needs to be simultaneously added to the selected set. This is a unique property of our filter-based UI problem, which is different from the independent retrieval of each document in standard IR system.

3.2.2 Transformation to a MILP.
In order to transform the above fractional optimization problem into a Mixed-integer Linear Programming (MILP) form (for which we have optimal solvers), we use the Charnes-Cooper method [4] and Glover linearization method [10] with big-M constraints, where auxiliary variables $w(j)$ and u must be introduced[1]. Here, $w(j)$ is defined as $w(j) = I(j) \times u$ with u defined as follows:

$$u = \frac{1}{\sum_{j=1}^{m} I(j) + C} \tag{9}$$

Then, the EF1 optimization problem is able to be transformed into the following MILP problem:

$$\begin{aligned} \underset{w,u}{\text{maximize}} \quad & \sum_{j=1}^{m} S(j)w(j) \\ s.t \quad & \sum_{j=1}^{m} w(j) + uC = 1 \\ & w(j) \leqslant u, \quad w(j) \leqslant M \times I(j) \\ & w(j) \geqslant u - M \times [1 - I(j)] \\ & u > 0, \quad I(j) \in \{0, 1\}, \quad w(j) \geqslant 0 \end{aligned} \tag{10}$$

3.2.3 Constraints.
As our goal is to select elements through three sub-filters, we add three constraints to the above optimization.

(1) Time: a two-element tuple (t_{start}, t_{end}) indicating respectively the start and the end of the time window.

$$I_{time}(j) = \begin{cases} 1, & \text{if } (t_{start} \leqslant t(j)) \wedge (t(j) \leqslant t_{end}) \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

(2) Position: a four-element tuple $(x_{min}, y_{min}, x_{max}, y_{max})$ to create a bounding box filter in visualization interface.

$$I_{pos}(j) = \begin{cases} 1, & \text{if } (x_{min} \leqslant x(j)) \wedge (x(j) \leqslant x_{max}) \wedge \\ & (y_{min} \leqslant y(j)) \wedge (y(j) \leqslant y_{max}) \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

(3) Keyword: a boolean vector of terms t_k^* with size m - the size of the dictionary of the global collection.

$$I_{term}(j) = \bigwedge_{t_k^* \in j} t_k^* \qquad \text{for s = 1, 2, } \cdots \text{, m} \tag{13}$$

All terms with $I_{term} = 0$ are included in the negation query.

(4) Multiple Filter: if an element j is simultaneously selected via the three sub-filters, all filter constraints have to be satisfied in this element. In others words, an AND operator is required between all the sub-filter constraints.

$$I(j) = I_{time}(j) \wedge I_{pos}(j) \wedge I_{keyword}(j) \tag{14}$$

[1] https://optimization.mccormick.northwestern.edu/index.php/Mixed-integer_linear_fractional_programming_(MILFP)

3.3 Multiple Filter Selection Wrapper

In practice a single filter chosen by the previously described algorithms will narrow the user in on a single "event". However, there will likely be multiple anomalous events and so the user should have a choice of multiple filters. In this work, we provide an initial greedy approach for providing a ranked list of multiple filters (that is a wrapper approach working with any of the previously defined filtering algorithms); we leave it to future work to develop improved filtering and ranking methods for multiple filters.

The algorithm itself is quite simple. After the first filter is produced, all selected elements by the filter have their scores zeroed out. The filtering algorithm is then run again, where it will inherently focus on a different content set. This procedure is repeated until the desired number of filters is reached, or a metric / coverage score for high scoring content is reached. The user should then be able to choose among the multiple filters.

4 EXPERIMENTAL SETUP

In this section we provide details of the three datasets that we evaluate in this paper, as well as the methodology used.

4.1 Dataset description

4.1.1 Email example (Malware detection). The first scenario evaluated in this article is related to the detection and analysis of malicious emails and therefore the detection of compromised user accounts. We assume having multiple emails, a spam classifier, and a display showing the graph of emails. We used ForceAtlas2, a Continuous Graph Layout Algorithm [18] to get the locations of emails in the display (See Figure 3(a)). We use the Enron email dataset, which has been made public by Cohen [19], and for which Shetty and Adibi [23] performed a set of cleansing tasks, mainly by removing a large number of duplicate emails. The final Enron dataset contains 252,759 emails sent from 17,527 users[2].

