

Structural Clustering of Machine-Generated Mail

Noa Avigdor-Elgrabli*, Mark Cwalinski†, Dotan Di Castro*, Iftah Gamzu*,

Irena Grabovitch-Zuyev*, Liane Lewin-Eytan*, Yoelle Maarek*

*Yahoo Research, Haifa, Israel {{noaa,dot,iftah,iragz,liane}@yahoo-inc.com, yoelle@ymail.com}

†Yahoo Inc., Sunnyvale, CA, USA {markcwal@yahoo-inc.com}

ABSTRACT

Several recent studies have presented different approaches for clustering and classifying machine-generated mail based on email headers. We propose to expand these approaches by considering email message bodies. We argue that our approach can help increase coverage and precision in several tasks, and is especially critical for mail extraction. We remind that mail extraction supports a variety of mail mining applications such as ad re-targeting, mail search, and mail summarization. We introduce new structural clustering methods that leverage the HTML structure that is common to messages generated by a same mass-sender script. We discuss how such structural clustering can be conducted at different levels of granularity, using either strict or flexible matching constraints, depending on the use cases.

We present large scale experiments carried over real Yahoo mail traffic. For our first use case of automatic mail extraction, we describe novel flexible-matching clustering methods that meet the key requirements of high intra-cluster similarity, adequate clusters size, and relatively small overall number of clusters. We identify the precise level of flexibility that is needed in order to achieve extremely high extraction precision (close to 100%), while producing relatively small number of clusters. For our second use case, namely, mail classification, we show that strict structural matching is more adequate, achieving precision and recall rates between 85%-90%, while converging to a stable classification after a short learning cycle. This represents an increase of 10%-20% compared to the sender-based method described in previous work, when run over the same period length. Our work has been deployed in production in Yahoo mail backend.

1. INTRODUCTION

Recent studies have demonstrated that more than 90% of non-spam traffic in Web mail is *machine-generated* [2, 17]. Our mail inboxes are thus filled with email messages generated by automated scripts that originate from commercial entities and organizations. Examples include shipment no-

tifications, flight itineraries, social events, bank statements, newsletters, etc. Consequently, a Web mail corpus today consists of mostly structured data (typically, HTML formatted), where variants of a same type of message are repeated at a large scale across numerous users. Clustering these messages into cohesive groups allows to approximate the script(s) that generated them and share a same intent and is an approach that has been used recently by several research studies, e.g., [2, 39].

Some of the authors of this work have previously introduced the notion of *templates* [2] in order to identify such clusters by analyzing message headers. Templates consists of a pair $\langle \text{sender}, \text{subject regex} \rangle$, where the subject regular expression captures variations of a same subject line (e.g., “Confirmation of order #.*”). These templates have been used for clustering similar messages so as to infer causality between similar messages [2], as well as for mail classification [39]. However, such header-only based approach clearly suffers from limitations. We have observed multiple cases of misrepresentation, such as messages with drastically different content associated with a same header-based template (e.g, a mass sender sending very different messages with the same subject line “update”, or messages with semantically similar content associated with different templates due to highly personalized subject lines (e.g., “1 new profile view, 4 new job offers, ...”). Having messages with drastically different content belonging to a same cluster becomes a show stopper in various application cases that extract information from the body of messages, such as flight alerts, automatic calendar updates, etc. We refer to this family of use cases as the “mail extraction” family.

We argue here that it is necessary to consider the body of messages in order to properly cluster machine-generated messages and support all use case families, from various mail classification schemes to mail extraction applications. As machine-generated mail typically use rich HTML formatting, we propose to leverage the *structural match* between messages when evaluating message similarity. Similarly to many cases of matching scenarios, we have the option to enforce either a perfect strict or a fuzzy flexible match.

Some of the authors of this work have previously introduced a strict structural match method specifically tailored to mail applications [16]. This method relies on the DOM tree of HTML messages, and represents each message as an ordered list of all the XPath expressions that lead to textual leaves in that tree. A *Mail-Hash signature* is then applied to the list, allowing to cluster messages according to their resulting (structure-based) signature. We refer to

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CIKM'16, October 24-28, 2016, Indianapolis, IN, USA

© 2016 ACM. ISBN 978-1-4503-4073-1/16/10...\$15.00

DOI: <http://dx.doi.org/10.1145/2983323.2983350>

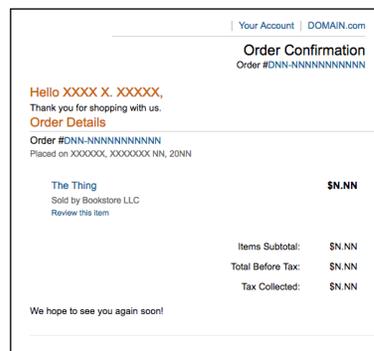
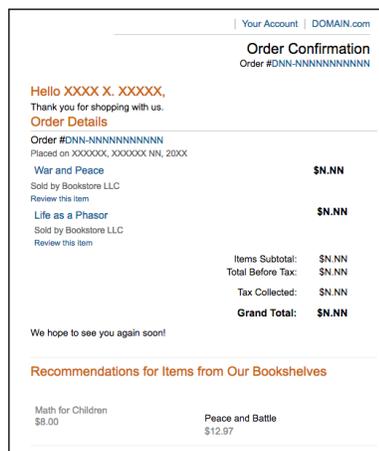


Figure 1: Anonymized email samples of an order confirmation. Note the the left sample consists of two items and a promotion section at the bottom, while the right sample has only one item and no promotion section.

the generated clusters as XPath clusters, or *x-clusters*. We suggest to reuse this method, which demonstrated its value for k-anonymization, in other scenarios that require strict matching. Examples include fully automated tasks that do not require interpretability during any intermediate phase of their cycle, like mail classification.

Note that, while being extremely efficient and precise, this method results in a very large number of often small x-clusters when applied to a huge corpora such as mail, greater by more than an order of magnitude than the number of clusters generated by template-based techniques. As a result, data becomes sparse, which becomes problematic in various family of applications, especially in domains where mass-senders proliferate, such as in e-commerce.

