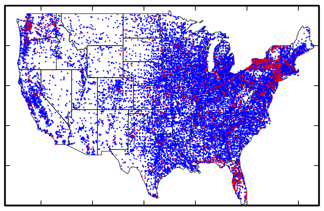
Relevance-driven F1-Score Optimization for Filtering in Visual Information Displays

Anonymous Author(s)

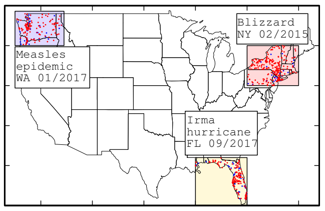
ABSTRACT

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 automated filtering for Visual Information Displays (VIDs) are required to help focus the user's attention by restricting the display of information relevant to their current task. In this paper, we make key contributions to the literature on information filtering for VIDs: (1) Given that a VID will typically support many types of filtering (spatial, temporal, and content), we provide a unified framework that abstracts presentation-specific details and facilitates an optimization perspective of filtering. (2) Within this optimization framework, we argue for the formulation of filtering as maximization of an expected F1-Score (EF1) objective subject to parameterized filter constraints. And (3) specifically for optimizing filters w.r.t. EF1, we contribute two efficient greedy algorithms as well as an *optimal* Mixed Integer Linear Programming (MILP) formulation intended to benchmark the performance of the greedy approximations on moderately sized datasets. We experiment with three different VID 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, this work provides the first formal and unified perspective of information filtering for VIDs from an F1-Score optimization perspective along with scalable and robust algorithms that demonstrate strong performance when benchmarked against optimal results.

Keywords: Visual Search Interfaces; Information Filtering; Constrained Optimization.



(a) Global display showing anomalies.



(b) Filtered and focused display.

Figure 1: (a) An unfiltered user interface with geolocated documents (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. This task is *not* simply clustering since this paper assumes that a relevance measure has been provided for each document, which is used to directly optimize the F1-Score of content presented in each displayed filter. As the data remains unchanged but the relevance measure changes, the displayed filters would change accordingly.

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 [22, 30, 34]. However, the massive volume of content to be presented on these Visual Information Displays (VIDs) prevents the simultaneous presentation of all information relevant to each alert or event [22]. Thus, it is critical to develop automated filters for VIDs that are capable of limiting displayed content to the most relevant context for each alert or event to initiate an appropriate course of action in a time-effective manner.

Going back to 1992, Belkin and Croft [4] realized that information filtering was simply a counterpart to information retrieval, albeit with a few key differences. Namely, Belkin and Croft pointed out that information filtering often occurs in the context of a long-term standing interest (represented implicitly through a relevance measure), as well as continuing interaction with the filtering system over a long period of time. While this accurately summarizes the filtering perspective we take in this paper, we remark that most work on information filtering displays has so far focused on *unsupervised* approaches such as (hierarchical) clustering [1, 5, 22, 24, 27, 28, 31, 37]. However, returning to Belkin and Croft's initial motivation and information retrieval analogy to information filtering, we assume a task-specific relevance measure and "supervised" relevance objective in order to generically view information filtering in VIDs as an expected F1-Score optimization of a result set given filter constraints.

To make the task of VID filtering more concrete, we introduce an example scenario. Consider the case of monitoring events that are discussed on Twitter for the purpose of identifying natural disasters.

WOODSTOCK'97, July 1997, El Paso, Texas USA
© 2016 Copyright held by the owner/author(s).
ACM ISBN 123-4567-24-567/08/06…$15.00
DOI: https://doi.org/10.475/123_4

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 points 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).

While the user would typically find it hard to manually set these filters to optimally reveal information about an anomaly (event) they are currently investigating, a VID that is 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 their investigation by identifying common properties of the elements involved in the alert, and eventually take a set of appropriate actions if warranted. It is critical to note in this work that we assume the prediction of content relevance to a user's information need 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 notably diverges from the standard information retrieval setting where ranked documents are chosen individually [3], these additional constraints do not change the overall objective to select relevant information given the user's information need as represented by the given relevance measure.

