Neurocomputing 135 (2014) 61–68

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

A class of neural-network-based transducers for web information extraction

Hassan A. Sleiman *, Rafael Corchuelo

University of Sevilla, ETSI Informática, 41012 Sevilla, Spain.

ARTICLE INFO

Article history:
Received 31 October 2012
Received in revised form
8 March 2013
Accepted 30 May 2013
Available online 18 January 2014

Keywords:
web wrappers
web information extraction
neural networks
finite automata
machine learning
supervised method

ABSTRACT

The Web is a huge and still growing information repository that has attracted the attention of many companies. Many such companies rely on information extractors to integrate information that is buried into semi-structured web documents into automatic business processes. Many information extractors build on extraction rules, which can be handcrafted or learned using supervised or unsupervised techniques. The literature provides a variety of techniques to learn information extraction rules that build on ad hoc machine learning techniques. In this paper, we propose a hybrid approach that explores the use of standard machine-learning techniques to extract web information. We have specifically explored using neural networks; our results show that our proposal outperforms three state-of-the-art techniques in the literature, which opens up quite a new approach to information extraction.

1. Introduction

The growth of the Web has raised many companies' interest in using the information that it provides to feed their business processes. Unfortunately, extracting information from web documents is not easy at all, which has motivated many researchers to work on the so-called information extractors. The common theme is to help transform web documents into structured information; that is, information for which there is an explicit model, so that it can be easily consumed by automatic business processes.

The information in the Web ranges from free-text to semi-structured information. Free-text web documents provide information that is buried into text that is written in natural language, with a few HTML tags that endow it with a little structure, e.g., headings, sections, or sidebars. Unfortunately, this little structure is not enough to characterise the information to be extracted; thus, natural language processing techniques are required to extract relevant information from these documents [22,31]. On the contrary, the information in a semi-structured web document is formatted using regular patterns, e.g., grids and lists. In these documents, HTML tags provide far more structure than in free-text documents since the pieces of information to extract are usually enclosed within formatting tags. Such pieces of information are usually referred to as slots in this context [4,27].

Our work focusses on information extraction from semi-structured web documents. The proposals in this field can be classified into two categories, namely: rule-based and heuristic-based. Rule-based information extractors rely on the so-called extraction rules, which range from regular expressions to context-free grammars, horn clauses, tree templates, or transducers, to mention a few. Heuristic-based proposals do not rely on extraction rules, but are based on a number of heuristics that have proven to work well on a large number of web sources [1,8,9,24,28]. Extraction rules can be handcrafted [6,21], but the costs involved motivated many researchers to work on proposals to learn them automatically using supervised and unsupervised learning techniques. Supervised learning techniques require the user to provide samples of the information to be extracted, aka annotations [10,14,19,29,30]. Unsupervised learning techniques learn extraction rules starting from one or more web documents; these rules extract as much prospective information as they can and the user then gathers the relevant information from the results [2,3,7,11,17,26]. Since typical web documents are growing in complexity, some authors are also working on techniques whose goal is to identify the region within a web document where the relevant information is most likely to reside [27]. Other authors have studied the problem of verifying the information extracted [12,13,15,18]; their focus was on determining when an information extractor stops performing well due to changes in the structure of the documents from which they have to extract information. Other authors have focussed on how to maintain information extractors automatically, that is, on how to adapt them to changes with minimum user intervention [16,20].

* Corresponding author.
E-mail addresses: hassansleiman@us.es (H.A. Sleiman), corchu@us.es (R. Corchuelo).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2013.05.057

Since automatic techniques build on machine learning techniques or heuristics [27], their precision and recall may not generally expected to be 100% in every case. This makes this quite an active research area. A user might also handcraft ad hoc information extractors that might have perfect precision and recall for a given web site, but the effort this task requires the high precision and recall that automatic techniques may achieve, render handcrafting rules not very appealing in general. We have found out that, without an exception, the automatic proposals in the literature build on ad hoc machine-learning techniques. As far as we know, the idea of using standard machine-learning techniques in this field remains largely unexplored. In many conversations with other researches in conferences world wide, we have found out that the usual problem that motivates researchers to work on ad hoc machine-learning techniques is that it is not easy to map the problem of information extraction onto the dataset tables that are required by standard machine-learning techniques. This has motivated us to work on this topic: we wished to explore it in an attempt to find out if ad hoc machine-learning techniques can be outperformed by means of standard machine-learning techniques.

