Automated Extractions for Machine Generated Mail

Dotan Di Castro*, Iftah Gamzu†,Irena Grabovitch-Zuyev*, Liane Lewin-Eytan†,
Abhinav Pundir**, Nil Ratan Sahoo**, Michael Viderman*
*Yahoo Research, Haifa, Israel, {dot,iragz,viderman}@oath.com
**Yahoo Inc., Sunnyvale, CA, USA {abhin,nilratan}@oath.com
†Amazon Research, Haifa, Israel, {iftah,lliane}@amazon.com

ABSTRACT

Mail extraction is a critical task whose objective is to extract valuable data from the content of mail messages. This task is key for many types of applications including re-targeting, mail search, and mail summarization, which utilize the important personal data pieces in mail messages to achieve their objectives. We focus on machine generated traffic, which comprises most of the Web mail traffic today, and use its structured and large-scale repetitive nature to devise a fully automated extraction method. Our solution builds on an advanced structural clustering technique previously presented by some of the authors of this work. The heart of our solution is an offline process that leverages the structural mail-specific characteristics of the clustering, and automatically creates extraction rules that are later applied online for each new arriving message. We provide of a full description of our process, which has been productized in Yahoo mail backend. We complete our work with large-scale experiments carried over real Yahoo mail traffic, and evaluate the performance of our automatic extraction method.

KEYWORDS

Mail Extraction, Automated Extraction, Machine Generated Mail, Mail Clustering

ACM Reference Format:
Dotan Di Castro*, Iftah Gamzu†,Irena Grabovitch-Zuyev*, Liane Lewin-Eytan†, Abhinav Pundir**, Nil Ratan Sahoo**, Michael Viderman*. 2018. Automated Extractions for Machine Generated Mail. In *WWW '18 Companion: The 2018 Web Conference Companion, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3184558.3186582

1 INTRODUCTION

During the last few years, there has been a surge of interest in analyzing machine generated mail. Machine generated messages are commonly created by scripts on behalf of commercial entities or organizations, and comprises more than 90% of non-spam Web mail traffic [2, 21]. There are numerous examples of such messages, including purchase receipts, travel reservations, events and social notifications, and more.

Most of the work that has been done in this context utilizes the special characteristics of this traffic. Arguably, the two main characteristics of such messages are that they are highly-structured, and that similar messages are sent at large scale across the users population. The identical message structure and repetitive parts of messages generated by the same script allow a dedicated analysis, and enable the application of automated data mining and learning methods at scale. Several purposes for such methods have been presented, including mail classification [4, 21, 37], mail anonymization [18], and mail extraction [4].

In this work, we focus on mail extraction, where the goal is to extract valuable information from the body of mail messages. More specifically, we concentrate on developing a fully automated method for identifying and extracting valuable data parts from machine generated traffic. Note that although machine generated messages are created by scripts, they typically include personal data pieces. These are usually the pieces of information we wish to extract. Examples of such might be the item that was purchased, its date of delivery, or the travel details in an itinerary.

Mail extraction is critical for many applications like re-targeting, mail search, and mail summarization used by various user facing features. For example, re-targeting can be optimized using the analysis of users interests based on their transactions; mail search can be improved by properly indexing extracted data to enrich search features, and mail summarization can be better performed after identifying the important parts of messages.

The general task of data extraction has been a major challenge in the Web domain, and many solutions that consider Web documents have been developed. However, these solutions either require types of similarities to which mail messages do not conform (e.g., semantic or statistical term similarities [8–10, 23]), or, as they tend to be rather generic, do not exploit valuable mail-specific characteristics that can greatly improve the quality of extractions (e.g., compare with the Web pages structural methods [6, 14]). There are also continuous efforts to formalize schemes for structured data on the Internet, including web pages, mail messages and others. The well-known *schema.org* defines a full hierarchy of data vocabulary for different mail types (e.g., "FlightReservation", "ParcelDelivery") with the goal of creating a standard that would be supported by major services to power extensible experiences. Despite these efforts, only a small part of mail traffic adheres to this schema[1], and thus, it does not provide sufficient coverage of the traffic.

This research was done while all authors were associated with Yahoo Research, Israel.

[1]Less than 1% of the mail clusters use a valuable schema type. The notion of clusters for machine generated traffic is explained later on.

Mail data extractions can be performed most efficiently at scale when based on a clustering of the traffic, rather than on a single message. The objective in clustering of mail traffic is to group together messages with similar structure and intent, so that extraction rules can be defined and applied for the entire cluster. Intuitively, such a clustering aims to capture the scope of the generating scripts. Different clustering techniques have been proposed in the context of mail, starting from clusters based on the message header [2, 37] to more advanced techniques based on the message structure [4, 18]. Some of the authors of this work have recently introduced new structural clustering methods that can be conducted at different levels of granularity, using strict or flexible matching constraints [4]. That work focuses on clustering and considers the mail extraction task as a primary use case. However, the extraction method employed in that work is based on manually defined rules.

