Easy to Find: A Natural Language Query Processing System on Advertisements Using an Automatically Populated Database

Yiu-Kai Ng[a]
3361 TMCB, Computer Science Department, Brigham Young University, Provo, Utah, USA
ng@compsci.byu.edu

Keywords: Query Processing, Databases, Classification, Data Extraction

Abstract: Many commercial websites, such as Target.com, which aspire to increase client's transactions and thus profits, offer users easy-to-use pull-down menus and/or keyword searching tools to locate advertisements (ads for short) posted at their sites. These websites, however, cannot handle natural language queries, which are formulated for specific information needs and can only be processed properly by natural language query processing (NLQP) systems. We have developed a novel NLQP system, denoted *AdProc*, which retrieves database records that match information specified in ads queries on multiple ads domains. *AdProc* relies on an underlying database (DB), which contains pre-processed (ads) records that provides the source of answers to users' queries. *AdProc* automates the process of populating a DB using online ads and answering user queries on multiple ads domains. Experimental results using ads queries collected through Facebook on a dataset of online ads extracted from Craigslist.org and Coupons.com show that *AdProc* is highly effective in (i) *classifying* online ads, (ii) *labeling*, *extracting*, and *populating* data from ads in natural language into an underlying database *D*, (iii) *assigning* ads queries into their corresponding domains to be processed, and (iv) *retrieving* records in *D* that satisfy the users' information needs.

1 INTRODUCTION

The Web is a perfect forum for information exchange and access, since most websites allow users to freely extract archived and newly-created documents anytime and anywhere. The number of (un-/semi-)structured web pages has made the Web a huge repository of information of various kinds, which include advertisements (ads for short). While users can search for information using keyword-based and phrase-based queries, or even advanced searches that include simple Boolean operators on websites, these websites cannot process complex queries, which can only be handled properly by natural language query processing (NLQP) systems. In this paper, we introduce a closed-domain NLQP system, denoted *AdProc*, which retrieves exactly matched answers to (Boolean, incomplete, or ambiguous) queries on ads. *AdProc* can answer queries on multiple ads domains using (i) an enhanced Naïve Bayes classifier to determine to which *ads domain* a user's query belongs and (ii) an underlying database (DB for short) with populated ads records, which serves as the source of answers to ads queries. Manually populating online data to a DB is not feasible, since it is a labor-intensive, time-consuming, and impractical process. *AdProc* applies the (i) Joint Beta-Binomial classifier to *classify* ads according to their domains, (ii) Support Vector Machine (SVM) to *label* non-stop keywords in ads based on their types, (iii) decision trees to *extract* previously-labeled keywords in an ad that are valid DB attribute values to generate a DB record, and (iv) feature set of attribute values in answering ads queries.

AdProc utilizes the *efficiency* of relational DB systems in managing data (records) and avoids the burden of *analyzing* the syntax and semantics of a user's natural language query Q such that interpreting the semantics of the keywords in Q employs at most the simple context-switching analysis, which facilitates the process of *transforming* the information needs expressed in Q into a SQL query to be evaluated against the underlying data(base records). Furthermore, *AdProc* combines the tasks of *classifying* and *extracting* data from ads into a single, automated process for populating a DB and extracting relevant data.

AdProc applies information retrieval techniques and machine learning approaches, which are simple and widely-adapted, in its design. Compared with using popular deep learning models, the latter require massive data for training, expect high computational power for processing data, and lack of transparency,

[a] https://orcid.org/0000-0002-5680-2796

which are excessive for designing *AdProc*.

The effectiveness and merits of *AdProc* have been experimentally verified in terms of achieving high accuracy of interpretation for complex queries. Furthermore, it has been validated that *QuePR* retrieves relevant results to the user with the accuracy ratio in the upper 90 percentile.

2 RELATED WORK

A number of methods have been proposed in the past in solving the classification problem. Dayanik et al. (Dayanik et al., 2006) perform classification using a Bayesian logistic regression framework that combines domain knowledge and supervised learning, whereas Xue et al. (Xue et al., 2008) extend the Probabilistic Latent Semantic Analysis algorithm by integrating labeled training data and unlabeled test data. As opposed to (Dayanik et al., 2006; Xue et al., 2008), the classifier of *AdProc* adopts a Beta-Binomial model, which considers the varying probability of word occurrence among documents in the same class for text classification. While *AdProc* can classify documents in various domains, the Relaxed Online SVM in (Sculley and Wachman, 2007) has only been evaluated for (spam) email classification, which involves only highly separable data.

Nguyen et al. (Nguyen et al., 2008) introduce a machine learning approach for labeling attributes from web form interfaces. Similar to our proposed tagging method, the labeling approach in (Nguyen et al., 2008) matches a form element (i.e., a keyword in an ad in our case) with its corresponding textual description (i.e., a type in our case). *AdProc*, however, avoids learning structural patterns required by the labeling approach in (Nguyen et al., 2008) for performing the keyword-tagging task.

AdProc relies on Support Vector Machine (SVM) for labeling keywords, which differs from the Hidden Markov Model (HMM) approach in (Khare and An, 2009) that labels components of web interfaces, such as text labels or text boxes. HMMs have been successfully adopted for extracting data from text documents which are later populated into DBs (Liu et al., 2003). Since training a HMM with data from multiple domains is ineffective, Liu et al. (Liu et al., 2003) first group analogous data and train separate HMMs, which however is a complex process.

Rajput and Haider (Rajput and Haider, 2009) apply ontologies, various information extraction techniques, and Bayesian Networks for extracting and predicting missing information from unstructured, ungrammatical, and incoherent data sources, such as online ads extracted from Craigslist.org. Although effective, the proposed model has not been validated for text classification using more than one ads domain.

In developing a closed-domain NLQP system for the construction business, Zhang et al. (Zhang et al., 2004) rely on a thesaurus, which consists of a set of domain-specific concepts and the relations among them organized into a hierarchy, whereas the NLQP system in (Demner-Fushman and Lin, 2007) re-ranks publications retrieved by the PubMed search engine for a given query Q so that the publication abstracts that contain medical terms relevant to the ones in Q are positioned higher in the answer set. Wang and Luo (Wang and Luo, 2009) present a Chinese NLQP system that answers queries in the telecom product domain by (i) applying semantic role tagging and (ii) using Hownet, an online knowledge based which describes the inter-conceptual and inter-attribute relations of Chinese and English words. On the contrary, *AdProc* depends only on a few predefined attribute types to retrieve potential answers to a user query.