Our proposal builds on transducers, which are regular automata that produce an output as they move from state to state; the outputs of our transducers are expected to be information that is extracted from input web documents or fragments. Transducers have proven to be very useful for information extraction in the past [5,10] since they have the ability to deal with web documents in which there is optional information, information that is displayed with different orderings, multi-valued information, and multiple formats for the same information. What is innovative in our proposal is that we have explored the idea of using neural networks to control how transitions take place. This resulted in a hybrid approach that can easily map the problem of web information extraction onto the dataset tables that are common to standard machine-learning techniques. The decision on why to focus on neural networks was because they are quite a powerful machine-learning technique that is characterised by its ability to find patterns in complex datasets.

The rest of the paper is organised as follows: Section 2 reports on the details of our proposal; Section 3 reports on our experimental evaluation, which proves that our proposal outperforms other state-of-the art techniques that build on ad hoc machine-learning techniques; Section 4 concludes our work and highlights a few future research directions.

2. Description of our proposal

Our proposal builds on using transducers, so we first provide a formal definition of this concept:

Definition 1. A transducer is a tuple of the form (S, i, f, T), where S is a set of states, i denotes a state in S that is referred to as the initial state, f denotes a state in S that is referred to as the final state, and T is a set of transitions of the form (p, n, q), where p and q denote states in S and n is a neural network that encodes the conditions required to move from state p to state q.

In the following subsections, we delve into the details regarding how to learn a transducer from a training set and how to run it on an input web document. The details are illustrated using the sample web document in Fig. 1. Note that this sample web document is intentionally simple and that it is fully annotated since it is intended for illustration purposes only. Our experience proves that it generally suffices to provide a few annotations of every possible formatting, like in other state-of-the-art supervised proposals.

2.1. Learning a transducer

Our proposal relies on learning a transducer from a training set that consists of annotated web documents. By annotation we refer to a slot to which the user has assigned an explicit label; by slot we refer to either a record or an attribute inside a record. We do not preclude the possibility that an attribute can also be a record since a transducer can be applied to a whole web document or to a fragment in that document that was extracted previously by means of another transducer. For instance, the sample web document in Fig. 1 has four record annotations (`Book`) and several attribute annotations (`title`, `author`, `price`, and `isbn`).

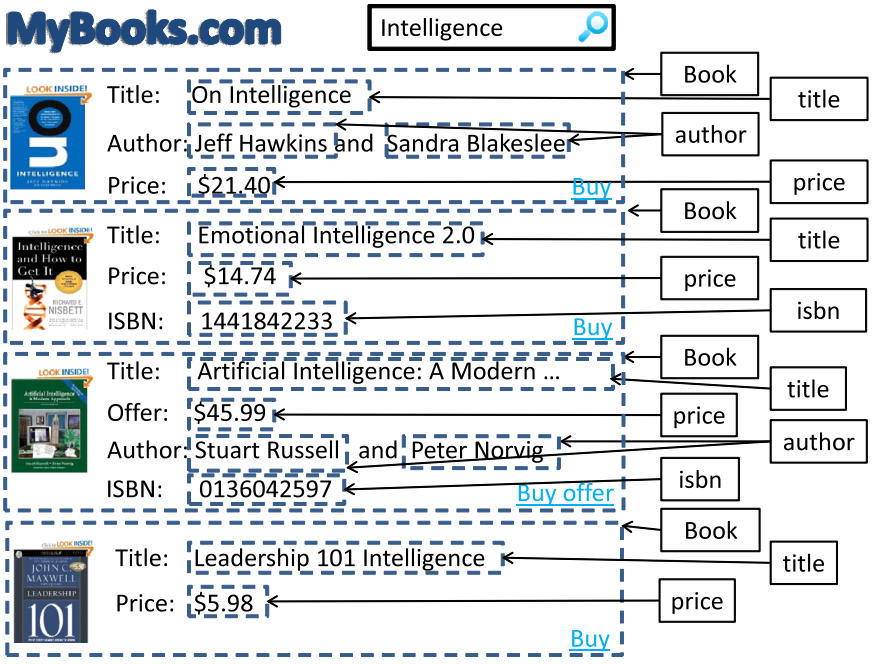
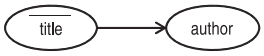