To the best of our knowledge, this work is the first to address information filtering in VIDs as a "supervised" optimization problem. In brief, we make the following contributions:

(1) We provide a unified framework that abstracts presentation-specific details of different information filters (spatial, temporal, and content) for VIDs and facilitates an optimization perspective of filtering w.r.t. a provided relevance measure.
(2) We propose new *expected* metrics for optimization based on conventional information retrieval metrics, i.e., precision, recall, and F1-Score. We argue that expected F1-Score (EF1) is the appropriate metric to optimize subject to parameterized filter constraints.
(3) For scalably and efficiently optimizing filters w.r.t. EF1, we contribute two greedy algorithms.
(4) To benchmark the performance of the greedy algorithms for EF1 optimization on moderately sized datasets, we provide an *optimal* Mixed Integer Linear Programming (MILP) formulation for EF1 derived from an initial fractional MILP formulation.
(5) Finally, we present a thorough evaluation on three different datasets to show that the greedy algorithms we propose are a good approximation of the optimal solution, and perform well in the presence of noise (i.e., corruption of the ground truth labels). We also 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 expected F1-Score.

2.1 Problem definition

In this paper, we work with the following three types of filters for visual information displays (VIDs):

- **Time Filter:** limits the display of time-stamped content according to two filter parameters for the lower and upper time bound.
- **Space Filter:** limits the display of 2D spatially annoted content according to four filter parameters for the upper left and lower right bounding box coordinates.
- **Keyword (or Discrete Attribute) Filter:** limits the display of content according to included or excluded keywords (or general discrete attributes of an information element).

Given an externally provided probabilistic measure of information relevance, the problem we study is how to efficiently optimize each of these individual filters given an appropriate objective, and if more than one filter is used, how to optimize a **Global Filter** that simultaneously optimizes over two or more of these above filters.

In general, we remark that the choice of which filters to use is up to the VID designer according to the display and (meta)data available; our proposed framework works for all filters or any subset of them. We further remark that this work is not limited to these three filters, but we believe time, space, and content constitute three of the most commonly used information filters in practice and hence are the ones we focus on in this work.

2.2 Mathematical Notation

With the problem defined above, we proceed to the formal portion of our presentation using the following mathematical notation:

- An information element j may have three types of associated metadata: (i) 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 with each information element j: $I(j)$ refers to whether an element j is displayed; $B(j)$ is a Boolean random variable indicating the relevance of an element j; $S(j)$ is a probabilistic score indicating the relevance of an element j. $B(j)$ follows a *Bernoulli* distribution with parameter $S(j)$, and hence, the expectation of $B(j)$ is $S(j)$, i.e., $\mathbb{E}_{\mathbb{S}}[B(j)] = S(i)$.
- We label GC as the global set of all information elements j with total size $|GC| = m$.
- The set of information elements displayed by a filter is labeled E with $E \subseteq GC$; we use E^* to refer to further subsets, i.e., $E^* \subseteq E$. Note that size of E can be represented as the

sum of $I(j)$ among the global collection GC. Therefore, we have $|E| = \sum_{j=1}^{m} I(j)$.

- We label the set of ground truth relevant information elements as the relevant set RS consisting of $|RS|$ elements. Note that $|RS|$ is equal to the sum of $B(j)$ among the global collection GC. Therefore, we have $|RS| = \sum_{j=1}^{m} B(j)$.
- Keyword parameters $Q_k = \{\neg t_1^*, \dots \neg t_k^*\}$, are composed of a set of query terms excluded from the keyword filter, i.e., elements displayed cannot contain the terms $t_1^*, \dots t_k^*$.
- Time parameters $Q_t = [t_{start}, t_{end}]$ express the lower bound t_{start} and upper bound t_{end} parameters of the time filter.
- Position parameters $Q_p = [(x_{min}, y_{min}), (x_{max}, y_{max})]$ express the upper left (x_{min}, y_{min}) and lower right (x_{max}, y_{max}) corners of the spatial filter, assuming $(0, 0)$ is in the upper left corner of the display window.
- A global query filter Q combines the three sub-filters Q_k, Q_t, and Q_p in a conjoined set of parameters $Q = [Q_k, Q_t, Q_p]$.

