Web Object Identification for Web Automation and Meta-Search *

Iraklis Kordomatis Christoph Herzog Ruslan R. Fayzrakhmanov

Bernhard Krüpl-Sypien Wolfgang Holzinger Robert Baumgartner†

Vienna University of Technology, Database and Artificial Intelligence Group
{kordomatis,cherzog,fayzrakh,kruepl,holzing,baumgart}@dbai.tuwien.ac.at

ABSTRACT

Web object identification plays an important role in research fields such as information extraction, web automation, and web form understanding for building meta-search engines. In contrast to other works, we approach this problem by analyzing various spatial, visual, functional and textual characteristics of web pages. We compute 49 unique features for all visible web page elements, which are then applied to machine learning classifiers in order to identify similar elements on other previously unexamined web pages. We evaluate our approach with different scenarios by analyzing the relevance of the chosen features and the classification rate of the applied classifiers. These scenarios focus on understanding search forms from the transportation domain, particularly flight, train, and bus connections. The results of the evaluation are very promising.

Keywords

web object identification; web automation; web accessibility; machine learning; big data; web page visual representation

1. INTRODUCTION

We propose a novel approach for identifying simple web objects. Web objects are contiguous parts of a web page that logically belong together, and the automatic detection of relevant web objects is crucial for web information extraction, web form understanding, web automation, and web accessibility. Our use case is supervised meta-search in the transportation domain, where we aim for automatic detection and operation of search forms for flights, trains and buses. For a small set of web pages that contain transport search forms and a small number of annotated examples, we want to automatically identify those web objects that are required for actually initiating the search for transport connections.

Web objects correspond to one or more CSS boxes and a certain rectangular area on a screen, which is the minimal bounding box of these CSS boxes. In our approach, we make use of visual information acquired from the CSS object model and calculate necessary features of the web objects. Leveraging the CSS object model of the rendered web pages enables the analysis of the visual characteristics perceived by users.

Depending on the scenario, some of these web objects play a special role. For example, in the flight search, we need to identify specific form control elements such as the departure airport name and the flight search submit button. This problem is well-known in web information extraction, but in our approach we compute 49 unique web object features from four different categories—including spatial and visual perception features. These features help in identifying relevant web objects in possibly large and complex web pages of arbitrary structure.

Our previous research [17] already indicated that there is a huge potential for considering visual and geometric characteristics in the problem of simple object identification. In the present paper, we improve our previous research and introduce a more efficient and universal approach for object identification. We focus on identifying exactly one target web object for each task, across different web pages, computing rankings of positively classified objects. Moreover, we also added more features and a very well performing **support vector machine (SVM)** classifier.

Core to our idea is the concept of a *feature matrix*, which contains all features of the web objects under consideration. We employ a pair-wise *distance calculation* to compute *feature distance vectors* between web objects. These vectors reflect how similar two objects are. We start by providing a number of positive examples and apply several classifiers to create rankings of likely matches on previously unseen web pages.

We believe that our feature set is particularly well suited to describe and distinguish relevant web objects in the context of complex web pages, especially more robust than ap-

*This work is funded by the Austrian Forschungsförderungsgesellschaft FFG under grant 829614 (TAMCROW [1]).

†This author is also affiliated with Lixto Software GmbH (baumgartner@lixto.com)

proaches which are based on source code or DOM tree. We distinguish between features that are inherent to the web object, local context features (in a rectangular area surrounding a web object), and global (document-wide) features. We evaluated our approach experimentally in three different test scenarios. The evaluation results are promising and illustrate the efficiency of our techniques. The datasets (flight, train and bus search domain) and feature distance matrices are available online.

The main contributions of the paper are: (i) an expressive description for web objects combining functional, spatial, visual and textual aspects into 49 features; (ii) a set of distance metrics between different feature types that can be used to calculate feature distance vectors; (iii) a workflow that employs various classifiers to determine rankings of most-likely matches.

The rest of the paper is organized as follows: In Section 2, we define the problem and present the overall architecture of our approach. Section 3 has a closer look on web objects and their features. Section 4 explains how we extract these features from the original web pages. Section 5 provides details about distance computation and metrics. Section 6 explains which machine learning techniques we use and discusses the different classifiers. Section 7 comprises the evaluation of our approach for four different scenarios. Section 8 provides an overview of related work. Finally, the paper concludes in Section 9 and gives an outlook to future work.

2. PROBLEM STATEMENT & APPROACH

In this paper we deal with the problem of finding specific web objects on previously unseen web pages. The problem consists of the sub problems of how to describe the desired objects (e.g., a specific submit button)and how to identify them on new web pages. This general problem is fundamental to many fields of research like Web Data Extraction, Web Form Understanding, Web Automation, etc.

Our approach uses features based on the visual perceivable characteristics of web objects for object description and machine learning techniques for object identification. Applied to a meta-search setting, this allows for a very universal solution. Here, the common elements of the considered search forms can be identified on previously unseen search pages, as long as they follow the same design principles. For evaluation, we use such a meta-search setup where we apply our approach to a number of heterogeneous transport search pages with very promising results (see Section 7.1).

Another application example would be to identify the different discussion threads and messages on web forums in order to rearrange them for optimized mobile access. We demonstrated the feasibility of such an application in [17].

As stated before, the two major problems we address are (a) how to model a target object and (b) how to identify the most similar candidates on previously unseen web pages. We defined features that describe web objects and correspond to visually perceptible properties on the rendered web page. Examples of such features are the foreground color or the number or number of horizontally aligned objects. See Section 3 for details on features.

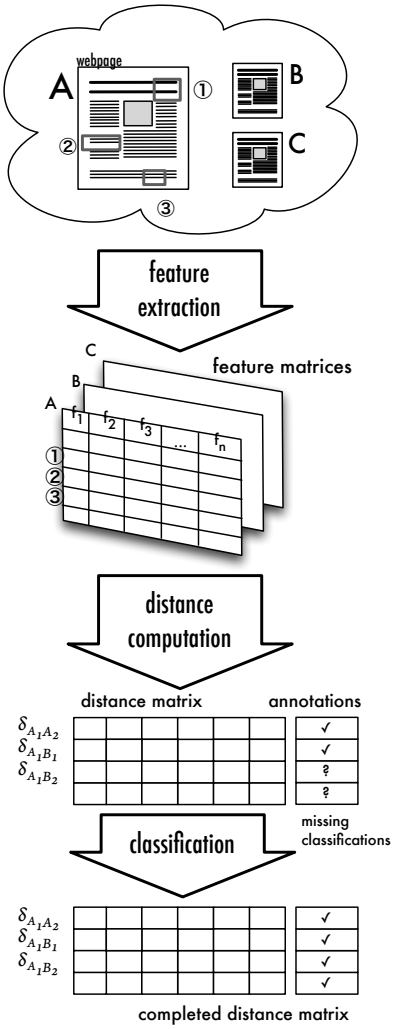


Figure 1: Web object identification workflow

The identification is based on the notion of feature distances, which reflect the degree of similarity between objects. To decide whether two objects are identical or not, we apply machine learning techniques (see Section 6 for details).