We therefore argue that there is a need for more flexible matching that would fulfill the three following requirements: (1) high intra-cluster message similarity, (2) an adequate cluster size, and (3) a maintainable number of clusters, where the thresholds for each requirement remain application-driven. We investigate the relaxation of strict matching by various means depending on the application domain. Consider as an illustrating example the two messages shown in Figure 1. A strict structural matching method would assign them to two distinct clusters, while a more flexible one that collapses recurring sub-structures of a message body into a single instance, would assign them to the same cluster, which seems to be reasonable here. Following this intuitive idea, we believe that the degree of flexibility in the similarity between structures should depend on the application domain. Consequently, in the general area of mail extraction, we consider not only collapsing substructures but also merging structures that are close enough in terms of edit distances. This results in grouping together several x-clusters into a same meta-cluster, or *m-cluster*, and allows us to apply the same extraction rule to both messages shown in Figure 1, and all other members of the same m-cluster.

The contributions of this paper are threefold:

1. We provide a first study of structural clustering of the body of machine-generated messages, departing from previous header-only approaches. We consider both strict and flexible structural matching techniques for

computing clusters, each associated with a different family of tasks.

2. We introduce the task of mail data extraction, supported by several flexible-match clustering methods. We empirically analyze the quality and applicability of those using a real large scale dataset of Yahoo Web mail service, and deduce the level of structural-flexibility tailored for the extraction task.
3. We develop a new mail classification approach that operates on strict-match based clusters. We conduct large scale experiments on real traffic of Yahoo Web mail in order to establish the superiority of our approach compared to previous sender-based classification.

We emphasize that all the experiments reported here have been conducted on fully anonymized samples at the cluster level, using the anonymization method presented in [16], and in accordance with Yahoo privacy policy.

The paper is organized as follows. We begin by presenting related work in Section 2 and introducing some preliminary notions in Section 3. In Section 4, we describe our flexible-match clustering methods and study their application to the novel mail extraction use case. In Section 5, we address strict-match clustering and present our mail classification process. Finally, we give our unified conclusions in Section 6.

2. RELATED WORK

Mail Classification. Various classification methods have been presented for the purpose of organizing users inbox, the main ones being either rule-based, information retrieval-based, or machine learning-based, e.g., [3, 12, 22, 23]. Two studies on classification experimented over Yahoo mail data are the works of Koren et al. [24] and Grbovic et al. [17]. The first is based on learning the most popular folders from the entire users population, while the latter is focused on mail senders, distinguishing between personal and machine senders and classifying senders into several categories.

Mail Extraction. Information extraction is a major research challenge at Web scale. Common techniques typically

build on clustering of the underlying documents. Algorithms for document clustering are widely used in information retrieval systems to group similar documents together. Web documents analysis stands at the core of this field, focusing mainly on HTML-based pages. Several approaches for identifying document similarity exist in this context. One main approach is based on the semantic or statistical term similarity of documents (see, e.g., [29] for an introduction of related techniques). Broder et al. [6, 7, 8, 9] reduce documents to a canonical sequence of syntactical tokens, upon which a small fixed size sketch is efficiently computed and utilized for clustering (see also [28, 19, 20]). Although machine-generated emails are typically documents in rich HTML format, they do not easily conform to the above types of similarity when structural considerations (like the ones required for data extraction) are meaningful. Identifying document similarity based on its structure, i.e., the underlying HTML or XML structure of the document, has also been given a lot of consideration in the past [5, 13, ?]. Although these methods show some similarity with our work, they are not dedicated to the mail domain and therefore do not exploit meaningful mail signals, which we propose to use here.

Information extraction commonly assumes that the generation of Web pages follows two main steps: the creation of a *layout* according to design requirements, and the placement of *content* into that layout. In a sense, most techniques try to reverse this generation approach by seeking to separate between the layout and the data. There is abundant body of research on extraction methods, where some of the main ones build on Web page structure [4, 21, 25, 38], pattern recognition [10], tree methods [14, 18], repetitive information identification [32, 33, 34], vision techniques [26], and more. One specifically relevant method is the one using tag path clustering [30]. This method while related is not tailored to the mail domain, and therefore, as previously mentioned, does not exploit mail specific signals.

3. PRELIMINARIES

We present the basic notions required for both our classification and extraction use cases.

3.1 X-Clusters

X-clusters are basic building blocks in both our meta-clustering methods, described in Section 4, and our classification process, presented in Section 5. An x-cluster is a group of mail messages from the same domain sharing the same structure. The key representative structure of an x-cluster is its *XPath-list*. This ordered list includes all XPath expressions leading to textual leaves in the DOM tree of the messages of the cluster (and is in fact a parallel representation to the DOM tree). This structure has been introduced in [16], where a succinct representation of it, coined the *Mail-Hash signature*, has been introduced. The Mail-Hash signature is computed by taking the lower 64-bits of the MD5 signature computed over all the XPaths in the XPath-list. In the current work, an x-cluster is formally characterized by a $\langle domain, XPath-list \rangle$ pair.

3.2 Extraction Rules

Extraction rules are a key element of the mail extraction task, the latter being an area of focus of this work. In general, extraction rules may be interpreted as short scripts or programs whose purpose is to gather data pieces from doc-

uments like mail messages. In the mail settings, they are commonly associated with clusters of messages that share a common structure. The data that needs to be extracted from a given cluster typically depends on the type of the cluster (e.g., order confirmation, travel itinerary), and is captured by a schema designating the different data pieces to extract (e.g., Schema.org¹ is commonly adapted for designated schemas of structured data). Clearly, different types imply different schemas. For example, an order confirmation schema may contain attributes such as *order date*, *order number*, *order item*, and more. Figure 1 shows an example of an order confirmation email.

An extraction rule consists of procedures (automatically or manually defined) that formalize the way to extract the important data pieces required by a schema. For instance, the order date attribute may be extracted from the example in Figure 1 by a procedure that first identifies the starting position of the order details part (i.e., `table[@id='main']/tbody/tr/td/table[@id='orderDetails']`), and then extracts the data in a concrete location relative to that starting position (i.e., `span[@class='orderDate']`). Note that the locations are defined using the XPath query language for selecting nodes from an HTML source. A procedure may apply additional post-processing operations on the data extracted from a node. For example, the procedure may execute a piece of code, trimming the “Placed on” prefix from the extracted order date. This allows attaining a well-formed date object. There are many other operations that can be supported by such procedures. For example, procedures can be repeatable, that is, they may run multiple times to extract data from multiple locations. This can be useful, for example, when extracting a variable number of ordered items from an order confirmation email.

4. USE CASE 1: MAIL EXTRACTION

We describe here our methods for generating the meta-clusters or *m-clusters* generated by applying flexible-matching constraints on the structure of message bodies. We also detail they can be applied to mail extraction. Our starting point are x-clusters defined in Section 3, each x-cluster being characterized by the pair $\langle domain, xpath-list \rangle$.