2.3 Deriving Expected F1-Score (EF1)

We adopt the Boolean relevance framework standard in information retrieval [3] and thus assume that any information element j has 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. However, as standard for both precision and recall, we note that they can be trivially optimized through pathological solutions. That is, the filter that selects all information elements (i.e., all time, all space, no excluded keywords) would trivially maximize (expected) recall. Similarly, the filter that selects the highest probability singleton information element would maximize expected precision. This leaves expected F1-score as a non-pathological objective to balance expected precision and recall.

To formally define expected F1-Score, we first begin with definitions of expected precision and recall. Recalling our previous definitions, given a displayed set of information elements E and a relevant set RS selected by a filter, the precision of E is defined as follows:

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

Given that $B(j)$ is a Boolean random variable, we can now take the expectation of $P(E)$ leading to 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 [33]. Hence, we can now define an *approximated expected recall*:

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

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

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

Due to space limitations, we focus on F1-score in this paper, however expected F_β scores follow directly from the above definitions.

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 $\lambda \to 1$) is strongly correlated with maximizing F1 score evaluated on the ground truth; we will further study the effect of a noisy classifier in Section 5. 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).

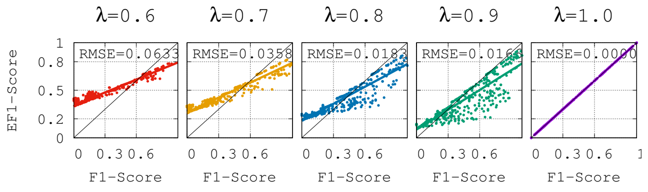


Figure 2: Scatter plot showing EF1-Score vs. F1-Score.

3 ALGORITHMS FOR FILTER SELECTION

Now that we have defined the EF1 objective for filter optimization, we need to find efficient ways to (approximately) optimize it. To this end, this section first describes two scalable and efficient greedy algorithms to optimize EF1. 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 discussed previously, we assume that three types of "sub-filters" are used to select a subset of relevant information in a VID: Keyword Filter, Time Filter, and Space Filter. A Global Filter is generated by *conjoining* these three sub-filters. In the following, we describe how to greedily build each of these sub-filters, and then how to combine them into a Global Filter.

3.1.1 Greedy Keyword Filter algorithm. Given a set of information elements, the greedy Keyword Filter 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 EF1-Score.

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 the EF1-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:

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

where E^* 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 to identify the keywords that are predictive of the "supervised" relevance measure. We remark that other metrics like frequency would be more appropriate for unsupervised tasks. We further remark that we have chosen to use a negation query as a means to effectively prune (or filter) the content as more terms are selected. The best indexing strategy to support this greedy search is the inverted index data structure [38].

3.1.2 *Greedy Time Filter algorithm.*

The idea behind the time-based greedy filter algorithm is as simple as finding a 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 the highest EF1-Score.

Formally, a list of elements $E = \{j_{t_1} \leq \dots \leq j_{t_n}\}$, where "≤" 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 EF1-Score. If the remaining set has a lower EF1-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 EF1-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 $EF1(E^* \text{ that satisfies } Q_t = [t_{start}, t_{end}]) \geq EF1(E)$.

Binary Partition Search algorithm: Large datasets with sparse 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 (BPS). Hence, instead of removing a single element j at each iteration, this algorithm operates by selecting between two distinct alternatives (binary partitions) at each iteration.

As an example of the BPS 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

Algorithm 1: Binary Partition Search (BPS) Algorithm

```
input  : A set of ordered elements $E = \{j_{v_1} \dots j_{v_n}\}$
output : A timestamp $t$;
1 $v_{min} = v_1; v_{max} = v_n; v_{mid} = \frac{v_1+v_n}{2}$;
2 while $v_{min}! = v_{mid}! = v_{max}$ do
3     if $[EF1(\{j_{v_{min}} \dots j_{v_n}\}) \geq EF1(\{j_{v_{mid}} \dots j_{v_n}\})]$ then
4         $v_{max} = v_{mid}; v_{mid} = \frac{v_{min}+v_{mid}}{2}$;
5     else
6         $v_{min} = v_{mid}; v_{mid} = \frac{v_{min}+v_{max}}{2}$;
7     end
8 end
9 return $v_{mid}$;
```

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 [13].