Fig. 1. A sample web document with annotations.

```
1: learnTransducer(W: set of AnnotatedWebDocument): Transducer
2:     i = create initial state
3:     f = create final state
4:     S = {i, f}
5:     M = ∅
6:     for each web document w in W do
7:         for each annotation a in w in increasing order of their offsets do
8:             S = S ∪ {a, ā}
9:             if a is the first annotation in w then
10:                t = (i, a)
11:                M = M ∪ {t ↦ M(t) ∪ compute windows of t}
12:            else
13:                b = previous annotation
14:                t = (b̄, a)
15:                M = M ∪ {t ↦ M(t) ∪ compute windows of t}
16:            end if
17:            if a is the last annotation in w then
18:                z = (ā, f)
19:                M = M ∪ {z ↦ M(z) ∪ compute windows of z}
20:            end if
21:            v = (a, ā)
22:            M = M ∪ {v ↦ M(v) ∪ compute windows of v}
23:        end for
24:    end for
25:    T = ∅
26:    for each t in domain M do
27:        m = transitions in domain M that have the same source as t
28:        n = learn a neural network from M(t) union M(m) as negative instances
29:        T = T ∪ {(source of t, n, target of t)}
30:    end for
31: return (S, i, f, T)
```

Fig. 2. Algorithm `learnTransducer`.

T1	T2	T3	T4	Class
On	Intelligence	 	Author	No
Intelligence	 	Author	:	No
 	Author	:	Jeff	Yes
Emotional	Intelligence	2.0	 	No
Intelligence	2.0	 	Price	No
...	...	...	...	...



T1	T2	T3	T4	Class
 	Author	:	Jeff	No
Author	:	Jeff	Hawkings	No
:	Jeff	Hawkings	and	No
Jeff	Hawkings	and	Sandra	Yes
 	Author	:	Stuart	No
...	...	...	...	...

Fig. 3. Sample windows tables.

Fig. 2 presents the algorithm that we have devised to learn a transducer from a set of annotated input documents.

The core of the algorithm is the loop at lines 6–24, which iterates over the set of input documents; for each such document, it iterates over its annotations, which must be sorted according to their offset. The inner loop first adds a couple of states to S (the set of states of the resulting transducer) for every annotation. We refer to these states as the positive and the negative states of the annotation and denote them as a and $\overline{a}$, for any given annotation a. Intuitively, a positive state represents a state in which the transducer detects the beginning of a slot and prepares to extract its tokens; similarly, the negative state represents a state in which the transducer detects the end of a slot and outputs the tokens that have been extracted so far. S is initialised to the initial and final states at lines 2–4. In our running example, the set of states includes `Book`, $\overline{\texttt{Book}}$, `title`, $\overline{\texttt{title}}$, `author`, $\overline{\texttt{author}}$, `price`, $\overline{\texttt{price}}$, `isbn`, and $\overline{\texttt{isbn}}$, plus the initial and the final states.

We also need connect some states by means of transitions. The problem is that each transition relies on a neural network that must be learned from sample data, and we cannot have that information until we have processed every annotation in every input web document. To solve this problem, we rely on an intermediate map to which we refer to as M in our algorithm. M maps pairs of states onto sets of so-called windows tables; by window we refer to a subsequence of tokens of an input web document. Fig. 3 illustrates part of the tables in our running

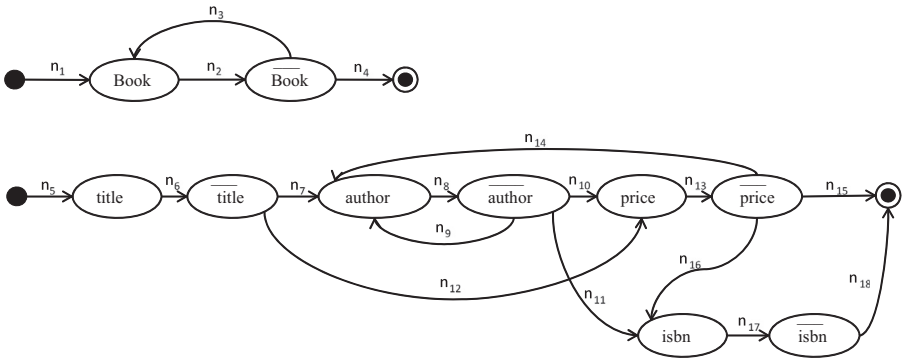


Fig. 4. Two sample transducers.

```
1: runTransducer(T: Transducer; F: Fragment): sequence of Slot
2:     s = initial state of T
3:     R = ⟨⟩
4:     e = ⟨⟩
5:     while F ≠ ⟨⟩ and s is not the final state of T do
6:         case for all (s, n, q) ∈ T, then n(window(F)) = false:
7:             if s is positive then
8:                 e = e + ⟨firstF⟩
9:             end if
10:            F = tail F
11:        otherwise, there exists (s, n, q) ∈ T such that n(window(F)) = true:
12:            if s is positive then
13:                R = R + ⟨(s, e)⟩
14:            end if
15:            s = q
16:            e = ⟨⟩
17:    end while
18: return R
```