Figure 1 shows an overview of our approach. In this example we consider only one target object, for example the submit button that starts a search process. We start from a set of web pages (A, B, C) where some of them have the target object manually marked (annotated). Then the basic steps are:

(1) *Feature Extraction:* We extract the features of all web objects from our set of web pages. The output of this step is a *feature matrix* for each web page, containing all web page objects together with their features. In addition, columns with meta information about the elements are added (e.g. web page id, annotation, etc.). Section 4 gives details about this step.

(2) *Distance Computation:* In the next step, the distances between web objects are calculated. The input data for this step are the feature matrices. The distances between objects from annotated web pages serve as train-

ing set for the classifier. They have a defined annotation class, in our example *match* or *1* when two examples of the target button are compared and *no match* or *0* otherwise. Furthermore, the distances between the target object and all objects from the non-annotated web pages are calculated. The resulting distances are stored in the *distance matrix*. See Section 5 for details on distance calculation.

(3) *Classification:* In the final step, we use that part of the distance matrix where the annotation class is known as input for machine learning algorithms. With these, we train classifiers which are able to estimate whether two objects are similar (i.e. both are target objects) or not. We apply these classifiers to the distances between the annotated target objects and the objects of the new web pages. This allows us to identify those objects of the new web pages which correspond to our targeted submit button. This process is detailed in Section 6.

For simplification, we only describe how we identify a single target object. Generally, we annotate and identify several target objects on our web pages. In the process of calculating the distances, a separate distance matrix is created for each of these "tasks", i.e., target objects. The identification problem is then solved independently for each of these tasks.

3. WEB OBJECTS & FEATURES

In this section, we introduce the main concepts which we use for describing the objects on a web page. We define several main structural elements of a web page (see Figure 2):

- A *document* is a web page rendered by the web browser's engine. It is formed by the set of X/HTML or XML files connected with each other by means of inclusion (e.g. by elements with the names `FRAME`, or `IFRAME`, or `OBJECT`). Width and height of the document correspond to the dimensions of a minimum bounding rectangle wrapping corresponding rendered files—*pages*.
- A *page* is a single rendered X/HTML or XML file of a document. A set of pages make a *hierarchy of pages* through the inclusion relations. From another point of view a page is a DOM-tree together with computed CSS attributes; it has the counterpart `Window` in the **Browser Object Model (BOM)** [18].
- We interpret a *selected object* as a certain visual element—web object—of the document that corresponds to one or several visible CSS boxes. And a *target object* is a selected object to be identified. A selected object has also a minimum bounding rectangle which is used for computing features for the object identification tasks. The following types of visual elements are considered in this work: `HtmlButton`, `HtmlCheckbox`, `HtmlFileUpload`, `HtmlImage`, `HtmlPasswordInput`, `HtmlRadiobutton`, `HtmlSelect`, `HtmlText`, `HtmlTextArea`, `HtmlTextInput`. All of these have corresponding object types in the CSS object model of a web page and in the **Unified Ontological Model (UOM)** accordingly.
- A *context* of the selected object is a rectangular area centered on the selected object and defines its neighborhood presented by the visible CSS boxes. In our evaluation, we define the context to be equal to $2h \times 1.4w$, where h is height and w is width of the selected object; the minimal allowed size is $h + 500 \times w + 500$ pixels [15].

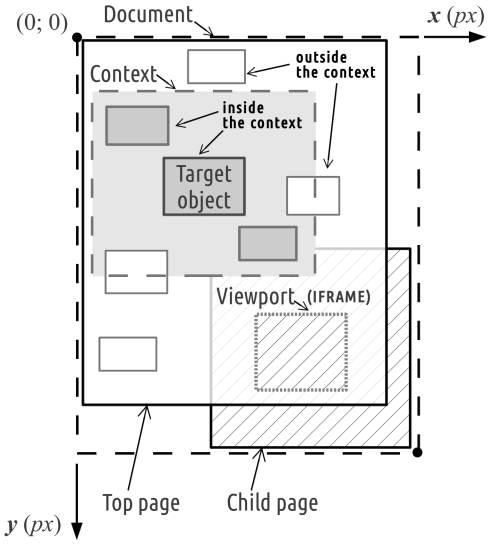


Figure 2: Elements of a web page

These structural elements have their counterparts in the **Physical Model (PhM)** of the UOM, which is acquired in the process of web page analysis. These objects and their associated descriptions provide us with the necessary information for computing the web objects' features considered in this work. The UOM is out of the scope of this paper, interested readers can refer to [13, 12, 14, 21]. For the spatial analysis we define the geometric space of a web page (web page canvas) as Euclidean space with pixels as unit of measure. The top-left corner of the top level page from the page hierarchy specifies the origin of coordinates. Abscissa is directed from left to right, and ordinate is from top to bottom (see Figure 2).

Computation of a set of *features* allows us to describe various aspects of web objects and provide us with the necessary information for the object identification tasks. Depending on the considered elements of a web page during the computation of web objects' features, we distinguish *inherent* and *relative* features. The former describe the characteristics of the web page elements themselves (e.g. of the selected object, context, or page) and are computed independently from other structural elements. Inherent features are only based on the computed styles of the contained CSS boxes or the attributes of their counterparts in the BOM. Examples are font color, tag name, height, or font size. Relative features on the other hand reflect the characteristics of several structural elements. Examples are the number of web objects in the context or the average color distance between the selected object and all other objects within the context. Relative features are considered for a pair of structural elements, e.g. a selected object and its context or a selected object and the document.

We distinguish three categories of features (a feature can be assigned to several categories):

- *Interface features* (IF) define the functional roles of objects (e.g. button, image, text) and structural types such as lists and tables.
- *Spatial features* (SF) reflect peculiarities of geometric configurations of web objects, for example, the number of elements aligned with the selected object, the number of orthogonally visible objects, the absolute position etc. Spatial features are computed based on several basic spatial features and relations, in particular alignment, distance, topology and direction [12].
- *Visual perception features* (VPF) relate to visual characteristics of objects, e.g. foreground and background color, emphasis or font size. They correspond to attributes used in the fields of graphical user interface design, computer graphics and computer vision.
- *Textual features* (TF), e.g., textual content of the selected object, text above the object, number of lines, number of tokens. Most of the textual features are adopted from the area of quantitative linguistics [28].

A comprehensive list of the features used in this work is presented in [15] and Table 3.

4. FEATURE EXTRACTION

In the process of feature extraction, we analyze specific set of web pages with manually annotated objects (see Figure 3) according to the task posed, i.e. identification of web form elements within the transportation domain (see Section 7.1). For each web page, the process of feature extraction consists of two phases: 1) Instantiating the PhM [13, 21, 12] for a certain web page and 2) Computing the feature matrix which for every object contains all the features required for the identification.