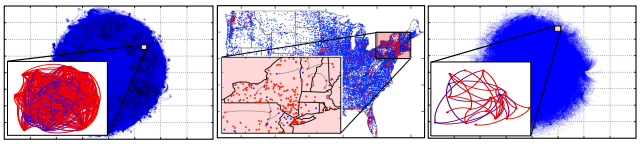
4.1.2 Twitter example (Detection of natural disasters). The second scenario is related to the detection and analysis of topical tweets on natural disasters. We assume having an agent monitoring tweets, a topical tweets classifier (e.g., [16]), and a display showing the locations of the tweets (See Figure 3(b)). We used the 2.5 TB of Twitter data described in [16], for which we restricted our analysis to the 9M tweets of January 2014.

4.1.3 Reddit example (Detection of illegal transactions). The third scenario is related to the detection and monitoring of discussions related to deals of illegal merchandise on SilkRoad via Reddit. Here, we assume again having a monitoring agent, a classifier to classify discussions as being related to illegal transactions or not, and a display showing all discussions (we also use the ForceAtlas2 algorithm as shown in Figure 3(c)). The Reddit dataset we used was crawled between October 2013 and January 2014, and contains 98,777 discussions.

4.2 Evaluation methodology

We assume that for each of the considered scenario, we have a third party application-specific tool (eventually a machine learning approach) that predicts the probability score $S(i)$ that each information element is relevant in the context of that scenario, i.e., an email being malicious, a tweet being related to a natural disaster, or a discussion being related to an illegal transaction. Therefore, this probability score is assumed to be a prior knowledge given, and the way it is computed is out of the scope of this paper.

[2]http://www.ahschulz.de/enron-email-data/



(a) Enron Networks. (b) Twitter Networks. (c) Reddit Networks.

Figure 3: Layouts used for the different datasets.

Rather than use an actual classifier for evaluation, we instead use a noisy corruption of ground truth in order to synthetically evaluate a range of scenarios in terms of classifier noise level. We also explicitly vary the number of relevant information elements in our ground truth to assess performance variation as a function of class imbalance. Hence, to setup a scenario with 20% of relevant elements, we randomly select 20% of the dataset and we assign a label value $l = 1$ for each element in that set, and $l = 0$ for each element out of that set. Then, for each element j, we assign the probability $S(j)$ of that element being relevant for scenario by introducing a random noise signal ratio as follows:

$$S(i) = \lambda * l + (1 - \lambda) * r \,, \tag{15}$$

where r is a random noise value chosen with uniform distribution in the range $[0, 1]$, and λ is a weighting parameter that satisfies $0.5 \leq \lambda \leq 1$, which controls the noise-to-signal ratio in the final probability value. Note that for $\lambda = 1$, $S(j)$ gives the ground truth probability value (perfect classifier), whereas for $\lambda = 0.5$, $S(j)$ returns a complete random probability value (random classifier).

Finally, for each dataset and a given ratio of positive examples, the evaluation was carried out by selecting 10 times independently random elements as being relevant, and then we report the average results using real F1-Score (given the know ground truth).

5 EXPERIMENTAL EVALUATION

We now report and discuss the main results of the experimental evaluation, considering both the accuracy and the effectiveness of the described algorithms. The configuration options that we have evaluated are the following: F1-Score vs. $\#data \in \{10, \ldots, 150, \ldots, 10^3\} \times \lambda \in \{0.6, 0.7, 0.8, 0.9, 1.0\} \times$ Rate of positive date $\in \{0.5\%, 1\%, 2\%, 3\%, 4\%, 5\%, 10\%, 20\%, 30\%, 50\%\}$. All algorithm have been implemented with Java while using the Gurobi Optimizer solver through its java interface for the MILP method [12].

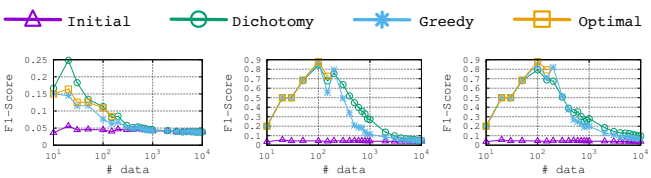
Below, we report only the most significant and interesting results while evaluating the optimization method up to #data=150 as it could not scale to a larger set. In the results shown, "Initial" refers to the performance of the initial global set of elements before applying any filtering. We show it as an uninformed recall-maximizing baseline.