Fig. 5. Algorithm `runTransducer`.

example. The table on the left corresponds to the transition between states `title` and $\overline{\texttt{title}}$, which occurs every time the next tokens in the input web document indicate that a title has just been recognised and thus has to be extracted. A window is a small subsequence of tokens, from which we learn the neural networks that help decide when a transition between two states must occur. The size of the window is a parameter that must be set beforehand. In our running example, we use four tokens for illustration purposes only. Similarly, the table on the right corresponds to the transition between states $\overline{\texttt{title}}$ and `author`. For illustration purposes, we use the actual tokens to represent the previous tables; in our implementation we use a hash of the actual tokens so that they are numeric and can thus be used to learn a neural network.

Intuitively, *M* records the windows table from which the neural network that corresponds to every transition shall be learned later. The algorithm proceeds as follows between lines 9 and 22:

1. If the annotation being processed is the first one in an input web document, then we record that there is a transition from the initial state to the positive state that corresponds to that annotation, cf. lines 10–11.
2. If it is not the first annotation, then we record that there is a transition from the negative state that corresponds to the previous annotation to the positive state that corresponds to the annotation being processed, cf. lines 13–15.
3. If it is the last annotation, then we record that there is a transition from its corresponding negative state to the final state, cf. lines 18–19.
4. In every case, we record that there is a transition between the positive and the negative state of the annotation being processed, cf. lines 21–22.

For the sake of simplicity, in our pseudocode, we assume that if *M* is applied to a pair that has not been recorded previously, then it returns an empty set.

Once the main loop at lines 6–24 has finished executing, *S* contains a positive and a negative state for each type of annotation in the training set and *M* maps every pair of states that must be connected by means of a transition to a windows table. The loop at lines 26–30 iterates over the domain of map *M* and then learns a neural network from those windows tables in order to create a new transition that is added to set *T*: line 27 collects the transitions that have the same source as the current transition *t* and saves them in *m*; then, line 28 creates a new table that contains the table created previously for the current transition *t* and the instances inside the tables of the transitions in *m* as negative instances; this new table is used to learn a neural network for the current transition *t*. Finally, the current transition and its neural network are stored in the transition set *T* at line 29. The idea that lies behind joining the table of the current transition with the tables of other transitions at lines 27 and 28 is to alleviate non-determinism, that is, situations in which several transitions get activated at the same time.

Once the previous loop finishes, (S, i, f, T) is returned at line 31 as the resulting transducer. Fig. 4 illustrates the two transducers that our algorithm learns to extract book records and their attributes, respectively. As usual, the initial state is represented as a black circle and the final state is represented as a black circle inside a white circle. Note that we have denoted the neural networks in the transitions as $n_1, n_2, \ldots, n_{18}$ since we have not committed to a particular network model in this example. Later, we provide more details on our experiments using several types of network models.

2.2. Running a transducer

Once a transducer is learned, it can be run on new web documents to extract information from them; obviously, the new documents must be similar to the documents that were used to learn the transducer or otherwise it shall fail to work. Fig. 5 presents the algorithm that we have devised to run a transducer on a fragment, which is the name that we use to refer to either a whole web document or a subsequence of tokens that was extracted previously by another transducer. The algorithm returns a sequence of slots, which is represented by means of pairs of the form (p, t), where p denotes a state and t a sequence of tokens; intuitively, p gives a label to t.

The core of the algorithm is the loop at lines 5–17, in which it iterates as long as the fragment is not empty and the current state is not the final state. There are two cases in each state, namely:

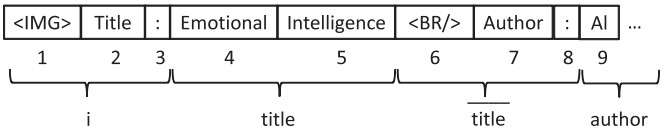
1. None of the transitions from the current state, s, can be executed (lines 6–10); that is, the neural networks that are associated with the outgoing transitions from state s return *false* on the current window of tokens from the input fragment, F. In this case, if the current state is positive, then we need add the first token in F to e, which is a temporary variable in which we accumulate the tokens that shall later be extracted when we reach the corresponding negative state. Furthermore, we then discard the current input token so that we can analyse the tail of the input fragment.
2. Otherwise, there is a transition (s, n, q) that allows us to move from the current state, s, to a new state q (lines 11–16); that is, the neural network that is associated with that transition, n, returns *true* when applied to the next window of tokens from the input fragment. In this case, if the current state is a positive state, then we need to add a new pair to variable R, which records the result of running the transducer; this pair consists of the current state, which provides a label, and the sequence of tokens that we have accumulated so far in variable e. Furthermore, we change the current state to q and reset e to an empty sequence.