3 AN ADS NLQP SYSTEM

In this section, we discuss ads data types (in Section 3.1) and describe the three consecutive tasks of *AdProc*: (i) *classifying* online ads into their respective domains (in Section 3.2), (ii) *tagging* keywords in online ads according to their types (in Section 3.3), and (iii) *extracting* the previously-tagged non-Types IV keywords in ads and populate them as attribute values in the corresponding DB records (in Section 3.4), which fully automate the process of extracting online ads data to generate the underlying DB records.

3.1 Data Types Used in *AdProc*

In populating online ads and answering queries in multiple domains, *AdProc* considers various attribute types proposed in this paper.

- *Type I* attribute values in an ad A yield the *unique identifier* of A that are required values to be included in A (its DB record, respectively). Type I attributes are *primary-indexed* fields of the relational schema which defines the corresponding ads domain. Sample Type I attributes in the Cars-for-Sale ads domain are "Maker" and "Model", and "Toyota" and "Camry" are one of their respective values.
- *Type II* attribute values describe the *properties* of an ad A, which are not required values in A. Type II attributes are *secondary-indexed* fields in the corresponding relational schema. "Color" is a Type II attribute in the Car-for-Sale ads domain, and "Blue" is one of its domain values.

- *Type III* attribute values specify the *quantitative values* in an ad. A sample *Type III* attribute is "Salary" in the Jobs ads domain, and $70,000 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 in ads, such as "large", which are not extracted as attribute values from any online ads.

3.2 Classification of Advertisements

In classifying online ads, we adopt a Naïve Bayes classifier, which is a popular text classification approach, since it is simple, easy to implement, robust, highly scalable, and domain independent.

3.2.1 Joint Beta-Binomial Sampling Model

The well-known Naïve Bayes' Theorem, which considers the conditional distribution of the class variable, computes the probability of assigning the natural class c to a document d as

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)} \quad (1)$$

To estimate $P(d \mid c)$ in the Bayes' Theorem, we use the *Joint Beta-Binomial Sampling Model* introduced in (Allison, 2008), denoted JBBSM, which considers the "burstiness" of a word, i.e., a word is more likely to occur again in d if it has already appeared once in d. JBBSM 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[1]:

$$P_{bb}(d_j \mid \alpha_j, \beta_j) = \binom{n}{d_j} \frac{B(d_j+\alpha_j, n-d_j+\beta_j)}{B(\alpha_j,\beta_j)} \quad (2)$$

where n is the length of d, which is the total number of keyword counts in d, B is the Beta function of JBBSM, and α_j and β_j are the parameters[2] that estimate the presence and absence of d_j in a particular class c, respectively.

The following equations estimate the parameter values of α_j and β_j in the Beta-Binomial model as defined in (Allison, 2008):

$$\frac{\alpha_j}{\alpha_j+\beta_j} = \frac{\sum_{i=1}^{k} \hat{\theta_{i,j}}}{k} \quad (3)$$

$$\frac{\alpha_j\beta_j}{(\alpha_j+\beta_j)^2(\alpha_j+\beta_j+1)} = \frac{\sum_{i=1}^{k} (\hat{\theta_{i,j}} - E[\theta_j])^2}{k} \quad (4)$$

where k is the size of the document collection used for training the classification model, $\theta_{i,j}$ is the expected distribution of the j^{th} keyword in the i^{th} $(1 \leq i \leq k)$ document in the collection, $\hat{\theta_{i,j}}$ is the maximum likelihood estimate of the parameter $\theta_{i,j}$, θ_j is the probability of selecting the j^{th} word in d, $E[\theta_j]$ $(= \frac{1}{k} \times \sum_{i=1}^{k} \theta_{i,j})$ is the expected value for the distribution over θ_j, and $(\hat{\theta_{i,j}} - E[\theta_j])$ yields the error in estimating the parameter $\theta_{i,j}$.

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

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

where α and β are parameters of the Beta-Binomial distribution[3] of d, and $P_{bb}(d_j \mid \alpha_j, \beta_j)$ is as defined in Equation 2.

In choosing the ads domain to which an ad d should be assigned, we compute the conditional probability $P(c \mid d)$ for each predefined ad domain. We assign to d the domain that yields the highest $P(c \mid d)$ among the set of predefined ads domains, C, i.e., $Class(d) = argmax_{c \in C} P(c \mid d)$.

3.2.2 Feature Selection

As claimed by Yang and Petersen (Yang and Pedersen, 1997), one of the main 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 classifier algorithms in terms of computational time. In solving this problem, we apply different feature selection strategies (given below), which select a subset of keywords to represent ads in a set of predefined ads domains, without affecting the accuracy of the chosen classifier. Furthermore, as claimed by Chouaib et al. (Chouaib et al., 2009), feature selection strategies are often applied to reduce irrelevant/misleading features.

During the feature selection process, we first remove *stopwords*, i.e., words with little meanings, which often do not represent the content of an ad, as well as numerical values. Hereafter, we apply the *gain* approach in (Yang and Pedersen, 1997), which defines a criterion for feature selection often adopted in the field of machine learning. Information gain, as defined below, determines the "goodness" of a keyword globally for classification, which yields keywords that are the most representative of ads in various predefined domains.

[1] Documents in JBBSM are represented as vectors of count-valued random variables, and thus in Equation 2, d_j denotes the frequency count of the j^{th} keyword in d.

[2] α_j and β_j are the parameters associated with a particular word j, which are computed using Equations 3 and 4 for each of the natural classes.

[3] α and β (α_j and β_j, respectively) are the parameters of the distribution of d (keyword j in d, respectively).

$$G(t) = -\sum_{i=1}^{m} P(c_i) log P(c_i) + P(t) \sum_{i=1}^{m} P(c_i|t) log P(c_i|t) + P(\bar{t}) \sum_{i=1}^{m} P(c_i|\bar{t}) \; log \; P(c_i|\bar{t}) \quad (6)$$

where m is the number of distinct ads domains, $P(c_i)$ is the probability of domain c_i, $P(t)$ ($P(\bar{t})$, respectively) is the probability of occurrence (absence, respectively) of keyword t, and $P(c_i \mid t)$ ($P(c_i \mid \bar{t})$, respectively) is the probability of domain c_i given that t is present (absent, respectively).

After computing the gain of all the distinct keywords in a collection of ads used for training purpose, we select the top n keywords that have the *highest gain* for representing ads in the set of predefined ads domains. In defining the appropriate n, we conducted an empirical study using a total of 80,000 randomly selected online ads belonged to the eight domains introduced in Section 5.1. We considered alternative values of n, such that $n \in \{50, 100, 200, 500, 1000, 2000\}$, and set n = 1000. By using 1000 keywords on the 80,000 ads we achieve the *highest* classification accuracy and still maintain the classification processing time on ads within a minute.