In the current work, we propose a fully-automated extraction solution. The progress that was made in clustering machine generated traffic underlies this new process. We utilize the structural clustering solution [4], and exploit its characteristics in the development of our solution. The heart of our solution is a fully-automated way for creation of extraction rules. The rules are generated in an offline manner per cluster, and are applied online upon an arrival of new mail messages. We present the complete solution, and analyze its performance using both professional editorial evaluation, and an offline extraction evaluation on a real large scale data-set of Yahoo mail service. The editorial evaluation is used to assess the quality of the extraction rules created automatically by our process, while the large scale offline evaluation is used to estimate the quality of the final extraction output. Using both types of evaluation, we derive some insights on the strengths and challenges of information extraction in the Web mail domain.

The rest of this paper is organized as follows. Section 2 covers some related work in the context of information extraction. Section 3 describes the automated mail extraction approach and the main steps of the rules creation process. Section 4 presents our system architecture as productized in Yahoo mail backend, along a traffic analysis. In Section 5, we present our evaluation, focusing on the travel domain, and considering the different phases of our automatic extraction process. Finally, we conclude in Section 6.

2 RELATED WORK

Information extraction is a major research challenge at Web scale. Traditional techniques range from *source-centric*, in which data is extract from an explicit source (like a specific website), to *web-centric*, where data is extracted from the entire Web [5, 11, 19, 20]. A web-centric approach is commonly confined to extracting simple entities (such as tables and lists) as well as some relational data. This approach finds it difficult to attach semantics to entities. As opposed, a source-centric or domain-centric approach, which considers a collection of sources, allows utilization of specific schemas to be populated (e.g., restaurant recommendations), as well as supervision at the domain level [7, 32]. In this sense, information extraction adapted to machine-generated mail traffic is much closer to the domain-centric approach, as it collects data according to domain-specific types. The programs that extract information are called *extractors* or *wrappers*, and the reader is referred to surveys [1, 12] for further reading about these concepts.

The task of information extraction processes online documents which are semi-structured and generated automatically by server-side applications. As such, it usually applies machine learning and pattern mining techniques to exploit the syntactical patterns or layout structures of the template-based documents. In a sense, most techniques seek to separate between the layout and the data. There is an abundant amount of research on extraction techniques, where some of the main ones build on Web page structure [3, 25, 26, 36], pattern recognition [13], tree methods [16, 22], repetitive information identification [33–35], vision techniques [27], and more. These techniques often build on clustering of the underlying documents, focusing mainly on HTML-based pages. Common approaches for identifying document similarity are based on the semantic or statistical term similarity of documents (see, e.g., [28] for an introduction of related techniques), or on methods for document reduction and canonical sequence representation [8, 10, 23]). 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 has been given a lot of consideration in the past [6, 14, 31]. However, those techniques are not dedicated to the mail domain, and therefore, do not exploit meaningful mail signals.

An important part of our automatic extraction process comprises of the identification and annotation of different information parts. Example of such are names, places, products, and more. Generally, identification of entities in documents is known as name entity recognition [30]. Some major methods in this field involve databases and statistical models, while others use machine learning or linguistic grammar-based techniques in order to identify entities using parts of speech. There are few papers that consider entity recognition in the context of informal documents such as mail messages, like the research on contacts information extraction [15], and the work of [29] that uses NLP methods to identify names. These works are not targeted to the mail domain, nor to machine-generated traffic. As a result, they do not make use of the ensemble of documents as provided by the clustering of this traffic and of the statistical power it provides. Our method allows treating any set of entities that are present in such traffic using a modular approach. This is partly achieved as our process is based on the horizontal analysis of each cluster of messages, rather than on a single message.

The only work we are aware of, which treats a similar problem as ours, is the work of Zhang et al. [38] that considers the extraction of structured data from emails. Our work can be seen as complementary to their work in several aspects. Zhang et al. [38] describes a general approach for extracting and annotating product names. It concentrates on the learning algorithm and optimization behind the annotation process, rather than on the implementation details and the deployment of the approach. For example, their extraction process is based on abstracted entities (clusters and templates), but specific details about their computation or level of abstraction are not provided, making the approach irreproducible. One contribution of our work lies in the description of the complete solution, which is based on a known notion of structural clustering. As part of our solution, we precisely describe and analyze how the structural characteristics of these cluster entities are processed and exploited for the creation of extraction rules. Furthermore, in [38], a separate learning model is computed per template, while in the current work, we compute one generic model which works for the entire set of clusters belonging to a specific category (e.g. travel, purchases, etc.). Note that the number of clusters is already in the low millions for travel-related domains. As scale is a crucial issue in a system processing email traffic, this is yet another key difference between the two works. This difference is also reflected in the traffic coverage of the experiments; Zhang et al. [38] considers about 20K messages (associated with 20 templates), while our work considers 10M messages (associated with 36K clusters). A final difference is that Zhang et al. focuses on the purchases domain while ours take the travel domain as an exemplary use case. These are indeed two different important business-consumer classes. Both of them have some support in all major Web mail services (e.g., the *Bulk Senders* category in AOL mail, the semantic *Bundles* in Inbox for Gmail, and the *Smart Views* in Yahoo mail).