Table 1
Experimental results.

Summary	MLP			VP			RBFN			SoftMealy			RoadRunner			FiVaTech		
Site	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Mean	0.95	0.96	0.96	0.54	0.42	0.46	0.94	0.95	0.94	0.84	0.61	0.65	0.49	0.54	0.50	0.79	0.90	0.82
Standard deviation	0.09	0.07	0.08	0.19	0.12	0.14	0.06	0.04	0.05	0.14	0.33	0.30	0.44	0.46	0.44	0.19	0.13	0.14
www.abebooks.com	0.68	0.77	0.72	0.64	0.44	0.52	0.84	0.91	0.87	0.87	0.58	0.69	–	–	–	0.92	0.99	0.95
www.awesomebooks.com	0.53	0.64	0.58	0.47	0.38	0.41	0.95	0.93	0.93	1.00	0.39	0.56	1.00	1.00	1.00	0.85	1.00	0.91
www.betterworldbooks.com	0.92	0.95	0.93	0.54	0.37	0.43	0.95	0.94	0.94	0.98	0.99	0.98	0.00	0.00	0.00	0.99	0.96	0.97
www.manybooks.net	1.00	1.00	1.00	0.55	0.53	0.54	1.00	1.00	1.00	0.99	0.99	0.99	–	–	–	0.77	0.97	0.85
www.waterstones.com	0.99	0.99	0.99	0.63	0.48	0.54	0.98	0.98	0.97	1.00	1.00	1.00	1.00	0.89	0.94	1.00	0.94	0.97
www.autotrader.com	0.98	0.98	0.97	0.25	0.35	0.29	0.96	0.97	0.96	0.89	0.87	0.87	0.00	0.00	0.00	–	–	–
www.carmax.com	1.00	1.00	1.00	0.25	0.27	0.25	1.00	1.00	1.00	0.89	0.89	0.89	0.00	0.00	0.00	0.45	0.89	0.59
www.carzone.ie	0.99	0.99	0.99	0.13	0.12	0.12	0.95	0.95	0.94	0.92	0.02	0.04	0.00	0.00	0.00	0.92	1.00	0.95
www.classiccarsforsale.co.uk	0.91	0.93	0.91	0.18	0.19	0.18	0.91	0.94	0.92	0.90	0.90	0.89	0.00	0.00	0.00	–	–	–
events.linkedin.com	0.99	1.00	0.99	0.73	0.39	0.50	0.85	0.92	0.88	0.87	0.55	0.67	–	–	–	–	–	–
www.allconferences.com	1.00	1.00	1.00	0.69	0.50	0.58	0.99	0.99	0.99	0.99	0.25	0.40	0.84	0.90	0.86	0.84	0.90	0.86
www.mbendi.com	1.00	1.00	1.00	0.70	0.57	0.62	0.94	0.97	0.95	0.60	0.60	0.60	0.90	1.00	0.94	0.90	1.00	0.94
www.netlib.org	0.90	0.94	0.91	0.68	0.46	0.54	0.98	0.98	0.97	0.87	0.44	0.58	0.39	0.50	0.43	0.39	0.50	0.43
www.rdlearning.org.uk	1.00	1.00	0.99	0.79	0.40	0.53	0.98	0.98	0.98	0.52	0.39	0.44	0.99	0.79	0.87	0.99	0.79	0.87
doctor.webmd.com	0.99	0.99	0.98	0.74	0.50	0.59	1.00	1.00	1.00	0.86	0.45	0.59	0.00	0.00	0.00	0.77	1.00	0.87
extapps.ama-assn.org	1.00	1.00	1.00	0.67	0.42	0.51	0.90	0.95	0.92	0.79	0.39	0.52	–	–	–	–	–	–
www.dentists.com	1.00	1.00	1.00	0.62	0.55	0.58	0.90	0.93	0.91	0.61	0.61	0.60	1.00	1.00	1.00	0.56	0.99	0.71
www.drscore.com	0.94	0.96	0.95	0.77	0.50	0.60	0.94	0.97	0.95	0.91	0.87	0.88	1.00	1.00	1.00	0.78	1.00	0.87