The PhM is a domain ontology which describes various aspects of a web page such as interface elements (e.g. buttons, text input fields, links, and images), layout (geometric characteristics of web page's elements), visual characteristics (foreground and background color, etc.). This model accumulates fundamental information which can be used in various tasks in the field of web information extraction and web page understanding. The WPPS framework [13] was used for generating the PhM for every considered web page. It has various useful functions; for instance, it allows a computation of the real background color for every element, taking transparency, overlapping, and painting order [2] into consideration. It also omits invisible objects that have a too small size or which are topologically inside and behind another object. During the computation of different qualitative characteristics (alignment, topology, direction, etc.), various inaccuracies which can be met in practice are automatically taken into account. For instance, CSS boxes can be visually perceived as aligned while according to the quantitative data they are not. Moreover, the framework conveys effective means for applying declarative and object-oriented paradigms. The former is realized by querying the PhM directly with SPARQL and performing an automatic reasoning over the logical rules, whereas the latter is done by means of the *abstraction layer* that makes object-oriented approach applicable and API that provides all the necessary functionality [13]. Thus SPARQL queries together with the provided API were used for computing additional features

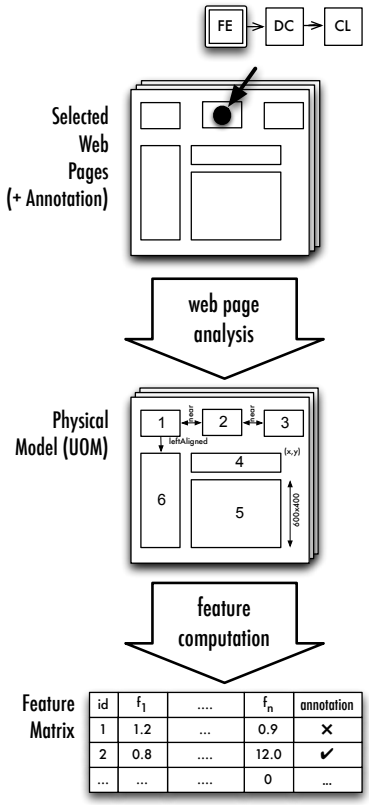


Figure 3: Feature extraction (FE) workflow

which are absent in the PhM, but are required for generating the feature matrices in the tasks considered in this work.

5. DISTANCE COMPUTATION

For finding a desired target object based on examples of how such an object looks like, we need means to compare web objects. In our system, all web objects are described by their features. In order to be comparable, the single feature values need to be related to each other. Figure 4 gives an overview of how distances are calculated. The feature matrix serves as input for the distance calculation. Distances are always calculated for two given rows from the distance matrix. For each feature f_i, a distance d_i is calculated for the two respective values from the feature matrix. The resulting distance vector is then added to the *distance matrix*.

In the simplest case we only have one target object we are looking for. We have several examples of this target object that have a 'positive' annotation and many examples of other objects with a 'negative' annotation. Distances are then computed for every pair of positive examples, which receive a 'positive' annotation. Additionally, each of these positive examples is compared to all negative examples, receiving a negative annotation. Finally, each positive example is compared to all non-annotated rows of the feature matrix, i.e., the objects on previously unseen web pages. The resulting distance vectors make up the distance matrix, which is used for further processing (see Section 6).

The following paragraphs explain the basic distance calculation formulas. Basically, each feature has a corresponding

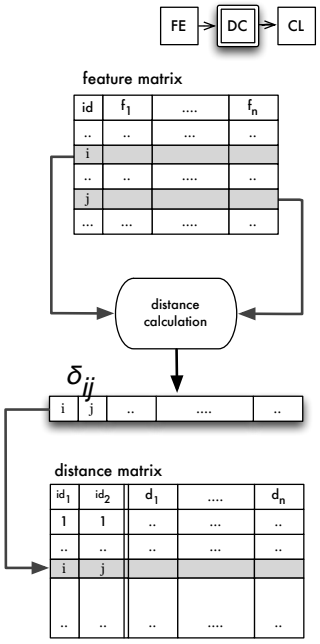


Figure 4: Distance computation (DC) workflow

feature distance. The *feature distance vector* is the vector that results from the calculation of each of the feature distances in two given feature vectors. Since not all features are applicable to all types of objects, we sometimes encounter `null` values in feature vectors. By definition, we assign the maximal distance to distance computations where one or more of the input values are `null`. The actual maximal distance value depends on the specific distance used. In this way we avoid `null` values in the resulting distance matrix.

For the different types of features, different formulas for calculating the distance are used. They are explained in the following paragraphs. Table 3 at the end of the document gives and overview of all features and their respective distance formulas.

- *Relative distance:* For values like pixel height, the pure numerical difference is not a good comparison criterion for our purpose. In this case, we rather calculate how much smaller the smaller value is than the larger value. In order to overcome several algorithmic issues, especially regarding possible divisions by zero and handling of negative values, we arrived at the following formula. f_1 and f_2 are two different values of a given feature:

$$\delta_{rel} = \frac{1}{1+e^{-max(f_1,f_2)}} - \frac{1}{1+e^{-min(f_1,f_2)}}$$

  The maximum value for this distance is 1.0, which is also applied when one or both of the input values are `null`.
- *Absolute distance:* Especially for features which already have a percent value, i.e. a value between 0 and 1, we calculate the distance between the two values as $|f_1 - f_2|$. The maximum value for this distance depends on the feature used, but normally it is also 1.0, assuming a feature value in the range of 0 to 1.
- *Boolean distance:* For features that take on boolean values, the distance between two feature values is simply calculated by assigning 0 if the values are identical and 1 if they are different. This is effectively a logical $\wedge$ operation with *true* being interpreted as 0 and *false* as 1.
- *Equality distance:* For features that can take on a value from a set of predefined enumerated values, we calculate the distance by assigning 0 ('equal') if both features have the same value and 1 ('not equal') if they have different values.
- *String edit distance:* For comparing text, we use the string edit distance with transpositions, also known as the Damerau–Levenshtein distance [7]. For handling `null` values in the input, we assume that a missing value equals an empty string.
- *Grid overlap distance:* The grid distance feature specifies roughly which areas of the web page a given object touches. The web page is divided into a 3×3 grid, resulting in 9 areas that the object can possibly touch (9-neighborhood). The grid overlap distance is then calculated as twice the number of grid areas touched in both features' grids divided by the total number of grid areas touched. This number is subtracted from 1 in order to have 0 as similarity and 1 as the maximum dissimilarity. A 3×3 grid was chosen as it proved most useful in preliminary experiments. Adding additional features for different grid layouts (e.g., 5×3) is planned in future versions.
- *Color distance:* All colors (e.g., foreground color) are represented in HSV color space, which roughly conforms to human perception. Accordingly, we calculate the color distance in HSV color space, with a minimum of 0 (equal) and the maximum distance in HSV color space at 2.

6. CLASSIFICATION

This section describes the classification process which is illustrated in Figure 5.

The workflow for the classification starts from the output of the distance computation process (see Section 5). First, the distance matrix is preprocessed by calculating the z-scores of its cells (1). We normalize each $f_{i,j}$ by replacing it with the corresponding z-value $z_{i,j}$.

...	d_1	...	d_n
...	...	...	...
...	$fd_{i,1}$	...	$fd_{i,n}$
...	...	...	...

$$z_{i,j} = \frac{fd_{i,j} - \hat{\mu}_{.j}}{\hat{\sigma}_{.j}} \quad (1)$$

Afterwards, the resulting matrix with the corresponding class annotations are used to train a classifier.

Finally, we compare each positively annotated object of the training set to all objects on a previously unseen web page. Then we use the trained classifier to estimate the class for each row in the corresponding subset of the distance matrix. From these results we can estimate which of a target web page's objects is the target object (see subsection 6.2).

