Web-based Closed-Domain Data Extraction on Online Advertisements

Maria S. Pera Rani Qumsiyeh Yiu-Kai Ng*

Computer Science Department
Brigham Young University
Provo, Utah 84602, U.S.A.

Abstract

Taking advantage of the popularity of the web, online marketplaces such as Ebay (.com), advertisements (ads for short) websites such as Craigslist(.org), and commercial websites such as Carmax(.com) (allow users to) post ads on a variety of products and services. Instead of browsing through numerous websites to locate ads of interest, web users would benefit from the existence of a single, fully integrated database (DB) with ads in multiple domains, such as Cars-for-Sale and Job-Postings, populated from various online sources so that ads of interest could be retrieved at a centralized site. Since existing ads websites impose their own structures and formats for storing and accessing ads, generating a uniform, integrated ads repository is not a trivial task. The challenges include (i) identifying ads domains, (ii) dealing with the diversity in structures of ads in various ads domains, and (iii) analyzing data with different meanings in each ads domain. To handle these problems, we introduce $ADEx$, a tool that relies on various machine learning approaches to automate the process of extracting (un-/semi-/fully-structured) data from online ads to create ads records archived in an underlying DB through *domain classification*, *keyword tagging*, and *identification* of valid attribute values. Experimental results generated using a dataset of 18,000 online ads originated from Craigslist, Ebay, and KSL(.com) show that $ADEx$ is superior in performance compared with existing text classification, keyword labeling, and data extraction approaches. Further evaluations verify that $ADEx$ either outperforms or performs at least as good as current state-of-the-art information extractors in mapping data from unstructured or (semi-)structured sources into DB records.

Keywords: Data extraction, classification, keyword tagging, advertisement

1 Introduction

The web is a perfect publication forum for advertisements (ads for short), since ads websites (allow sellers to) post ads for potential buyers worldwide who can freely access archived and newly-created ads anytime and anywhere, which cannot be provided by any traditional

*Corresponding Author; Contact Email: ng@compsci.byu.edu

publication media. According to a report from eMarketer(.com), online advertising surpassed newspaper marketing in 2010 and the margin is widened in 2011, which indicates that online ads are popular and proliferating. Even though these days (online) information access has gone through evolutionary changes, most of the tools employed for accessing ads information still rely on an underlying database (DB) to maintain ads records. We recognize that existing ads information providers employ their own ads structures and formats for information processing. As a result, web users are forced to access ads archived at individual websites using a variety of searching tools provided by each website to look for ads of interest. A *unified* framework that integrates online ads available from various sources into a single underlying DB should facilitate the process of querying, question answering, and performing various data mining tasks on ads data. Creating an underlying DB from multiple sources, however, is a non-trivial task due to the diversity in the formats and contents of ads in various domains, a problem that we address and solve in this paper.

We introduce $ADEx$, a machine learning-based tool that automatically extracts and populates ads available at various sources into a unified DB. During the extraction process, $ADEx$ employs effective supervised learning approaches to (i) *categorize* ads according to their *domains*, (ii) *label* non-stop keywords[1] in classified ads based on their *types* such that the essential keywords are either (a) unique identifiers of a product/service P in an ad, (b) properties of P, or (c) qualitative values associated with P, which facilitates the process of identifying valid attribute values in each ad, and (iii) populate the tagged keywords as attribute values in an underlying DB. To populate various ads regardless of their structures/formats and ads domains to which they belong, $ADEx$ analyzes, filters, and extracts (ir)relevant data from online ads using easy-to-implement algorithms to generate a single, unified source of information on ads data.

$ADEx$ advances the current data extraction techniques. Unlike most of the existing information extraction approaches, it extracts data from un-/semi-/fully-structured ads data sources without altering its design. $ADEx$ generalizes the data extraction process by labeling keywords in ads based on their *types*, as opposed to relying on domain-specific vocabularies or ontologies to identify keywords associated with DB attributes which are only applicable to the respective domains. Conducted empirical studies (see Section 4) have verified that $ADEx$ is highly effective in automating the process of populating a DB with online ads from multiple sources.

$ADEx$ is unique, since it (i) provides a unified tagging framework using its own *type definitions* on ads, (ii) develops an elegant set of empirically-verified *features* to distinguish essential from useless data in ads, and (iii) introduces an optimal and effective approach which *combines* the tagging and extraction mechanisms (based on support vector machines and decision trees, respectively) to accurately populate the underlying DB. Moreover, $ADEx$ either outperforms or is comparable with the current state-of-the-art data extraction tools.

The remaining of this paper is organized as follows. In Section 2, we discuss previous work on data extraction, a task performed by $ADEx$. In Section 3, we detail the design of

[1] *Stopwords* are words with little meaning, such as articles, prepositions, and conjunctions, which often do not represent the content of an ad.

$ADEx$. In Section 4, we present the empirical study conducted for verifying the effectiveness of $ADEx$ in populating ads and compare its performance with other state-of-the-art data extraction approaches. In Section 5, we give a conclusion and discuss future work.

2 Related Work

In this section, we discuss recently proposed methodologies for extracting unstructured or (semi-)structured data from online data sources that are closely related to the design of $ADEx$.

A number of supervised learning approaches [14, 15, 22] have been developed for extracting data from online sources. The machine learning approach in [14] extracts labeled attributes from web form interfaces. It matches a form element, which is an identified value type in our case, with its corresponding textual description, which is a keyword in an ad in our case, by using classifiers that label form elements through learned structural patterns on the form. $ADEx$, which relies solely on pre-defined features that are independent of ads domains, avoids learning structural patterns for identifying attribute values in ads to be extracted. Similar to $ADEx$, which first labels keywords in online ads and then extracts labeled data, the tool proposed by Zhu et al. [22] captures the contents of web pages as semantic trees, labels attributes in the trees, and then extracts attribute values using the semantics provided by attribute labels and a Dynamic Hierarchical Conditional Random Field model. Raeymaekers et al. [15] also represent the content of a web page in a tree structure and use the (k, l)-contextual tree languages to (i) decompose a tree into sub-trees, each of which contains at most k children with a maximal depth of l, and (ii) induce domain-specific wrappers for identifying and extracting information from the tree structure. Unlike [15, 22], $ADEx$ does not impose an overhead on structuring the *contents* of web pages (online ads in our case) as trees for data extraction.

Ontologies, along with various probabilistic models, have been constructed for extracting data from online data sources. Khare and An [7] rely on a Hidden Markov Model (HMM) to label major components of a web interface, such as text-labels, text-boxes, and selection lists, and extract information from the interface. The authors in [7] train different HMMs, one for each available template of a web page, using training data that are grouped according to the templates. This approach is similar to $ADEx$, since $ADEx$ constructs a decision tree classifier for extracting data from ads in a particular domain D using training ads data belonged to D. Even though HMM is effective for data extraction, it is slower than existing supervised algorithms, including the decision tree employed by $ADEx$. Rajput and Haider [16] apply ontologies, various information extraction techniques, and Bayesian Networks to extract and predict missing information from unstructured, ungrammatical, and incoherent data sources, such as online ads at Craigslist(.org). Although effective, the model has been verified only for extracting data from a single ads domain.

Instead of relying on supervised learning methodologies, probabilistic models, and/or ontologies, some of the data extraction tools [11, 18] depend on the HTML structure of web pages to perform the information extraction task. Miao et al. [11] represent each web

page as a sequence of binary visual signals, which capture the patterns of HTML tag paths[2] so that paths that satisfy the structure of data records of interest are extracted. Unlike $ADEx$ which analyzes the contents of ads to identify attribute values to be extracted, the information extraction methodology in [11] relies solely on web page structures and thus can only be applied for extracting data from structured web pages. Song et al. [18] introduce MiBAT (Mining data records Based on Anchor Tree), an information extraction tool that automatically creates data records from semi-structured (HTML) web pages that include free-format, user-generated content such as posts, comments, or reviews. MiBAT considers domain constraints of web data records and anchor points (enclosed in HTML structures) to identify the structured part of a semi-structured web page, which contains DB attributes and values (embedded in XML/HTML tags) to be extracted. Although MiBAT has been evaluated only on forums, the authors claim that it can also be employed to extract data from other semi-structured web sources.

Wrappers have also been commonly employed for extracting information from online documents. Phoebus [12] creates data records from unstructured and ungrammatical online ads based on extractors constructed using labeled information in documents. In performing the extraction task, Phoebus automatically creates "reference sets" using trained wrappers. Reference sets, which are collections of known (DB) attributes with their values, serve as a source of supervised assistance in creating the information extractor. Phoebus aligns a car, hotel, or restaurant ad to a reference set to determine the schema for the ad and then extracts attribute values in the ad. Dalvi et al. [5], on the other hand, develop a generic noise-tolerant framework (NTW), which facilitates the process of constructing wrappers in an unsupervised manner using automatically acquired training data. The proposed framework combines (i) supervised learning wrapper induction techniques, (ii) domain knowledge, i.e., dictionaries, and (iii) unsupervised grammar induction methods, i.e., regular expressions, to perform the wrapper induction task.

Unlike MiBAT, Phoebus, and NTW, $ADEx$ does not require any manually-created regular expressions and/or ontologies for mapping text from un-/semi-structured data sources into data records. In addition, $ADEx$ defines features applicable to any ads domains, which is not applicable to wrappers, reference sets, and domain constraints on which NTW, Phoebus, and MiBAT rely, respectively.

3 $ADEx$

In this section, we present the overall process of $ADEx$, as shown in Figure 1. We describe the three major, automated, consecutive tasks performed by $ADEx$ during the process of extracting online ads data to populate a DB: (i) $classifying$ online ads into their respective domains (as detailed in Section 3.1), (ii) $tagging$ keywords in classified ads according to their types (as presented in Section 3.2), and (iii) $extracting$ the tagged keywords of different types in ads and populating them as attribute values of the corresponding DB records (as discussed in Section 3.3).

[2] A *tag path* is a path from the root to a leaf in a Document Object Model (DOM) representation of a web page.