www.steadyhealth.com	0.99	0.98	0.98	0.74	0.69	0.71	0.99	0.98	0.98	0.75	0.25	0.37	0.00	0.00	0.00	0.83	0.83	0.83
careers.insightintodiversity.com	0.98	0.98	0.98	0.60	0.44	0.50	0.96	0.98	0.96	0.57	0.47	0.51	0.70	0.70	0.70	1.00	0.74	0.85
www.4jobs.com	0.88	0.93	0.90	0.59	0.28	0.38	0.94	0.94	0.93	0.45	0.25	0.32	0.00	0.00	0.00	–	–	–
www.6figurejobs.com	0.99	1.00	0.99	0.68	0.39	0.49	0.99	0.99	0.99	0.75	0.00	0.00	1.00	1.00	1.00	1.00	0.98	0.98
www.careerbuilder.com	0.99	0.99	0.99	0.79	0.70	0.74	0.94	0.88	0.90	0.75	0.00	0.00	0.00	0.00	0.00	0.80	0.83	0.81
www.jobofmine.com	1.00	1.00	1.00	0.68	0.43	0.52	0.98	0.97	0.97	0.75	0.03	0.06	0.86	1.00	0.92	–	–	–
www.albaniam.com	0.95	0.93	0.93	0.44	0.33	0.37	0.94	0.93	0.93	0.85	0.40	0.54	0.81	1.00	0.89	0.82	0.81	0.81
www.allmovie.com	1.00	0.99	0.99	0.41	0.41	0.41	0.98	0.97	0.97	0.93	0.29	0.43	0.27	0.30	0.28	0.79	0.74	0.76
www.citwf.com	0.91	0.93	0.91	0.43	0.34	0.37	0.92	0.93	0.92	0.92	0.72	0.80	1.00	1.00	1.00	1.00	1.00	1.00
www.disneymovieslist.com	0.97	0.97	0.96	0.53	0.43	0.47	0.97	0.97	0.97	0.97	0.97	0.96	0.00	0.00	0.00	0.71	0.67	0.69
www.imdb.com	0.98	0.98	0.97	0.58	0.43	0.49	0.93	0.93	0.92	0.88	0.85	0.86	0.00	0.00	0.00	–	–	–
www.soulfilms.com	0.96	0.96	0.96	0.37	0.36	0.36	0.95	0.96	0.95	0.99	0.95	0.96	0.00	0.00	0.00	0.59	1.00	0.74
realestate.yahoo.com	1.00	1.00	1.00	0.63	0.35	0.44	0.82	0.87	0.84	1.00	1.00	1.00	0.77	0.97	0.85	0.77	0.97	0.85
www.homes.com	0.95	0.95	0.95	0.67	0.43	0.52	0.88	0.92	0.89	0.80	0.79	0.79	–	–	–	–	–	–
www.remax.com	0.99	1.00	0.99	0.67	0.41	0.51	0.77	0.87	0.81	0.84	0.84	0.84	0.77	0.99	0.86	0.77	0.99	0.86
www.trulia.com	1.00	1.00	1.00	0.62	0.62	0.61	0.86	0.92	0.89	0.88	0.92	0.90	–	–	–	–	–	–
baseball.playerprofiles.com	0.98	0.97	0.97	0.59	0.39	0.47	0.98	0.98	0.98	0.83	0.13	0.22	0.36	0.99	0.52	0.36	0.99	0.52
en.uefa.com	0.99	0.99	0.99	0.41	0.51	0.45	1.00	1.00	1.00	1.00	1.00	1.00	–	–	–	–	–	–
www.atpworldtour.com	0.94	0.96	0.94	0.29	0.37	0.32	0.96	0.94	0.95	0.94	0.94	0.94	0.99	0.88	0.93	0.99	0.88	0.93
www.nfl.com	0.98	0.99	0.98	0.28	0.30	0.29	0.97	0.99	0.98	0.71	0.71	0.71	0.53	0.81	0.64	0.53	0.81	0.64
www.soccerbase.com	0.97	0.98	0.97	0.17	0.25	0.19	0.87	0.91	0.88	0.89	0.89	0.88	–	–	–	–	–	–