3.1.3 *Greedy Spatial Filter algorithm.*

The aim of this algorithm is to return coordinates $Q_p = [(x_{min}, y_{min}, (x_{max}, y_{max})]$ representing the EF1-Score maximizing bounding box represented by the lower and upper bound coordinates – respectively (x_{min}, y_{min}) and (x_{max}, y_{max}). This 2D 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 use the R-tree as a data structure for indexing multi-dimensional continuous data [15].

3.1.4 *Global Filter algorithm.*

To obtain a Global Filter combining all sub-filters, we propose a greedy algorithm, which at each iteration applies all sub-filters and chooses the one that most reduces EF1. The selected filter is updated with its new setting and the iteration continues. Iterations termination when no sub-filter can unilaterally improve EF1 and the final filter settings are returned as the Global Filter.

Finally, we note that the Global Filter algorithm can use the Greedy Keyword Filter with the naive Greedy Time and Spatial Filters, which we refer to experimentally as **Greedy**, or the Binary Partition Search variants, which we refer to experimentally as **BPS**.

3.2 Optimal MILP Solutions for Benchmarking

Next, we propose an exact Mixed Integer Linear Progamming (MILP) optimization-based formulation to maximize EF1 and provide a benchmark for evaluating the two previous Global Filter algorithms (Greedy and BPS).

3.2.1 *Fractional MILP Formulation.*

We begin by reformulating the EF1 objective to prepare for further optimization steps by replacing the global sum of scores of all information elements with a constant $C = \sum_{j=1}^{m} S(j)$:

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

In order to obtain the EF1-optimal Global Filter, we let binary variables $I_{filter}(j) \in \{0, 1\}$ indicate whether an information element j is selected in each sub-filter and constrain that to be selected in the Global Filter (i.e., $I(j) = 1$), j must be selected in all filters (i.e., a conjunction). This leads to the following fractional MILP formulation with sub-filter constraints to be defined later:

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

3.2.2 Transformation to a MILP.

While there are no direct solvers for fractional MILPs, we can transform (8) into a pure MILP form for which we have efficient and optimal solvers. To do this, we use the Charnes-Cooper method [6] and Glover linearization method [12] with big-M constraints, where auxiliary variables $w(j)$ and u are 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 Filter Constraint:** 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) **Spatial Filter Constraint:** 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 Filter Constraint:** 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) **Global Filter Constraint:** for information element j to be selected globally, it must be simultaneously selected by the three sub-filters.

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

We refer to the above MILP formulation in (10) with all filter constraints (11)–(14) as the **Optimal** Global Filter.

3.3 Multiple Filter Selection Wrapper

In practice a single Global 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. Consider Figure 1: this actually shows three different spatial bounding boxes corresponding to three different events provided by Global 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 – Greedy, BPS, or Optimal); 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 $S(j)$ 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 in the VID.

4 EXPERIMENTAL SETUP

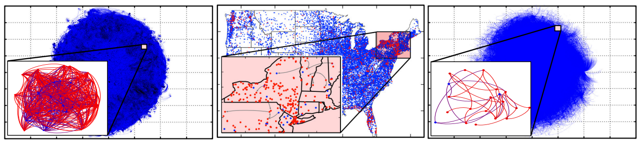
4.1 Dataset description

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 [20] 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 [21], and for which Shetty and Adibi [26] 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., [18]), and a display showing the locations of the tweets (See Figure 3(b)). We used the 2.5 TB of Twitter data described in [18], for which we restricted our analysis to the 9M tweets of January 2014.

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

[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.

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 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. This probability scoring function is task-specific and assumed to be given.

For relevance prediction in our experiments, we use a noisy corruption of ground truth in order to synthetically evaluate a range of scenarios in terms of predicted relevance 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 signal-to-noise 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).

For each dataset and given proportion of positive examples, the evaluation was carried out by averaging over 10 independent runs that each select random relevant documents according to the designated proportion. We report the average ground truth F1-Score (i.e., ground truth is known in the experimental setting).