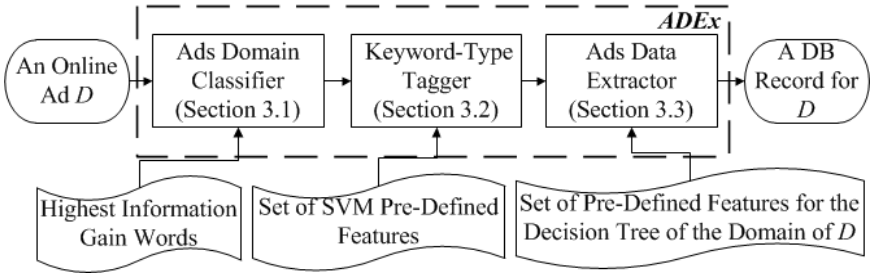

Figure 1: The overall process of $ADEx$ in extracting data from an online ad with unknown domain to create a DB record

3.1 Advertisements Classification

Given an online ad, $ADEx$ first classifies the ad according to its domain, since it is assumed that the domains of ads from multiple online sources are not always known in advance.

3.1.1 Keyword Selection

As claimed by Yang and Pedersen [21], one of the major problems in document classification is the high dimensionality of the feature space, i.e., the large number of unique keywords in documents, which affects the performance of classifiers in terms of computational time. In solving this problem, we first identify and select the set of *most representative keywords*, denoted MRs, in ads belonged to different ads domains based on a set of randomly-selected ads S (which yields the dataset used for keyword selection described in Section 4.3). This set of keywords captures the contents of the ads in S. Using the keywords in MRs to represent (the contents of) ads for classification, the processing time is minimized without affecting the accuracy of the classifier.

During the keyword selection process, we start with removing *stopwords*[3] and numerical values from the ads in S and then apply *information gain* (as defined in Equation 1) to determine the "suitability" of a keyword for classification.

$$G(w) = -\sum_{i=1}^{m} P(c_i) log P(c_i) + P(w) \sum_{i=1}^{m} P(c_i|w) log P(c_i|w) + P(\bar{w}) \sum_{i=1}^{m} P(c_i|\bar{w}) log P(c_i|\bar{w}) \quad (1)$$

where w is a keyword in S, m is the number of distinct natural classes, i.e., ads domains in our case, $P(c_i)$ is the probability of class c_i (an ads domain in our case), which is computed as the number of ads belonged to c_i over the total number of ads in S, $P(w)$ ($P(\bar{w})$, respectively) is the percentage of ads in S in which w occurs (does not occur, respectively), and $P(c_i \mid w)$ ($P(c_i \mid \bar{w})$, respectively) is the probability of assigning class c_i to an ad in S, given that w is present (absent, respectively) in the ad.

After computing the information gain of each distinct keyword in S, we select the top 2,500 keywords that have the *highest* information gain for representing ads to be classified by $ADEx$. (See Section 4.3 for details.)

[3]We compiled our own list of 531 stopwords using multiple stopword lists posted online. Online stopword lists are widely available on the web these days.

3.1.2 Naive Bayes Classifier

In classifying online ads that are represented using the selected keywords, we adopt a Naive Bayes classifier, which is simple, easy to implement, robust, highly scalable, and domain independent. The classifier relies on the probability of assigning the natural class c to a given document d, which is the well-known Bayes' Theorem.

$$P(c \mid d) = \frac{P(c)P(d \mid c)}{P(d)} \tag{2}$$

where $P(d)$ is the probability of a given document d, $P(c)$ is the probability of a particular natural class c, and $P(d \mid c)$ is the probability of d given c. From now on, unless stated otherwise, d and c in Equation 2 denote an online ad and ads domain, respectively.

In choosing the domain to which an ad d should be assigned, we compute the conditional probability $P(c_i \mid d)$ as defined in Equation 2 for each one of the possible ads domains c_i ($1 \leq i \leq m$). We assign to d the ads domain c_i that yields the *highest* $P(c_i \mid d)$ among all the ads domains $c_1, \ldots, c_m$ (as shown in Equation 3).

$$Class(d) = argmax_{c_i \in C}\ P(c_i \mid d) \tag{3}$$

where C is the set of ads domains $c_1, \ldots, c_m$.

3.1.3 Joint Beta-Binomial Sampling Model

In estimating $P(d \mid c)$ in Equation 2, we have chosen the Joint Beta-Binomial Sampling Model (JBBSM) introduced in [1], which considers the "burstiness" of a (representative) keyword in d, i.e., a keyword is more likely to occur again in d if it has already appeared once in d. JBBSM represents d as a vector of count-valued variables and computes $P(d \mid c)$ as a sequence of probabilities of the form $P(d_j \mid c)$, which is the probability of the j^{th} ($1 \leq j \leq n$) keyword in d, i.e., d_j, given a particular class c as shown in Equation 4.

$$P_{bb}(d_j \mid \alpha_j, \beta_j) = \left(\begin{array}{c} n \\ d_j \end{array} \right) \frac{B(d_j + \alpha_j, n - d_j + \beta_j)}{B(\alpha_j, \beta_j)} \tag{4}$$

where n is the total number of keywords in d, B is the Beta function of JBBSM, and α_j and β_j are the parameters of B which estimate the *presence* and *absence* of d_j in d belonged to c, respectively. (Detailed discussion on B and the estimation of parameters α_j and β_j in JBBSM, which are numerical values empirically determined, can be found in [1]. As previously stated, these parameters vary depending on the class c to which d belongs. In other words, the parameter distributions for d_j in d, i.e., α_j and β_j, affect the values computed by using Equations 4 and 5, which differ depending on the class c assigned to d.)

Using JBBSM, the probability $P(d \mid c)$ is computed as

$$P(d \mid \alpha, \beta) = \prod_j P_{bb}(d_j \mid \alpha_j, \beta j) \tag{5}$$

where α and β are parameters of the Beta-Binomial distribution of d (in c) as defined in [1], and $P_{bb}(d_j|\alpha_j, \beta_j)$ is as defined in Equation 4.

3.2 Keyword Tagging Based on Types

Ads in different domains include various attributes and their corresponding values. In designing *ADEx*, we (i) define a number of attribute types for keywords (in ads) which capture or indicate valid attribute values, and (ii) develop a *domain independent* tool to identify attribute values in different ads (domains).

Regardless of its domain, each ad showcases a particular product or service PS, which can be recognized by its *unique identifier* in the ad. In addition, each ad often includes (i) a number of *properties* that describe PS and/or (ii) *quantitative values* that identify the measurable substances of PS. With that in mind, we have defined the following ads *data types*, of which the corresponding data items are alpha-numerical strings.

- A *Type I* attribute value in an ad (a DB record, respectively) is a *single, unique identifier* of PS, which is not a numerical value (as some of the Type III attribute values are). A sample Type I attribute in the Jewelry ads domain is "Category", and "necklace" is one of its values.

- *Type II* attribute values describe the *properties* of PS in an ad (a DB record, respectively). "Features" is a Type II attribute in the Houses-for-Sale ads domain, and "central air" is one of its possible values.

- *Type III* attribute values are the *quantitative values* of PS in an ad (a DB record, respectively). A sample *Type III* attribute is "Salary" in the Jobs ads domain, and $50,000 is one of its values. In addition, "usd" is also a *Type III* attribute value, which identifies the unit of "Price" (a *Type III* attribute) in the Cars-for-Sale ads domain.

- *Type IV* attribute values are *non-essential, cosmetic* keywords included in ads, such as "large," which are non-attribute values in any ad.

Keyword/Attribute value types yield a universal tagging mechanism that allows *ADEx* to handle any types of ad data regardless of its domain and source. The tagging mechanism (i) minimizes the overhead and time required to identify different data included in each ad domain, (ii) allows for training and testing to be conducted on completely separate domains while maintaining a high tagging accuracy on test results, and (iii) avoids manual intervention required to identify the data in each separate ad domain during the data extraction/training process.

By default, *ADEx* assigns a Type III tag to each numerical value in an online ad d, which are potential attribute values to be included in the DB record generated for d. To tag each of the non-stop, non-numerical keywords in the ads according to their types, we rely on *SVM*, which is a robust classifier. In describing SVMs, we adopt the notations used in [17], which define a set S of training instances as a collection of N ($= |S|$) labeled input vectors of the form $\{(x_1, y_1), \ldots, (x_N, y_N)\}$, where x_i ($1 \leq i \leq N$) is an input vector of *features* (which are introduced in Section 3.2.1) used for describing a given keyword i in an ad, and $y_i \in \{-1, 1\}$ is the (binary) class label of x_i, i.e., a keyword type in our case. Furthermore, $\varphi(x_i)$ is the *kernel mapping* that generates the corresponding vector x_i in the *feature space*, and $K(x_i, x_j) = \varphi(x_i).\varphi(x_j)$ is the *kernel function* that computes the dot/scalar product

between $\varphi(x_i)$ and $\varphi(x_j)$ to determine the *distance* between the two vectors in the feature space. To identify the *support vectors* of the (soft-margin) SVM, i.e., the *points* that lie close to the decision boundary, which are employed during the classification process of a new instance, the following optimization problem must first be solved[4]:

$$max_\alpha\{\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j y_i y_j K(x_i,x_j)\ \},\ 0 \leq \alpha_i \leq C, \sum_{i}^{N}\alpha_i y_i = 0 \tag{6}$$

where C is a regularization parameter controlling the penalty for misclassification which we have empirically established as 1.0, α_i (α_j, respectively) is the Lagrange multiplier associated with (i.e., the weight of) x_i (x_j, respectively), and y_i (y_j, respectively) is the corresponding class of x_i (x_j, respectively). The goal of solving Equation 6 is to identify all the input vectors with coefficients $\alpha_i > 0$ ($1 \leq i \leq N$), which yield the *support vectors* of the SVM.

We have adopted the Radial Basis Function (RBF) kernel in Equation 7, which is one of the most typical kernels [19], as the kernel function K in Equations 6 and 8 for the SVM.

$$K(x_i,x_j) = exp(-\frac{||x_i - x_j||^2}{\sigma^2}) \tag{7}$$

where $||x_i - x_j||$ is the Euclidean distance computed between vectors x_i and x_j[5], and σ is the parameter that determines the area of influence of the corresponding support vector. A large σ yields a smoother decision surface, since an RBF with a large σ allows a support vector to have a larger area of influence. We have experimentally determined σ to be 500.