3 THE MAIL EXTRACTION PROCESS

The mail extraction process comprises two parts, as presented in Figure 1. The first part is an offline process during which extraction rules are automatically created. The second part is an online process where upon the arrival of a new message, the appropriate rules are identified and applied, resulting in extracted information.

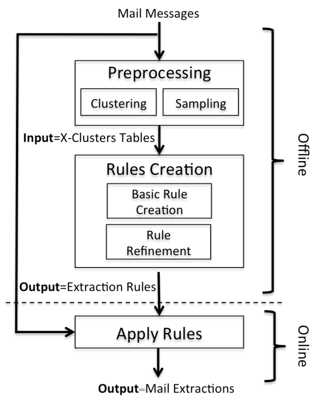

Figure 1: Flow logic of automatic mail extraction process.

We focus on the automatic offline process.Our approach is modular, and can be applied to any class of mail messages such as travel, social or finance. The steps of the process that require adaptation to a specific class are mentioned along the section.

3.1 Preliminaries

An essential and first building block in our approach is a clustering method that groups together machine generated emails having the exact same structure. This method is known as *x-clustering* [18]. Without delving into technicalities, the method represents each message as an ordered list of all the XPath expressions in its DOM tree, and then applies a so-called Mail-Hash signature to that list. This implies a clustering of messages according to their resulting (structure-based) signature. The generated clusters are referred to as XPath clusters, or *x-clusters*.

Importantly, this structure-based clustering enables us to easily collect multiple (different) message instances that result from the same generative script. We process and analyze those messages to identify the concrete locations that hold information of interest, that is, information to be extracted. Typically, this kind of information is present in *variable XPaths*, which are XPaths whose values vary between messages in the same cluster. Those variable data pieces commonly hold the most valuable information of the message. Xpaths that hold fixed information over the entire cluster are called *constant XPaths*, and are typically of low value. Subsequently, we define rules that can extract variable information and annotate it. The resulting extraction rules are then used to extract data online, as mentioned earlier. Specifically, once a message arrives to the mail system, its x-cluster signature is computed, and the associated extraction rules are retrieved and applied to the message.

3.2 Overview: Automatic Rules Creation

Given a large corpus of email messages collected over several (not necessarily consecutive) days, our process begins by grouping together messages into x-clusters. Then, it processes each x-cluster independently to create extraction rules per cluster. The process applied to the x-clusters is described below.

Input & Output Given an x-cluster, we gather a random sample of email messages that are associated with the cluster, namely, messages having the exact same structure. We require the sample size to be sufficiently large, usually around several dozens of messages (parameterized by a threshold Γ), to allow effective and precise identification of variable data. Note that the difference between constant and variable data is sometimes delicate and far from being absolute. For example, there are sometimes variations in what we consider constant data to account for identical message structure under different languages. This technical issue is of secondary importance, and therefore, we neglect it from the current discussion.

It is instructive to represent the sample data as a table whose rows correspond to different messages while its columns correspond to different XPaths. Such a tabular representation is called an *x-cluster table*, an example of which is presented in Table 1. Note that some of the XPaths in this example are variable (e.g., XPath_1, XPath_4, and XPath_7), while others are constant (e.g., XPath_2, XPath_3, and XPath_6). In particular, note that XPaths that consist of both constant and variable sub-parts (like, XPath_1) are regarded as variable. Those mixed XPaths are usually the most challenging for defining extraction rules. In the remainder of this work, we concentrate only on extraction rules for variable XPaths.

As already noted, the output of the process are extraction rules for the underlying x-cluster. Each extraction rule is associated with some XPath of the x-cluster. At the most intuitive level, a rule provides a description of the way to extract variable data pieces from the XPath, and suggests an *annotation* for each of those pieces. An annotation is chosen from a pre-defined set of information types to be extracted, e.g., <passenger-name>, <confirmation-code>, <date-depart>. Figure 2 shows an example of an extraction rule that was generated for a specific XPath, based on a set of values collected

	XPath$_1$	XPath$_2$	XPath$_3$	XPath$_4$	XPath$_5$	XPath$_6$	XPath$_7$	...
msg 1	Thank you **John**!,	Below is ...	Upcoming Trip:	**07/08/16 - Ontario**	...	AIR Confi...	**ABC123**	...
msg 2	Thank you **Arya**!,	Below is ...	Upcoming Trip:	**06/09/16 - New York**	...	AIR Confi...	**Z5T8Q2**	...
msg 3	Thank you **Khal**!,	Below is ...	Upcoming Trip:	**23/08/16 - Boston**	...	AIR Confi...	**ER46T1**	...
msg 4	Thank you **Lucy**!,	Below is ...	Upcoming Trip:	**13/08/16 - Seattle**	...	AIR Confi...	**61FXC1**	...
...	...	...	...	...	...	...	...	...

Table 1: An example of an input table for the rules creation process.

from different messages. In this case, constant (repeating) sub-parts are identified, while variable sub-parts are given a placeholder and annotation. The concrete way by which this rule can be utilized for future extractions is a matter of implementation. One possibility is to interpret this rule as a regular expression by replacing the placeholders with regexes that can match the XPath values.