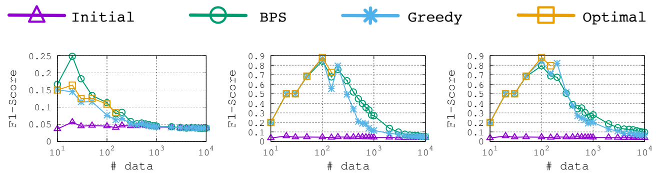
5 EXPERIMENTAL EVALUATION

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 [14].

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 below, **Initial** refers to the performance of the initial global set of elements before applying any filtering. We show it as a recall-maximizing baseline.

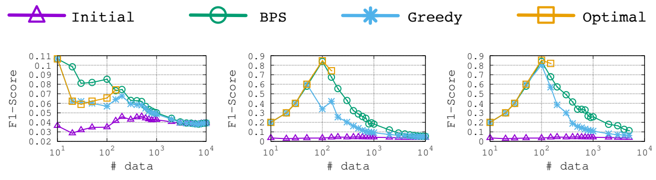
5.1 Performance analysis

Varying #data: Here we aim to understand how the different F1-Score optimization algorithms perform as the amount of data varies. Analysis is provided in Figures 4, 5 and 6 respectively for 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.



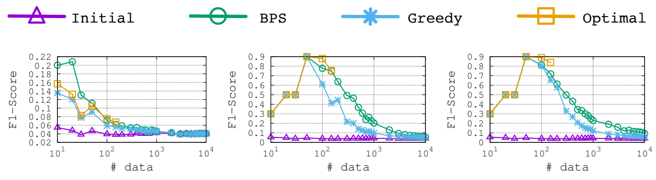
(a) $\lambda = 0.6$. (b) $\lambda = 0.9$. (c) $\lambda = 1$.

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.



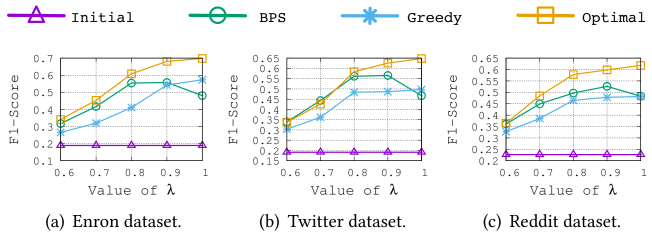
(a) $\lambda = 0.6$. (b) $\lambda = 0.9$. (c) $\lambda = 1$.

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

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 observe that beyond a certain #data threshold, it generally becomes harder to optimize F1 as the size of the data grows as it is harder to separate signal from noise. We also notice that in general, the BPS and greedy algorithms are a good approximation of the optimal solution, with the BPS method doing better for some cases ($\lambda = 0.6$ and #data$\in [10, 100]$) since the coarse splits of BPS serve as a guard against "overfitting" to noise seen in Optimal.

Varying relevance noise (λ): Here we aim to understand how the different F1-Score optimization algorithms perform as the amount of noise λ (defined previously) in the relevance prediction varies. 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 $\lambda \in [0.6, 1.0]$.

Again, all curves of the same algorithm have roughly the same shape across the different datasets, which indicates that the algorithms are consistent. The problem becomes easier (higher F1-Score value) as λ increases because EF1 becomes closer to the ground truth F1-Score. We also note that the BPS 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. We conclude by observing that for low noise, the BPS algorithm is a good approximation of the optimal method.

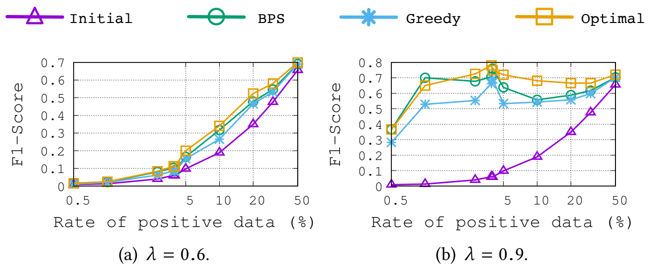


(a) Enron dataset. (b) Twitter dataset. (c) Reddit dataset.