3.2.1 Feature Representation

To train our SVM, each training instance is a feature-vector with a sequence of "0" and "1" assigned to a particular non-stop, non-numerical keyword w in an ad, such that a '1' is given to the corresponding feature f (defined as follows) if f applies to, i.e., describes, w and is given a '0', otherwise.

- **Is-Plural**: This feature is set to '1' if w is in *plural* form and is '0', otherwise. Types I and II attribute values tend to be expressed in the *singular* form.
- **Is-Capitalized**: The first character in w that is a Type I attribute value is often *capitalized*, and this feature is assigned '1' if the first letter of w is capitalized and is '0', otherwise.
- **Is-Style**: This feature is set to '1' if w is either *bold* or *italicized* in an ad and is '0', otherwise. Types I and II attribute values in an ad tend to be either bold or italicized.

[4]As stated in [20], in solving constraint optimization problems it is a common practice to adopt the Lagrange multiplier method, since the method reduces a problem complexity with a minimal impact on its optimality.

[5]In *ADEx*, each vector represents the heuristics, i.e., *features*, of a keyword in an ad.

- **In-Text**: Since the most descriptive, i.e., Types I and II, attribute values of an ad d often appear in the title or first sentence of d, this feature is assigned a value '0' if w is in the title or first sentence of d and is given a '1', otherwise, i.e., w appears in the text of d.

- **Is-Adjective**: This feature is implemented using Stanford's Part Of Speech (POS) tagger[6], which assigns parts of speech, such as nouns, verbs, or adjectives, to keywords. This feature is set to '1' if w is given an "adjective" POS tag and is '0', otherwise. Most Type II attribute values are adjectives, such as color "blue" in a car ad, which describe the *properties* of an ad.

- **Is-Measurement**: This feature takes on a value of '1' if w is a *unit of measurement*, e.g., usd, mile, square feet, or inches, and is '0', otherwise. A set of measurement terms was extracted from the Electronic Hobby Projects[7], a website that lists units of measurements for different categories, such as length, area, power, and speed. This feature is a clear indicator of a Type III attribute value.

- **Is-Alphanumeric**: This feature is assigned a '1' if w contains both *numbers* and *letters* and is assigned a '0', otherwise. This feature assists in identifying a Type III attribute value.

- **Is-Location**: This feature depends on a set of locations. If w is a *location*, i.e., a Type IV attribute value which is non-essential in $ADEx$, the feature is set to '1' and is '0', otherwise. The list of locations, i.e., US cities, was extracted from Wikipedia[8].

- **Is-Acronym**: This feature is set to '1' if w is an *acronym* and is '0', otherwise. In determining an acronym, we adapt the approach in Chieu and Ng [2] which looks for sequences of capitalized words in a document d, i.e., sequence of words in which the first letter of each word is capitalized, that match (potential) acronyms in d. If the concatenation of the first capitalized letter of each word in a sequence of words in d matches w (in d), then we treat w as an acronym. Acronyms in general are Type I attribute values.

To verify that the chosen features listed above are accurate in identifying different types of keywords in ads, we conducted an empirical study using a newly-created dataset, denoted $Feature\text{-}DS$, which does not overlap with the dataset used in Section 4.1 for analyzing the performance of $ADEx$. $Feature\text{-}DS$ consists of 15,000 ads, out of which 5,000 ads are extracted from each one of three ads websites, Craigslist, Ebay(.com), and KSL(.com), and are evenly distributed over the eight distinct ads domains defined in Section 4.1.

Figure 2 shows the percentages of keywords (in $Feature\text{-}DS$) of each type that are recognized by each feature, which is designed to identify its corresponding type and accurately do so (as shown in **bold** in Figure 2). For example, 83% (87% respectively) of Type III keywords are identified by "Is-Alphanumeric" ("Is-Measurement", respectively),

[6] nlp.stanford.edu/software/tagger.shtml

[7] hobbyprojects.com/dictionary_of_units.html

[8] en.wikipedia.org/wiki/List_of_cities,_towns,_and_villages_in_the_United_States

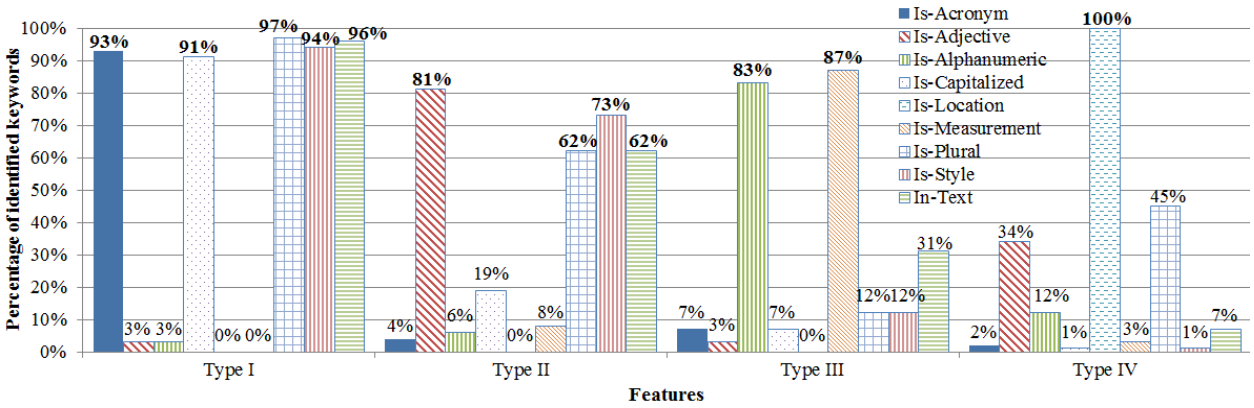

Figure 2: Percentages of keywords (grouped by types) in *Feature-DS* identified by each of the nine features defined by *ADEx* for its SVM

whereas the percentage of Type III keywords identified by the remaining features are at or below 31%, which implies that "Is-Alphanumeric" and "Is-Measurement" are indeed features indicative of Type III keywords in ads. The same applies to the remaining keyword types. This empirical study validates that the chosen features used by the SVM of *ADEx* adequately identify the types of keywords for which they are intended.

3.2.2 Multiclass-SVM

In tagging keywords based on their types, using a single binary SVM is insufficient, since the adopted SVM must handle more than two types of attribute values. We apply the one-against-all strategy [8] to solve the multi-class problem using a number of binary SVMs.

Given j (> 2) different classes, the one-against-all approach constructs j binary SVM classifiers, each of which separates one class from the rest. The j^{th} SVM is trained using the training instances in which the ones belonged to the j^{th} class are given *positive* labels and the remaining instances *negative* ones [8]. Using the multi-class SVM, the classification of a new instance v, which is represented as a feature-vector associated with a keyword in an ad, is a task to determine among each of the pre-defined attribute types t (i.e., Type I-IV) the one for which the corresponding $f_t(v)$ is the highest, as shown in Equation 8.

$$argmax_{\,t \in T}\, f_t(v) = \sum_{i=1}^{N'} \alpha_i y_i K(x_i, v) + b \tag{8}$$

where $f_t(v)$ is the predicted score for v computed for attribute type t, T is the set of all the possible attribute types, b is the bias term[9] defined during the training of the SVM, $N^{'}$ ($\leq N$) is the number of *support vectors* of the SVM, and N, α_i, y_i, and $K(x_i, v)$ are as defined in Equation 6.

[9]In a soft-margin SVM, the *bias term* dictates the *distance* between the origin and hyperplane so that if the bias term is *decreased*, the hyperplane moves *closer* to the origin.

3.3 Populating an Underlying DB

Having identified the *domain* to which an online ad d belongs (as discussed in Section 3.1) and assigned the corresponding *type* to each non-stop keyword w in d (as explained in Section 3.2), $ADEx$ proceeds to extract the data in d and create a DB record[10] for d to be included in the underlying DB[11]. Extracting information from unstructured data sources is a *classification* process, since w is either assigned as a value to its corresponding attribute in the DB record of d or a "not-valid" label which indicates that w is not included in the DB record of d.

We apply the C4.5 decision tree algorithm [13] to construct a decision tree, one for each ads domain, for extracting data from ads of the same domain. The algorithm applies the divide-and-conquer strategy which recursively partitions the training instances into subsets according to a splitting criterion (separation test), i.e., the feature (as defined below) with the highest information gain (as defined in Equation 9).

1. **Keyword-Type** is the type of a keyword w.
2. **Previous-Keyword-Type** is the type of the non-stop keyword immediately preceding w in d, if it exists.
3. **Post-Keyword-Type** is the type of the non-stop keyword immediately following w in d, if it exists.
4. **Previous-Keyword-Attribute** is the DB attribute of the non-stop keyword immediately preceding w in d, if it exists.
5. **Closest-Type-III** is a non-stop keyword of Type III in d, if it exists, that is closest to w.

The features are defined to accurately identify w as an attribute value, and are based on the *context* in which w appears, i.e., based on w and other keywords that appear $before$ and/or $after$ w, in an online ad d. Moreover, the $Closest\text{-}Type\text{-}III$ feature identifies keywords commonly associated with numerical values for data extraction. For example, given "25 acres" in d, we rely on the keyword "acres", i.e., the keyword that is closest to the Type III keyword "25", in assigning the value 25 to its corresponding attribute in a DB record. Furthermore, the possible values of $Keyword\text{-}Type$, $Previous\text{-}Keyword\text{-}Type$, and $Post\text{-}Keyword\text{-}Type$ are either Type I, II, III, or IV, whereas $Previous\text{-}Keyword\text{-}Attribute$ is assigned either (i) an attribute in the schema of the domain of d or (ii) the label "not-valid".