4.1 Collapsing into Stripped Structures

Our most basic meta-clustering method, also referred to as *stripped clustering*, groups together x-clusters by collapsing repetitions of sub-structures in the XPath-list into one instantiation of each sub-structure. Formally, the stripped clustering method is quite similar to the x-clusters method, but instead of grouping together emails having the same $\langle domain, xpath-list \rangle$ pair, it groups together emails having the same $\langle domain, stripped-xpath-list \rangle$ pair. The *stripped-xpath-list* of an email can be generated by going through its ordered XPath-list and recording any new XPath encountered within the list. Note that the index values of XPath predicates are neglected during this stripping process. An illustration of this process is shown in Figure 2. Interpreting this approach as a meta-clustering method on top of x-clusters, this method groups x-clusters that share the same “plain” *stripped structure*.

The intuition behind this clustering approach is rather straightforward. Most substructure repetitions in machine-

¹<http://schema.org/>

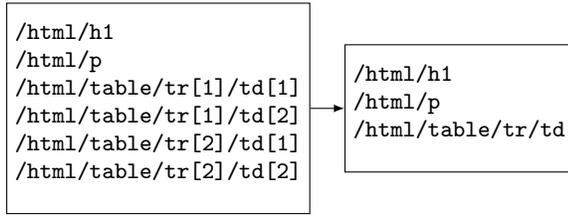


Figure 2: An example of an XPath-list (left) and its corresponding stripped XPath-list (right).

generated mail messages have the same semantic interpretation, e.g., product order lines in an order confirmation or shipping notification messages, flight lines in itinerary messages, etc. The same data extraction procedure can be applied to this recurrent substructure across all its occurrences. This hints that the extraction essence in such messages is already captured by their stripped structure.

Note that computing the stripped structure of an email can be done efficiently, like in the x-clustering approach. In addition, constructing a list of m-clusters for a given dataset can be easily done in a single pass over the entire dataset of emails. This makes the m-clustering method extremely efficient and scalable in practice.

4.2 Stripped Clustering with Edit Distance

The properties of the stripped clustering approach, and especially the simplicity of constructing the clusters, make it a natural candidate for meta-clustering. Nevertheless, in many cases of interest, one needs a method that is more potent in reducing the number of m-clusters for a given dataset. A key requirement is that the generated m-clusters capture core extraction components and support high quality extractions. In many realistic scenarios, mail messages with different stripped structures still share common structures used for extraction, while only varying in secondary structures that are not valuable for extraction. For example, product promotions or recommendations are sometimes affixed to purchase confirmation messages. Although these additional components change the stripped structure of a message, they do not introduce additional essential data parts to be extracted on top of those in the “core” message structure. It clearly makes sense to group such similar stripped structures together under the same m-cluster.

This observation immediately gives rise to a natural generalization of the stripped clustering approach, allowing more flexibility by grouping together stripped structures having small pairwise edit distance. This is materialized in algorithm **Stripped-Edit-Distance-Clustering**, formally described below. This algorithm begins by forming a m-cluster for each stripped structure. Then, it computes the edit distance between all m-cluster pairs, i.e., the underlying stripped structures. Note that the XPath paths are the basic comparison unit in the edit distance computation. In particular, the computation is oblivious to specific tags inside XPath paths when testing for equality. Any pair that has a sufficiently small edit distance (smaller than a given parameter D) is added to a distance-based priority queue. The algorithm continues by iteratively extracting from the queue m-cluster pairs with minimum distance and merging them. Whenever two m-clusters are merged, their edit distance to any other m-cluster is updated. Note that the input provided to

the algorithm is a set S of stripped structures (i.e., stripped-xpath-list), generated from a given dataset of messages using the process described in Section 4.1. We assume without loss of generality that all the messages in the dataset share the same domain, and thus, the output of the algorithm is a m-clustering for that domain. This assumption is sound since one can apply the algorithm for each domain separately.

Algorithm 1 Stripped-Edit-Distance-Clustering

Input: A set $S = \{s_1, s_2, \dots\}$ of stripped structures
 An edit distance bound parameter $D \in \mathbb{N}_+$
Output: A m-clustering $\mathcal{C} : S \rightarrow \{C_1, C_2, \dots\}$

▷ Step 1: identify mergeable stripped clusters

- 1: PQ \leftarrow empty distance-based priority queue
- 2: **for** $s_i \in S$ **do** $C_i \leftarrow \{s_i\}$, and let $\mathcal{C} \leftarrow \{C_1, C_2, \dots\}$
- 3: **for** $(s_i, s_j) \in S \times S$ such that $s_i \neq s_j$ **do**
- 4: $d \leftarrow \text{Edit-Distance}(s_i, s_j)$
- 5: **if** $d \leq D$ **do** PQ.insert($(C_i, C_j), d$)

▷ Step 2: iteratively merge clusters

- 6: **while** PQ.isNotEmpty() **do**
- 7: $(C_i, C_j) \leftarrow \text{PQ.extractMin}()$
- 8: $C_i \leftarrow C_i \cup C_j$, and $\mathcal{C} \leftarrow \mathcal{C} \setminus \{C_j\}$
- 9: **for** $C_k \in \mathcal{C} \setminus \{C_i, C_j\}$ **do**
- 10: $d_i \leftarrow \text{PQ.extract}(C_i, C_k)$
- 11: $d_j \leftarrow \text{PQ.extract}(C_j, C_k)$
- 12: **if** both d_i and d_j exist **do**
- 13: PQ.insert($(C_i, C_k), \max\{d_i, d_j\}$)

Few remarks are in place with respect to the algorithm. The first is about its running time and space usage. Clearly, both time and space complexity depend on the concrete implementation of the priority queue and the way the edit distance is computed. Arguably, the simplest design decisions here are to implement the priority queue as a heap, and utilize the well-known dynamic programming algorithm for edit distance calculation [37, 27]. This allows supporting each queue operation with a worst case running time of $O(\log |S|)$, and each edit distance calculation with running time and space of $O(\ell_1 \ell_2)$, where ℓ_1 and ℓ_2 are the length (i.e., number of XPath paths) of the underlying stripped structures. This implies that the respective running time and space usage of the entire algorithm (both steps) are $O(|S|^2 \cdot (\log |S| + \ell^2))$ and $O(\max\{|S|^2, \ell^2\})$, where ℓ is the maximum length of any stripped structure. In particular, note that the number of iterations in Step 2 is at most $O(|S|^2)$ since the algorithm adds a m-cluster pair to the queue only if it previously extracted two m-cluster pairs. Both the running time and space usage terms can be improved by employing better implementation techniques. For example, as our algorithm is parameterized with an upper bound on the edit distance of mergeable m-clusters, we can use the edit distance algorithm of Ukkonen [35] whose running time is $O(D \cdot \min\{\ell_1, \ell_2\})$ and space utilization is $O(D \cdot \min\{D, \ell_1, \ell_2\})$. Note that we always use a small constant value of D in our experiments.