3.3 Keyword Tagging Based on Types

To tag each non-stop, non-numerical[4] keyword in online ads according to their types, we rely on the Support Vector Machine (SVM) approach, since SVM is a robust methodology which has been shown to yield state of the art performance on classification (Sculley and Wachman, 2007). SVM constructs hyper-planes as decision surfaces in a higher dimension space so that data becomes linearly separable and maintains a maximum margin of separation between positive and negative examples, i.e., binary training instances.

3.3.1 RBF Kernel

In implementing SVM for tagging ads keywords according to their types, we adopted Radial Basis Function (RBF) in Equation 7 as the kernel function for the SVM, since RBF is one of the most typical kernels.

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

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. In our implementation of SVM, we have empirically determined the value of the parameter σ to be 500, which yields a smoother decision surface, since an RBF with σ set to be 500 allows a support vector to have a larger area of influence.

3.3.2 Multiclass-SVM

In tagging keywords based on their types, we cannot apply directly a binary SVM, since the adopted SVM must handle more than two types of attribute values. We consider one-against-all and one-against-one for solving the multi-class problem using binary SVM. As claimed by (Liu and Zheng, 2005), the most common strategy, which is the one we adopt, in implementing SVM is one-against-all.

Given j (≥ 1) different predefined types, the one-against-all approach constructs j binary SVM classifiers, each of which separates one type from the rest. Hereafter, the j^{th} SVM is trained using the training instances in which the ones belonged to the j^{th} type are given positive labels and the remaining instances negative ones (Liu and Zheng, 2005).

3.3.3 Feature Representation

To train our multi-class SVM, each training instance, i.e., input vector, is a feature-vector associated with a non-stop, non-numerical keyword k in an ad, in which a "1" is assigned to each feature f (introduced below) if f applies to k and is assigned a value "0", otherwise.

- **Is-Plural**. This feature is set to '1' if a keyword is in a *plural* form and is '0', otherwise. Type I attribute values tend to be expressed in the singular form.
- **Is-Capitalized**. We have observed that, in general, the first character in keywords that represent Type I attribute values is *capitalized*, and this feature is assigned the value of '1', and '0', otherwise.
- **Is-Style**. This feature is set to '1' if the keyword being evaluated is either *bolded* or *italicized* in an ad and is '0', otherwise. The most important attribute values in an ad tend to be either bolded or italicized and are Type I attribute values.
- **In-Title**. Since the most descriptive attribute values of an ad d appear in the *title* or *first sentence* of d, this feature is assigned a value '1' if the keyword being analyzed is in the title or in the first sentence of d and is given a '0', otherwise. In-Title values are Type 1 attribute values.
- **Is-Adjective**. In implementing this feature we rely on the part-of-speech (POS) tagger[6], which assigns POS, such as noun or verb, to (key)words (in ads).

[4] Numerical keywords are tagged as Type III by default.

[5] x_i and x_j are two input vectors. In our case, each input vector represents the heuristics, i.e., features (to be introduced in Section 3.3.3), of a keyword in an ad.

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

A feature is assigned a '1' if the word is given an *"adjective"* POS tag and '0', otherwise. Most of Type II attribute values are adjectives which describe essential information of an ad.

- **Is-Measurement**. This feature takes a value of '1' if the keyword being evaluated represents a *measurement*, e.g., $, usd, miles, square feet, or inches, and '0', otherwise. The set of measurement terms was extracted from Hobby[7], a website that lists units of measurements for different categories, such as length, area, power, or speed. This feature is a clear indicator of Type III attribute values.
- **Is-Alphanumeric**. This feature is assigned a '1' if the keyword being evaluated is alphanumerical and is assigned a '0', otherwise. This feature assists in identifying Type II attribute values.
- **Is-Location**. This feature relies on a set of predefined locations to decide whether the keyword being evaluated is a *location*, in which case the feature is set to '1' and '0', otherwise. The list of locations, e.g., US cities, was extracted from Wikipedia[8].
- **Is-Acronym**. This feature is set to '1' if the word being evaluated represents an *acronym* and is '0', otherwise. In determining whether a word is an acronym, we adapt the approach proposed by Chieu and Ng (Chieu and Ng, 2003) to look for sequences of capitalized words in an ad d that match (potential) acronyms found in d. If a sequence of capitalized letters in an ad matches the word k being evaluated, then we treat k as an acronym.

3.4 Extracting Data for Query Answering

Having identified the domain of an ad d (in Section 3.2) and assigned the corresponding type to each (non-stop) keyword k in d (in Section 3.3), *AdProc* proceeds to extract data in d and populate the underlying DB[9] employed by *AdProc* in answering queries on ads. In this regard, extracting information from unstructured data sources is a *classification* process, since k is either assigned as a value to its corresponding attribute in the DB record of d or a "not-valid" label, indicating that k will not be populated to the DB.

We apply the C4.5 decision tree algorithm (Mitchell, 1997) to construct a decision tree for extracting data from online ads. Decision trees are widely-used and employ a simple classification technique for inductive inference which utilizes the decision process as a set of if-then rules. The algorithm applies the divide-and-conquer strategy and recursively partitions the training instances into subsets according to a splitting criterion (test separation), which is predefined prior to constructing the tree. We construct a decision tree for each previously-defined ad domain (schema).

Prior to extracting and populating data in online ads to the underlying DB, we first define the set of training instances S for constructing the corresponding decision tree. A training instance in S, which is associated with a keyword k in an online ad d, is a sextuple of the form $<f_1, f_2, f_3, f_4, f_5, A>$, where f_i $(1 \leq i \leq 5)$ is one of the possible values which can be assigned to i, where $i \in$ *Feature-Set* with features defined below, and A is one of the DB attributes in the corresponding schema for which k is a valid value of A or A is the label "not-valid" (when k is not a valid attribute value). Each feature i in *Feature-Set* is defined below.

1. **Keyword-Type** is the attribute type of k.
2. **Previous-Keyword-Type** is the attribute type of the keyword immediately preceding k in d.
3. **Post-Keyword-Type** is the attribute type of the keyword immediately following k in d.
4. **Previous-Keyword-Attribute** is the DB attribute of the keyword immediately preceding k in d.
5. **Closest-Type-IV** is the Type IV keyword in d that is closest to k.

The possible values of *Keyword-Type*, *Previous-Keyword-Type*, and *Post-Keyword-Type* are Type I - IV, whereas the possible values of *Previous-Keyword-Attribute* include the set of attributes in the DB schema corresponding to the domain of d and the label "not-valid".

The features in *Feature-Set* are defined for capturing essential information to accurately identify each keyword k as an attribute value, which are based on the context in which k appears, i.e., based on other keywords that appear before and after k, in an online ad. Moreover, the *Closest-Type-IV* feature identifies keywords commonly associated with numerical values for data extraction. For example, given the phrase "25 acres", we rely on the keyword "acres", i.e., the closest Type IV keyword to "25", in assigning the numerical value "25" to its corresponding DB attribute.