A training instance in a training set S (used for constructing the decision tree of an ads domain), which represents a keyword w in an online ad d, is a sextuple of the form $<f_1$,

[10]Besides data extracted from an ad d, a DB record for d includes an additional attribute $LINK$ that points to the URL of the original ad and is not an attribute in the DB schema for the domain of d. The URL provides users other details of d, such as a picture of the product showcased in d.

[11]The schema of each ads domain is pre-defined, which is based on the schemas provided at various popular ads websites, such as Ebay, KSL, Carmax, etc., prior to invoking $ADEx$ to classify, tag, and extract data from online ads of the domain.

f_2, f_3, f_4, f_5, $Att>$, where f_i $(1 \leq i \leq 5)$ is one of the possible values that can be assigned to the corresponding feature i such that $i \in Feature\text{-}Set$ = {Keyword-Type, Previous-Keyword-Type, Post-Keyword-Type, Previous-Keyword-Attribute, Closest-Type-III}, and Att is either an attribute in the corresponding schema for which w is a valid attribute value or the label "not-valid" (which implies that w is not a valid attribute value).

One of the major tasks in constructing a decision tree is to establish the criterion used for identifying the *most effective* feature (in $Feature\text{-}Set$) in splitting training instances into different groups. The criterion we adopt is *Information Gain* defined as

$$Information\ Gain(S, F) = Entropy(S)\ - \sum_{f \in Values(F)} \frac{|S_f|}{|S|} Entropy(S_f) \tag{9}$$

where F is a feature in $Feature\text{-}Set$, $Values(F)$ is the set of all possible values of F, S_f is the subset of training instances in S in which the value of F is f, $|S_f|$ ($|S|$, respectively) is the number of training instances in S_f (S, respectively), and *Entropy* is defined as

$$Entropy(S_f) = \sum_{i=1}^{|Att|} -P_i\ log_2\ P_i \tag{10}$$

where P_i is the percentage of instances in S_f such that the value of their corresponding attribute is an i^{th} attribute value in the DB schema of the respective ads domain, and $|Att|$ is the total number of DB attributes in the schema of the corresponding ads domain plus one, i.e., the "not-valid" label. $Entropy(S)$ in Equation 10 is defined accordingly.

Example 1 Figure 3 shows an ad extracted from Craigslist[12]. Using the Naive Bayes classifier of $ADEx$, the ad is correctly identified as an ad in the *Cars-for-Sale* domain. After $ADEx$ has removed stopwords and employed the multi-class SVM, each of the remaining keywords in the ad is tagged according to its type, i.e., Types I-IV. As shown in Table 1, $ADEx$ correctly assigns to each of the non-stop keywords in the ad their corresponding types. (Recall that each numerical keyword in an ad is automatically assigned a Type III tag.) Furthermore, $ADEx$ uses the decision tree of the Cars-for-Sale ads domain to determine the DB attribute (in the schema of the *Cars-for-Sale* domain) to which each of the Types I, II, and III keywords in the ad should be assigned. Table 2 shows a portion of the DB record created by $ADEx$ for the ad in Figure 3, with each non-stop keyword in the ad correctly assigned as attribute value in the created DB record[13]. □

Example 2 An ad HS downloaded from KSL is shown in Figure 4, with all the contact information in HS again suppressed. Unlike the ad shown in Figure 3, which is a plain (unstructured) text, HS is organized according to descriptive terms pertinent to *Houses-for-Sale* ads, such as "Acres" and "Year Built", a format which is commonly used by KSL for *Houses-for-Sale* ads. $ADEx$ correctly identifies HS as a Houses-for-Sale ad. Table 3

[12]To protect the privacy of the creator of the ad, we have omitted the contact information of the creator by replacing them with "..." in Figure 3.

[13]Since the decision tree of $ADEx$ can assign the value "not-valid" to some of the Types I-III tagged keywords, not all the Types I-III keywords as shown in Table 1 are attribute values in the DB record of the ad as shown in Table 2.

2008 Toyota Scion xD - $ 13000 (orem ut)

Date: 2010-07-19, 6:53PM MDT ....

ONLY 31K miles on it!!! priced as bluebook value (13k). Can't lower price cuz I am only asking what I still owe on it. Its been an great car and did great through 5 blizzard last year. I was very impressed since it only has front wheel drive. Its a great sporty car. seats 5. Gas mileage 32 mpg is what I have been getting. Hatch back. A Fun Fun car. Bought it brand new from brent brown Toyota and still has warranty coverage. Contact me by phone call or text for more questions ... '08 toyota scion xD – 5 door

Figure 3: A *Cars-for-Sale* ad from Craigslist

Keyword	
Type	in Ad
Type I	Toyota, Scion
Type II	xD, front, wheel, drive, sporty, hatch
Type III	2008, $, 13000, 31k, miles, 13k, seats, 32, mpg, '08, 5, door
Type IV	price, bluebook, value, owe, car, blizzard, year, gas, mileage, fun, bought, brand, brent, brown, warranty, coverage, contact, phone, call, text, questions, ...

Table 1: Types assigned by $ADEx$ to non-stop keywords in the ad shown in Figure 3

displays the types assigned by $ADEx$ to the non-stop keywords in HS, whereas Table 4 shows the tagged Types I-III keywords assigned to their corresponding DB attributes (in the *Houses-for-Sale* schema as partially shown in the table). Note that all the keywords in HS, except "500k" and "garage", are correctly tagged. $ADEx$ incorrectly assigns "500k" as $Price$, which should be excluded, whereas "garage" should not be tagged as a Type II keyword, since in the original ad it states "Garage: None". Even though the DB record of HS includes incorrectly assigned DB attribute values, the quality of the created record is not significantly affected, since the majority of the ad data in HS are correctly populated. □

4 Experimental Results

In this section, we assess the overall performance of $ADEx$. In Sections 4.1 and 4.2, we introduce the dataset and metrics employed for performance evaluation, respectively. In Section 4.3, we determine the ideal number of keywords, i.e., the size of the vocabulary, for capturing the content of ads in their respective domains for classification. Hereafter, we evaluate the effectiveness of each major task of $ADEx$, which include *classifying* ads (in Section 4.4), *tagging* keywords (in Section 4.5), and *extracting* ads data to populate the underlying DB (in Section 4.6). We also assess the overall performance of $ADEx$ (in Section 4.7) and compare (the performance of) $ADEx$ with existing state-of-the-art information extractors (in Section 4.8).

DB Attribute	Attribute Values
Make	Toyota
Model	Scion
Price	13000
Mileage	31K
Year	2008
Number of Doors	5
Number of Seats	5
Mpg	32
Features	xD, front, wheel, drive, sporty, hatch
...	...

Table 2: A portion of the DB record of the ad shown in Figure 3 with previously-tagged keywords (as shown in Table 1) assigned to their corresponding DB attributes

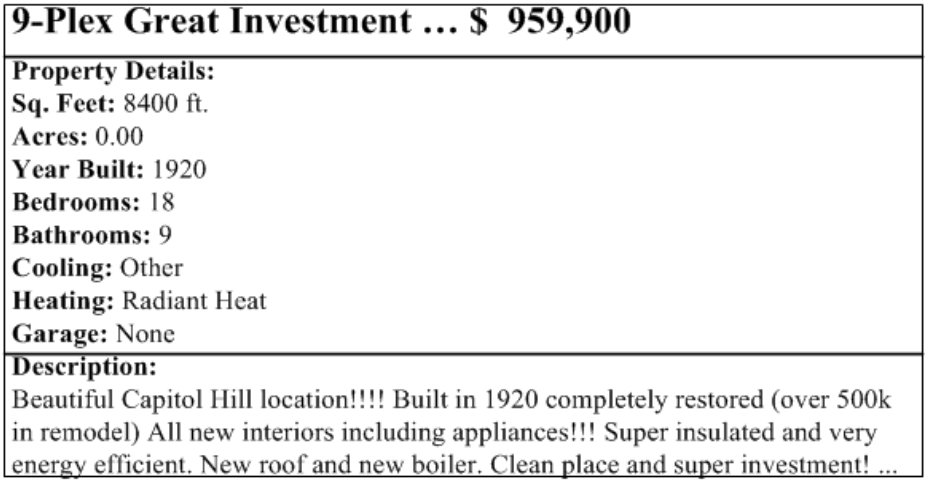

Figure 4: A Houses-for-Sale ad from KSL

4.1 The Dataset

To the best of our knowledge, there is no dataset available for evaluating classification, labeling, or data extraction approaches on online ads. For this reason, we have created our own dataset, denoted *EData*, for assessing the performance of *ADEx*.

EData consists of 18,000 online ads, which were extracted from Craigslist, Ebay, and KSL. The ads in *EData* are uniformly distributed among the eight chosen domains, which are Cars-for-Sale, Computer Science (CS) Jobs, Food Coupons, Furniture, Houses-for-Sale, Jewelry, Motorcycles-for-Sale, and Musical Instruments, and there are 750 ads in *each* of the eight ads domains extracted from *each* of the three ads websites. The ads domains in *EData* vary in terms of their (i) *diversity*, which include ads in jobs, transportation, food, housing, and entertainment that are essential to our daily lives, (ii) ads *sizes*, from arbitrary long ads (such as Houses-for-Sale ads) to relatively short ones (such as Jewelry ads), and

Keyword	
Type	in Ad
Type I	9-Plex
Type II	cooling, heating, radiant, heat, garage, capitol, hill, insulated, energy, efficient, clean, boiler
Type III	$, 959,900, sq., feet, 8400, ft, year, 1920, acres, 0.00, bedrooms, 18, bathrooms, 9, 500k
Type IV	property, details, built, beautiful, location, completely, restored, remodel, interiors, appliances, insulated, roof, place, investment, ...

Table 3: Keyword types assigned by *ADEx* to non-stop keywords in the ad shown in Figure 4

DB Attribute	Attribute Values
Category	9-Plex
Price	959,900, 500k
Lot Size	0.00
House Size	8400
Year Built	1920
Number of Bedrooms	18
Number of Bathrooms	9
Features	cooling, heating, radiant, heat, garage, capitol, hill, insulated, energy, efficient, boiler, clean
...	...