Note that an alternative to the algorithm *runTransducer* presented in Fig. 5 is to devise a recursive algorithm that tries every outgoing transition at each state until reaching the final state in the transducer or the input fragment is exhausted. We have chosen the current implementation since, according to our experimental results, the recursive implementation of this algorithm did not achieve significant improvements; that proves that the technique used to learn a transition in the context of the others works quite effectively to alleviate non-determinism.

To illustrate how the algorithm works, we use the following sample fragment, which represents the initial tokens of a sample book record:



and the transducer at the bottom of Fig. 4. It starts running at the initial state *i*, where it analyses the first window of the input fragment: ⟨⟨`IMG`⟩, `Title`, :, `Emotional`⟩ (recall that we are using four tokens in our running example). For illustration purposes, we may safely assume that neural network n_5, which controls the transition between the initial state and state `title`, returns *false*, which indicates that the window being analysed does not activate this transition. Since the initial state is not a positive state, then the input fragment is deprived of its first token and the process is repeated again. No transition shall be activated until the transducer reaches window ⟨`Emotional`, `Intelligence`, ⟨`BR/`⟩, `Author`⟩, which delimits the beginning of a title. Then we can assume that neural network n_5 returns *true*, which causes the current state to shift to `title`, but note that the fragment being analysed remains the same. In other words, we shall re-analyse it in state `title`; in this case, we can assume that neural network n_6 shall not return *true* until the transducer reaches window ⟨⟨`BR/`⟩, `Author`, :, `Al`⟩, in which a transition to state $\overline{\texttt{title}}$ occurs; since this transition happens from a positive to a negative state, that implies that the tokens in between are returned by the transducer using the positive state as a label, i.e., it returns (`title`, ⟨`Emotional`, `Intelligence`⟩).

3. Experimental results

In this section, we present the results of the experiments we have carried out to compare our proposal to other techniques in the literature from an empirical point of view. We first describe our experimentation environment and then present our results and analyse them statistically.

3.1. Experimentation environment

We have developed a Java 1.7 prototype of our proposal using the CEDAR framework [25]. We performed a series of experiments on a virtual computer that runs on our University Cloud infrastructure; it was equipped with a four-threaded Intel Core i7 processor that ran at 2.93 GHz, had 4 GiB of RAM, Windows 7 Pro 64-bit, Oracle's Java Development Kit 1.7.0_02, and Weka 3.6.8. The configuration parameters of the Java Virtual Machine were set to their default values. The size of the windows in our algorithms was set to 10 tokens. Note that 10 tokens is not a universal value; there are web sites in our experiments for which three tokens is sufficient to recognise when an information slot starts, but 10 is a size that, according to our experiments, is a good threshold; using less tokens had a negative impact on the average effectiveness on our datasets, and adding more tokens did not contribute to improving it significantly.

We performed our experiments on a collection of 39 datasets that provide a total of 1170 web documents that were gathered from 39 real-world web sites. It contains datasets on books, cars, conferences, doctors, jobs, movies, real estates, and sports. These categories were randomly sampled from The Open Directory subcategories, and the web sites inside each category were randomly selected from the 100 best ranked web sites between December 2010 and March 2011 according to Google's search engine. We downloaded 30 web documents from each web site and handcrafted a set of annotations with the slots that we would like to extract from each document. We fully annotated every document so that we could compute precision and recall. In general, the same results may be achieved by annotating just a few samples; for instance, in documents with several records, it generally suffices to annotate the first record, the last record and a record in the middle for the technique to learn a good transducer. The

Table 2
Statistical ranking.

Measure	Sample ranking		Iman-Davenport's test	Hommel's test						Statistical ranking	
	Technique	Rank	*P*-value	*P*-value	RBFN	VP	SoftMealy	RoadRunner	FiVaTech	Technique	Rank
P	MLP	1.90	6.66E-16	MLP	0.250	2.92E-14	4.48E-03	5.62E-10	0.000	MLP	1
	RBFN	2.37								RBFN	1
	FT	3.13								FiVaTech	2
	SM	4.15								SoftMealy	2
	RR	4.45								RoadRunner	2
	VP	5.00								VP	2
R	MLP	1.86	0.00E+00	MLP	0.226	2.53E-15	3.58E-08	1.30E-07	0.024	MLP	1
	RBFN	2.32								RBFN	1
	FT	3.22								FiVaTech	1
	RR	4.00								RoadRunner	2
	SM	4.55								SoftMealy	2
	VP	5.05								VP	2
F1	MLP	1.65	2.22E-16	MLP	0.364	1.19E-15	1.52E-06	1.11E-10	8.80E-05	MLP	1
	RBFN	2.12								RBFN	1
	FT	3.42								FiVaTech	2
	RR	4.40								RoadRunner	2
	SM	4.46								SoftMealy	2
	VP	4.95								VP	2

cases in which this does not suffice are cases in which records are displayed using alternate formattings; in such cases, the user must annotate at least one example of every possible formatting. The results regarding precision and recall were calculated using 10-fold cross validation.

We searched the Web and asked the authors of other proposals for implementations of their techniques to compare them with ours. Unfortunately, we only managed to get three techniques, namely: SoftMealy [10], RoadRunner [7], and FiVaTech [11]. We compared the effectiveness of these techniques to our proposal using three different neural networks, namely: Multi-Layer Perceptron (MLP), Voted Perceptron (VP), and Radial Basis Function Network (RBFN). We used the default Weka configuration for each kind of network.