An additional important observation regarding the above algorithm is that it can be easily extended and enhanced for improving its performance in practice. For instance, the algorithm can be extended to incorporate more involved prioritization considerations (e.g., give precedence for merging m-clusters whose underlying stripped structures have more XPath paths), or enhanced to use intricate merging criteria (e.g., merge m-clusters if the edit distance between them is smaller

than some fraction of the number of XPath nodes in their underlying stripped structures). Such modifications help to tailor the algorithm for different scenarios.

4.3 Improved Clustering with Edit Distance

An alternative to algorithm Stripped-Edit-Distance-Clustering with improved time and space utilization is formally described below as Pivot-Stripped-Edit-Distance-Clustering. This simple algorithm iteratively selects a random (pivot) stripped structure, and builds a m-cluster around it. Constructing clusters around randomly selected pivot nodes has been shown to provide good theoretical and practical guarantees in different clustering settings (see, e.g., [1, 11, 36]). We show that this algorithm also works well in our case, although it employs a more relaxed approach. Specifically, we empirically demonstrate that this algorithm achieves a comparable solution quality to that of algorithm Stripped-Edit-Distance-Clustering, while being more resource efficient. We believe this finding is interesting on its own right.

Algorithm 2 Pivot-Stripped-Edit-Distance-Clustering

Input: A set $S = \{s_1, s_2, \dots\}$ of stripped structures
 An edit distance bound parameter $D \in \mathbb{N}_+$

Output: A m-clustering $\mathcal{C} : S \rightarrow \{C_1, C_2, \dots\}$

```

1:  $i \leftarrow 1$ 
2: while  $S \neq \emptyset$  do
3:    $s \leftarrow$  uniformly random stripped structure from  $S$ 
4:    $C_i \leftarrow \{s\}$ 
5:   for  $t \in S \setminus \{s\}$  do
6:     if  $\text{Edit-Distance}(s, t) \leq D$  do
7:        $C_i \leftarrow C_i \cup \{t\}$ 
8:        $S \leftarrow S \setminus \{t\}$ 
9:    $i \leftarrow i + 1$ 

```

One can easily validate that even when the edit distance calculation is implemented by the well-known dynamic programming algorithm, the worst case time and space utilization is $O(|S|^2 \ell^2)$ and $O(\max\{|S|, \ell^2\})$, respectively. This can be improved by utilizing an improved edit distance calculation. For example, the algorithm can have optimal linear-space usage by employing Ukkonen’s algorithm [35].

4.4 Experimental Evaluation

We present empirical evaluation for both algorithms, that is, algorithms Stripped-Edit-Distance-Clustering (referred to as SC) and Pivot-Stripped-Edit-Distance-Clustering (PSC). We use $\text{SC}(D)$ and $\text{PSC}(D)$ to denote the parameterization of the algorithms with an edit distance bound of D . Note that both $\text{SC}(0)$ and $\text{PSC}(0)$ reduce to the simple stripped clustering method described in Section 4.1.

4.4.1 Evaluation Setup

We test our algorithms on the actual traffic of Yahoo Web mail service. The evaluation of each algorithm consists of three main steps:

- We first generate meta-clusters for one day of mail data (say, d_1) using the algorithm under evaluation.
- We associate extraction rules with some of the generated m-clusters. The way extraction rules are constructed and assigned to m-clusters is described below.

- We apply an extraction rule associated with a m-cluster over all its messages (gathered from a different day, d_2), and analyze the outcome. We consider an application of a rule on an email message as successful if it extracted the relevant data pieces from that message.

We primarily compare between the outcomes of both algorithms with different edit distance bound parameterization. We use the x-clustering approach as a reference point for comparing the overall number of clusters. Due to the relatively high number of x-clusters, one can not generate extraction rules for all of them. However, it is intuitive to assume that the success ratio of such extraction rules would have been close to 100% as all the messages in each x-cluster have an identical structure.

4.4.2 Rules Assignment

The process of assigning extraction rules to m-clusters consists of two steps. Initially, a collection of nearly 2500 stripped structures was selected, and given to professional quality insurance staff, referred to as “editors”, for manual definition of extraction rules. Those structures were chosen from multiple domains, and various business-justified category types (e.g., Travel, Finance, etc.). Note that the classification process presented in Section 5 was utilized to ensure the selection of structures with corresponding category types. Furthermore, the structures were selected so that they had similar daily mail traffic to avoid bias due to different traffic distributions.

Then, for each meta-clustering algorithm, we constructed a mapping from each m-cluster to at most one extraction rule by considering all the rules assigned to the stripped structures within that cluster, and identifying the best rule in terms of extraction success ratio and traffic coverage of the structure.

4.4.3 Extraction Evaluation

We present the extraction results achieved for the different meta-clustering methods. We are mainly interested in understanding the trade-off between the number of m-clusters created by our methods and the ratio of successful email extractions. We are also interested in the overall number of email extraction attempts (i.e., coverage of the clustering method in terms of number of messages). Note that our measurements for the randomized algorithm PSC indicate the average values attained for 10 experiments. A further analysis of these experiments demonstrates that the true values of all the measurements under consideration lie in an interval of no more than $\pm 1\%$ around the averages with high probability (i.e., confidence of above 95%).