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

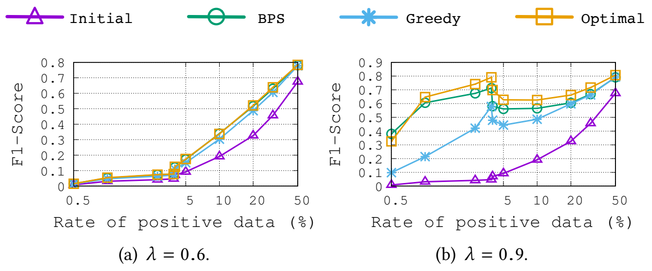
Varying rate of positive data: Here we aim to understand how the different F1-Score optimization algorithms perform as we vary the proportion of positively labeled ground truth data. Figures 8, 9, 10 show the results of this analysis while fixing the the size of the data to 150 and $\lambda \in \{0.6, 0.9\}$, 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.



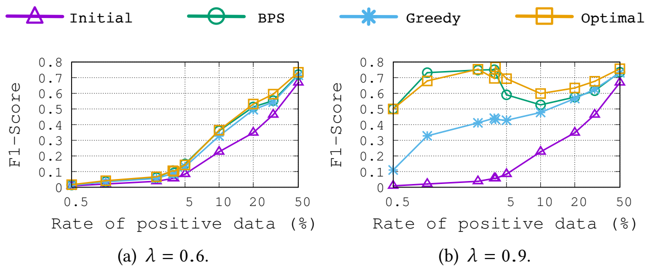
(a) $\lambda = 0.6$. (b) $\lambda = 0.9$.

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 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.

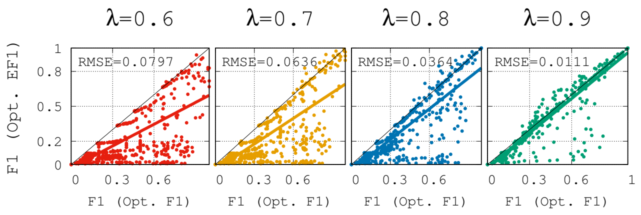
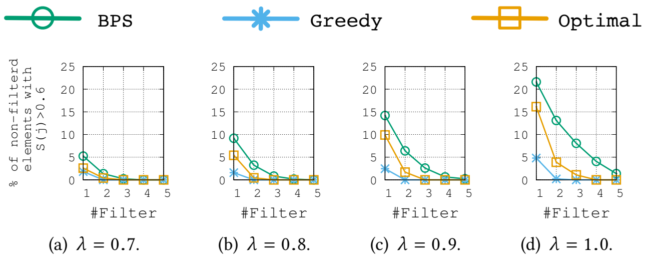


Figure 11: Scatter plot showing EF1-Score vs. F1-Score on the Enron dataset.

Analysis of the multiple filter selection wrapper strategy: Figure 12 shows how many filters it takes (x-axis) to reduce the number of non-filtered emails above the score threshold of $S(j) = 0.6$ to zero (y-axis). A faster reduction to zero indicates better coverage with fewer filters. We observe the following: (1) for higher relevance noise, fewer filters are needed since all methods tend to return most elements in the first filter, (2) BPS tends to return the highest number of filters, followed by the Optimal MILP method, then 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.



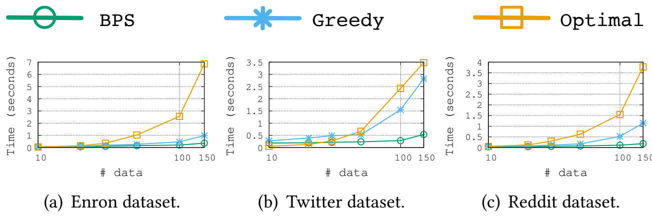
(a) $\lambda = 0.7$. (b) $\lambda = 0.8$. (c) $\lambda = 0.9$. (d) $\lambda = 1.0$.

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 experimental results under the previous settings in terms of time complexity to analyze the scalability of the various algorithms.

Time vs. #data: A critical question for scalability is how time complexity of filter optimization scales vs. the amount of 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. Naturally, the problem becomes harder as the size of the data increases. Overall, we critically notice that the BPS algorithm tends to have a lower time complexity than the greedy algorithm making it the most scalable of those compared.