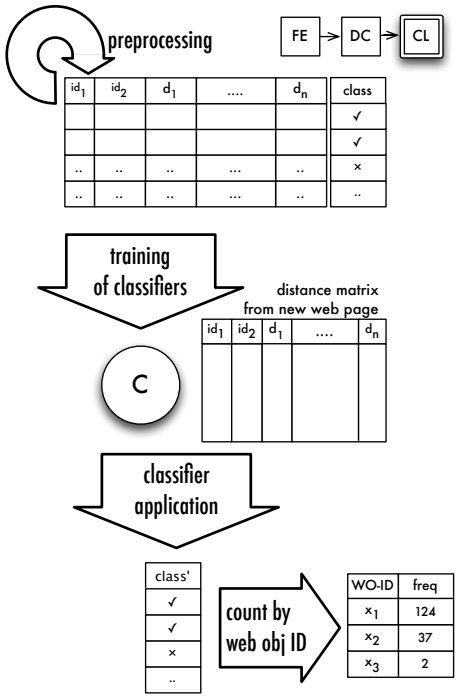


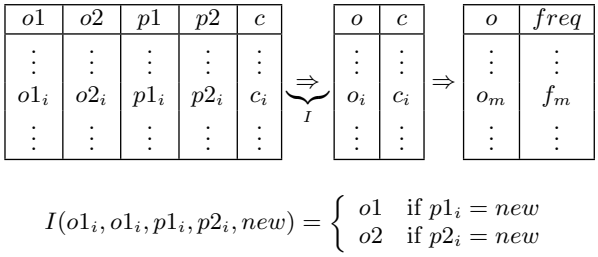
Figure 5: Classification (CL) workflow

6.1 Machine Learning Techniques

In this subsection, different machine learning techniques that we evaluate in this work are discussed. The idea is to use classification techniques of different types (linear, non-linear, etc.), since the results can be expected to be more diverse compared to using only classifiers of similar type.

- *Logistic Regression* is a commonly used classification technique. What makes it very convenient is the fact that it returns probabilities which can be interpreted as a certain class affiliation. Another advantage over similar classification methods (e.g. **linear discriminant analysis (LDA)**) is that it provides the user with inferential statistics. This is useful for analyzing the importance of certain features. However, this raises the question of model selection. In general, small models help reducing overfitting and make the classifier more robust. In our experiments, it shows that models which were reduced stepwise on the basis of the **Akaike information criterion (AIC)** [4] perform worse than logistic regressions with the full model. This is surprising, since normally reduced models are used to reduce the influence of overfitting and make the regressions more robust. Further information on logistic regressions can be found in [23].
- *M5PM* is a method provided by WEKA[1]. It is an improved M5 algorithm from Wang & Witten [29]. This classifier is a decision tree that provides the user with a regression function. The resulting value can be used as proxy for the probability, since a higher result value would indicate a higher probability to belong to a certain class.
- *SVM* [27] is a common technique in **Machine Learning (ML)** which attempts to find a linear separation between different classes. This separation is achieved by transforming the input data into a higher dimensional feature space. The transformation is done via kernel functions. Commonly used kernel functions are linear, polynomial, radial[2] and sigmoid[3].

[1] http://www.cs.waikato.ac.nz/ml/



$$I(o1_i, o1_i, p1_i, p2_i, new) = \begin{cases} o1 & \text{if } p1_i = new \\ o2 & \text{if } p2_i = new \end{cases}$$

Figure 6: From the classifiers' output to object identification

6.2 Postprocessing

This subsection deals with the postprocessing of a classifier's output. The output of a classifier is its "suggestions" regarding the class affiliations of the distance pairs of web objects. However, the goal is to find the web object (only one) which is required for a certain task (e.g. the submit button for a specific web form). This object is determined by the task. Figure 6 illustrates how the output can be transformed in order to identify the most probable target object.

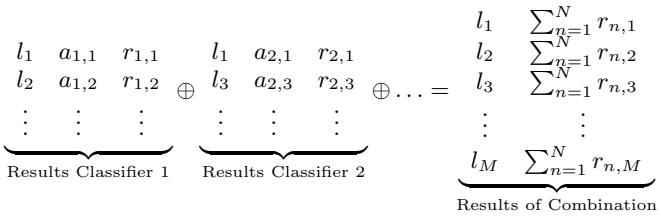
Firstly, the identification function (I) determines which web object id (column $o1$ and $o2$) is the id on the *new*[4] web page. It uses the information of the column $p1$, $p2$ and *new* where the first two contain the web page id for $o1$ and $o2$ respectively; *new* is the web page id of the new web page.

Secondly, with the resulting table it is possible to count the number of appearances for each unique web object (also known as absolute frequency). The highest frequency in this list provides the web object which is most likely to be the searched target object.

6.3 Combining Results of Different Classifiers

In order to receive better classification results, we try to combine the results of base classifiers. Therefore, we compute the relative frequency from the classifier outputs. The combined results are the sums of the relative frequencies of the base classifier for each unique web object. Equation (2) gives an overview of how the calculation works. $a_{i,j}$ denotes the absolute frequency for classifier i and unique web object j. $r_{i,j}$ stands for the relative frequency, where the indices are equal to $a_{i,j}$. l_i is the label of the unique web object. An interesting observation is that classifier 1 and 2 (from the illustration) have not necessarily the same elements in the same order nor the same elements in the list at all. Therefore, it is required to merge the resulting list accordingly.

[2] alias Gaussian Radial Basis
[3] also known as the hyperbolic tangent
[4] *new* in this context means the web page on which we want to detect web objects



$$r_{i,j} = \frac{a_{i,j}}{\sum_{m=1}^{M} a_{i,m}} \quad (2)$$

6.4 Dealing with unbalanced classes

This subsection refers to the classes we want estimate with the classifiers (sometimes referred to as the dependent variable in a regression model). Unfortunately, the class affiliations of this dependent variable are far from balanced. This means that a vast majority of elements belong to one class and minor number to the other class. For example, in our test scenarios described in Section 7.1, the majority are negative examples.

When training a classifier to the entire test set, it would tend to overfit towards to the major class. To avoid this, there has to be a procedure to deal with this undesirable behavior. In our previous work [17], we decided to use all elements of the minor class and a sample of size N from the other class. The size of N for the experiments in Section 7 is 10 times the minor class. We found this ratio to return the best results for our application. As we choose the samples randomly, this approach is not deterministic for each run. To average the results, we repeat the sampling 10 times. We train the classifier and measure the classification rate again for each run.

7. EVALUATION

In this section we present the results of the experimental evaluation of our method. We decided to test our approach on a meta-search setting, for which we collected a number of real life web pages from the area of transportation search (see Section 7.2). We annotated these sample pages manually in order to describe the most important input fields for this type of search—target objects. With this input we implemented our approach described in Section 2. Annotation and generation of the feature matrices was performed using an application, Objident (see Figure 7), which is build based on the WPPS framework [13] and has Firefox (XULRunner 1.9.2) integrated. Thus the process of generating PhMs and computing the required features is hidden from the user.

7.1 Test Scenarios

In order to test our approach under realistic conditions, we have defined four *scenarios*, namely searches for bus, flight and train connections, as well as a last one, which combines them all referring to it as the 'combined' or 'all' scenario. The latter is especially interesting as it is actually a meta-search over multiple different connection types. In all four scenarios, we focused on finding the relevant elements in the main search forms.