3.3 Step 1: Basic Rule Creation

Our first objective is to create a basic rule that identifies constant and variable sub-parts within an XPath. We start by performing a first round of annotations of the tokens in the sample set. This first round is based on *light annotations* that are general and independent of the context. For instance, we may use a date annotation without a concrete contextual interpretation indicating that the date corresponds to a flight arrival, package delivery, service expiration, or else. Essentially, this views the set of annotations as a two-level hierarchy in which first level annotations are general (e.g., <date>), and second level annotations are refined (e.g, <date-arrive> for a flight). By replacing tokens with annotations, we are able to identify popular tokens that are part of variable XPaths, and are not supposed to be considered constants. For example, for an XPath with a large number of samples referring to SFO airport, the token "SFO" would be annotated as <airport> and identified as valuable, even if appearing in most of the samples.

The task of annotating tokens clearly depends on the set of supported annotations and the definition of the information that ought to be extracted. Many techniques can be used here. Examples range from brute-force dictionary search and regular expression patterns matching, to more involved machine learning based classification procedures. A dictionary comprises a set of words related to a specific subject. For instance, one can build a dictionary of personal names. Given a dictionary and a term (token), we search the dictionary, and replace any identified token with a respective annotation. Consider the text "`Hello Jack, Thank you for flying with us!`". Using the dictionary method, this sentence will be annotated as "`Hello <name>, Thank you for flying with us!`". Similarly, one can define regular expressions that identify annotations such as date or time. For example, given the text "`Departing at 9:00 AM (PST)`", if we use time-related regexes, the sentence will be annotated as "`Departing at <time>`". In addition, it is likely one would need build more involved classification procedures for complicated annotation tasks such as identification of product names. The specific annotation types used for flights are specified in Table 3.

We note that a given technique will likely not be accurate enough to identify all tokens in all sampled phrases. Since our sample is large enough and chosen at random from the cluster, we are still expected to identify many (if not most) of the tokens. The approach described relies on the strength of the clustering phase, allowing successful annotations over a large number of similar messages, but not necessarily over the entire sample. This way, patterns can be identified even if for some cases the annotation fails.

3.4 Step 2: Rule Refinement

In the second step, we refine the light annotations from the first step to include contextual meaning. For example, we'd like to understand whether an airport code stands for the departure airport or the arrival airport in a flight itinerary. This requires departing from the local XPath view, and considering connections between multiple XPaths, sometimes within the entire x-cluster. As annotations take context into consideration, the supported annotations are specifically tailored to the use case, e.g., annotations for flights would be different than for purchases. A date token in an itinerary may stand for a departure or arrival date, while it may represent the expected delivery date when the message is a shipping notification. We assume to know the classification of the underlying x-cluster, which can be generated using known techniques [4].

Our approach is based on learning the context of each annotation by the use of machine learning. Given an x-cluster table of cluster c, we associate each light annotation of a variable XPath XP_i with a vector of features. These features are naturally based on the characteristics of XP_i, but also on the other XPaths in the x-cluster, i.e., $\{XP_k\}$ where $k = 1, \ldots, i-1, i+1, \ldots K$, K being the number of XPaths in c. The features used for can be divided into two main types: one includes the light annotations, and the other includes indicative words. For example, in the travel domain, light annotations could be <time>, <airport> or <name>, while indicative words could be "passenger", "depart" or "arrive". The collection of inductive words are identified using both human experts and automatic statistical techniques (e.g., tf-idf weighting schemes). The annotations and words considered relevant for some XP_i are those that can be found at its proximity. Proximity is defined in two ways. The first is the distance between XPaths according to the DOM tree representation of c. The second way by which proximity is defined is based on the visualization of the cluster's structure and uses the CSS[2] of the HTML. It can be specifically tailored to HTML tables, where a feature is associated with the relative position of its table entry with respect to XP_i.

Based on these features, we can classify each light annotation to a contextual annotation. We follow standard machine learning procedures for training the classifier, and use Random Forest [24] after carefully comparing with other classifiers. Details regarding the train and test sets used for the classifier are presented in Section 5.

[2]Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in HTML or any other markup language.

Check-in is open for **TP438**, departing from **Lisbon** (**LIS**) to **Paris** (**CDG**), on **09JAN16** at **16:20**.
Check-in is open for **TP954**, departing from **Lisbon** (**LIS**) to **Geneva** (**GVA**), on **06JAN16** at **19:25**.
Check-in is open for **TP569**, departing from **Hamburg** (**HAM**) to **Lisbon** (**LIS**), on **05JAN16** at **12:45**.
...

$\downarrow$

Check-in is open for <**flight-code**>, departing from <**place-depart**> (<**airport-depart**>) to <**place-arrive**> (<**airport-arrive**>), on <**date-depart**> at <**time-depart**>.

Figure 2: Example of an extraction rule.

4 SYSTEM & TRAFFIC ANALYSIS

The mail extraction process has been productized in Yahoo mail backend. We provide details regarding its deployment along with some traffic analysis, focusing on flight itineraries use case.