Table 4: A portion of the DB record of the ad shown in Figure 4 with previously-tagged keywords (as shown in Table 3) assigned to their corresponding DB attributes

(iii) *word distribution*, i.e., different word usage in closely-related ads which are similar in contents and nature, such as Cars- and Motorcycles-for-Sale that are two different means of transportation. Moreover, ads in *EData* were extracted from various online sources with different structures and formats, which further shows that the dataset is generic for assessing *ADEx*.

4.2 Evaluation Measures

To evaluate the effectiveness of *ADEx* in classifying ads, tagging keywords, and extracting ads data, we rely on *Precision* (P), *Recall* (R), and *F-Measure* ($= \frac{2 \times P \times R}{P+R}$), which are popular metrics for analyzing the performance of information retrieval tools.

To perform an unbiased evaluation, we adopted the five-fold cross-validation approach [10] so that in each of the five repetitions, 80% of the instances in *EData* were used for training and the remaining 20% for testing. From now on, whenever we refer to Precision, Recall, or F-Measure, we mean the *averaged* Precision, Recall, or F-Measure, respectively

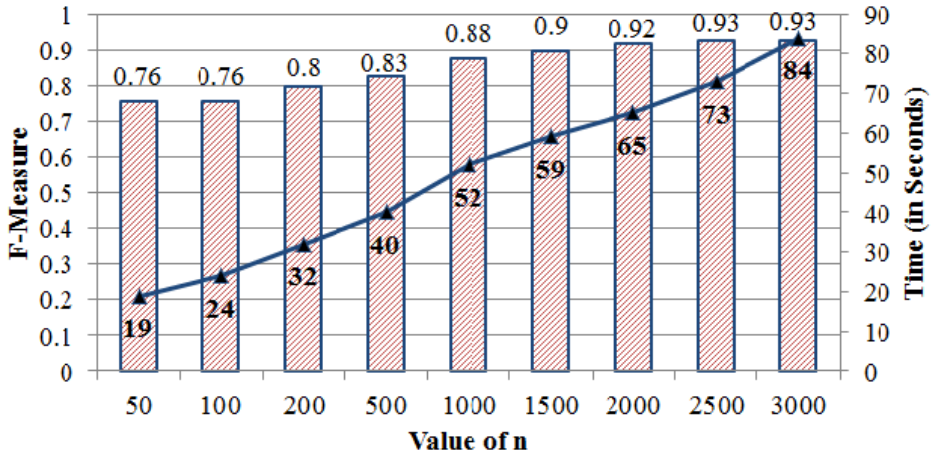

Figure 5: The F-Measure ratios, along with the processing time, achieved by $ADEx$ using different vocabulary sizes for classification on 3,000 ads

generated by the five repetitions of the validation.

4.3 Keyword Selection

Prior to performing the classification task, the *keyword-selection* process discussed in Section 3.1.1 is applied to reduce the *size* of the *vocabulary*, i.e., the number of distinct keywords in $EData$, so that the top n (≥ 1) keywords, which are neither stopwords nor numerical values, are chosen for representing ads in various domains. To determine the appropriate n, we conducted an empirical study using a total of 3,000 randomly-selected online ads (not included in $EData$), which were extracted from Craigslist, Ebay, and KSL and belong to the eight domains introduced in Section 4.1. We considered alternative values of n, such that $n \in \{50, 100, 200, 500, 1000, 1500, 2000, 2500, 3000\}$, and set n to be *2,500*. As shown in Figure 5, using 2,500 keywords on the 3,000 ads, we achieve the *highest F-Measure* for *classification* and still maintain the classification processing time just slightly over a minute.

4.4 Classification of Ads

In this section, we detail the empirical study we have conducted to assess the effectiveness of $ADEx$ in classifying ads and compare its performance with other classification approaches.

4.4.1 Classification Effectiveness

To verify the effectiveness of the proposed classifier of $ADEx$, we computed the precision and recall achieved by $ADEx$ for classifying ads in $EData$ to their corresponding domains. As illustrated in Figure 6, $ADEx$ achieves high precision and recall in classifying ads, and most of the classification errors occur when two ads domains share very *similar* probability distribution on a considerable number of keywords, such as in Motorcycles- and Cars-for-Sale. Even though the precision and recall ratios for car and motorcycle ads are the lowest

Figure 6: Precision, Recall, and F-Measure achieved by $ADEx$ on ads classification using $EData$

among all the eight domains, they are still in the ninety percentile.

Figure 6 also shows the F-Measure of classifying $EData$ ads into each one of the eight ads domains previously introduced, as well as the overall F-Measure achieved by $ADEx$ for the classification, which are in the (upper) ninety percentile. Among the eight ads domains in $EData$, $ADEx$ achieves the *highest* F-Measure for classification on Food (Coupons), Jewelry, CS Jobs, and Music(al Instruments), and the *lowest* F-Measure on Cars(-for-Sale) and Motorcycles(-for-Sale). Based on the conducted empirical study, we conclude that a domain in which its word usage is *similar* to the one used in another domain tends to yield *lower F-Measure* for classification than the ones dissimilar in word usage due to the existence of *common* keywords, as anticipated.

4.4.2 Comparison of Classifiers

We have further verified the effectiveness of the classifier of $ADEx$ to a greater extent by comparing the classification performance of $ADEx$ with two other well-known classifiers: the Multinomial Naive Bayes classifier, denoted MNB, and the (implementation of the one-against-all) SVM extracted from RapidMiner[14], which are two widely-used text classification approaches.

MNB follows the premises of the Naive Bayes classifier (as discussed in Section 3.1.2) in assigning a given document to its class. As opposed to JBBSM introduced in Section 3.1.3, MNB determines the probability of a keyword w in a natural class c_j, denoted $P(w \mid c_j)$, using Equation 11, which is based on the *frequency* of keyword *occurrence*.

$$P(w|c_j) = \frac{1 + \sum_{i=1}^{|D|} N_{i,w} P(c_j|d_i)}{|V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N_{i,s} P(c_j|d_i)} \tag{11}$$

where $|D|$ is the number of labeled documents in a collection D, which is the number of training instances in $EData$ in our empirical study, $|V|$ is the number of distinct keywords

[14]RapidMiner (sourceforge.net/projects/yale/) is an open-source system that implements a wide variety of machine learning and data mining algorithms.

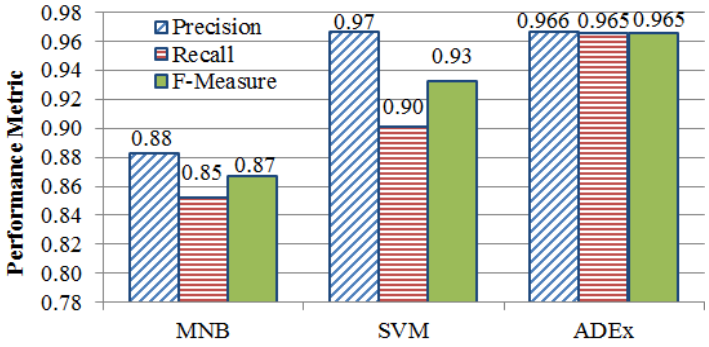

Figure 7: Precision, Recall, and F-Measure scores for ads classification achieved by MNB, SVM, and the JBBSM-based classifier of $ADEx$ on $EData$, respectively

Classification Approach	F-Measure
MNB	88.7 +/- 0.6
SVM	90.9 +/- 1.0
$ADEx$	97.4 +/- 0.8

Table 5: Averaged F-Measure for $ADEx$, as well as for the classification approaches MNB and SVM, used for the comparison purpose

in D, $N_{i,w}$ ($N_{i,s}$, respectively) denotes the *frequency* of *occurrence* of keyword w (w_s, respectively) in a labeled document (i.e., online ad in our case) d_i, and $P(c_j \mid d_i)$ is '1' if c_j is the class label of d_i and '0', otherwise.

Using Equation 2, the probability of assigning a class c_j (i.e., c in Equation 2) to a given document d is computed. In MNB, $P(d \mid c_j)$ is calculated as

$$P(d|c_j) = P(|d|)|d|! \prod_{s=1}^{|V|} \frac{P(w_s|c_j)^{N_{s,d}}}{N_{s,d}!} \tag{12}$$

where $|d|$ denotes the number of keywords in d, $N_{s,d}$ is the frequency of occurrence of keyword w_s in d, and $P(w_s|c_j)$ and $|V|$ are as defined in Equation 11.

The SVM classifier, on the other hand, is a vector-space-based method as introduced in Section 3.2, which determines a decision boundary between classes for classification [9].

As shown in Figure 7, the classifier of $ADEx$ outperforms both MNB and SVM, in terms of F-Measure, in assigning online ads in $EData$ to their corresponding ads domains. To further validate the accuracy of the F-Measure achieved by $ADEx$ in ads classification, we repeated the conducted experiments two more times, using two new, disjoint subsets of 5,000 ads each extracted from Craigslist, Ebay, and KSL, which are uniformly distributed among the eight ads domains. The overall F-Measure achieved by $ADEx$'s classifier, as well as MNB and SVM, is shown in Table 5, which yield F-Measure ratios consistent with the ratios displayed in Figure 7.