Every considered web page within the scenarios can have following types of objects: (1) *departure location*, (2) *arrival location*, (3) *departure date*, (4) *one-way trip* which corresponds to the control that defines the trip as one-way, (5) *adult passengers* which is the number of adult travelers, (6) *submit button* which is the button to send a request form and (7) *other* which correspond to all other elements on a web page. We consider every item of this list as a *task*. Thus in each *scenario* each input field (*task*) has to be found on the different websites.

In the annotation process, the user specifies a scenario for every web page and a task for every target object by means of the Objident. She only assigns the labels for the objects to be identified (i.e. of the types 1–6), whereas all irrelevant elements are automatically recognized and labeled with *other*.

7.2 Selected Web Pages

For the aforementioned scenarios we have created an evaluation dataset we call the ATW[5] dataset [3]. It contains the following information required for the method introduced in this paper: (1) web pages from the transportation domain, (2) annotations of web page elements with aforementioned object classes, (3) computed features [15] listed in Table 3 at the end of the paper and (4) computed feature distances for each pair of elements (see section 5).

The selected web pages were arbitrarily chosen. In order to test an efficiency of our method over web pages authored in different languages, we decided to select an equal number of English, German and Russian web pages. The following list shows the selected pages per scenario:

- *Flight:* Aeroflot, Air Berlin, Air Canada, American Airlines, Austrian Airlines, Avia Sales, British Airways, Checkfelix.com, Emirates, Lufthansa, Germanwings, Quantas, Rossiya, Tatarstan, Trip.ru,
- *Bus:* Avperm.ru, Avtovokzal.ru, Avtovokzal73.ru, BerlinLinienBus.de BusEireann.ie, Eurolines.at, GoToBus.com, Greyhound.com, Matkahuolto.fi, Postbus.at, Postbus.ch, PublicExpress.de, Regiobus.ch, Turistua.com
- *Train:* Eurostar.com, IrishRail.ie, OEBB.at, Poezdato.ru, Portal-Poisk.ru, RZD.ru, s-bahn-berlin.de, saarbahn.de, SBB.ch, TGV-Europe.com, Trenitalia.com, Tutu.ru, Voyages-sncf.co.uk
- *Combined:* All of the above mentioned web pages.

A detailed table with links and a list of available web form fields can be found in [3, 15].

7.3 Cross-Validation

For evaluation purposes, a k-fold cross-validation is commonly used in order to validate the estimated models and avoid overfitting. The k-fold cross-validation separates the available data into k partitions of the same size. Thereafter, a model is estimated with all data except that from the first partition. The excluded data is used to validate the estimated model. This procedure is repeated until every partition has been used for validation.

[5] Annotated Transport Web Forms

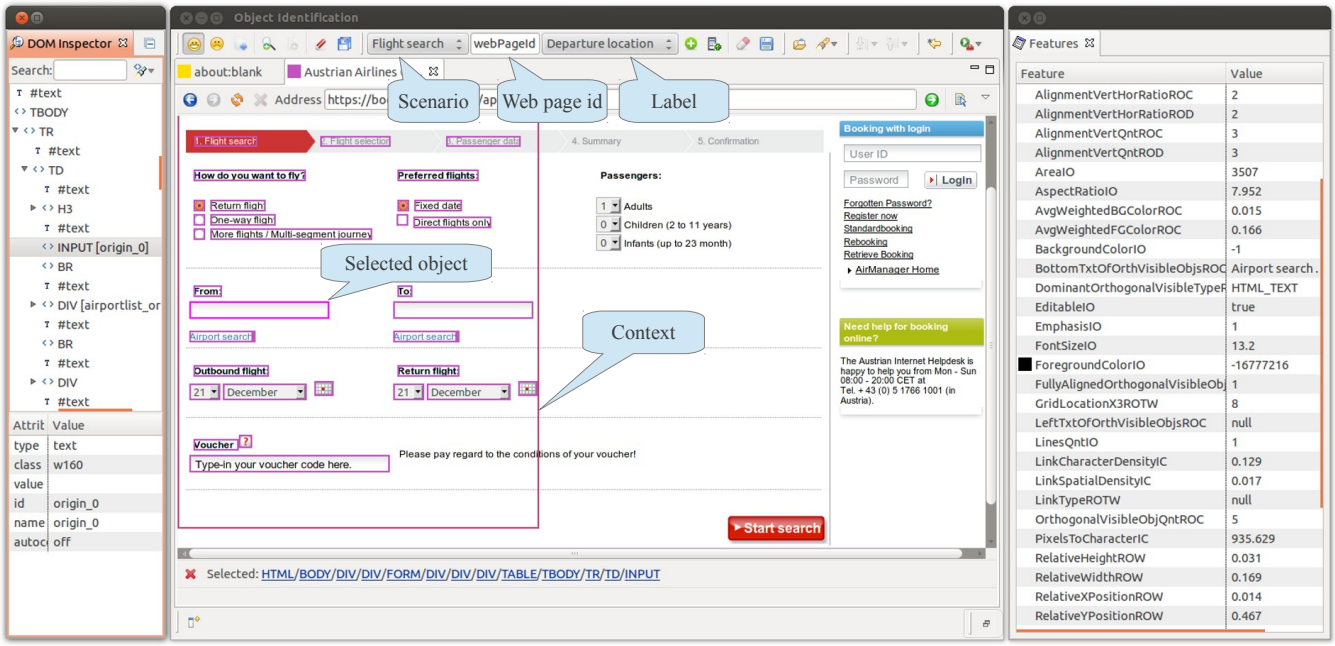


Figure 7: Screenshot of Objident, the feature extraction tool

We have adopted this principle in order to validate the different classifiers. In contrast to the procedure mentioned above, where the input data is separated into equally large parts, we have separated our input data into the S different web pages. As the input data are pairs of elements which can be from different web pages, we can exclude all distance pairs of web page x[6] from the training set and use it for the validation.

7.4 Performance Measures

There are several well accepted performance measures for classification problems. Since our goal is to classify exactly one positive web object correctly, the positive precision is the relevant measure for our approach. This value is either 100% in case the one object is classified correctly or 0% in case we did not find the correct element. Precision is defined in equation (3). We compute the average precision over all our test results. The resulting number is the ratio of instances in which we correctly identified the target object.

$$precision = \frac{TP}{TP + FN} \tag{3}$$

7.5 Classification Results

This subsection illustrates the classification results of the different scenarios. First, we present a benchmark of the classifiers for the above mentioned web pages and scenarios. Then we go deeper and discuss the different outcomes between the tasks.

7.5.1 Result Overview

[6] x represents the id of the currently selected web page in the whole cross-validation process

Table 1: Classification rate in % per classifier

Technique	All	Bus	Flight	Train
Log Regression	71.71	75.08	88.97	71.55
M5PM	75.96	70.46	89.87	60.17
SVM linear	72.89	74.46	85.38	66.21
SVM polynomial	46.71	34.31	37.05	9.14
SVM radial	62.32	32.77	54.74	36.03
SVM sigmoid	**79.78**	64.92	80.13	68.97
C[7] **(All)**	62.06	45.08	71.03	29.31
C[7] **(LReg,Sigm.)**	77.63	**80.31**	**90.77**	**77.07**

Table 1 gives an overview of the results. It provides the accuracy per classifier for each scenario. Note that the All-column is not the average sum of the other three but a separate scenario. From the base classifiers the logistic regression has almost always the best results for the Bus, Flight and Train scenario. However, its performance in the All scenario is only average. On contrary, the SVM with the sigmoid kernel does well on the All scenario. Therefore we combined the two of them and did indeed receive better results than with the base classifiers alone, except for the All scenario. But even there the result of the combined classifiers is still better than their average.