4.1 Deployment

The architecture of the mail extraction system, as deployed in Yahoo mail backend, is described in Figure 3. The classification, clustering, sampling and rule creation modules are all part of the offline process, leading to the creation of the extraction rules. The rules creation module is the heart of our process, and is fully covered in Section 3. Sampling is performed for clusters of predefined classes that are supported by the extraction process (e.g. flights, coupons, purchases, etc.) as annotations are specific for each use case. The classification is attained using the method described in [4]. Thus, sampling is based on the output of the classification and clustering modules. The output of the sampling module are x-cluster tables for all x-clusters that have enough samples, based on the threshold Γ set in the system. The concrete value of Γ depends on the desired level of trade-off between quality and coverage, as discussed later on. The rules, the samples, and all information related to their respective x-clusters and annotations are stored in the *Rules Management Framework* (RMF). The RMF has a UI interface that allows human intervention for quality assurance purposes, or modification of certain rules in case of need. The online process is applied over the incoming messages stream. For each message, its x-cluster is first computed. Then, the respective rule is fetched from the RMF and applied to the message. Finally, the extraction data is stored in a dedicated storage, to be further used by various applications.

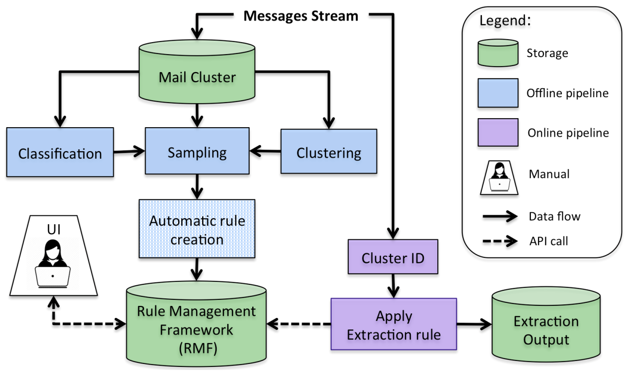

Figure 3: The architecture of the mail extraction system.

The online extraction process is extremely light-weight by design. When it is applied to a message under consideration it is essentially instantaneous, that is, it takes only few milliseconds to complete in most cases. The more involved computational part of our approach is the offline rules creation process. This process is implemented in Hadoop MapReduce [17]. This enables creating the extraction rules efficiently in parallel as an extraction rule for one x-clusters is independent of the rules generated for other x-clusters. The most computation-intensive step in our approach is the random sampling which requires going over the entire inbound mail corpus during some time period. Providing concrete time measure for each of the modules seems redundant as they depend on the number of messages that are taken into consideration, the number of computational units used in the parallel processing, additional load on the MapReduce system, and more. Still, we note that the time consumed by the automatic rules creation module is smaller by roughly an order of magnitude than the time of the preliminary steps, i.e., clustering, classification and sampling. For example, when the preliminary steps take few hours, the rules creation takes only few minutes.

4.2 Traffic Analysis

We analyze the characteristics of the traffic corresponding to flight itineraries with respect to the phases preceding the rules creation, namely, classification, clustering and sampling. The analysis was performed over one week of traffic during April 2017, leading to about 500 domains that contained x-clusters classified as flight itineraries. We first look at the distribution of traffic and x-clusters per domain, presented in Figure 4. The domains in the x-axis are ordered according to decreasing number of x-clusters in each domain. Thus, the curve descending exponentially corresponds to the distribution of x-clusters number per domain. The second curve corresponds to the distribution of the traffic (number of messages) per domain, and is represented as percentage of the overall traffic classified as flight itineraries (that is, overall number of messages belonging to x-clusters classified as itineraries). The top-10 domains with respect to the number of x-clusters are listed in Table 2. For each domain, we specify the number of x-clusters and traffic coverage. As can be seen, there is a long tail of flight itinerary domains, where the number of x-clusters varies between only dozens, to hundreds or thousands, up to above 60K. It can also be seen that there is no correlation between this number and the traffic coverage of a domain. This leads to the understanding that there is a need for a robust method that can handle different 'types' of domains, and can reach high coverage and extraction quality with large as

Domain	x-clusters	coverage
united.com	67,835	4.37%
e.delta.com	34,804	5.68%
luv.southwest.com	30,926	8.02%
amadeus.com	22,169	2.01%
aa.globalnotifications.com	20,532	1.74%
cheapoair.com	16,473	0.73%
expediamail.com	12,638	1.88%
t.spiritairlines.com	12,332	4.70%
cs.tuiuk.com	10,504	0.45%
alaskaair.com	8,284	0.62%

Table 2: Top-10 flight domains wrt. x-clusters number.

well as small amounts of traffic per domain and x-cluster. This is further emphasized in the next figure as well as in Section 5.3.

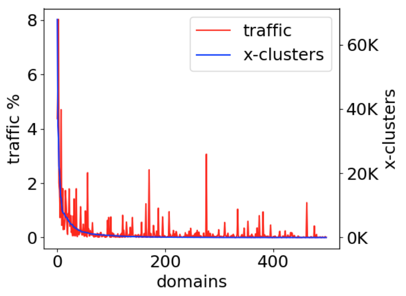

Figure 4: Domains x-clusters number and traffic.