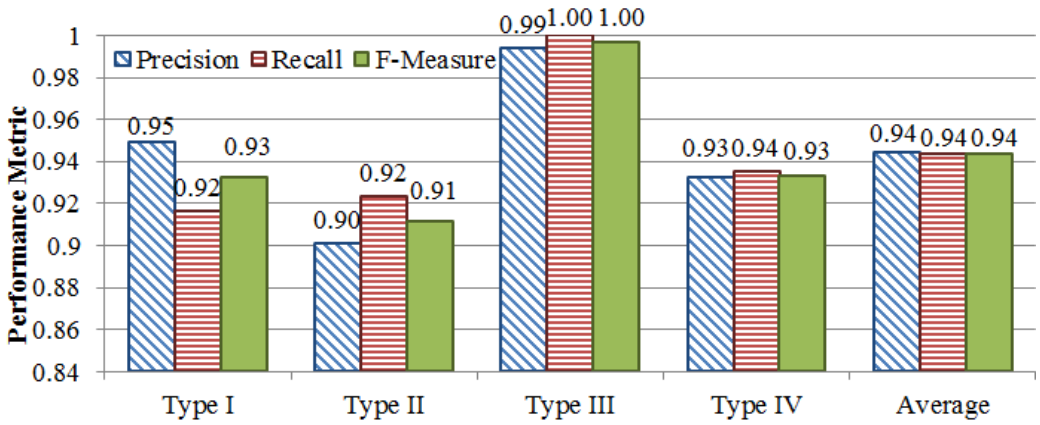

Figure 8: Precision, Recall, and F-Measure achieved by *ADEx* on tagging non-stop, non-numerical keywords in *EData*

4.5 Type-Based Keyword Tagging

To assess the accuracy of the multi-class SVM (introduced in Section 3.2.2) in tagging keywords in ads according to their respective types, we first created training and test instances using the ads in *EData*. Hereafter, for each non-stop keyword w in each ad in *EData*, we determined the features (as defined in Section 3.2.1) that apply to w, which yielded a set of 173,541 instances. Using the five-fold, cross-validation strategy (as discussed in Section 4.2), 80% of the instances were reserved for training the SVM, whereas the remaining 20% were employed as test instances in each validation step.

4.5.1 Tagging Accuracy

As shown in Figure 8, the overall Precision, Recall, and F-Measures of assigning Types I-IV tags to keywords in ads are in the *ninety percentile*. We have observed that most of the *misclassification errors* occur when attribute values that should be assigned a Type I tag are incorrectly labeled as Type II. It is because if none of the keywords in an ad is in *bold* or *italicized*, or is *capitalized*, the values assigned to features such as *Is-Style* or *Is-capitalized* are the same for keywords representing Type I or Type II attribute values, which causes the misclassification. Moreover, Type II attributes values are sometimes mislabeled as Type IV due to their relative positions in online ads and their proximity, and thus tagging Type II attribute values yields the lowest F-Measure.

4.5.2 Comparison of Taggers

To further validate the effectiveness of the keyword tagger of *ADEx*, we have compared its performance with two other well-known approaches, the C4.5 decision tree classifier and an artificial neural network, since as mentioned in [13], decision trees and neural networks are two frequently-adopted approaches in solving the labeling problem.

As defined in [13], a *decision tree* classifier categorizes instances, i.e., feature-vectors associated with a particular keyword in our case, by organizing them down the tree from

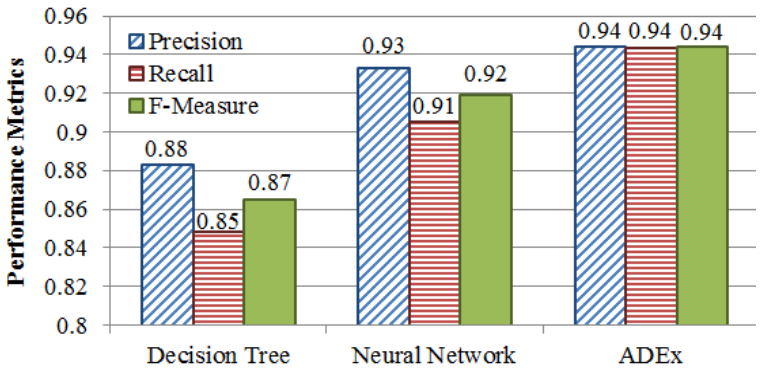

Figure 9: Performance evaluation on the multi-class SVM of *ADEx*, as well as other alternative machine learning approaches, for tagging keywords

the root to a leaf node, which provides the label of the instance, i.e., type associated with a keyword in our case. *Neural network*, on the other hand, is a robust approach that approximates real-, discrete-, or vector-valued target functions [13]. The training of a neural network invokes an iterative process in which for each training instance, i.e., a feature-vector representing a particular keyword in our case, the correct class, i.e., keyword type in our case, is known and thus it is possible to compare the output predicted by the network with the "known" one. The Neural Network proceeds to adjust the weights of the internal nodes of the network so that during the subsequent iteration process, the predicted output classes are closer to the "known" classes. We employed the implementations of the aforementioned taggers provided by WEKA[15] for comparing the effectiveness of *ADEx*'s multi-class SVM with the decision tree and neural network classifiers.

As shown in Figure 9, the multi-class SVM of *ADEx* outperforms the alternative approaches used for tagging keywords in ads according to their types.

4.6 Populating the DB

We have verified the effectiveness of the decision-tree-based approach of *ADEx* (introduced in Section 3.3) which assigns non-stop keywords in ads that are valid attribute values to their corresponding DB attributes using the ads in *EData* as training and test instances. In constructing the instances, we considered (i) the domain assigned to each ad in *EData*, (ii) the type of each non-stop keyword in the ads, and (iii) the features defined in Section 3.3. The set of training and test instances includes 173,541 feature-vectors, one for each non-stop keyword in *EData* ads (as mentioned in Section 4.5). As stated in Section 4.2, for each iteration in the cross-validation strategy, 80% of the instances were reserved for training the decision tree of *ADEx* and the remaining 20% for testing.

[15]WEKA (cs.waikato.ac.nz/ml/weka/) is another open source collection of implemented machine learning algorithms.

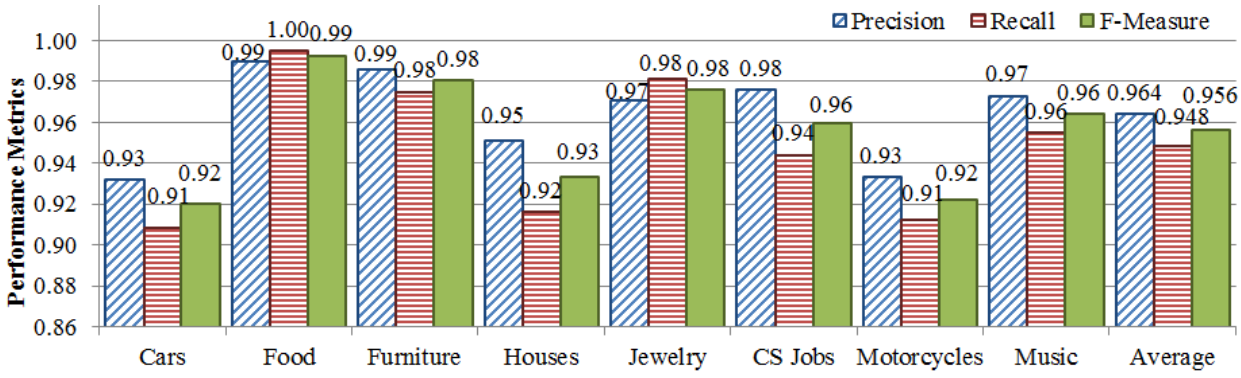

Figure 10: Precision, Recall, and F-Measure achieved by *ADEx* in extracting data on ads belonged to the eight ads domains in our empirical study

4.6.1 Extraction Accuracy

Figure 10 shows the accuracy of *ADEx* in assigning valid attribute values to their corresponding DB attributes in different ads domains. On the average, the F-Measure of the decision-tree-based approach is 0.956. Based on the conducted empirical study, we have observed that the F-Measure on data extraction for an ads domain that contains a *large* number of attributes in its schema is *lower* compared with others with *smaller* number of attributes. This is due to the fact that the *larger* the number of DB attributes, proportionally the *lower* the number of available instances in any training set of the same size compared with other ads domains with a smaller number of attributes to train the decision tree. This translates into *lower* precision and recall, which yield a lower F-Measure ratio, in correctly assigning values to the attributes. Moreover, (i) keywords of Types I and II in the Cars- and Motorcycles-for-Sale ads domains are often correctly assigned to their corresponding DB attributes, and (ii) keywords of Type IV are not assigned to any DB attribute, as anticipated. However, the overall F-Measure of Cars-for-Sale (Motorcycle-for-Sale, respectively) domain is among the lowest of the eight ads domains. This is caused by the common (numerical) Type III attribute values which are assigned to incorrect DB attributes with the same or compatible attribute domain values. For example, in Cars-for-Sale ads, '2009' is assigned to the attribute 'Mileage', instead of the correct attribute 'Year'.

4.6.2 Comparison of Extractors

We have compared the performance of *ADEx*, in terms of extracting data from online ads to populate the DB, with the WEKA implementation of two machine learning approaches commonly employed for defining classification rules: the Decision Tables Naive Bayes approach (DTNB) in [6] and the Rule Induction approach in [3], denoted JRIP. DTNB is a hybrid method that combines two well-established extraction strategies: *decision tables*, which define rules that determine to which DB attribute a particular keyword should be assigned, and *Naive Bayes classifiers*, which determine the probability of occurrence of a particular rule given a particular DB attribute. Given a keyword w in an ad, DTNB applies the rule with the highest probability in assigning the DB attribute for which w is its value.