AdProc constructs a decision tree for each ads domain of interest to perform the data extraction task. A major task in constructing a decision tree is to establish the criterion used for identifying the feature (in *Feature-Set*) that is the most effective in splitting training instances into different groups, i.e., various DB attributes in our case. A goodness measure used

[7] www.hobbyprojects.com/dictionary_of_units.html

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

[9] The DB schema for each ad domain is defined prior to invoking *AdProc* to automate the data extraction process.

for indicating the classification power of a split is the *information gain*, which we adapt.

$Information\ Gain(S, F)$ =

$$Entropy(S) - \sum_{f \in Values(F)} \frac{|S_f|}{S} Entropy(S_f) \quad (8)$$

where F is a feature, $Values(F)$ 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 $Entropy(S)$ is defined in Equation 9.

$$Entropy(S) = \sum_{i=1}^{|A|} -p_i \ log_2 \ p_i \quad (9)$$

where p_i is the number of instances in S that include attribute i, and $|A|$ is the number of attributes in the corresponding schema plus one, the "not-valid" label.

4 GENERATING SQL QUERIES

Constraints specified by users in natural language queries must be translated into database query languages for execution. For the implementation of *AdProc*, we selected SQL as the query language of translation output. There are three common components to a *AdProc* SQL query: the SELECT-FROM, WHERE, and ORDER BY clauses.

4.1 The SELECT-FROM Clause

During the classification process, *AdProc* determines the category of the given query Q, which dictates the corresponding table name in the underlying database scheme. To retrieve all the relevant information to the user, *AdProc* extracts all the columns of the table for each ad that satisfies the constraints specified in Q using the wildcard (*), i.e., SELECT *.

4.2 The WHERE Clause

The WHERE clause is a logical combination of each of the parsed constraints stated in a user query. *AdProc* links each constraint to the corresponding attribute/column in the database table in the WHERE clause. Since in a natural language query Q the user often does not identify the correct table column, *AdProc* make inferences of the correct attribute based on the information provided in Q. *AdProc* performs the matching based on the "Type" constraints.

- For Type I and Type II constraints, the corresponding table column can be determined by a local search of the value, called *value matching*, using a *trie* data structure. For example, if "Toyota" and "Camry" are specified as constraints, *AdProc* looks up the corresponding trie (that is periodically updated), which is created according to the column values of the DB tables, to determine whether the values appear and their corresponding column names, i.e., "Make" and "Model", respectively. Value matching has the potential to return multiple rows for a single value, since a constraint value may appear in different rows in the corresponding table.
- For Type III constraints, they cannot be identified by using their corresponding attributes based on value matching due to the infinite nature of numbers. A similar approach to the value matching, however, can be performed by using the unit of the Type III constraint instead of the value alone. For this strategy to be effective, two conditions must be satisfied: (i) each Type III attribute in the DB table includes a list of related units (which is anticipated) and (ii) there is an attached unit for each Type III constraint in a query (which is often the case).

The WHERE clause of the SQL query is composed of the translated constraints of Types I, II, and/or III. The values of Type I and Type II constraints are compared with the column values in SQL using the `LIKE` operator, which requires that the given substring appears in all matches. For Type III comparison constraints, if a comparison operator is not given, the equality operation (=) is assumed. For *range* Type III comparisons, the `BETWEEN` operator is used.

4.3 The ORDER BY Clause

The "ORDER BY" clause is created for *implicit sorting order* of Type III comparisons. These implicit constraints come from sorting each result by its distance from the optimal value of the comparison. Equality comparisons have an optimal value of the number to be matched on, and they are sorted by `ABS(x-y)`, where x is a Type III attribute value and y is an expected value. $<$ or $\leq$ operations have an optimal value of 0, whereas $>$ or $\geq$ operations have an optimal value of infinity, i.e., ∞. *Ranges* have no optimal value, and therefore do not create an implicit sorting constraint. Consider the constraints specified in Q, "*Toyota blue Camry or white Corolla 2015 less than $15000 in NY*". In the WHERE clause, the constraint "Toyota" is determined to be car `Make`, "blue" and "white" are `colors`, whereas "Camry" and "Corolla" are car `Models`. "2015" is assumed to be a year, since it is within the range of `Years`. The constraint "less than $15K" matches with `Cost` because of the dollar sign. Lastly, "NY", an abbreviation of "New York", is matched in the `Cars` table as a `Location` value. Hereafter, the ORDER BY clause is constructed in which

"2015" is the optimal value for `Years` and `Price` is minimized. The constructed SQL statement for Q is

```
SELECT * FROM Cars
WHERE (make LIKE "Toyota" AND ((color LIKE
  "blue" AND model LIKE "Camry") OR (color
  LIKE "white" AND model LIKE "Corolla"))
  AND year = 2015 AND price < 15000 AND
  location LIKE "New York")
ORDER BY ABS(year - 2015), price;
```

5 EXPERIMENTAL RESULTS

In this section, we assess the performance of *AdProc* by evaluating its major tasks in populating ads and answering ads queries. We (i) first introduce the dataset and metrics used for the performance evaluation (in Sections 5.1 and 5.2, respectively), and (ii) analyze its accuracy in classifying (in Section 5.3), tagging (in Section 5.4), extracting ads data (in Section 5.5), and evaluating its populating process (in Section 5.6), besides retrieving answers to queries (in Section 5.7).

5.1 The Dataset

To the best of our knowledge, there is no existing dataset for evaluating classification, labeling, and/or data extraction of online ads. Thus, we have created our own dataset, denoted *EData*, for assessing the performance of *AdProc* based on it.

EData consists of 80,000 uniformly distributed online ads, with 10,000 ads in each of the eight different ads domains, which were randomly extracted from Craigslist.org, a popular source of unstructured ads in various domains, and Coupons.com. The eight domains are cars(-for-sale), food (coupons), furniture, houses(-for-sale), jewelry, C(omputer) S(cience) jobs, motorcycles(-for-sale), and music(al instruments). These chosen ads domains vary in terms of their (i) diversity, which include ads in jobs, food, housing, transportation, and entertainment that offer our daily needs, (ii) ad size, from arbitrary long (such as houses) ads to relatively short (such as jewelry) ads, and (iii) word distribution, i.e., different word usage associated with different types of ads. Moreover, to test the versatility of *AdProc*, domains that are closely related, e.g., cars and motorcycles, and diverse, e.g., food and jobs, in content and nature were chosen.