Figure 5 exhibits the coverage obtained for different threshold values for the sample size. That is, it represents the relative coverage percentage when taking into consideration only x-clusters that have more than Γ messages. The three curves in the figures correspond to coverage in terms of number of x-clusters, number of mail messages, and number of domains. It is important to note that while the number of covered x-clusters drops drastically for values of $\Gamma \geq 20$, the coverage in terms of number of messages and domains remain close to or above 80%. This affirms the strength of our solution, which allows a relatively high coverage for a wide range of threshold values. Remark that for small values of Γ, the coverage is higher, but the small number of samples might negatively influence the quality of the extractions. Notably, our solution can be easily adjusted to optimize between quality and coverage. This trade-off is further explored in Section 5.3.

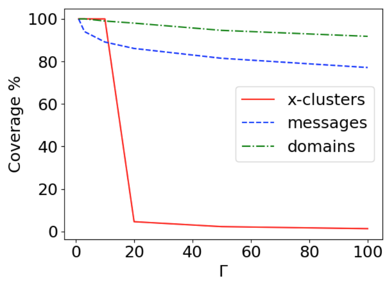

Figure 5: Trade-off between sample size and coverage.

5 EXPERIMENTAL EVALUATION

We demonstrate the value of our approach through an offline evaluation based on Yahoo Web mail traffic. As in Section 4, we adapt our general approach, detailed in Section 3, to the travel domain, and more specifically, to flight itineraries. We divide our evaluation into two parts. In the first part (Section 5.1), we evaluate the extraction rules, while in the second part (Section 5.2) we consider our entire extraction process and evaluate its final output. The first part is further partitioned into two sub-parts, one for the evaluation of the basic rules, and the other for the evaluation of the refined rules.

The first evaluation part is based on a manually labeled set, allowing us to evaluate independently each phase of the rules creation. Conversely, the second part evaluates the final output of the end-to-end process, resulting from the consecutive application of all phases described over real Yahoo mail traffic at large scale. We further note that when considering the rules, the evaluation is performed at the XPath level, and when considering the output of the extractions, the evaluation is performed at the message level.

5.1 Evaluation of the Extraction Rules

5.1.1 Dataset. The evaluation of this phase is based on a manually labeled set, referred to as *golden set*. The golden set was created by selecting 850 x-clusters belonging to 60 different domains that were classified as flights.

For each x-cluster, we presented three mail samples to professional quality assurance staff, referred to as *editors*, for manual labeling. We note that this process was conducted under full privacy preservation in accordance with Yahoo privacy policy. The manual labeling was performed at the XPath level. For each XPath, editors selected labels from a predefined set of contextual annotations as presented in Table 4. Note that an XPath could be associated with more than a single label in case its value contains more than one annotation, e.g., an XPath containing both an airport name and a place. For constant XPaths, no labels were provided.

5.1.2 Evaluation of the Basic Rules. We built several annotators for identifying light annotations for flight itineraries. Those annotators were computed using the two approaches presented in section 3.3, that is, dictionaries and regular expression patterns. We evaluated the basic (light annotation) rules as follows. For each XPath, we compared the light annotations resulting from the manual labeling, to the light annotation assigned by our automated method. The results are presented in Table 3 in terms of precision, recall and *F*-measure (F_1-score).

Looking at the results, one can see that almost all light annotations achieve precision and recall above 90%. The single caveat is the recall of the place annotator. As it turns out, places have high variability both in absolute number and different ways to spell the (essentially) same place. Any dictionary-based annotator has high-dependency on the richness of the dictionary and its coverage of unpopular terms. In accordance, extending the places dictionary that underlies this annotator and adding more advanced canonization techniques are likely to significantly improve this metric.

5.1.3 Evaluation of the Refined Rules. In order to evaluate the rules refinement step, without any dependency on the preceding step of basic rules creation, we trained our model on the light

Annotation	Type	Precision	Recall	F-score
confirmation-code	Regex	100%	96%	0.98
airport	Dict.	97%	87%	0.92
date	Regex	96%	92%	0.95
name	Dict.	92%	90%	0.91
place	Dict.	98%	77%	0.86
time	Regex	99%	94%	0.97

Table 3: Evaluation of light annotations for flights.

annotations obtained from the manual labeling of the golden set. We trained a model for the classification of each type of light annotation into its optional contextual annotations, as described in Section 3.4. Our labeled set consisted of about 17K anonymized XPaths[3]. We applied cross-validation on the data with 5-folds in order to assess the classifiers performance. The evaluation results are presented in Table 4. As can be seen, the precision and recall of most refined annotations are above 90%.

Annotation	Precision	Recall	F-score
airport-depart	91%	96%	0.93
airport-arrive	95%	89%	0.92
date-booking	100%	78%	0.87
date-flight	90%	99%	0.94
name-passenger	97%	99%	0.98
name-user	96%	86%	0.91
place-depart	94%	95%	0.94
place-arrive	94%	93%	0.94
time-depart	91%	94%	0.93
time-arrive	93%	90%	0.91

Table 4: Evaluation of contextual annotations for flights.