It is intuitive to expect that as the edit distance increases in each of our algorithms, the number of m-clusters and the success ratio should decrease, and the coverage should increase. Indeed, as can be seen in Table 1 and Figure 3, this is generally the trend. One interesting and important observation regarding the success ratio is that both our methods have a concrete edit distance bound beyond which the extraction performance deteriorates sharply. For algorithm SC, this edit distance bound is 3, while for PSC, it is 2. In both cases, crossing this bound results in a deterioration of more than 10% in the extraction success ratio. The reason for this deterioration is rather clear: when the edit distance between the structures in a m-cluster becomes too large, the structural variability becomes too high for a single ex-

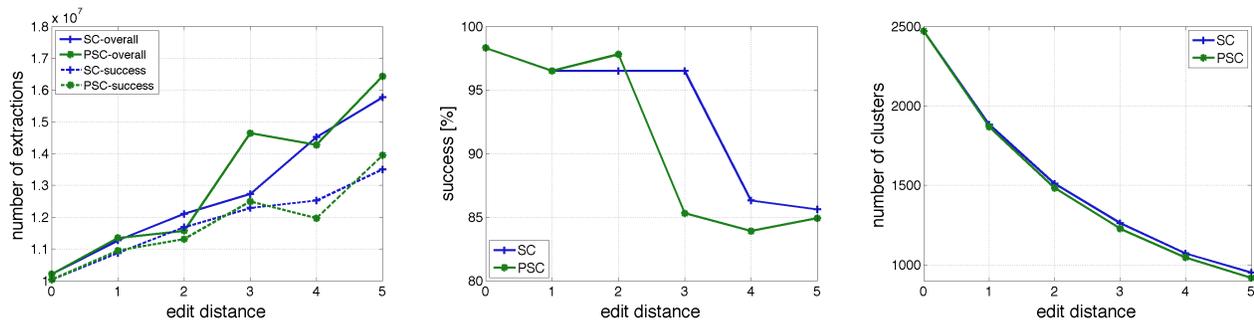


Figure 3: The extraction results achieved for the different meta-clustering methods. The results are presented with respect to varying levels of edit distance. Note that the number of x-clusters follows from the selection of the stripped structures.

Clustering method	Extraction attempts	Successful extractions	Success ratio	Number of clusters
x-clustering				20,018
SC(0)	10,204,003	10,029,671	98.3%	2470
SC(1)	11,264,858	10,867,187	96.5%	1882
SC(2)	12,102,625	11,678,910	96.5%	1510
SC(3)	12,725,416	12,284,170	96.5%	1262
SC(4)	14,510,407	12,518,141	86.3%	1071
SC(5)	15,764,210	13,498,539	85.6%	951
PSC(0)	10,204,003	10,029,671	98.3%	2470
PSC(1)	11,345,523	10,948,429	96.5%	1868
PSC(2)	11,567,372	11,307,139	97.8%	1483
PSC(3)	14,640,723	12,490,332	85.3%	1227
PSC(4)	14,264,943	11,966,334	83.9%	1046
PSC(5)	16,423,536	13,946,311	84.9%	917
PSC(6)	18,547,277	15,837,179	85.3%	821
PSC(7)	18,901,811	15,853,518	83.9%	746
PSC(8)	20,013,552	16,123,143	80.6%	702
PSC(9)	20,779,379	16,255,188	78.2%	661
PSC(10)	21,209,923	16,489,105	77.7%	633

Table 1: The extraction results achieved for the different meta-clustering methods.

traction rule to handle. The concrete bound is mostly an artifact of the extraction technique and its properties. Also note that the fact it is smaller for algorithm PSC is also expected since an edit distance bound of D for this algorithm may result in m-clusters with edit distance of $2D$ between some of their stripped structures.

Another valuable observation is that both clustering methods have similar behavior with algorithm PSC being more jittery. This latter property is expected as algorithm PSC uses randomization and acts less conservatively when merging stripped structures.

As we focus on method SC(3), we can see that its coverage is greater than the coverage of the simple stripped clustering method by roughly 25%, while its success ratio is less than 2% lower than that of stripped clustering. Furthermore, the overall number of clusters is nearly half of that in stripped clustering, and more than an order of magnitude smaller than the number of clusters generated by the x-clustering approach. These properties indicate high extraction performance with a practically maintainable number of clusters. This is exactly what we aimed to achieve.

5. USE CASE 2: MAIL CLASSIFICATION

We describe here our approach for clustering mail based on body structure for the task of mail classification. Note that this case being fully automated, there is no need to

relax constraints so as to reduce the number of clusters and we use strict matching techniques. We then demonstrate its superiority over a sender-only based method by performing a thorough comparison with [17].

5.1 Classification Process

Unlike previous mail classification methods, which cluster messages based on their header (sender identity in [17], and header-based template in [39]), we propose to cluster messages based on the structure of their body, according to the x-clusters they belong to. This method is tailored to machine generated messages, as messages generated by the same script will have a similar structure. Additionally, this clustering method also provides a natural way to extract structural features that are often highly indicative of the category of the message, as detailed in this section.

We investigate the classification of x-clusters into the following categories: *Purchases*, *Finance*, *Travel*, *Services*, *Events*, and *Marketing*. These categories are closely related to those used in [17, 39]. *Purchases* include notifications related to online purchases, such as receipts and shipping notifications. *Finance* refers to financial activities such as transactions related to stocks, credit cards, banking, and more. *Travel* includes all activities related to travel, the main ones being flights, but also car rentals, hotel reservations and other trip rentals. *Services* comprises of periodical payments to service providers such as phone, cables, and Internet companies. *Events* includes reservations, receipts and subscriptions related to events such as restaurant reservations, sporting events, beauty and health appointments, etc. Finally, *Marketing* applies to all promoted products and similar content, which could be either general or personalized with respect to the user.

We follow a general categorization framework, where numerical features are extracted at the granularity of x-clusters. The generated features, together with labeled examples, are fed to a learning mechanism to create a classification model.

5.2 Features

In general, our features may be either based on messages of the x-cluster or apply directly to the structure of the x-cluster. Features that apply to messages are aggregated at the x-cluster level over the whole mail traffic. Such features include user actions performed over the messages, or content features related to the messages. Features applying directly to the x-cluster structure represent attributes of the x-cluster, and use the strength of the structural clustering method used for classification. As detailed later in this sec-

tion, structural attributes of an x-cluster are often valuable indicators of its nature. We list below the various types of features we use and explain the way they are computed. The complete list of features is summarized in Table 2.

Content features. Some of these features are derived from the “bag of words” extracted from the subject and body of the messages in the x-cluster, following the content analysis performed over the x-cluster traffic. We calculate the frequency of each word, eliminating the most frequent words, as per *idf*, found across most x-clusters (not contributing to any differentiation), stop words, and words with low frequency that accrue in very few x-clusters. As a last step, we perform a stemming process using the classic Porter stemmer [31] over our lexicon.