To obtain a representative set of queries for verifying the *accuracy* of *AdProc* from (i) classifying ads queries to their corresponding domains to (ii) retrieving answers in multiple ads domains, we collected 1,750 queries on the eight ads domains from *Facebook users* between January 9, 2023 and May 11, 2023. These Facebook users were recruited by the (friends of the) authors for the empirical studies.

5.2 Evaluation Measures

To evaluate the effectiveness of *AdProc* in classifying ads, tagging keywords, extracting ads data, and assigning users' queries to their corresponding domains, we rely on the accuracy ratio defined below.

$$Accuracy = \frac{Correctly\_classified\_instances}{|Dataset|} \quad (10)$$

where $|Dataset|$ is the size of a given dataset, i.e., the total number of instances considered for evaluation, and $Correctly\_classified\_instances$ is the number of instances correctly classified/labeled/extracted by the corresponding evaluated method.

To measure the accuracy of *AdProc* in *retrieving* correct answers, we compare the answers generated by *AdProc* on each one of the 1,750 Facebook queries with the ones on the same answer set provided by a group of 350 independent Facebook appraisers, which serve as the gold-standard, using the Spearman Rank Correlation Coefficient (SRCC) (Callan and Connell, 2001). SRCC returns -1 or 1, where '1' indicates that the two given results to be compared are identical and '-1' implies that the results are not related.

$$SRCC = \frac{1 - \frac{6}{n^3-n} \times (\sum d_i^2 + \frac{1}{12}\sum(f_k^3 - f_k))}{\sqrt{1 - \frac{\sum f_k^3 - f_k}{n^3-n}}} \quad (11)$$

where d_i is the difference between the two results for the same query i, n is the total number of queries, f_k is the number of ties in the k^{th} (≥ 1) group of ties created by the appraisers.

To measure the performance of *AdProc* in correctly transforming the information needs expressed in a user's query into a correct SQL statement, we use *precision* as the evaluation metric.

$$Precision = \frac{\text{\#Correct Matches Retrieved by } AdProc}{\text{\#Records Retrieved by } SQL} \quad (12)$$

where a *correct match* is a retrieved answer that satisfies all the search criteria specified in a query.

5.3 Classification of Ads

Prior to performing the classification task, feature selection introduced in Section 3.2.2 is applied to reduce the size of the vocabulary, i.e., the number of distinct keywords in *EData*, so that the top n (≥ 1) non-stopwords and alphanumerical keywords are chosen for representing ads in the corresponding domains. Figure 1 depicts the accuracy ratio and classification time for determining the ideal value n (= 1,000).

Figure 2 shows that the accuracy ratio of classifying *EData* ads in each domain, and the average achieved by *AdProc* are in the ninety percentile.

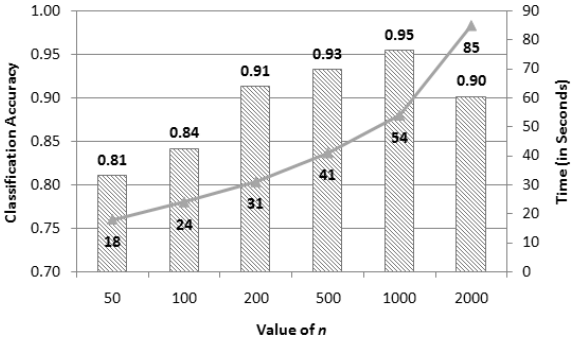

Figure 1: Accuracy ratio and classification time for the 80,000 ads used for determining the ideal value of n, the size of the reduced vocabulary for ads classification.

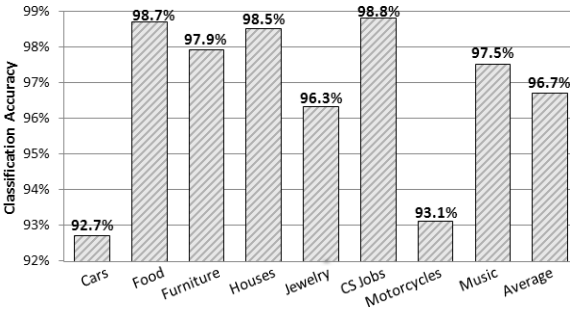

Figure 2: Classification accuracy of ads in *EData*.

Ads in the jobs, food coupons, and houses domains achieve higher classification accuracy, whereas cars and motorcycles achieve lower accuracy ratios. We observed that domains in which (i) ads descriptions are *short*, such as jewelry ads, and (ii) their word usage is *similar* to the one in others, such as cars and motorcycles ads, tend to yield *lower* accuracy.

To analyze the effectiveness of the proposed classifier in assigning ads to their corresponding domains, we used a *confusion matrix*, denoted CF. CF (= $[I_{i,j}]$) displays the total number of instances $I_{i,j}$ ($i \neq j$) in class c_i which have been *misclassified* into class c_j, and the principal diagonal of CF denotes the total number of instances *correctly* classified. Table 1 shows that most of the classification errors occur when any two ads domains share a similar probability distribution on a considerable number of keywords, e.g., cars and motorcycles, as mentioned earlier.

We have verified the effectiveness of the classifier used by *AdProc* to a greater extent by comparing its classification accuracy with two other well-known classifiers, the multinomial Naïve Bayes (*MNB*) classifier (McCallum and Nigam, 1998) and the SVM extracted from WEKA[10], an open-source collection of machine learning algorithms, based on the 80,000 Craigslist.org and Coupons.com ads in *EData*.

MNB follows the premises of the Naïve Bayes

[10]www.cs.waikato.ac.nz/ml/weka/

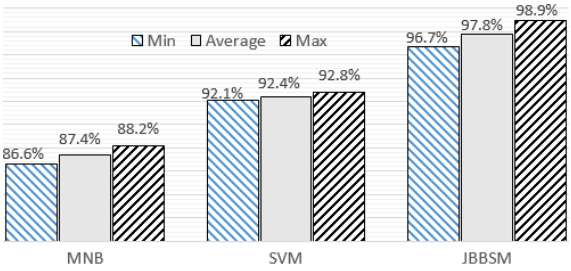

Figure 3: Classification accuracy obtained by using MNB, SVM, and the classifier of *AdProc*, JBBSM, on *EData*.

classifier (as discussed in Section 3.2) in assigning a document to a class. As opposed to the Joint Beta-Binomial Sampling model (JBBSM) introduced in Section 3.2.1, MNB computes the probability of a keyword w_t in a class c_j, $P(w_t|c_j)$, by considering the frequency of keyword occurrence in each document, whereas SVM is as defined in Section 3.3. Figure 3 shows that the classifier of *AdProc*, i.e., JBBSM, outperforms MNB and SVM in assigning ads in *EData* to their corresponding domain, and the results are statistically significant ($p < 0.05$) based on the Wilcoxon Signed-Rank test (Rey and Neuhäuser, 2011).