Another interesting notion is that the SVM with the polynomial and radial kernel performed rather poorly. Furthermore, the combination of all classifiers did also not perform better than the combination of the logistic regression and the SVM with the sigmoid kernel, indicating that they indeed complement each other very well. Moreover, the M5PM, SVM with the linear and sigmoid kernel appear relatively unstable compared to the logistic regression.

[7] C stands for a combined classifier. The names in the brack-

Table 2: Classification rate in %, excl. the *adult passengers* and *one-way trip* in the bus and train scenarios, since too few examples are available

Technique	All	Bus	Flight	Train
Log Regression	85.53	86.15	88.97	86.46
M5PM	92.24	82.88	89.87	72.71
SVM linear	87.70	85.58	85.38	80.00
SVM polynomial	57.96	35.38	37.05	11.04
SVM radial	74.67	40.77	54.74	43.54
SVM sigmoid	**97.04**	80.96	80.13	83.33
C(All)	77.70	52.31	71.03	35.42
C(LReg,Sigm.)	93.55	**92.69**	**90.77**	**93.13**

7.5.2 *Detailed Results*

Figure 8 shows the classification results per scenario and task. The y-axis shows the average accuracy for each run. The classification results for each test run is either 100% or 0% since there is only one web object to be correctly classified. Therefore, the arithmetic average is a better indicator than the median. The numbers at the x-axis stand for the following classifiers: (1) logistic regression, (2) M5PM, (3) SVM with linear kernel, (4) SVM with polynomial kernel, (5) SVM with redial kernel, (6) SVM with sigmoid kernel, (7) combined classifier with all above mentioned classifiers and (8) combination of the logistic regression with the sigmoid kernel of the SVM. In addition, the 90% confidence interval is shown per bar in form of two whiskers.

Figure 8 shows that the adult passenger and the one-way tasks perform rather poorly (except the flight scenario). This is due to the fact that there are too few examples to learn from in these test sets. When looking at the aggregated results in Table 1 we can also see that the SVM with the polynomial kernel has most often the worst results in each task (followed by the radial kernel SVM). However, it shows that the locations (arrival and departure) and the submit button achieve excellent results with the logistic regression, M5PM, the linear and the sigmoid kernel SVMs as well as the combination of the first and last base classifier. Furthermore, the results of this combined classifier (logistic regression and sigmoid) is in almost every case above 90% for every task (except adult passengers and one-way). This makes it the most favorable classifier for our meta-search setting, achieving an impressive classification rate. It is interesting to note, that not all of the above mentioned observation are statistically significant[9].

As pointed out above, some tasks perform rather poorly due to a too small number of examples. To give an impression of how good the results would be if we had more examples available, we provide the results excluding these undersampled tasks in Table 2.

7.6 Feature Discussion

This section discusses the importance of the single features for the classification. Out of the used techniques only the logistic regression provides detailed information about the influence of each input feature. Here, the change of explanatory power by setting its variable coefficient to 0 is measured. If the explanatory power of the dependent variable drops significantly, this variable is considered very important for describing the dependent variable. This measurement is in general applied only per feature, which means that it will not consider interdependencies between several features. Therefore, it can happen that a single feature is not significant but that it actually is important in combination with others.

ets indicate the combined classifiers.

[9]This status appears if the confidence intervals of two bars are not overlapping. Then one is significantly different from the others.

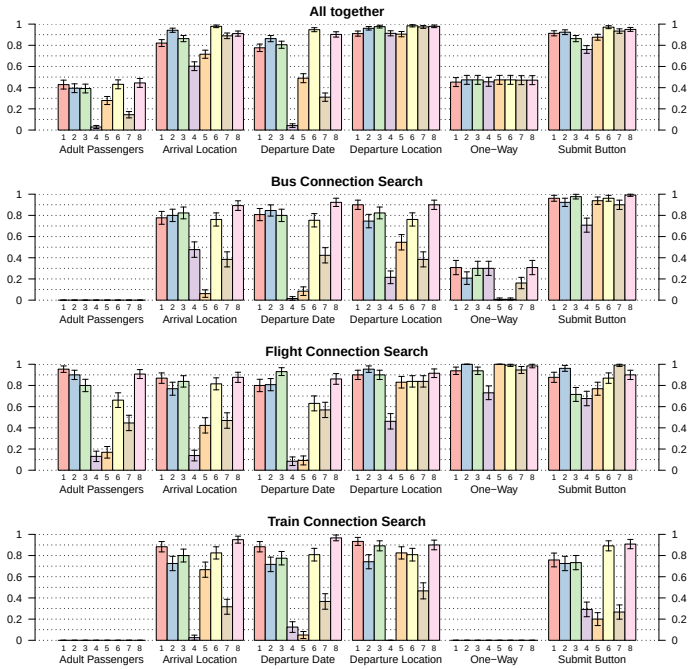


Figure 8: Detailed results with 90% confidence intervals

We expected to be able to find a small subset of features that would be sufficient for the web object identification within the scope of specific task, scenario, or the transportation domain. However, this challenge appeared to be unsolvable. Indeed, considering every feature, we found out that no single feature dominates over all others. Furthermore, in the evaluations across various test cases, it appears that almost every feature is significantly important. This seems surprising at first, but this also describes the fact, that a reduced model by means of the AIC is in general not better than the complete model.

8. RELATED WORK

The web object identification problem is met in different areas of science and technology, either expressly or by implication, be it web GUI testing, web accessibility, web data extraction, or web form understanding for querying meta-search engines.

In the field of GUI regression testing for web applications [24], the definition of interactive GUI elements and their identification play an important role, particularly in the phases of GUI model learning and evaluation. During this first phase, various AJAX-aware crawlers can be applied for

traversing all possible interactions on a web page, building corresponding graph-based interface models (e.g., CRAWLJAX [25], AJAX Crawler [10], [8]). The second phase is a simulation of necessary interactions from generated test cases. This is often realized using light-weight approaches, such as Selenium[11], Sahi[12] or Watij[13] [5, 6].

In the area of web automation and web accessibility, there are several tools to automate repetitive interactions (e.g., Ubiquity [11], CoScripter [22]) or to assist disabled persons in performing certain tasks. While they have a rich language for expressing interactions and tasks, they are highly dependent on the DOM tree, resulting in a lack of robustness.

Actually, most methods and approaches that refer to the problem of web page object identification consider either the source code of a web page or its DOM tree. They do not reflect visual and spatial characteristics, which are less volatile and more closely resemble the way humans look for relevant objects. Furthermore, these approaches are implicitly sensitive to differences in wording and language. In contrast, our approach considers visual characteristics, which makes it independent from source code, more robust against changes, and applicable to a wide range of web pages.