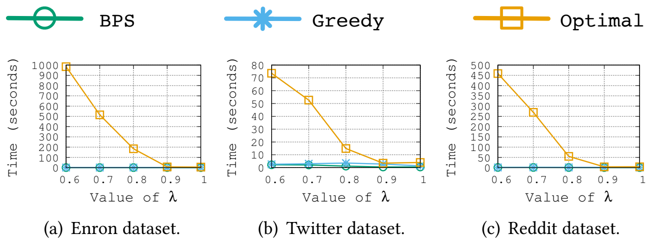


(a) Enron dataset. (b) Twitter dataset. (c) Reddit dataset.

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

Time vs. λ: Another question is how noise levels in relevance prediction affect the time complexity of filter optimization. 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 $\lambda \in [0.6, 1.0]$.

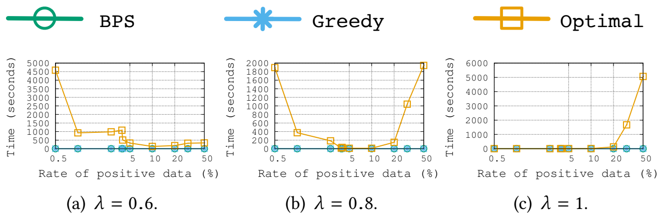
In brief, we first notice that clearly, for this configuration with $\lambda = 0.6$, the optimization algorithm has a very high time complexity (up to 16 minutes to find the optimal solution). Hence, the optimal solution is clearly not scalable. Moreover, we observe that this time complexity reduces 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.

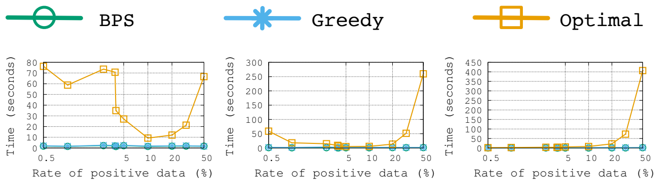
Figure 14: Time complexity: 2% positive data and #data = 150.

Time vs. rate of positive data: Finally we ask how changes in the proportion of positive data affect the time complexity of filter optimization. Figures 15, 16, and 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].



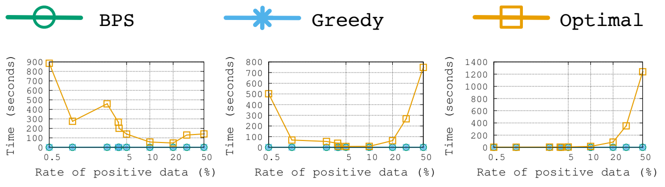
(a) $\lambda = 0.6$. (b) $\lambda = 0.8$. (c) $\lambda = 1$.

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 we observe both effects.

5.3 Summary of findings

In conclusion, the evaluations have shown that the BPS and Greedy algorithms are a good approximation of the Optimal MILP formulation and may outperform the MILP in high noise settings (especially the BPS approach which tends to overfit less to noise). We have also shown that the BPS 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 [8, 19, 36]. 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 [7], k-NN [2], rule learning [23] to deep learning [29]. 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 [25, 29] or by defining personalized re-ranking rules [9]. Finally, some recent advanced but very preliminary work has considered sequential optimization of interface adaptations based on user interactions [29]. 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 [35] proposed to use an expected score approximation to optimize Average Precision, Discounted Cumulative Gain, and Reciprocal Rank. However, the authors didn't propose a way to optimize recall and F1-Score, which are 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 filter settings that optimize the Boolean metric of expected F1-Score – not a ranking metric, although this may be part of our future work. As for using MILPs to optimize IR metrics, to the best of our knowledge, there is only one paper describing such a method proposed by Hansen et al. [16]. The drawback in this work is the need for ground truth relevance labels of all documents. Finally, a lot of work has been done for visualizing large-scale information graphs using filters selected via clustering-based approaches [1, 5, 22, 24, 27, 28, 31, 37]. However, the main difference with our work is that we have a "supervised" objective based on the relevance predictions, whereas clustering is inherently unsupervised.