To further assess the effectiveness of *AdProc* in ads classification, we repeated the conducted experiments two more times, using two new subsets of 80,000 ads uniformly distributed among the eight domains. The overall evaluation of *AdProc* in terms ads classification is shown in Table 2.

5.4 Tagging Keywords Based on Types

To assess the effectiveness of the multi-class SVM approach (introduced in Section 3.3) on tagging keywords in ads according to their corresponding types, we first created training/test instances using each of the ads in *EData* and for each non-stop, non-numerical keyword in each ad we extracted the corresponding features, as discussed in Section 3.3.3.

As shown in Figure 4, the overall accuracy of our SVM in assigning Types I-IV tags to keywords in ads is in the 90 percentile. Most of the misclassification errors occur when attribute values that should be assigned a Type I tag are incorrectly labeled as Type II. When none of the keywords in an ad are bolded, italicized, or capitalized, the values assigned to features, such as *Is-Style* or *Is-capitalized*, are the same for keywords of Types I and II, which causes the misclassification. Figure 4 also reveals that Type II attribute values yields the lowest accuracy, since they are often labeled as Type IV due to their proximity in ads.

To further assess *AdProc*'s tagger, we have compared its performance with two other widely-used approaches, C4.5 decision tree classifier and an artificial neural network, since decision trees and neural net-

Table 1: The confusion matrix generated according to (in)correctly classified ads in *EData*

Ads Domain	Cars	Food	Furniture	Houses	Jewelry	CS Jobs	Motorcycles	Music
Cars	927	1	3	2	0	0	67	0
Food	0	987	3	2	6	1	1	0
Furniture	8	2	979	5	1	0	0	5
Houses	1	1	6	985	1	4	0	2
Jewelry	4	4	5	8	963	5	5	6
CS Jobs	0	1	0	2	2	988	0	7
Motorcycles	58	1	3	3	1	1	931	2
Music	1	3	4	5	3	7	2	975

Table 2: Averaged accuracy for *AdProc*, as well as the classification approaches used for comparison.

Classification Approach	Accuracy
MNB	87.4 +/- 0.8
SVM	92.4 +/- 0.4
AdProc	97.8 +/- 1.1

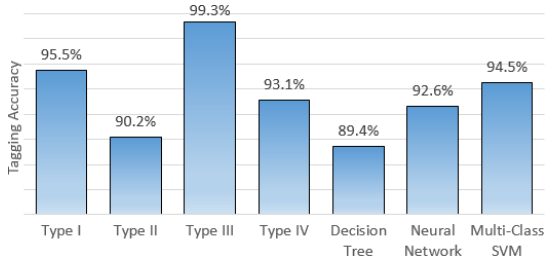

Figure 4: Performance evaluation using *AdProc*'s multi-class SVM, as well as alternative machine learning approaches, for tagging keywords.

works are frequently adopted to solve machine learning tasks, such as for text classification and labeling (Mitchell, 1997). Figure 4 shows that the multi-class SVM of *AdProc* outperforms the two methods for tagging keywords in ads according to their types, and based on the Wilcoxon Signed-Rank test, the results are statistically significant ($p < 0.04$).

5.5 Extracting Ads Data

To determine the effectiveness of the decision-tree based approach (introduced in Section 3.4) which assigns non-stop keywords in ads that are valid attribute values to their corresponding DB attributes, we first created training/test instances using the ads in *EData*. In constructing the instances, we considered the (i) domain assigned to each ad in *EData*, (ii) type of each non-stop keyword in the ads, and (iii) features defined in Section 3.4. The set of training/test instances includes approximately 135,400 feature vectors, one for each non-stop keyword in *EData* ads.

Figure 5 shows the high accuracy ratio of *AdProc* in assigning valid attribute values to their corresponding DB attributes in different ads domains. On the average, the decision-tree based approach achieves

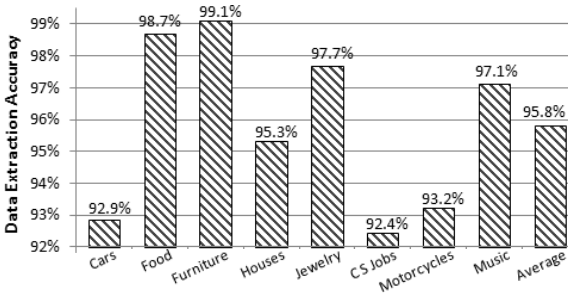

Figure 5: Accuracy ratios of data extraction computed for the eight domains in our empirical study.

95.8% accuracy. Based on the conducted empirical study, we have observed that the accuracy ratios for ads domains that contain a *large* number of attributes are *lower* compared with others with a *smaller* number. This is because the *larger* the number of DB attributes, proportionally the *lower* the number of available instances in any training set that describe a particular DB attribute to train the decision tree, which translates into *lower* accuracy ratios in correctly assigning values to the attribute. Furthermore, (i) keywords of Types I and II in the cars and motorcycles 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 accuracy of each of these two domains is among the lowest of the eight domains. This is caused by the common (numerical) Type III attribute values which are assigned to incorrect DB attributes with the same or compatible domain(s), e.g., in motorcycles ads, '2000' is assigned to the attribute 'Year', instead of the attribute 'Miles'.

We have compared the performance of *AdProc*, in terms of extracting data from online ads to populate the DB, with the WEKA implementation of two machine learning approaches: the Decision Tables Naïve Bayes approach (DTNB) in (Hall and Frank, 2008) and the Rule Induction approach in (Cohen, 1995), denoted JRIP. DTNB is a hybrid method that combines two well-established approaches: decision tables and Naïve Bayes classifiers. JRIP, on the other hand, is a bottom-up method such that given a class,

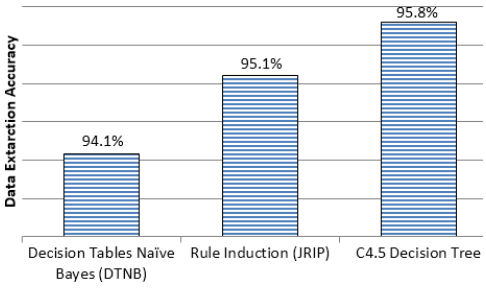

Figure 6: Performance evaluation using the decision trees of *AdProc* and other machine learning methods for extracting data from ads to create DB records.