Other types of features are statistics computed over the length of the subject and body of the messages, similarity between the x-clusters messages with respect to the same bag of words, and statistics on the number of URLs in the body. We also generate features corresponding to special symbols and strings in the subject or body. For example, we noticed that punctuation marks such as ‘!’ and ‘?’ in the subject often indicate a marketing x-cluster, probably due to the appealing personal style favored by ads campaigns. Other strings are ‘Re’ and ‘Fwd’, usually indicating personal correspondence, and helping us identify they do not belong to any machine generated category. Typical symbols being valuable indicators of our categories are also currency symbols, credit card names, airport identifiers, and Schema.org types such as “FlightReservation” or “OrderProcessing”.

Behavioral features. These features include signals indicating the actions that users performed on the messages of the x-cluster. We consider the following actions: read, reply, forward, delete, mark-spam, mark-ham (not-spam), flag, and move-to-folder. The actions are a signal of the engagement level of the users at the x-cluster level, and help distinguish between significant and non-significant correspondence. More details on the importance of message actions when analyzing Web mail traffic can be found in [15]. Further, in case of a move-to-folder action, the folder name is a valuable signal with respect to the x-cluster category. For example, a high amount of messages in the x-cluster moved to a folder named “shopping” is a good indicator that this x-cluster belongs to the *Purchases* category. The list of folder names is generated similarly to the bag of words used for content features.

Traffic features. These features reflect the x-cluster traffic distribution over time. As part of these features, we collect the total number of messages sent per week, per day of the week, as well as per every hour of the day. We do so in order to understand the traffic behavior of the x-cluster, to identify “rush hours” or any significant days. From these statistics, we extract both counter and indicator features (e.g., an indicator of whether more than 80% of the x-cluster’s messages are sent during a specific hour of the day). We note that bursts are a good indicator of marketing emails.

Address features. We extract features corresponding to substrings in the sender address, similarly to the content features. Note that an x-cluster always corresponds to a single domain, but might correspond to more than one sender in this domain. For example, for an x-cluster with messages from `ticketreservations@lufthansa.com`, we will have indicator features for “ticket”, “reservation”, and “lufthansa”.

As the domain by itself carries significant classification information, we also have features corresponding directly to large domains such as `amazon.com`, `mit.edu`, and more.

Structural features. These features are calculated over a sample of about a hundred messages of the x-cluster, and do not require examination of the entire x-cluster traffic. The sample is large enough to deduce structural characteristics due to their high structural similarity. The structural features correspond to the number of XPath paths in the structure as well as the XPath depth of the structure. These are both valuable indicators to the x-cluster nature. For example, an x-cluster of an order confirmation (thus classified as ‘Purchase’) usually has deep XPath paths corresponding to the specification of the purchase, typically presented in a table (e.g., `/html/body/table/tbody/tr[3]/td[1]/p/text()` with a depth of eight HTML tags). On the other hand, the structure of messages composed by persons usually contain only small amount of shallow XPath paths, as no complex structure will be established (e.g., `/html/body/p/text()` with a depth of three tags).

Another type of structural feature is structural similarity, represented by a value in the range of $[0, 1]$. The similarity is computed by measuring the deduplication level of the sample using PACK chunking [40]. A similarity of 1.0 indicates that all the messages are identical, which will typically be the case for clusters corresponding to commercial ads in the marketing category.

The last structural feature type represents the ratio of *constant XPath values*. A constant XPath value is a textual string pointed by the same XPath which is identical for most of the messages in the x-cluster. For example, the string “Thanks for shopping with us, we have received the order you placed on MyShoppings.com”, appearing at the same HTML location in most of the messages of a corresponding x-cluster, is considered constant. The *constant level* of an XPath determines the ratio of messages with an identical value for that XPath, while its *variability level* determines the ratio of messages with different values. We consider different thresholds for constant and variability levels.

5.3 Experimental Evaluation

5.3.1 Evaluation Setup

Our experiment was conducted over a period of one week covering billions of messages. To form our ground truth, we sampled 6.5K messages, each corresponding to a different x-cluster. The sample was collected gradually in order to obtain sufficient coverage for each category, and was based on a thorough manual labeling process conducted by professional staff. Figure 4(a) presents the labels distribution of a randomly chosen x-clusters sample. As can be observed, taking a random sample of x-clusters does not provide a good representation of all categories and would have harmed our ability to train and test our models. Our sample was thus produced via several classification rounds, each time using examples that were previously (manually) labeled. Each round generated newly classified data, that was then sampled to cover evenly all categories, and passed on to the manual labeling process in order to enlarge our ground truth dataset.

Our test set was chosen as 15% of our labeled set, sampled at random equally for each category. A model was trained for each category separately as “one-vs-all” using the same training set, where the labels of the category considered were

Type	Sub-Type	Feature Name	Description
Content	Subject/Body	words length repetitiveness urls	words appearance in subject/body (bag of words) average/difference/min/max of subject/body length over all messages percentage of repetitive words represented by bins (subject/body) average/difference/min/max number of URLs in the body
	Special Symbols	money symbols credit card airport names schema.org punctuation reply/forward	money symbols counter/percentage credit card names counter/percentage airport identifiers counter/percentage binary indicator of the schema.org type appearance of '?'/'!' marks in the subject - counter/percentage appearance of 'Re'/'Fwd' in the subject - counter
Behavioral	User Actions	action seen replied forwarded flagged delete spam ham	(any) actions counter/percentage read messages counter/percentage replied messages counter/percentage forwarded messages counter/percentage flagged by star messages counter/percentage deleted messages counter/percentage marked as Spam messages counter/percentage marked as Ham (not spam) messages counter/percentage
	Folder	move to folder folder type	move to folder counter/percentage set of binary features indicating names of the folders (bag of names)
Traffic	Coverage	inbound traffic	number of weekly incoming messages
	Temporal	time by hour/day	traffic distribution within a day/week, represented by bins
Address		address words	words appearance in sender address (bag of words)
Structural		XPath count XPath depth similarity constant XPaths ratio variable XPaths ratio	XPaths number, indicators for different max thresholds in [5, 100] average/difference/min/max XPaths depth similarity indicators for different max thresholds in [0, 1.0] ratio of constant XPaths for different constant levels ratio of variable XPaths for different variability levels

Table 2: The set of features used by the classification process.

set to be true while the labels of the other categories were set to false. The learning model was generated using the highly scalable and commonly used Vowpal Wabbit logistic regression algorithm². As a final step, the predictions that each category received from the logistic regression algorithm were merged into one prediction based on the category with the highest score.