5.2 Evaluation of the Extraction Output

We evaluate the performance of the entire automatic extraction process by comparing its output to the output of the extraction process presented in [4]. This latter process was based on manually-defined rules, and hence, we refer to it as the *manual rules* process[4]. The process was applied to 10M messages received in December 2016 from 100 flight domains, and associated with 36K x-clusters. The intersection between those domains and the domains of the golden set described in Section 5.1 leads to 40 domains, allowing the validation of our process on domains that were not necessarily part of the training set.

As a first step, we gathered samples for the 36K x-clusters during a two weeks period in December 2016. Extraction rules were created using our automated process for each of those x-clusters. The rules created by our process were applied to the same traffic as the manual rules process. Note that we ignored x-clusters for which we could not gather enough samples, as defined by the threshold $\Gamma = 50$.

The results of the comparison between the output of our automatic process and the output of the manual rules process are presented in Table 5. Here, the comparison was performed over

Annotation	Precision	Recall	F-score
airport	94%	94%	0.94
airport-depart	85%	91%	0.88
airport-arrive	88%	89%	0.88
confirmation-code	88%	88%	0.88
date	82%	96%	0.88
date-flight	82%	96%	0.88
name	90%	93%	0.91
name-passenger	75%	78%	0.76
place	94%	92%	0.93
place-depart	78%	84%	0.81
place-arrive	86%	88%	0.87
time	94%	96%	0.95
time-depart	91%	95%	0.93
time-arrive	73%	73%	0.73

Table 5: Evaluation of entire extraction process for flights.

the set of extractions per message. An extraction is considered successful if the output strings match between the two processes and get the same annotation. The results are computed per type of annotation, considering both light and contextual annotations. As expected, the precision and recall for the entire process are usually lower compared to its rules creation sub-parts, and mostly range between 80%-90% for both precision and recall. Note that we present light as well as contextual annotations, even if considering only one type of contextual annotation (e.g., date and date-flight, where date-flight comprises a subset of date). This provides further understanding of the process, since for both annotation types, we now evaluate the quality of the actual output, rather than the quality of the rules as presented previously.

While our method could be further enhanced for specific annotations, there are other less obvious reasons for the performance decrease that ought to be discussed. First and foremost, the manual rules process is far from being perfect. It is noisy and not all its extractions are successful, and thus, it only approximates the ground truth and cannot be considered as truly clean. It is important to note that failures in the manual process translate to (false) failures accounted for our automatic process since our process usually does not make the same errors as the manual process. Second, our process was specifically tuned for samples in English, and therefore, had decreased performance metrics for messages in other languages. Lastly, our process was influenced by a skewness of the data that resulted in a decreased performance of the classifier for arrival time.

5.3 Trade-off Between Quality and Coverage

We investigate the trade-off between the quality of our process and the traffic coverage. Both measures rely primarily on the threshold Γ, which defines the minimal size of an x-cluster, that is, the minimal number of samples required for creating extraction rules for an x-cluster. This value influences the quality of the light annotations step, as it is based on the horizontal analysis of each x-cluster and XPath, according to their samples. Conversely, the step of contextual annotations is based on learning, where for each annotation, training examples are gathered from our entire set of samples for all x-clusters, and thus does not rely on Γ. As demonstrated in Section 4.2, the threshold Γ has also a direct influence on the coverage we

[3]We emphasize that all the experiments reported here have been conducted on fully anonymized data, in accordance with Yahoo strict privacy policy.

[4]A note is added to avoid confusion between two manually created sets: (1) the golden set used for training the contextual annotation model, as well as for the evaluation in Section 5.1; and (2) the manual process presented in [4] used for creating extraction rules, against which our automatic end-to-end process is evaluated.

can achieve, as extractions are not performed for x-clusters that do not meet the threshold.

Figure 6 presents the F-score of the light annotation step in our process for different values of Γ when the process is applied to the mail data from the previous section. Recall that this mail corpus consisted of about 10M messages. As can be seen, the higher the value of Γ, the higher the quality of the annotations. This is a natural outcome, as the more samples we have per x-cluster, the more values we have per XPath, and the more precise is our process. For example, annotations relying on dictionaries (airport, name, place) benefit from a large sample since unpopular terms have smaller influence. One can also observe that the improvement in quality becomes smaller when Γ is larger than 20. This demonstrates that our process does not require a high number of samples per x-cluster to achieve high quality.

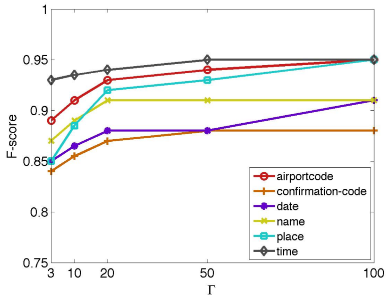

Figure 6: Extraction quality as a function of Γ.

6 CONCLUSIONS

We presented a fully-automated approach for mail extraction of machine generated messages. We leveraged a structural clustering method for messages, and devised a modular automated method for creation of extraction rules at the cluster level. We presented the entire solution along its architecture, performance considerations, and an analysis of the traffic related to its different phases.