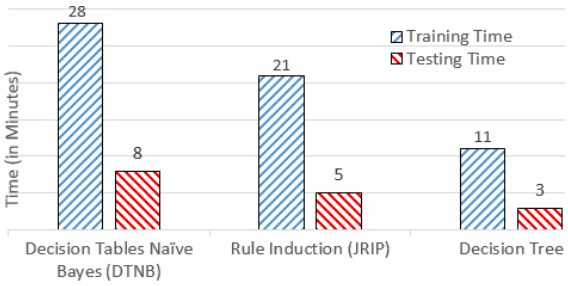

Figure 7: Training/Testing time of DTNB, JRIP, and decision tree using the instances created for non-stop keywords in *EData* for data extraction.

i.e., a DB attribute A, it finds the set of rules that cover all the members of that class, i.e., all the training instances associated with A.

As shown in Figure 6, the decision-tree based approach of *AdProc* outperforms the alternative approaches for extracting data from online ads. Although the difference in accuracy between JRIP and Decision Tree is less than 1%, *AdProc* is *simpler* in terms of implementation, which has been verified. Even though *AdProc* does not outperform JRIP, it surpasses DTNB and the results are statistical significant ($p < 0.05$) based on the Wilcoxon Signed-Rank test.

In comparing with DTNB and JRIP on extracting data from online ads to populate the underlying DB, Figure 7 shows that the training and testing time of Decision Tree on *EData* are reduced on an average by 45% and 46%, respectively. These results verify that *AdPorc* is an efficient tool for data extraction.

5.6 Accuracy in Populating the DB

To assess the overall performance of *AdProc*, in terms of its *accuracy* in populating the DB with extracted ads data, we first created a new collection of ads, denoted *TData*. *TData* consist of 8,000 Craigslist.org and Coupons.com ads, uniformly distributed among the eight ads domains. A DB record d created by *AdProc* is treated as *incorrect* if (i) at least one valid attribute value in the ad used for creating d is assigned

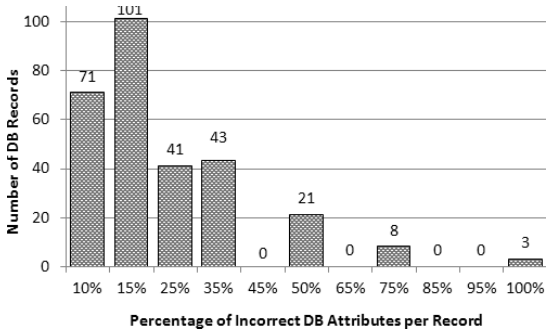

Figure 8: Error distribution in terms of percentages of attribute values incorrectly assigned to DB attributes in the 288 incorrectly created DB records.

to a wrong DB attribute in d or not assigned to d, or (ii) a Type IV, non-essential attribute, value in an ad is assigned to an attribute in d. The calculated *accuracy ratio* of correct DB records generated by *AdProc* on the ads in *TData* is 80.3%.

Besides measuring the accuracy 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 corresponding DB attributes. The experimental results show that *AdProc* correctly assigned 96.4% ($= \frac{8{,}000-288}{8{,}000}$) of the attribute values in *TData*. As shown in Figure 8[11], most of the incorrect DB records include a low percentage, i.e., between 10% and 15%, of incorrectly assigned attributes values. In fact, each incorrectly created record includes at least 85% correctly assigned attribute values.

Based on the conducted experiments, we draw the conclusion that *AdProc* is *highly accurate* in assigning keywords in an ad A to its corresponding attribute in the DB record of A (according to the predefined ad schema to which A belongs), since close to 90% of the DB records are either correctly created (i.e., 80.3%) or have at most 15% of invalid attribute values in their DB attributes as shown in Figure 8.

5.7 Accuracy on Processing Queries

In verifying the accuracy on answers retrieved by *AdProc*, we computed the SRCC values on the answers to each one of the 1,750 Facebook queries in multiple ads domains compiled by the appraisers. As shown in Figure 9, regardless of the ads domain being evaluated, *AdProc* achieves an accuracy for each ads domain that is alike the *gold standard* established by the 350 appraisers. The average SRCC value on the

[11] If the percentage of incorrect attribute values in a given DB record R falls in between the interval of any two percentages shown in Figure 8, R is counted towards the incorrect number of records rounded to the nearest percentage.

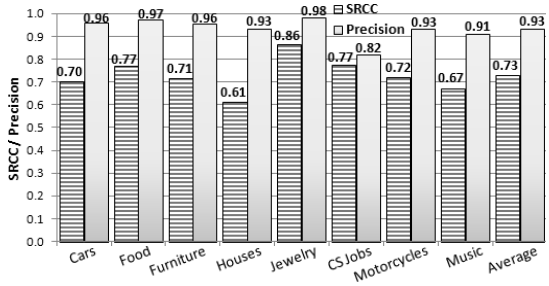

Figure 9: Average SRCC values and Precision computed for the eight ads domains.

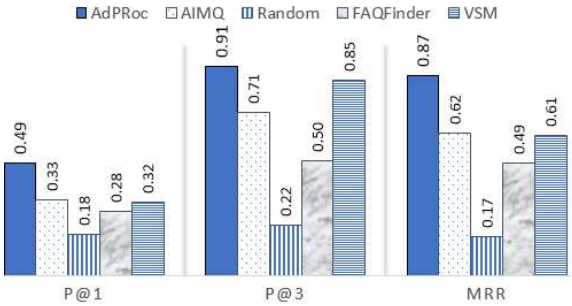

Figure 10: *Precision@K* (K = 1, 3) and *MRR* scores on the (top-3) answers achieved by *AdProc* and other query processing approaches for the 1,750 test queries

eight domains, which is in the 73%, verifies that the retrieval strategy adopted by *AdProc* is reliable. Figure 9 also includes the *average precision* of the DB records retrieved (up till the 15^{th}) for each one of the 1,750 Facebook queries in the eight ads domains with an *average precision* of 93%.

To further measure the performance of *AdProc* in retrieving answers that match the constraints specified in users' natural language queries, we randomly selected one third of the queries created by our Facebook users, i.e., 1,750, and processed them using *AdProc*. The evaluation metrics for measuring the correctness of retrieving exactly-matched answers to a user query are *precision* (P), *recall* (R), and *F-measure*, where a *correct match* is a retrieved (up till the 15^{th}) DB record that satisfies all the search criteria specified in a question. We measured *precision* based on the correct (up till the 15^{th}) DB records retrieved, and *recall* without restricting the total number of DB records retrieved. Since it can be biased to measure the effectiveness of *AdProc* using *precision* and *recall* alone, we consider the *F-measure* which combines *precision* and *recall* with the same weight assigned to the two measures. *AdProc* achieves 94.2%, 93.3%, and 93.9% for the averaged P, R, and *F-measure*, respectively. We found that most of the test questions yield 100% for P and R, whereas a few yield 0%, i.e., answers are either correct or incorrect.