In this paper, we consider the problem of web form understanding in the context of building meta-search engines, which plays an important role in extracting heterogeneous data from the Deep Web. Existing approaches target on different web page representations [20]. Since we focus on visual features in this paper, research works which take the web page's layout into account are most relevant for us. We distinguish rule/heuristics-based approaches [16, 30, 9] and machine learning approaches [26, 19]. The first group depends on a set of predefined rules, which can be web- or domain-specific. These manually constructed rules and heuristics usually reflect semantics hidden in spatial relations between web form elements. Machine learning approaches focus on automatically deriving a model, which describes specific aspects or features of web forms. In LabelEx [26], the authors leverage a domain-specific classifier ensemble for labeling every web from element. In [19], the authors use a two-layered Hidden Markov Model for automatic search interface segmentation. In contrast to all these approaches, we are not aimed on manually modeling specific objects to be identified. Also, we do not target on learning complicated domain-specific relations for web forms. Instead, we provide a novel, universal solution for identifying basic objects based on annotated examples. Our solution can work with relatively few examples and provides very efficient and robust classifiers that achieve impressive success rates.

In contrast to our previous work [17], in this paper we focussed on the problem of identifying the *most similar objects* on a web page, given a set of examples for a specific target object. We achieved this by defining a *ranking* across all positively classified objects. Solving this specific problem can help to considerably increase precision and recall of our object identification method and allows us to apply it to use cases like web automation and meta-search. Moreover, we introduce various additional features, referring to different aspects of web page representation [15], and test additional machine learning techniques for object identification. We evaluate our approach on real life examples in a transport meta-search setting, where we achieved very promising results.

[11] http://seleniumhq.org/
[12] http://sahi.co.in/
[13] http://watij.com/

9. CONCLUSION & FUTURE WORK

The challenge of developing approaches and means for basic web object identification is well-known in the fields of web automation, web data extraction and meta-search. Taking into account the promising results acquired in our previous research [17], we introduced a more robust and universal approach for describing web objects and identifying them on previously unseen web pages.

The method proposed in this paper is based on the analysis of the visual characteristics of web page's elements and the application of various machine learning techniques to identify specific objects on new web pages. The overall approach consists of the following steps: feature extraction, distance computation, and classification. During the extraction phase 49 various features are generated for the objects to be compared. The features reflect different aspects of a web page's visual representation, such as interface, layout, textual, and visual perception. Feature distance computation provides a means to compare objects. Finally, feature distances are used to train and apply classifiers. Three main types of classifiers are considered in the evaluation: logistic regression, decision tree (M5PM) and SVM with four different kernel functions. The classifiers are trained to find class affiliations depending on their feature distances. In this way we can identify the most similar object on a web page with respect to a set of sample objects.

We evaluated our approach with real life scenarios from the area of transportation meta-search. The evaluation problem was formulated as the task of identifying the expected set of web form fields on previously unseen transportation search pages. Using real life examples of flight, train and bus search pages, our method achieved outstanding results in identifying the required search form elements. However, we want to experiment with several combined classifiers. This should address the question, which combination works best. Therefore, it could be useful to add further base classifiers to find a good combined one.

Knowing the peculiarities of the considered domain, its semantics, linguistic characteristics and constraints, different complementary techniques can be applied for improving the efficiency of the method proposed in this paper. For instance, for the transport search domain, we can expect to have some compulsory fields (e.g. departure and arrival location, submit button), which are always grouped together in the same HTML form. This fact can be taken into account for a heuristic post-processing when several candidates for a specific object (e.g., departure location) are found on the web page. Furthermore, knowledge from the web design domain can be applied to create even more efficient heuristics. For example, if departure and arrival location are nearly indiscernible by the classifier, they can be distinguished based

on the spatial order of their appearance on the web page. Usually, we expect the departure field to appear above or left of the arrival field. We believe that the chance of accurately identifying all required fields of a web form can be increased even further when applying such domain dependent knowledge. This would be of benefit for specific applications like meta-search.

10. REFERENCES

[1] TAMCROW — Task mining and crowd sourcing. FFG Fit-IT Project 829614, 2011–2012. http://www.dbai.tuwien.ac.at/proj/tamcrow/.

[2] CSS Level 2 Revision 1 (CSS 2.1) Specification (W3C Recommendation 07 June 2011), 2011.

[3] ATW Dataset. http://www.dbai.tuwien.ac.at/proj/tamcrow/atw/, 2012.

[4] H. Akaike. A new look at the statistical model identification. *IEEE Trans. on Automatic Control*, 19(6):716–723, 1974.

[5] A. Bartoli, E. Medvet, and M. Mauri. Recording and replaying navigations on AJAX web sites. *Web Engineering*, 7387:370–377, 2012.

[6] J. Byrne, C. Heavey, and P. Byrne. A review of Web-based simulation and supporting tools. *Simulation Modelling Practice and Theory*, 18(3):253–276, Mar. 2010.

[7] F. J. Damerau. A technique for computer detection and correction of spelling errors. *Commun. of the ACM*, 7(3):171–176, Mar. 1964.

[8] M. E. Dincturk, S. Choudhary, G. von Bochmann, G.-V. Jourdan, and I. V. Onut. A statistical approach for efficient crawling of rich internet applications. In *Proc. of the 12th Int. Conf. on Web Engineering*, pages 362–369, Berlin, 2012. Springer.

[9] E. C. Dragut, T. Kabisch, C. Yu, and U. Leser. A hierarchical approach to model web query interfaces for web source integration. In *Proc. of VLDB Endowment*, volume 2, pages 325–336. VLDB, 2009.

[10] C. Duda, G. Frey, D. Kossmann, R. Matter, and C. Zhou. AJAX Crawl: Making AJAX applications searchable. In *Proc. of the IEEE 25th Int. Conf. on Data Engineering*, pages 78–89. IEEE, 2009.

[11] M. Y. Erlewine. Ubiquity: Designing a multilingual natural language interface features of a natural syntax. In *SIGIR Workshop on IAMW*, page 4, Boston, 2009.

[12] R. R. Fayzrakhmanov. A blocks-based geometric model of web pages for automatic processing and information extraction. *Science and Business: Development Ways*, 15(9):56–64, 2012.

[13] R. R. Fayzrakhmanov. WPPS: A novel and comprehensive framework for web page understanding and information extraction. In *Proc of IADIS WWW/Internet*, pages 19–26, Madrid, 2012. IADIS.

[14] R. R. Fayzrakhmanov, M. C. Göbel, W. Holzinger, B. Krüpl, and R. Baumgartner. A Unified ontology-based web page model for improving accessibility. In *Proc. of the World Wide Web 2010*, pages 1087–1088, New York, NY, US, 2010. ACM.

[15] R. R. Fayzrakhmanov, C. Herzog, and I. Kordomatis. Web objects identification for web automation: objects and their features. Technical report DBAI-TR-2013-80, Institute of Information Systems, TU Vienna, Vienna, 2013.

[16] T. Furche, G. Gottlob, G. Grasso, X. Guo, G. Orsi, and C. Schallhart. OPAL: Automated form understanding for the deep web. In *Proc. of WWW 2012*, pages 829–838, New York, 2012. ACM.

[17] C. Herzog, I. Kordomatis, W. Holzinger, R. R. Fayzrakhmanov, and B. Krüpl-Sypien. Feature-based object identification for web automation. In *Proc. of the 28th Annual ACM SAC'13*, pages 742–749, Coimbra, Portugal, 2013. ACM.