5.1 Performance analysis

We first report the experimental results in terms of F1-Score.

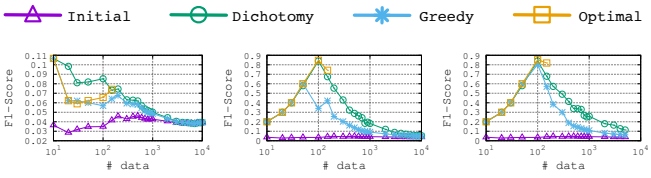
F1-Score vs. #data: The results of this analysis are shown in Figures 4, 5 and 6 for respectively Enron, Twitter, and Reddit while fixing the rate of positive data to 2% and $\lambda \in \{0.6, 0.9, 1.0\}$, and varying the size of the data.

At first glance, all curves of the same algorithm with the same parameters have the same shape. This indicates that the algorithms are consistent across different datasets while varying the size of the data. We explain the apparent peak in all figures as a statistical anomaly as small datasets often don't have enough positive data (for low positive data rate), so this is just a statistical artefact and can be ignored. It does disappear as the rate of positive data increases. Also, we observe that in general it becomes harder to optimize F1 as the size of the data grows as it is harder to separate signal from noise with existing filters. Finally, we also notice that in general, the dichotomy and greedy algorithms are a good approximation of the optimal solution, with the dichotomy method doing better for some cases ($\lambda = 0.6$ and #data$\in [10, 100]$). For example, we observe up to 42% improvement of Greedy w.r.t Optimal for a classifier with an accuracy of 60%.



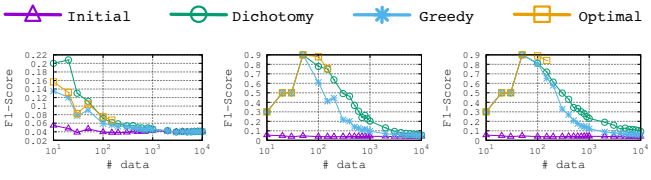
(a) $\lambda = 0.6$. (b) $\lambda = 0.9$. (c) $\lambda = 1$.

Figure 4: Performance on Enron with 2% of positive data.



(a) $\lambda = 0.6$. (b) $\lambda = 0.9$. (c) $\lambda = 1$.

Figure 5: Performance on Twitter with 2% of positive data.

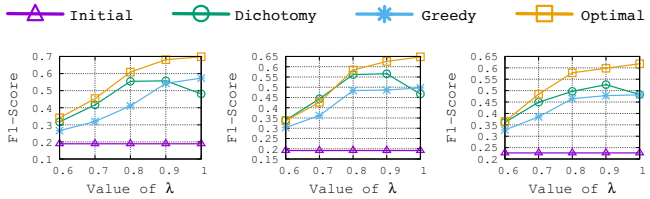


(a) $\lambda = 0.6$. (b) $\lambda = 0.9$. (c) $\lambda = 1$.

Figure 6: Performance on Reddit with 2% of positive data.

F1-Score vs. λ: The results of this analysis are shown in Figure 7 for the three datasets while fixing the rate of positive data to 10% and the size of the data to 150, and varying the amount of noise of the classifier $\lambda \in [0.6, 1.0]$.

Again, all curves of the same algorithm have roughly the same shape across the different datasets, which indicates that the algorithms are consistent. Obviously, we note that the problem becomes easier (higher F1-Score value) as λ increases because EF1 becomes closer to real F1-Score, and so we end up with optimizing the real F1-Score. We also note that the Dichotomy algorithm does better for high noise since it does not overfit to singletons as easily as Greedy; in contrast Greedy does overfit which is better for low noise since it can optimize small details that are actually valid. Finally, we conclude by observing that for low noise, the Dichotomy algorithm is a good approximation of the optimal method.