5.7.1 Existing Approaches to be Compared

Besides verifying the accuracy of *AdProc* based on *SRCC*, P, R, and *F-measure* for answering natural language queries, we present four query processing approaches to be compared with *AdProc* below.

Random Processing (Meng et al., 2009) shuffles all the retrieved DB records into some potentially new order by using a *random number generator*. In this case, we implemented the number generator using the pseudorandomness from Python's random module, which served as a useful baseline measure.

FAQFinder (Burke et al., 1997) uses *TF-IDF* for computing the *similarity* of different results to the constraints specified in user queries. Generally, *TF* is the number of times an ad fulfills some constraint, and *IDF* is the relative rarity of that constraint being fulfilled by other ads.

Cosine similarity (Li and Han, 2013) relies on the Vector Space Model (VSM). Each ad is represented as a vector of numeric values, where each value indicates whether the correspondent constraint is satisfied, which is '1' if it is, and '0' otherwise, and the score of the ad is calculated as the cosine similarity between the ad vector and the query constraint vector.

AIMQ (Nambiar and Kambhampati, 2006) depends on attribute-value pairs (denoted AV-pairs) to generate the associated supertuple of each attribute. A *supertuple* is an inferred DB tuple A that contains a set of attribute values, each of which includes a summary of values in the corresponding table column, and is used for calculating the similarity of categorical attributes. AIMQ determines the similarity between query Q and a DB tuple (i.e., record) A.

5.7.2 Comparison of Performance Evaluation

To avoid bias, the top-3 answers for each one of the 1,750 randomly-selected test queries generated by each of the five natural language query processing approaches were evaluated by the 350 Facebook appraisers. The appraisers were asked to determine the *relevance* of the 15 chosen results for the corresponding query. The study was performed between May 20 and May 27, 2023. The metric $P@1$, $P@3$, and *MRR* were computed based on the evaluation provided by the 135 Facebook users on the 1,750 test cases, which serve as the *ground truth* for this empirical study.

As shown in Figure 10, *QuePR* outperforms the other four querying systems based on $P@1$, $P@3$, and *MRR*, which verifies the *effectiveness* of *AdProc*. The results are *statistically significant* based on the Wilcoxon Signed-Ranks Test ($p < 0.01$).

Among all the five approaches, the $P@1$, $P@3$, and *MRR* values for FAQFinder are the *lowest*, ex-

cept the Random approach, since FAQFinder uses a simple method that does not compare numerical attributes. On individual category, we observed that the lowest scores on the three measures for *AdProc* occur in the jobs category. For this category, appraisers did not consider the answers based on their similarity to the original query. For example, a Java programmer job is closely related to a C++ programmer job, but the appraisers considered the answers based on which result is more relevant to their own expertise and experience, which is different from one user to another.

6 CONCLUSION

We have introduced *AdProc*, a closed domain natural language query processing system on multiple ads domains, which (i) automates the process of *classifying*, *extracting*, and *populating* data from online ads to its underlying database, (ii) relies on simple probabilistic models to determine the *domain* an ad query belongs, and (iii) generates answers that match the information needs expressed in an ad query. Empirical studies conducted on a set of 80,000 online ads show that *AdProc* is highly effective in classifying ads in multiple domains and labeling and extracting their data, with accuracy in the ninety percentile. Furthermore, the approaches adopted by *AdProc* outperform other machine learning approaches (up to 9%) in accomplishing the same task. In addition, a conducted study has verified the effectiveness of *AdProc* in answering natural language queries in multiple ads domains.

For future work, we intend to further enhance *AdProc* so that it can (i) automatically define the *schema* of the underlying database for storing ads from multiple domains, and (ii) handle online ads that include multiple products within the same ad, such as video games ads.

REFERENCES

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

Burke, R., Hammond, K., Kulyukin, V., Lytinen, S., Tomuro, N., and Schoenberg, S. (1997). Question Answering from Frequently Asked Question Files: Experiences with the FAQ Finder System. *AI Magazine*, 18(2):57–57.

Callan, J. and Connell, M. (2001). Query-Based Sampling of Text Databases. *ACM TOIS*, 19(2):97–130.

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

Chouaib, H., Cloppet, F., and Tabbone, S. (2009). Generic Feature Selection and Document Processing. In *Proc. of ICDAR*, pages 356–360.

Cohen, W. (1995). Fast and Effective Rule Induction. In *Proc. of ICML*, pages 115–123.

Dayanik, A., Lewis, D., Madigan, D., Menkov, V., and Genkin, A. (2006). Constructing Informative Prior Distri- butions from Domain Knowledge in Text Classifica- tion. In *Proc. of SIGIR*, pages 493–500.

Demner-Fushman, D. and Lin, J. (2007). Answering Clinical Questions with Knowledge-Based and Statistical Techniques. *Comp. Linguistics*, 33(1):63–103.

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

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

Li, B. and Han, L. (2013). Distance Weighted Cosine Similarity Measure for Text Classification. In *Proc. of IDEAL*, pages 611–618. Springer.

Liu, Y., Lin, Y., and Chen, Z. (2003). Using Hidden Markov Model for Information Extraction Based on Multiple Templates. In *Proc. of NLP-KE*, pages 394–399.

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

McCallum, A. and Nigam, K. (1998). A Comparison of Event Models for Naive Bayes Text Classification. In *Proc. of AAAI Workshop on LTC*, pages 41–48.

Meng, X., Ma, Z., and Yan, L. (2009). Answering approximate queries over autonomous web databases. In *Proceedings of WWW*, pages 1021–1030.

Mitchell, T. (1997). *Machine Learning*. McGraw Hill.

Nambiar, U. and Kambhampati, S. (2006). Answering Imprecise Queries over Autonomous Web Databases. In *Proceedings of ICDE*, pages 45–45. IEEE.

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

Rajput, Q. and Haider, S. (2009). Use of Bayesian Network in Information Extraction from Unstructured Data Sources. In *Proc. of WASET*, pages 325–331.

Rey, D. and Neuhäuser, M. (2011). Wilcoxon-signed-rank test. In *International Encyclopedia of Statistical Science*, pages 1658–1659. Springer.

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

Wang, Z. and Luo, X. (2009). A Semantic Pattern for Restricted Domain Chinese Question Answering. In *Proc. of ICMLC*, pages 1333–1338.

Xue, G., Dai, W., Yang, Q., and Yu, Y. (2008). Topic-bridged PLSA for Cross-domain Text Classification. In *Proc. of ACM SIGIR*, pages 627–634.

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

Zhang, Z., DaSylva, L., Davidson, C., Lizarralde, G., and Nie, J. (2004). Domain-Specific QA for the Construction Sector. In*Proc. of IR4QA Workshop*, pages 65–71.