Evaluation: Usually, visual information displays (VIDs) for search or exploration-oriented tasks are evaluated based on the system effectiveness of retrieving relevant elements, e.g., alarms [2], images [7, 9], textual items [32], news documents [25], or based on the usability of the interface [10, 11, 17]. Also, VIDs are usually evaluated through an off-line evaluation using specific benchmarks [2, 7, 29] or through a user study [2, 9–11, 17, 32]. Instead, the purpose of this paper was to introduce relevance-driven EF1 optimization algorithms to perform filter selection for VIDs; hence, our paper focuses on the evaluation of filter performance and computational time complexity for this difficult optimization problem.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we have expanded on the information filtering paradigm of Belkin and Croft [4], by introducing a novel formulation of filtering for visual information displays (VIDs) as a "supervised" optimization problem given an external measure of relevance for information elements. We have motivated and derived the expected F1-Score as an objective criteria for filter optimization and we have experimentally demonstrated that the proposed expected F1-Score metric is a good surrogate of the ground truth F1-Score.

We introduced novel filter optimization algorithms for expected F1-Score based on two different greedy strategies (Greedy and BPS). We have also proposed a method for optimization of expected F1-score using a MILP approach that has critically allowed us to benchmark the greedy algorithms on moderate-sized datasets. The evaluations we have performed on a variety of datasets have shown that the Greedy and BPS algorithms we have proposed are relatively efficient and provide a good approximation of the optimal MILP solution. Somewhat surprisingly, we remark that greedy methods like BPS may even do better than exact expected F1-score optimization in high noise cases due to their more restricted search space, which prevents overfitting to this noise.

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] C. Ahlberg and B. Shneiderman. Visual information seeking: Tight coupling of dynamic query filters with starfield displays. In R. M. Baecker, J. Grudin, W. A. Buxton, and S. Greenberg, editors, *Readings in Human-Computer Interaction (Second Edition)*, Interactive Technologies, pages 450 – 456. Morgan Kaufmann, second edition edition, 1995.

[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] N. J. Belkin and W. B. Croft. Information filtering and information retrieval: Two sides of the same coin? *Commun. ACM*, 35(12):29–38, Dec. 1992.

[5] A. Bennamane, H. Hacid, A. Ansiaux, and A. Cagnati. Vizpicious: A visual user-adaptive tool for communication logs analysis and suspicious behavior detection. In *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 1, pages 641–642, Dec 2012.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

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

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

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

[19] 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)*, pages 13–16, Aug 2016.

[20] 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.

[21] B. Klimt and Y. Yang. Introducing the enron corpus. In *CEAS*, 2004.

[22] 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.

[23] 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.

[24] J. Sankaranarayanan, H. Samet, B. E. Teitler, M. D. Lieberman, and J. Sperling. Twitterstand: News in tweets. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '09, pages 42–51, New York, NY, USA, 2009. ACM.

[25] 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.

[26] J. Shetty and J. Adibi. The enron email dataset database schema and brief statistical report. Technical report, 2004.

[27] B. Shneiderman and C. Dunne. Interactive network exploration to derive insights: Filtering, clustering, grouping, and simplification. In W. Didimo and M. Patrignani, editors, *Graph Drawing*, pages 2–18, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[28] M. A. Smith, B. Shneiderman, N. Milic-Frayling, E. Mendes Rodrigues, V. Barash, C. Dunne, T. Capone, A. Perer, and E. Gleave. Analyzing (social media) networks with nodexl. In *Proceedings of the Fourth International Conference on Communities and Technologies*, C&#38;T '09, pages 255–264, New York, NY, USA, 2009. ACM.

[29] 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.

[30] 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.

[31] B. E. Teitler, M. D. Lieberman, D. Panozzo, J. Sankaranarayanan, H. Samet, and J. Sperling. Newsstand: A new view on news. In *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '08, pages 18:1–18:10, New York, NY, USA, 2008. ACM.

[32] 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.

[33] G. van Kempen and L. van Vliet. Mean and variance of ratio estimators used in fluorescence ratio imaging. *Cytometry*, 39(4):300–305, 2000.

[34] 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.

[35] 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.

[36] 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.

[37] H. Yifan and S. Lei. Visualizing large graphs. *Wiley Interdisciplinary Reviews: Computational Statistics*, 7(2):115–136, 2015.

[38] J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Comput. Surv.*, 38(2), July 2006.