In parallel, we ran the sender-based classification process from [17], using exactly the same code developed by the authors of that work. In order to conduct a fair comparison, we used exactly the same six categories that were used in the x-clusters classification³. We created the sender-based labeled set using the same traffic covered by our x-cluster labeled set, resulting in 5M senders (corresponding to the labeled x-clusters). From our 5M labeled senders, we sampled about 6.5K senders covering evenly all categories, out of which 15% were chosen for testing. The orders of magnitude difference between the number of senders and number of x-clusters needed to cover the same traffic is worth noting, and hints on the difference that our comparison will point to. This follows the much higher traffic coverage that can be obtained by x-clusters as compared to senders.

5.3.2 Evaluation Results

Our results are given in Figures 4(b) and 4(c). Specifically, Figure 4(b) presents the precision/recall curve of the x-clusters classification, while Figure 4(c) presents the precision/recall curve of the senders classification.

It can be observed that based on a period of one week only, our classifier achieves precision and recall rates between

85% – 90% for almost all categories, while the sender-based classifier achieves precision and recall rates close to 75%. Table 3 describes balanced precision-recall results and AUC measurements (i.e., Area Under the ROC Curve), for both classifiers. All results presented in the table were averaged over three different runs, each run performed with a different sample of test set from our ground truth data.

category	X-Clusters			Senders		
	precision	recall	AUC	precision	recall	AUC
event	0.90	0.88	0.97	0.54	0.42	0.54
finance	0.85	0.83	0.92	0.78	0.77	0.84
marketing	0.86	0.86	0.95	0.65	0.64	0.73
purchase	0.86	0.85	0.91	0.72	0.72	0.84
service	0.81	0.83	0.95	0.70	0.68	0.74
travel	0.93	0.92	0.95	0.81	0.78	0.77

Table 3: Evaluation for x-clusters and senders classification.

Our classification process achieves results higher by 10%-20% compared to those achieved for the sender-based classification process presented in [17], when running over one week period. The results of the sender-based approach become competitive when relying on data collected over a period of six months, as presented in [17]. This is due to several reasons. First, the strength of the strict structural clustering method differs naturally between personal and machine-generated traffic. Thus, it inherently identifies structures without the need of applying complicated learning processes. In addition, as described previously, the average traffic coverage achieved per x-cluster is higher by orders of magnitude than the coverage achieved per sender. Finally, a sender is less stable in the sense that it might send more than one category of messages, and its behavior might change over time. A specific x-cluster, however, achieves message similarity by definition, due to the structural similarity on which it is based. Thus, features collected on an x-cluster basis are

²https://github.com/JohnLangford/vowpal_wabbit/wiki

³We note that the accuracy of the evaluation is not harmed, since the classification process and collected features are independent from the categories semantics.

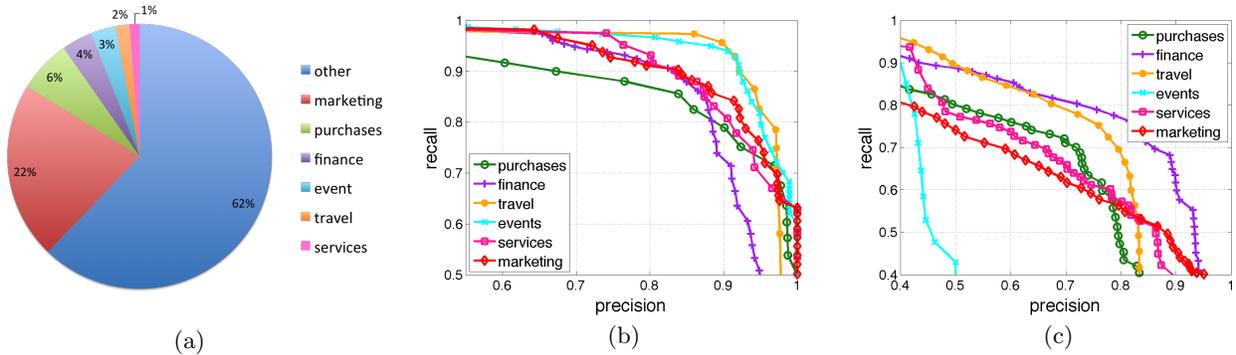


Figure 4: (a) Distribution of x-cluster categories, (b) Precision-recall curve of the x-clusters classification, and (c) Precision-recall curve of the senders classification (using the classification process from [17]).

relatively stable and remain a valuable representation of the x-cluster for a long period of time. This is demonstrated by an experiment conducted over three different weeks taken from three consecutive months. Comparing the classification of the x-clusters in each week, we got that 97% of the classifications remain stable in all three weeks. This stability property is highly important in a production environment, as we only need to collect features for newly arriving x-clusters while relying on previously computed features for existing x-clusters.

We note the large gap between the x-cluster and sender-based results in the *Events* and *Marketing* categories. In both categories, the sender-based clustering does not accurately reflect the generating scripts. In fact, these are categories that cannot be easily assigned to senders (as opposed to x-clusters), exactly due to the misrepresentation cases mentioned in Section 1. The events category typically comprise of highly structured similar messages sent from a large amount of senders. In this case, messages created by the same scripts are sent on behalf of different senders, for example event invitations, calendar events etc. In the marketing category, the opposite effect is observed, where the same sender might send various types of promotions and notifications, differing in both their structure and content.

6. CONCLUSIONS

We presented the first study of clustering machine-generated mail using email message body structure, departing from the previous approaches, which were based on the message header. We studied both strict structural match and flexible structural match in the context of different use cases.

We presented flexible structural match clustering methods adapted for mail extraction applications. These methods create an additional level of abstraction on top of x-clusters, allowing a maintainable and interpretable extraction process. Using large scale, privacy-preserving, experiments over Yahoo mail traffic, we evaluated the quality of our methods. Our experiments precisely identified the level of structural flexibility that should be used in order to achieve both high extraction quality (i.e., a success ratio of about 97%) and a maintainable number of clusters, which represents a reduction by an order of magnitude of the number of produced clusters when compared to strict-matching methods.

We also showed that strict structural match clustering methods retain value in some fully automated tasks, specifically considering the use case of mail classification. We

developed a new classification approach operating over x-clusters and showed that it achieves precision and recall rates between 85% – 90% for almost all categories using a large scale experiment conducted over Yahoo mail traffic. This result implies an increase of 10%-20% compared to previous work. One additional strength of our approach is in the short cycle required for our learning phase. This follows from the fact that strict structural clustering provides stable properties at the x-cluster level, and naturally differs between personal and machine-generated traffic. Specifically, it inherently identifies structures without the need of applying complicated learning processes to extract information from machine-generated traffic.