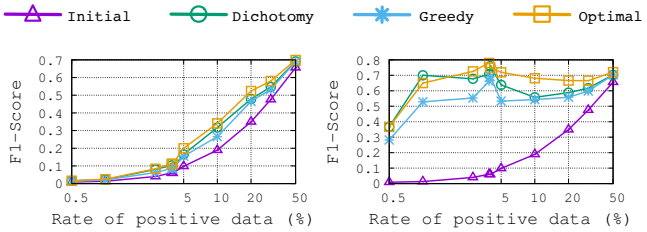


(a) Enron dataset. (b) Twitter dataset. (c) Reddit dataset.

Figure 7: Performance: 10.0% of positive data and #data=150.

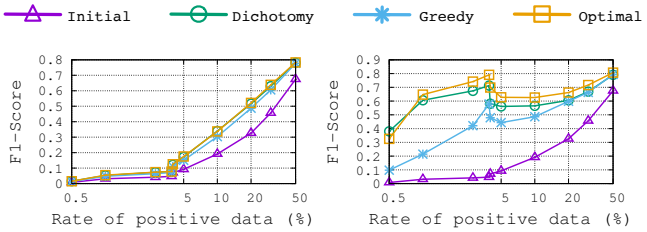
F1-Score vs. rate of positive data: Figures 8, 9, 10 show the results of this analysis while fixing the the size of the data to 150 and $\lambda \in \{0.6, 0.9\}$, and varying the rate of positive data in $[0.5, 50]$.

Briefly, we note again that all curves of the same algorithm with the same parameters have the same shape across the different datasets. We observe that the problem becomes easier as the number of positive examples increases as well as the value of λ increases (i.e., noise decreases). This is explained by the fact that a high rate of positive examples allows the algorithms to be able to select a subset of elements with a large number of positive examples thus maximizing F1-Score. On the other hand, a low value of λ tends to lead the algorithms to select all content (close to Initial unfiltered set) due to the level of noise, while a high value of λ tends to lead the algorithms to select the actual positive examples and thus maximize F1-Score. This is shown through the increasing separation of all curves from the Initial curve as λ increases.



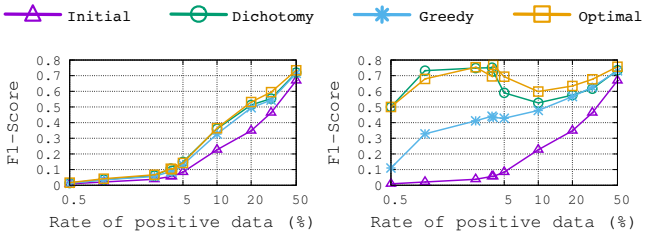
(a) $\lambda = 0.6$. (b) $\lambda = 0.9$.

Figure 8: Performance on Enron dataset with #data=150.



(a) $\lambda = 0.6$. (b) $\lambda = 0.9$.

Figure 9: Performance on Twitter dataset with #data=150.



(a) $\lambda = 0.6$. (b) $\lambda = 0.9$.

Figure 10: Performance on Reddit dataset with #data=150.

F1-Score vs. EF1-Score: This analysis aims to answer the following question: "*Under what conditions is EF1 is a good surrogate for F1?*". The results of this analysis are shown in Figure 11 using a set of four scatter plots on the Enron dataset. We show a comparison of the results obtained when optimizing the real F-Score vs. those obtained when optimizing expected F1-Score with different values for λ. To provide more clarity, the line $y = x$ is included to convey whether the optimization of a particular set gave the same F1-Score using the different optimization strategies or not. A point above the line corresponds to a set for which the optimization with EF1-Score gave a higher F1-Score, while a point below the line refers to a set for which the optimization with EF1-Score hurts the F1-Score. We also include the plot of the function that fits the data points to show how far is the expected F1-Score with respect to the real F1-Score. In addition, we provide a global Root Mean Square Error (RMSE) value to get an overview of the approximation.

Briefly, it is clear that for high values of λ (e.g., 0.9), the optimization with EF1 gives a good approximation of the real F1-Score, with RMSE=0.01 for $\lambda = 0.9$. For a lower $\lambda = 0.8$ and $\lambda = 0.6$ we do observe significantly more noise (as expected), but we also note from the assymetry that EF1 effectively lower bounds the true F1 score indicating the desirable optimization property that a high EF1 does imply a high F1-score.