[18] J. Keith. *DOM Scripting: Web design with JavaScript and the Document Object Model*. Springer, New York, the USA, 2005.

[19] R. Khare and Y. An. An empirical study on using hidden markov model for search interface segmentation. In *Proc. of the 18th ACM CIKM '09*, page 17, New York, 2009. ACM.

[20] R. Khare, Y. An, and I.-Y. Song. Understanding deep web search interfaces: A survey. *ACM SIGMOD Record*, 39(1):33–40, 2010.

[21] B. Krüpl-Sypien, R. R. Fayzrakhmanov, W. Holzinger, M. Panzenböck, and R. Baumgartner. A versatile model for web page representation, information extraction and content re-packaging. In *Proc. of the DocEng'11*, pages 129–138, 2011.

[22] G. Leshed, E. M. Haber, T. Matthews, and T. Lau. CoScripter: automating & sharing how-to knowledge in the enterprise. In *Proc. of the SIGCHI '08*, pages 1719–1728, Florence, 2008. ACM.

[23] P. McCullagh and J. Nelder. *Generalized linear models*, volume 37. 1989.

[24] A. Memon, I. Banerjee, and A. Nagarajan. GUI ripping: reverse engineering of graphical user interfaces for testing. In *Proc. of the 10th WCRE '03*, pages 260–269, Washington, 2003. IEEE.

[25] A. Mesbah, E. Bozdag, and A. V. Deursen. Crawling AJAX by inferring user interface state changes. In *Proc. of the ICWE '08*, pages 122–134, Washington, July 2008. IEEE.

[26] H. Nguyen, T. Nguyen, and J. Freire. Learning to extract form labels. *Proc. of the VLDB Endowment*, 1(1):684–694, 2008.

[27] V. Vapnik. Statistical learning theory, 1998.

[28] R. Vulanović and R. Köhler. Syntactic units and structures. In *Quantitative Linguistics*, pages 274–291. de Gruyter, Berlin, 2005.

[29] Y. Wang and I. Witten. Induction of model trees for predicting continuous classes. 1996.

[30] Z. Zhang, B. He, and K. C.-C. Chang. Understanding web query interfaces: best-effort parsing with hidden syntax. In *Proc. of the ACM COMAD'04*, pages 107–118, 2004.

Table 3: Object features and distances.

	Feature	T	Description	Dist.	
Selected Objects	Object type	E	Type of object, e.g., button, input field.	Equ.	**IF**
	Editable	B	Whether the object is editable.	Bool.	
	Selection	B	Whether a checkbox or radio button is selected	Bool.	
	Area	R	The area of the bounding box in pixels.	Rel.	**SF**
	Aspect ratio	R	The aspect ratio between width and height.	Rel.	
	Foreground color	C	Foreground color of the object in HSV color space.	Color	**VPF**
	Background color	C	Background color of the object in HSV color space.	Color	
	Emphasis	R	Value representing the level of emphasis (font weight, style, etc.).	Rel.	
	Font size	R	Font size of the text.	Rel.	
	Text	S	Text of the object.	Edit	**TF**
	Number of lines	I	How many rows the object contains.	Rel.	
	Number of tokens	I	How many tokens (usually words) the text contains.	Rel.	
Selected object – Context	Type of dominant object	E	Type of the dominant orthogonally object.	Equ .	**IF**
	Objects of same type	I	Number of objects in the context that have the same type.	Rel.	
	Aligned objects (context)	I	Number of objects horizontally or vertically aligned with the selected object.	Rel.	**SF**
	Horizontally aligned objects	I	Number of objects horizontally aligned with the selected object.	Rel.	
	Vertically aligned objects	I	Number of objects vertically aligned with the selected object.	Rel.	
	Horizontal index (context)	I	Index of the selected object in the set of horizontally aligned objects.	Abs.	
	Vertical index (context)	I	Index of the selected object in the set of vertically aligned objects.	Abs.	
	Alignment factor	R	Ratio of objects aligned to the selected object to unaligned ones.	Rel.	
	V/H alignment ratio (context)	R	Ratio of number of vertically aligned objects to number of horizontally aligned ones.	Rel.	
	Orthogonally visible obj.	I	Number of orthogonally visible objects.	Rel.	
	Aligned orth. visible obj.		Number of orthogonally visible objects aligned with the target object.	Rel.	
	Fully aligned orth. visible obj.	I	Number of orthogonally visible objects fully aligned with the target object.	Rel.	
	Pixels to character ratio	R	Average are in the context that is occupied by characters.	Rel.	
	Average foreground color distance	R	Average HSV foreground color distance between the target object and all other objects in the context.	Abs.	**VPF**
	Average background color distance	R	Average HSV background color distance between the target object and all other objects in the context.	Abs.	
	Upper text	S	Merged text of the upper orthogonal visible objects.	Edit	**TF**
	Right text	S	Merged text of the orthogonal visible objects on the right.	Edit	
	Lower text	S	Merged text of the lower orthogonal visible objects.	Edit	
	Left text	S	Merged text of the orthogonal visible objects on the left.	Edit	
	Most similar text distance	–	Distance comparing the upper, lower, left and right text, taking the most similar ones.	–	
	Orthogonal nearest text	S	Text of the nearest ortogonal visible object in the context.	Edit	
	Nearest text	S	Text of the nearest object in the context.	Edit	
Selected object – Page	Relative Width	R	Width of the selected object in relation to the page.	Abs.	**SF**
	Relative Height	R	Height of the selected object in relation to the page.	Abs.	
	Relative X	R	X position of the the selected object in relation to the page.	Abs.	
	Relative Y	R	Y position of the selected object in relation to the page.	Abs.	
Selected object – Top page	Link Type	E	Specifies whether the target of a link is within the same page, same domain or outside.	Equ.	**IF**
	3x3 Grid Location	M	Dividing the web page in a 3x3 grid, this specifies which areas are touched.	Grid	**SF**
Selected object – Document	Alignments (document)	I	Number of objects in the document which are in any alignment with the selected object.	Rel.	**SF**
	Horizontal Alignments (document)	I	Number of objects in the document which are horizontally aligned with the selected object.	Rel.	
	Vertical Alignments (document)	I	Number of objects in the document which are vertically aligned with the selected object.	Rel.	
	Horizontal Index (document)	I	Index of the selected object in the sequence of horizontally aligned objects.	Abs.	
	Vertical Index (document)	I	Index of the selected object in the sequence of vertically aligned objects.	Abs.	
	V/H Alignment Ratio (document)	R	Ratio of objects vertically to objects horizontally aligned with the selected one.	Rel.	
Context	Objects	R	Total number of objects contained in the context.	Rel.	**IF**
	Text density	R	Area of the context used by text divided by the total context area.	Abs.	**SF**
	Link density	R	Area of the context used by links divided by the total context area.	Abs.	
	Link character density	R	Ratio of characters in links to all characters in the context.	Abs.	**TF**

Data Type (T): Real(R), Integer(I), Enumeration(E), Boolean(B), RGBA Color(C), Bitmap(M), String(S).
Feature groups: interface feature (IF), spatial feature (SF), visual perception feature (VPF), textual feature (TF).