Summarizing, we demonstrated the strength of structural clustering on mail bodies for tasks involving data mining of machine-generated mail. Our techniques have been deployed in production in Yahoo mail backend, allowing to tackle both advanced tasks such as mail extraction, as well as improving traditional tasks such as mail classification. More ambitiously, in the context of mail extraction, we believe that this work serves as a stepping-stone towards assigning extraction rules to entire clusters of machine-generated messages in a fully automated manner.

7. REFERENCES

- [1] N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: Ranking and clustering. *J. ACM*, 55(5), 2008.
- [2] N. Ailon, Z. S. Karnin, E. Liberty, and Y. Maarek. Threading machine generated email. In *WSDM*, pages 405–414, 2013.
- [3] I. Alberts and D. Forest. Email pragmatics and automatic classification: A study in the organizational context. *JASIST*, 63(5):904–922, 2012.
- [4] A. Arasu and H. Garcia-Molina. Extracting structured data from web pages. In *ACM SIGMOD*, pages 337–348, 2003.
- [5] L. Blanco, N. Dalvi, and A. Machanavajjhala. Highly efficient algorithms for structural clustering of large websites. In *20th WWW*, pages 437–446, 2011.
- [6] A. Z. Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences*, pages 21–29, 1997.
- [7] A. Z. Broder. Identifying and filtering near-duplicate documents. In *11th CPM*, pages 1–10, 2000.

- [8] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60(3):630–659, 2000.
- [9] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. *Computer Networks and ISDN Systems*, 29(8):1157–1166, 1997.
- [10] C.-H. Chang and S.-C. Lui. Iepad: information extraction based on pattern discovery. In *10th WWW*, pages 681–688, 2001.
- [11] F. Chierichetti, N. N. Dalvi, and R. Kumar. Correlation clustering in mapreduce. In *20th KDD*, pages 641–650, 2014.
- [12] W. Cohen. Learning rules that classify e-mail. In *AAAI on Machine Learning in Information Access*, pages 18–25, 1996.
- [13] V. Crescenzi, P. Merialdo, and P. Missier. Clustering web pages based on their structure. *Data & Knowledge Engineering*, 54(3):279–299, 2005.
- [14] N. Dalvi, P. Bohannon, and F. Sha. Robust web extraction: an approach based on a probabilistic tree-edit model. In *ACM SIGMOD*, pages 335–348, 2009.
- [15] D. Di Castro, Z. Karnin, L. Lewin-Eytan, and Y. Maarek. You’ve got mail, and here is what you could do with it!: Analyzing and predicting actions on email messages. In *WSDM’16*, San Francisco, California, USA, 2016.
- [16] D. Di Castro, L. Lewin-Eytan, Y. Maarek, R. Wolff, and E. Zohar. Enforcing k-anonymity in web mail auditing. In *WSDM’16*, San Francisco, California, USA, 2016.
- [17] M. Grbovic, G. Halawi, Z. Karnin, and Y. Maarek. How many folders do you really need? classifying email into a handful of categories. In *CIKM*, 2014.
- [18] Q. Hao, R. Cai, Y. Pang, and L. Zhang. From one tree to a forest: a unified solution for structured web data extraction. In *34th ACM SIGIR*, pages 775–784, 2011.
- [19] N. Heintze. Scalable document fingerprinting. In *USENIX Workshop on Electronic Commerce*, 1996.
- [20] M. Henzinger. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *29th ACM SIGIR*, pages 284–291, 2006.
- [21] M. Kaye and C.-H. Chang. Fivatech: Page-level web data extraction from template pages. *Knowledge and Data Engineering, IEEE Transactions on*, 22(2):249–263, 2010.
- [22] S. Kiritchenko and S. Matwin. Email classification with co-training. In *Conference of the Center for Advanced Studies on Collaborative Research*, pages 301–312, 2011.
- [23] B. Klimt and Y. Yang. The enron corpus: A new dataset for email classification research. In *ECML*, 2004.
- [24] Y. Koren, E. Liberty, Y. Maarek, and R. Sandler. Automatically tagging email by leveraging other users’ folders. In *KDD*, 2011.
- [25] B. Liu, R. Grossman, and Y. Zhai. Mining data records in web pages. In *9th ACM SIGKDD*, pages 601–606, 2003.
- [26] W. Liu, X. Meng, and W. Meng. Vide: A vision-based approach for deep web data extraction. *IEEE Transactions on Knowledge and Data Engineering*, 22(3):447–460, 2010.
- [27] R. Lowrance and R. A. Wagner. An extension of the string-to-string correction problem. *J. ACM*, 22(2):177–183, 1975.
- [28] U. Manber. Finding similar files in a large file system. In *USENIX winter technical conference*, pages 1–10, 1994.
- [29] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*, volume 1. 2008.
- [30] G. Miao, J. Tatemura, W.-P. Hsiung, A. Sawires, and L. E. Moser. Extracting data records from the web using tag path clustering. In *18th WWW*, pages 981–990, 2009.
- [31] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
- [32] H. A. Sleiman and R. Corchuelo. Tex: An efficient and effective unsupervised web information extractor. *Knowledge-Based Systems*, 39:109–123, 2013.
- [33] H. A. Sleiman and R. Corchuelo. Trinity: on using trinary trees for unsupervised web data extraction. *IEEE Transactions on Knowledge and Data Engineering*, 26(6):1544–1556, 2014.
- [34] W. Thamviset and S. Wongthanavas. Information extraction for deep web using repetitive subject pattern. *World Wide Web*, 17(5):1109–1139, 2014.
- [35] E. Ukkonen. Algorithms for approximate string matching. *Information and Control*, 64(1-3):100–118, 1985.
- [36] A. van Zuylen and D. P. Williamson. Deterministic pivoting algorithms for constrained ranking and clustering problems. *Math. Oper. Res.*, 34(3):594–620, 2009.
- [37] R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *J. ACM*, 21(1):168–173, 1974.
- [38] J. Wang and F. H. Lochovsky. Data extraction and label assignment for web databases. In *12th WWW*, pages 187–196, 2003.
- [39] J. B. Wendt, M. Bendersky, L. Garcia-Pueyo, V. Josifovski, B. Miklos, and I. Krka. Hierarchical label propagation and discovery for machine generated email. In *WSDM*, 2016.
- [40] E. Zohar, I. Cidon, and O. O. Mokryn. The power of prediction: Cloud bandwidth and cost reduction. In *ACM SIGCOMM*, volume 41, pages 86–97, 2011.