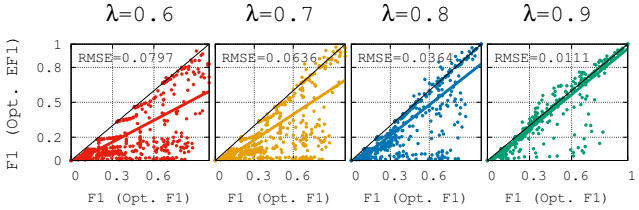
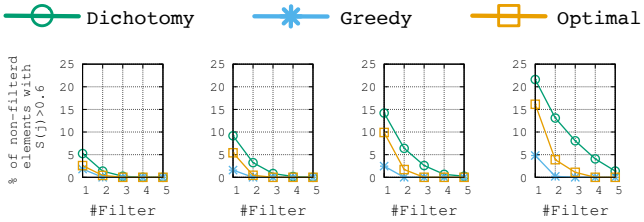


Figure 11: Scatter plot showing EF1-Score vs. F1-Score on the Enron dataset.

Analysis of the multiple filter selection wrapper strategy : Figure 12 shows the number of non-filtered emails above the score threshold of $S(j) = 0.6$ vs. the number of filters produced.

The main outcome of this analysis are: (1) the less accurate the classifier is, the lower the number of the filters produced as the methods tend to return all elements in the first filter, (2) the Dichotomy tends to return the highest number of filters, followed by the Optimal MILP method, than the Greedy method, and (3) all methods tend to quickly return filters (at most 5 filters) that cover the relevant elements with a probability score higher than 0.6.



(a) $\lambda = 0.7$. (b) $\lambda = 0.8$. (c) $\lambda = 0.9$. (d) $\lambda = 1.0$.

Figure 12: Effect of the filter wrapper strategy on the Twitter dataset with 30% of positive data and #data=150.

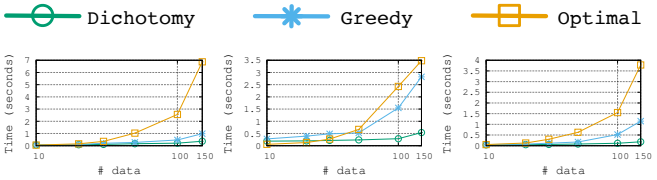
5.2 Computational time complexity analysis

We now report the experimental results in terms of time complexity.

Time vs. #data: Figure 13 reports the results of this analysis on the three datasets while fixing the rate of positive data to 2% and $\lambda = 0.9$, and varying the size of the data.

First, we obviously notice that the problem becomes harder as the size of the data increases. We also observe that Enron has highest time complexity followed by the Reddit dataset, and then the Twitter dataset. This is certainly due to the fact that the size of the text associated with each entity in each dataset is different, ranging from dozens of thousands of characters for emails in Enron

to 144 characters for tweets in Twitter. Hence, more words increase the time complexity. Finally, we also notice that the dichotomy algorithm tends to have a lower time complexity than the greedy algorithm.

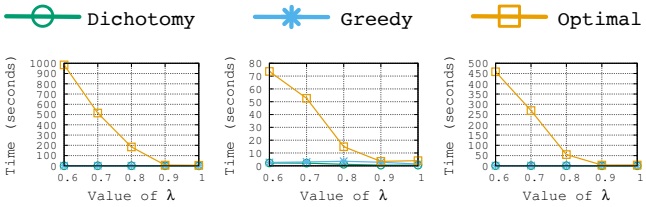


(a) Enron dataset. (b) Twitter dataset. (c) Reddit dataset.

Figure 13: Time complexity: 2% positive data and $\lambda = 0.9$.

Time vs. λ: The results of this analysis are shown in Figure 14 for the three datasets while fixing the rate of positive data to 2% and the size of the data to 150, and varying the amount of noise of the classifier $\lambda \in [0.6, 1.0]$.

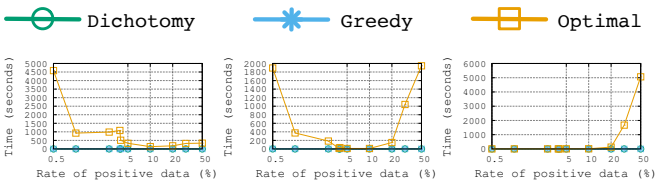
In brief, we first notice that clearly, for this configuration with $\lambda = 0.6$, the optimization algorithm has a very high time complexity (up to 16minutes to find the optimal solution). Hence, the optimal solution is clearly unscalable. Moreover, we observe that this time complexity reduces as the value of λ increases. We explain that by the fact that the algorithms need to make a large number of iterations to find the best solution with high noise (low λ).