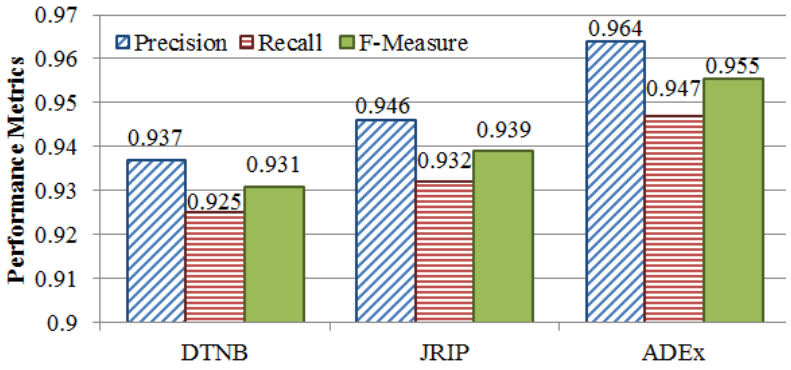

Figure 11: Performance evaluation on the decision tree of $ADEx$, as well as alternative machine learning approaches, for extracting data from online ads to create DB records

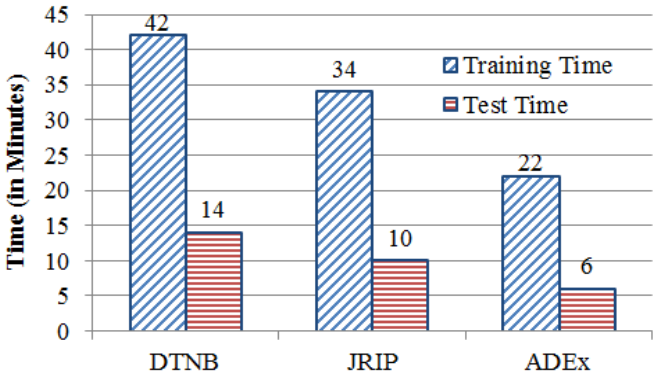

Figure 12: Training and Testing time of DTNB, JRIP, and $ADEx$ achieved by using the instances created for non-stop keywords in $EData$ for data extraction, respectively

JRIP, on the other hand, discovers rules that cover or partition the training examples, i.e., the set of feature vectors discussed in Section 3.3 in our case. JRIP is a bottom-up method such that for a given class, i.e., a DB attribute in our case, it finds the set of rules that cover all the members of that class, i.e., all the training instances associated with the given DB attribute in our case. Thereafter, JRIP repeats the process of defining rules for each class until all possible classes have been covered, i.e., until rules for each of the possible DB attributes are created in our case.

As shown in Figure 11, the decision tree-based extractor of $ADEx$ outperforms the alternative approaches for extracting data from online ads in terms of Precision, Recall, and F-Measure. Even though the difference in F-Measure between JRIP and $ADEx$ is less than 2%, $ADEx$ is *simpler* to implement, which has been verified. As shown in Figure 12, by using $ADEx$, compared with DTNB and JRIP, for extracting ads data to populate the underlying DB, the training (testing, respectively) time on $EData$ is reduced on an average by 58% (50%, respectively).

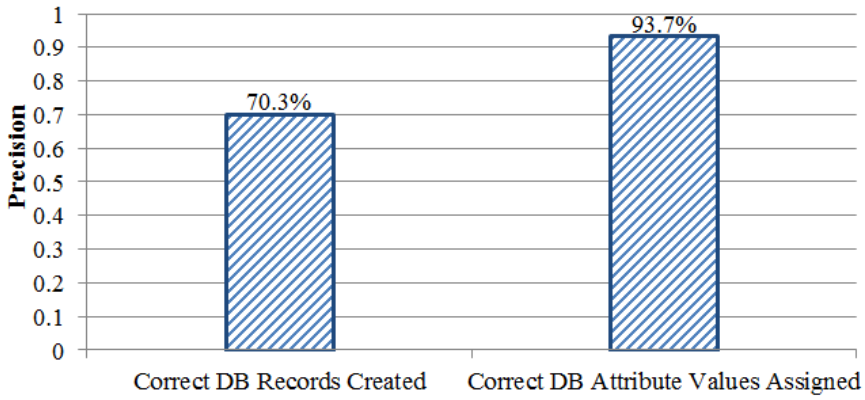

Figure 13: Overall precision on (i) correctly created DB records and (ii) DB attribute values correctly assigned to DB records by $ADEx$ based on ads in $TData$

4.7 Overall Evaluation of $ADEx$

To assess the overall performance of $ADEx$, in terms of its *effectiveness* in classifying, labeling, and extracting data to be populated into an underlying DB using online ads data, we first created a new collection of online ads, denoted $TData$, which is disjointed from $EData$ (as introduced in Section 4.1). $TData$ consist of 3,000 ads uniformly extracted from Craigslist, Ebay, and KSL, which are evenly distributed among the same eight ads domains previously introduced in Section 4.1.

4.7.1 Record Level Precision

The overall performance evaluation of $ADEx$ is based on the assumption that a DB record D created by $ADEx$ is treated as *incorrect* if (i) at least one valid attribute value in the corresponding ad from which D was created was assigned to the wrong DB attribute in, or not assigned to, D, or (ii) a Type IV attribute value, i.e., an invalid attribute value, in the corresponding ad was assigned to an attribute in D. We computed the *precision* on generating *correct* DB records, i.e., the proportion of ads for which correct records are created by $ADEx$ among all the ads in $TData$. The conducted empirical study shows that $ADEx$ achieves a precision ratio of 70.3% (see Figure 13).

4.7.2 Attribute Level Precision

Besides measuring the precision ratio at the *record* level, we conducted the same evaluation at the DB *attribute* level. In doing so, we determined the portion of attribute values that were correctly assigned to their designated DB attributes. The experimental results show that $ADEx$ correctly assigned close to 94% of the attribute values in $TData$, as shown in Figure 13. Furthermore, as shown in Figure 14, most of the incorrect DB records (528 = 225 + 303 in total) include a low percentage, i.e., between 10% and 15%, of incorrectly assigned attribute values[16].

[16]If the percentage of incorrect attribute values in a record R falls in between the interval of any two percentages, the error percentage is rounded to the nearest percentage point P, and R is assigned to the

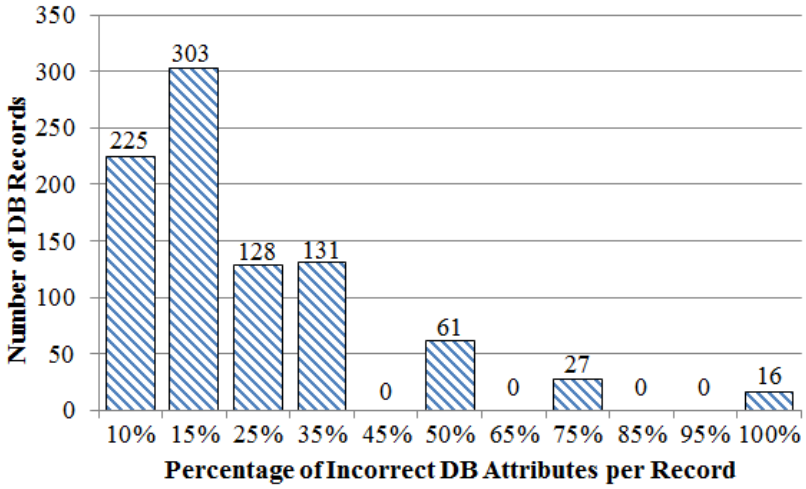

Figure 14: Error distribution in terms of percentages of attribute values incorrectly assigned to DB attributes in the 891 incorrectly created records out of 3,000 ads (records)

4.7.3 Records with Incorrectly Assigned Attribute Values

Based on the conducted experiments, we draw the conclusion that *ADEx* is *highly effective* in assigning keywords in an ad d to its corresponding attributes in the DB record of d (according to the pre-defined schema of the ads domain to which d belongs). Our claim is supported by the fact that close to 88% of the DB records are either correctly created (i.e., 70.3% as shown in Figure 13) or have at most 15% of invalid attribute values in their DB attributes (i.e., $\frac{225+303}{3000} = 17.6\%$ as shown in Figure 14).

4.8 Comparing *ADEx* with Other Data Extraction Approaches

To further assess the overall data-extraction method of *ADEx*, we compare its performance with other existing state-of-the-art data-extraction approaches. We have conducted two separate evaluations, one on unstructured data (in Section 4.8.1) and another one on semi-structured data (in Section 4.8.2), since *ADEx* is capable of extracting data from both data sources.

4.8.1 Evaluations on Unstructured Data

To extend the evaluation on the efficiency of *ADEx* in extracting data from unstructured data sources, we have compared the performance of *ADEx* with Phoebus [12] and ONDUX [4]. While Phoebus was introduced in Section 2, ONDUX is an information-extraction tool based on an unsupervised probabilistic model that segments text. ONDUX first defines a reference set, i.e., domain-specific knowledge base, for each domain based on which ONDUX is applied to extract data. Each reference set consists of sets of attribute values extracted from pre-determined data sources. To extract attribute values from a textual source in domain D, ONDUX relies on the knowledge base of D and employs generic matching

incorrect DB record with P% error.

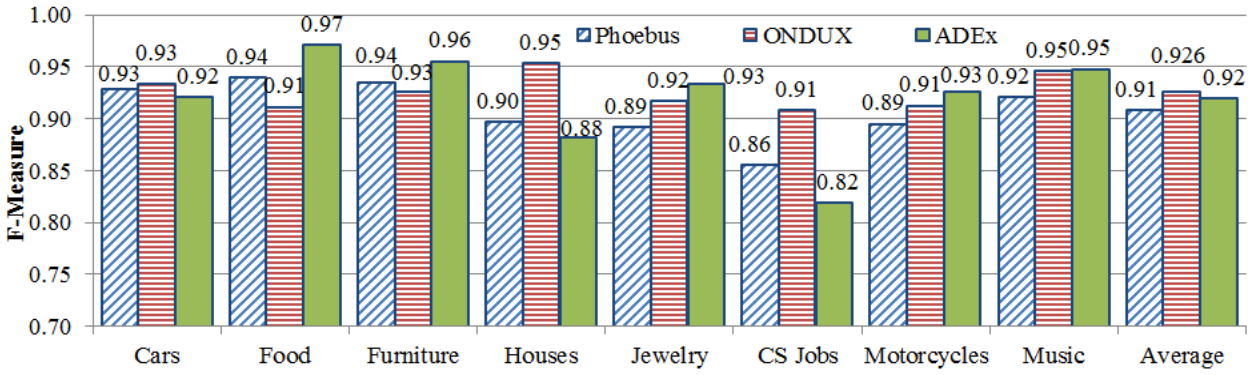

Figure 15: (Average) F-Measure achieved by Phoebus, ONDUX, and *ADEx*, respectively on identifying attribute values using the 1,444 unstructured ads in *TData*

functions to compute a score that determines the likelihood of a text segment as a value of a DB attribute. Thereafter, ONDUX applies a "reinforcement" step to verify and correct, if necessary, the assignment of a particular DB attribute label to a text segment, i.e., an attribute value.