Then, we performed thorough offline experiments for evaluating our automated extraction method over real Yahoo mail traffic, considering both the quality of the extraction rules as well as the quality of the entire extraction pipeline. We adapted our solution to the travel domain, and specifically to flight itineraries, and provided clear evaluation. For the rules evaluation, we achieved a performance of above 90% in precision and recall, while for the extraction output evaluation, our numbers mostly range between 80%-90%. We further discussed some data insights and ideas for improvement. We believe that this work is an important step in the field of information extraction in the Web mail domain, and more specifically, for the task of reaching fully-automated mail extractions, adapted to the large scale of this domain.

REFERENCES

[1] C. C. Aggarwal and C. X. Zhai. *Mining Text Data.* Springer Publishing Company, Incorporated, 2012.
[2] N. Ailon, Z. S. Karnin, E. Liberty, and Y. Maarek. Threading machine generated email. In *WSDM*, pages 405–414, 2013.
[3] A. Arasu and H. Garcia-Molina. Extracting structured data from web pages. In *ACM SIGMOD*, pages 337–348, 2003.
[4] N. Avigdor-Elgrabli, M. Cwalinski, D. D. Castro, I. Gamzu, I. Grabovitch-Zuyev, L. Lewin-Eytan, and Y. Maarek. Structural clustering of machine-generated mail. In *CIKM*, pages 217–226, 2016.
[5] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the web. In *20th IJCAI*, pages 2670–2676, 2007.
[6] L. Blanco, N. Dalvi, and A. Machanavajjhala. Highly efficient algorithms for structural clustering of large websites. In *20th WWW*, pages 437–446, 2011.
[7] P. Bohannon, N. Dalvi, Y. Filmus, N. Jacoby, S. Keerthi, and A. Kirpal. Automatic web-scale information extraction. In *ACM SIGMOD*, pages 609–612, 2012.
[8] A. Z. Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences*, pages 21–29, 1997.
[9] A. Z. Broder. Identifying and filtering near-duplicate documents. In *11th CPM*, pages 1–10, 2000.
[10] 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.
[11] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. Webtables: Exploring the power of tables on the web. *Proc. VLDB Endow.*, 1(1):538–549, 2008.
[12] C.-H. Chang, M. Kayed, M. R. Girgis, and K. F. Shaalan. A survey of web information extraction systems. *IEEE Trans. on Knowl. and Data Eng.*, 18(10):1411–1428, Oct. 2006.
[13] C.-H. Chang and S.-C. Lui. Iepad: information extraction based on pattern discovery. In *10th WWW*, pages 681–688, 2001.
[14] V. Crescenzi, P. Merialdo, and P. Missier. Clustering web pages based on their structure. *Data & Knowledge Engineering*, 54(3):279–299, 2005.
[15] A. Culotta, R. Bekkerman, and A.McCallum. Extracting social networks and contact information from email and the Web. In *CEAS*, 2004.
[16] 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.
[17] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, 2008.
[18] D. Di Castro, L. Lewin-Eytan, Y. Maarek, R. Wolff, and E. Zohar. Enforcing k-anonymity in web mail auditing. In *WSDM*, 2016.
[19] H. Elmeleegy, J. Madhavan, and A. Halevy. Harvesting relational tables from lists on the web. *The VLDB Journal*, 20(2):209–226, 2011.
[20] W. Gatterbauer, P. Bohunsky, M. Herzog, B. Krüpl, and B. Pollak. Towards domain-independent information extraction from web tables. In *16th WWW*, pages 71–80, 2007.
[21] 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.
[22] 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.
[23] M. Henzinger. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *29th ACM SIGIR*, pages 284–291, 2006.
[24] T. K. Ho. Random decision forests. In *Document Analysis and Recognition, 1995*, volume 1, pages 278–282. IEEE, 1995.
[25] M. Kayed 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.
[26] B. Liu, R. Grossman, and Y. Zhai. Mining data records in web pages. In *9th ACM SIGKDD*, pages 601–606, 2003.
[27] 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.
[28] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*, volume 1. 2008.
[29] E. Minkov, R. C. Wang, and W. W. Cohen. Extracting personal names from email: applying named entity recognition to informal text. In *HLT/EMNLP*, pages 443–450, 2005.
[30] D. Nadeau and S. Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.
[31] A. Nierman and H. Jagadish. Evaluating structural similarity in xml documents. In *WebDB*, volume 2, pages 61–66, 2002.
[32] P. Senellart, A. Mittal, D. Muschick, R. Gilleron, and M. Tommasi. Automatic wrapper induction from hidden-web sources with domain knowledge. In *10th WIDM*, pages 9–16, 2008.
[33] H. A. Sleiman and R. Corchuelo. Tex: An efficient and effective unsupervised web information extractor. *Knowledge-Based Systems*, 39:109–123, 2013.
[34] 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.
[35] W. Thamviset and S. Wongthanavasu. Information extraction for deep web using repetitive subject pattern. *World Wide Web*, pages 1109–1139, 2014.
[36] J. Wang and F. H. Lochovsky. Data extraction and label assignment for web databases. In *12th WWW*, pages 187–196, 2003.
[37] 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.
[38] W. Zhang, A. Ahmed, J. Yang, V. Josifovski, and A. J. Smola. Annotating needles in the haystack without looking: Product information extraction from emails. In *21th ACM SIGKDD*, pages 2257–2266, 2015.