(a) Enron dataset. (b) Twitter dataset. (c) Reddit dataset.

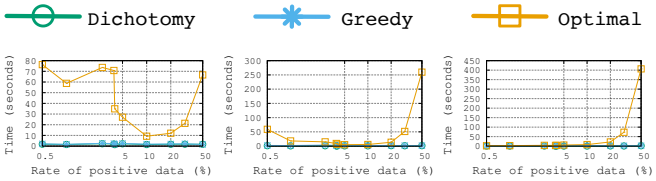
Figure 14: Time complexity: 2% positive data and #data = 150.

Time vs. rate of positive data: Figures 15, 16, 17 show the results of this analysis while fixing the the size of the data to 150 and $\lambda \in \{0.6, 0.8, 1.0\}$, and varying the rate of positive data in $[0.5, 50]$.



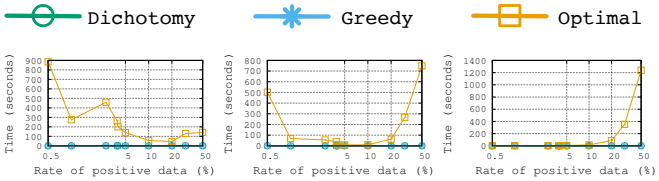
(a) $\lambda = 0.6$. (b) $\lambda = 0.8$. (c) $\lambda = 1$.

Figure 15: Time complexity on Enron with #data=150.



(a) $\lambda = 0.6$. (b) $\lambda = 0.8$. (c) $\lambda = 1$.

Figure 16: Time complexity on Twitter with #data=150.



(a) $\lambda = 0.6$. (b) $\lambda = 0.8$. (c) $\lambda = 1$.

Figure 17: Time complexity on Reddit with #data=150.

Here again we observe that the results are consistent across the different datasets. At first glance, we observe that high noise (low λ) makes a hard problem for the optimization algorithm, but increasing the rate of positive examples causes it to select everything (making the problem easier). For low noise (high λ), only a few elements are selected for a low rate of positive data, but the optimization algorithm has to work hard to select the right 50% for high rate of positive data. For moderate noise see we observe both effects.

5.3 Summary of findings

In conclusion, the evaluations have shown that the Dichotomy and Greedy algorithms are a good approximation of the Optimal MILP formulation and may outperform the MILP in highly noise settings (especially the Dichotomy approach which tends to overfit filters less to noise). We have also shown that the Dichotomy performs well in terms of objective optimization and is most efficient for scaling to large datasets. Finally, we have demonstrated that the expected F1-Score metric is a good surrogate of the F1-Score metric.

6 RELATED WORK

There is a substantial body of research related to UI systems [6, 17, 30]. Below, we review the major works related to aspects of this paper including information element scoring, filter selection and optimization, and evaluation of AUIs.

Element scoring: Different systems for managing information in visual interfaces rely on some method to score information for viewing. In this work we assumed the scoring system was given and focused our contributions on novel filter optimization and evaluation criteria and supporting algorithms. However, other works have focused primarily on the scoring method and could be integrated as the scoring component in our work. For example, other approaches rely on machine learning methods for scoring, that range from decision trees [5], k-NN [2], rule learning [21], deep learning [24],

etc. Some approaches go one step further to leverage user interactions as a means of relevance feedback for refining the scoring system by interacting with the interface [22, 24] or by defining personalized re-ranking rules [7]. Finally, some recent advanced but very preliminary work has considered sequential optimization of interface adaptations based on user interactions [24]. While all of the above provided contributions orthogonal to this paper, they offer interesting potential for a tighter integration of the learning and user interaction loop in future extensions of this work.