In comparing the performance of *ADEx*, Phoebus, and ONDUX, we selected the portion of *TData* (as introduced in Section 4.7) that consists of unstructured ads, which yields a set of 1,444 ads extracted from Craigslist, Ebay, and KSL. As shown in Figure 15, for most of the ads domains the F-Measures of *ADEx* are higher than the one achieved by Phoebus and ONDUX, respectively. For the remaining ads domains, the F-Measures of *ADEx* are comparable to the ones of Phoebus and ONDUX, with the exception of ads in the House(-for-Sale) and CS Jobs, for which the F-Measures of *ADEx* are in the eighty percentile. Overall, *ADEx* outperforms Phoebus, in terms of averaged F-measure, and the difference in averaged F-Measure between *ADEx* and ONDUX is 0.6%, which is not significant.

4.8.2 Evaluations on Semi-Structured Data

Besides evaluating the performance of *ADEx* against information-extraction tools on unstructured data sources, we have also compared the performance of *ADEx* in extracting ads records from semi-structured sources with the two information extraction approaches, MiBAT [18] and NTW [5], respectively, which have previously been introduced in Section 2.

In comparing the performance of *ADEx*, MiBAT, and NTW, we gathered semi-structured ads in *TData*, which yields a set of 1,556 ads extracted from Ebay and KSL. Even though MiBAT (NTW, respectively) achieves higher F-Measure than *ADEx* in 2 (3, respectively) out of the eight ads domains, i.e., Cars and CS Jobs (Cars, Furniture, and Houses, respectively), the overall F-Measure of *ADEx* is higher than MiBAT and NTW, respectively, as shown in Figure 16.

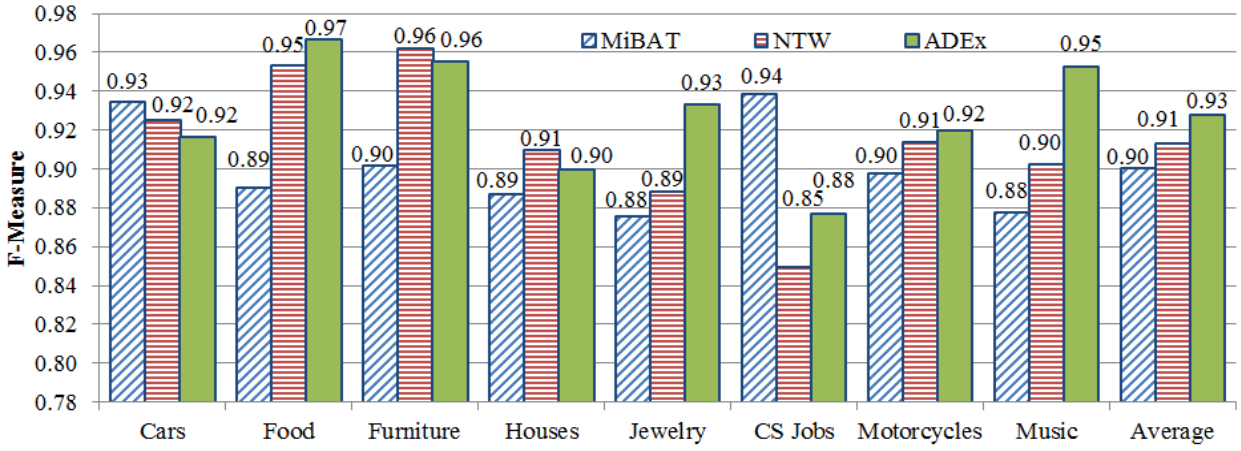

Figure 16: (Average) F-Measure achieved by MiBAT, NTW, and *ADEx*, respectively on identifying attribute values using the 1,556 semi-structured ads in $TData$

5 Conclusions

With the rise in popularity on online advertising, more web users turn to online sources to locate advertisements (ads for short) of interest. Since these web sources are independently operated and structured using different ads formats on various ads domains tailored for specific information processing, web users are required to access ads archived at different sites by employing a wide variety of searching tools individually. The existence of a unified database (DB), which integrates ads in multiple domains extracted from various online sources, should greatly simplify the process of inquiring ads data. To create such a unified DB, we have developed *ADEx*, a tool that automatically extracts data from online ads to generate ads records which are populated to an underlying ads DB. *ADEx* is implemented using well-established and easy-to-use machine learning algorithms and combines the tasks of *classifying* ads, *tagging* keywords in ads data, and *extracting* data in ads into a single process.

ADEx is unique, since it analyzes, filters, and extracts (ir)relevant data from online ads with different structures in a simple manner. As part of the extraction process, *ADEx* (i) categorizes ads according to their *domains*, since the domain to which an ad belongs is not always known in advance, and (ii) labels keywords in ads according to a set of pre-defined *types*, which generalizes and facilitates the process of identifying essential keywords, which are valid *attribute values*, in ads. More importantly, *ADEx* handles ads originated from unstructured, as well as (semi-)structured, sources without modifying its core design, which differs from existing information extraction approaches.

Empirical studies conducted on a set of 18,000 online ads, which belong to eight different ads domains retrieved from multiple web sources, show that *ADEx* is highly effective in classifying ads, as well as labeling and extracting their data, with F-Measure ratios in the *ninety percentile*. We have compared the performance of the various machine learning approaches adopted by *ADEx* with other existing machine learning approaches, which include using the Naive Bayes classifier based on JBBSM versus Multinomial Naive Bayes

and SVM for *text classification*, SVM versus C4.5 Decision Tree and Neural Networks for *keyword labeling*, and Decision Tree versus Decision Tables Naive Bayes and Rule Inference for *data extraction*. The results of the empirical study show that *ADEx outperforms* other machine learning approaches in accomplishing the same task. We have further assessed the overall performance of *ADEx* using other sets of online ads. The experimental results indicate that *ADEx* is consistent in performance or outperforms existing state-of-the-art information extractors that were designed solely for unstructured (semi-structured, respectively) data sources, in extracting attribute values from un-/semi-structured ads.

ADEx is currently designed to extract data from ads that include a single product/service in an ad. As part of our future work, we intent to enhance *ADEx* so that it can handle any online ads that include multiple products, such as in video games ads. Furthermore, even though *ADEx* currently handles ads from various domains, it is less accurate in distinguishing ads in closely-related domains, such as ads on different means of transportation, or ads that advertise professional jobs that cross multiple disciplines, such as Bioinformatics. We intent to improve *ADEx* so that it can discern ads with closely-related domains.

References

[1] B. Allison. An Improved Hierarchical Bayesian Model of Language for Document Classification. In *Proc. of COLING*, pages 25–32, 2008.

[2] H. Chieu and H. Ng. Named Entity Recognition with a Maximum Entropy Approach. In *Proc. of Conf. on Natural Language Learning*, pages 160–163, 2003.

[3] W. Cohen. Fast and Effective Rule Induction. In *Proc. of ICML*, pages 115–123, 1995.

[4] E. Cortez, A. da Silva, M Goncalves, and E. de Moura. ONDUX: On-Demand Unsupervised Learning for Information Extraction. In *Proc. of SIGMOD*, pages 807–818, 2010.

[5] N. Dalvi, R. Kumar, and M. Soliman. Automatic Wrappers for Large Scale Web Extraction. *VLDB Endowment*, 4(4):219–230, 2011.

[6] M. Hall and E. Frank. Combining Naive Bayes and Decision Tables. In *Proc. of Florida Artificial Intelligence Research Society Conf.*, 2008.

[7] R. Khare and Y. An. An Empirical Study on Using Hidden Markov Model for Search Interface Segmentation. In *Proc. of ACM CIKM*, pages 17–26, 2009.

[8] Y. Liu and Y. Zheng. One-Against-All Multi-Class SVM Classification Using Reliability Measures. In *Proc. of IJCNN*, pages 849–854, 2005.

[9] C. Manning, P. Raghavan, and H. Schutze. *Introduction to Information Retrieval*. Cambridge University, 2008.

[10] C. Manning and H. Schutze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 2003.

[11] G. Miao, J. Tatemura, W. Hsiung, A. Sawires, and L. Moser. Extracting Data Records from the Web Using Tag Path Clustering. In *Proc. of WWW*, pages 981–990, 2009.

[12] M. Michelson and C. Knoblock. Creating Relational Data from Unstructured and Ungrammatical Data Sources. *Journal of Artificial Intelligence Research*, 31(1):543–590, 2008.

[13] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[14] H. Nguyen, E. Kang, and J. Freire. Automatically Extracting Form Labels. In *Proc. of IEEE ICDE*, pages 1498–1500, 2008.

[15] S. Raeymaekers, M. Bruynooghe, and J. Bussche. Learning (k, l)-Contextual Tree Languages for Information Extraction from Web Pages. *Machine Learning*, 71(2-3):155–183, 2008.

[16] Q. Rajput and S. Haider. Use of Bayesian Network in Information Extraction from Unstructured Data Sources. *World Academy of Science, Engineering and Technology (WASET)*, 52:325–331, 2009.

[17] D. Sculley and G. Wachman. Relaxed Online SVMs for Spam Filtering. In *Proc. of ACM SIGIR*, pages 415–422, 2007.

[18] X. Song, J. Liu, Y. Cao, C.-Y. Lin, and H.-W. Hon. Automatic Extraction of Web Data Records Containing User-Generated Content. In *Proc. of ACM CIKM*, pages 39–48, 2010.

[19] B. Tang and D. Mazzoni. Multiclass Reduced-Set Support Vector Machines. In *Proc. of ICML*, pages 921–928, 2006.

[20] W. Xu, X. Liu, and Y. Gong. Document Clustering Based on Non-negative Matrix Factorization. In *Proc. of ACM SIGIR*, pages 267–273, 2003.

[21] Y. Yang and J. Pedersen. A Comparative Study on Feature Selection in Text Categorization. In *Proc. of ICML*, pages 412–420, 1997.

[22] J. Zhu, Z. Nie, B. Zhang, and J. Wen. Dynamic Hierarchical Markov Random Fields for Integrated Web Data Extraction. *Machine Learning Research*, 9:1583–1614, 2008.