Filter selection: The filter selection algorithms we presented in this paper are based on the optimization of IR evaluation metrics. Given the trivial solution of optimizing the precision (singleton) and the recall (the whole collection), we considered in this work only the optimization of F1-Score (trade-off between precision and recall). However, in the context of conventional IR models, other works have focused on the optimization of other metrics using different strategies. For examples, Wang and Zhu [29] proposed to use score approximation using expectation to optimize Average Precision, Discounted Cumulative Gain, and Reciprocal Rank. However, authors didn't propose a way to optimize recall and F1-Score, which is critical for our filtering problem. Also, machine learning has been explored to optimize different metrics such as NDCG or MAP through Learning to Rank (L2R) [3]. However, L2R cannot be directly applied in our filter optimization problem, as the task we address is to find filters that fit optimal sets — although this may be part of our future work. As for using MILP to optimize IR metrics, to the best of our knowledge, there is only one paper describing such a method proposed by Hansen et al. [14]. The drawback in this work is the need for real relevance labels of all documents.

Evaluation: Usually, AUIs are evaluated based on the system effectiveness of retrieving relevant elements, e.g., alarms [2], images [5, 7], textual items [26], news documents [22], or based on the usability of the interface [8, 9, 15]. Also, AUIs are usually evaluated through an off-line evaluation using specific benchmarks [2, 5, 24] or through a user study [2, 7–9, 15, 26]. Instead, the purpose of this paper was to introduce a general information retrieval methodology to the design and evaluation of filter selection algorithms for adaptive user interfaces; hence, our paper focuses mainly on the evaluation of filter objectives with respect to a ground truth Boolean relevance criterion. Further evaluating our general approach through user studies in specific scenarios as done by the previously cited papers can help refine the novel information retrieval methodology introduced here for specific application use cases.

7 CONCLUSIONS AND FUTURE WORK

Overall, we aim for this work to bring a formal IR perspective to the important research area of AUIs while opening new research directions for the application of IR methodology. As information retrieval has transformed our web experience, we believe there is a similar opportunity to transform the present nascent state of AUIs through optimization and information retrieval principles to make them more ubiquitous in our daily lives.

In this paper we have introduced a novel information retrieval perspective of filtering for Adaptive User Interfaces. In this context, we introduced new filter search algorithms for expected F1-Score based on two different Greedy strategies. We have also proposed a method for optimization of expected F1-score using a MILP approach for purposes of benchmarking. The evaluations we have performed on a variety of datasets have shown that the Greedy algorithms we have proposed are relatively efficient and provide a good approximation of the optimal MILP solution; Greedy methods may even do better than exact expected F1-score optimization in high noise cases due to their more restricted search space. Furthermore, we have experimentally demonstrated that the proposed expected F1-Score metric is a good surrogate of the ground truth F1-Score.

Important areas of future work include consideration of the role of (pseudo-)relevance and other explicit or implicit feedback methods to create a tighter and more responsive user interaction loop. Furthermore, in combination with user studies and consideration of human factors, future work should also consider novel application-specific objectives, e.g., in specific visualization frameworks or based on a ranking theory of results presentation (e.g., using size or color for visual ranking emphasis).

REFERENCES

[1] Removed for double-blind review.

[2] S. Amershi, B. Lee, A. Kapoor, R. Mahajan, and B. Christian. Human-guided machine learning for fast and accurate network alarm triage. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, pages 2564–2569. AAAI Press, 2011.

[3] R. A. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., 2 edition, 2010.

[4] A. Charnes and W. W. Cooper. Programming with linear fractional functionals. *Naval Research Logistics Quarterly*, 9(3-4):181–186, 1962.

[5] J. Cui, F. Wen, and X. Tang. Intentsearch: Interactive on-line image search re-ranking. In *Proceedings of the 16th ACM International Conference on Multimedia*, MM '08, pages 997–998, New York, NY, USA, 2008. ACM.

[6] S. Dumais, E. Cutrell, J. J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins. Stuff i've seen: A system for personal information retrieval and re-use. *SIGIR Forum*, 49(2):28–35, Jan. 2016.

[7] J. Fogarty, D. Tan, A. Kapoor, and S. Winder. Cueflik: Interactive concept learning in image search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 29–38, New York, NY, USA, 2008. ACM.

[8] K. Z. Gajos, K. Everitt, D. S. Tan, M. Czerwinski, and D. S. Weld. Predictability and accuracy in adaptive user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 1271–1274, New York, NY, USA, 2008. ACM.

[9] K. Z. Gajos, D. S. Weld, and J. O. Wobbrock. Automatically generating personalized user interfaces with supple. *Artificial Intelligence*, 174(12):910 – 950, 2010.

[10] F. Glover. Improved linear integer programming formulations of nonlinear integer problems. *Management Science*, 22(4):455–460, 1975.

[11] L. J. Guibas and R. Sedgewick. A dichromatic framework for balanced trees. In *19th Annual Symposium on Foundations of Computer Science (sfcs 1978)*, pages 8–21, Oct 1978.

[12] I. Gurobi Optimization. Gurobi optimizer reference manual; 2017. *http://www.gurobi.com/documentation/7.5/refman/refman.html*, 2017.

[13] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, SIGMOD '84, pages 47–57, New York, NY, USA, 1984. ACM.

[14] P. Hansen, M. V. Poggi de Aragão, and C. C. Ribeiro. Hyperbolic 0–1 programming and query optimization in information retrieval. *Mathematical Programming*, 52(1):255–263, May 1991.

[15] M. Hartmann and D. Schreiber. Proactively adapting interfaces to individual users for mobile devices. In W. Nejdl, J. Kay, P. Pu, and E. Herder, editors, *Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 300–303, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[16] Z. Iman, S. Sanner, M. R. Bouadjenek, and L. Xie. A longitudinal study of topic classification on twitter. In *ICWSM*, pages 552–555, 2017.

[17] C. Inibhunu, S. Langevin, S. Ralph, N. Kronefeld, H. Soh, G. A. Jamieson, S. Sanner, S. W. Kortschot, C. Carrasco, and M. White. Adapting level of detail in user interfaces for cybersecurity operations. In *2016 Resilience Week (RWS)*.

[18] M. Jacomy, T. Venturini, S. Heymann, and M. Bastian. Forceatlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software. *PLOS ONE*, 9(6):1–12, 06 2014.

[19] B. Klimt and Y. Yang. Introducing the enron corpus. In *CEAS*, 2004.
[20] S. Liu, W. Cui, Y. Wu, and M. Liu. A survey on information visualization: recent advances and challenges. *The Visual Computer*, 30(12):1373–1393, Dec 2014.
[21] N. Mezhoudi. User interface adaptation based on user feedback and machine learning. In *Proceedings of the Companion Publication of the 2013 International Conference on Intelligent User Interfaces Companion*, IUI '13 Companion, pages 25–28, New York, NY, USA, 2013. ACM.
[22] E. Schrier, M. Dontcheva, C. Jacobs, G. Wade, and D. Salesin. Adaptive layout for dynamically aggregated documents. In *Proceedings of the 13th International Conference on Intelligent User Interfaces*, IUI '08, pages 99–108, New York, NY, USA, 2008. ACM.
[23] J. Shetty and J. Adibi. The enron email dataset database schema and brief statistical report. Technical report, 2004.
[24] H. Soh, S. Sanner, M. White, and G. Jamieson. Deep sequential recommendation for personalized adaptive user interfaces. In *the 22nd ACM International Conference on Intelligent User Interfaces (IUI-17)*, 2017.
[25] G.-D. Sun, Y.-C. Wu, R.-H. Liang, and S.-X. Liu. A survey of visual analytics techniques and applications: State-of-the-art research and future challenges. *Journal of Computer Science and Technology*, 28(5):852–867, Sep 2013.
[26] T. Tsandilas and m. c. schraefel. An empirical assessment of adaptation techniques. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '05, pages 2009–2012, New York, NY, USA, 2005. ACM.
[27] G. van Kempen and L. van Vliet. Mean and variance of ratio estimators used in fluorescence ratio imaging. *Cytometry*, 39(4):300–305, 2000.
[28] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. van Wijk, J.-D. Fekete, and D. Fellner. Visual analysis of large graphs: State-of-the-art and future research challenges. *Computer Graphics Forum*, 30(6):1719–1749, 2011.
[29] J. Wang and J. Zhu. On statistical analysis and optimization of information retrieval effectiveness metrics. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 226–233, 2010.
[30] S. Yamada and F. Murase. Adaptive user interface of a web search engine by organizing page information agents. *Transactions of the Japanese Society for Artificial Intelligence*, 16(1):46–54, 2001.
[31] J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Comput. Surv.*, 38(2), July 2006.