3.2. Experimental results

We ran our proposal (using MLP, VP, and RBFN), SoftMealy, RoadRunner, and FiVaTech on our datasets in order to learn extraction rules. Then, we measured the standard effectiveness measures (precision, recall, and the $F1$ measure). Since we handcrafted annotations for every web document in our datasets, we could easily compare the extracted slots by each of the supervised techniques to the annotated samples and calculate the number of true positives (tp), false negatives (fn), and false positives (fp). This allowed us to compute precision as $P = tp/(tp+fp)$, recall as $R = tp/(tp+fn)$, and the $F1$ measure as $F1 = 2PR/(P+R)$. In the case of RoadRunner and FiVaTech, which are unsupervised, we compared each group of extracted slots to the annotations we handcrafted and considered the group that achieves the highest $F1$ measure as the correct group.

Table 1 reports on the results of our experiments. The columns report on the web site from which the web documents were downloaded, precision (P), recall (R), and the $F1$ measure ($F1$) for each technique. The first two rows provide a summary of these measures in terms of mean values and standard deviations. A dash in a cell means that the corresponding technique was not able to learn an extraction rule in 15 CPU minutes. Note that the results seem to suggest that our proposal performs better than the other state-of-the-art techniques, which is an experimental confirmation that our hybrid approach based on transducers and neural networks is quite appealing to extract information from semi-structured web documents.

Unfortunately, there are a few cases in which our technique does not perform well. We have analysed these cases and we have characterised them as follows:

- Datasets in which the number of training data for a certain transition is too small: the transducer learned by our technique should have at least one example for each possible transition. Since we performed cross-validation, the small number of training data for some transitions in some datasets has decreased the efficiency in some cases, i.e., these special cases appeared in the test set but not in the training set during cross validation, which had an impact on the result.
- In the cases in which the tokens that appear before different attributes are similar, i.e., they are encoded identically in the windows tables. This, has generated ambiguities while choosing the next transition, and reduced the effectiveness in some cases.

3.3. Statistical analysis

To confirm that the conclusions we have drawn from our empirical evaluation are valid, we need to perform a statistical analysis [23], which consists in performing a statistical ranking regarding the effectiveness measures.

The first step is to determine if the evaluation results are normally distributed and have equal variances; in such a case we must perform a parametrical analysis; otherwise, we would perform a non-parametrical analysis. We have conducted a Shapiro-Wilk test at the standard significance level $\alpha = 0.05$ on every measure and we have found out that none of them behave normally. As a conclusion, we have performed a non-parametric analysis, which consists of the following steps: (i) compute the rank of each technique from the experimental data; (ii) determine if the differences in ranks are significant or not using Iman-Davenport's test; (iii) if the differences are significant, then use Hommel's test to compare the best-ranking technique to the others in order to find out which ones rank equal and which ones do not.

Table 2 presents the results of the statistical analysis. Note that the P-value of Iman-Davenport's statistic is nearly zero in every case, which is a strong indication that there are statistically significant differences in the ranks we have computed from our experiments. It then proceeds to rank the techniques using Hommel's test. For the sake of readability, we also provide an explicit ranking in the last column. Note that the MLP and RBFN variations of our proposal rank the first regarding every effectiveness measure. As a conclusion, our experiments prove that there is enough statistical evidence to conclude that if we use MLP or RBFN with our proposal, we can outperform the other techniques.

4. Conclusions

In this paper, we have presented a new class of transducers for web information extraction. It is a hybrid approach that combines a transducer that represents the structure of the information to be extracted and neural networks that model the conditions that allow the transducer to move from one state to another. We have implemented three variations of our proposal that build on Multi-Layer Perceptrons, Voted Perceptrons, and Radial Basis Function Networks. We have evaluated them empirically on a large collection of actual web documents, and we have found out that it outperforms three-state of the art techniques in the literature that build on ad hoc machine-learning techniques. Our results are very promising since, as far as we know, hybridising transducers and neural networks to implement web information extractors has been an unexplored field so far. In future, we plan on extending our results to explore how to combine different standard machine-learning techniques in order to improve the effectiveness of our transducers.

Acknowledgements

We greatly appreciate the help and the comments of Dr. Achim Rettinger from Karlsruhe Institute of Technology (KIT), with whom we exchanged some interesting preliminary ideas that led to the results in this article. This work was supported by the European Commission (FEDER), the Spanish and the Andalusian R&D&I programmes (Grant nos. TIN2007-64119, P07-TIC-2602, P08-TIC-4100, TIN2008-04718-E, TIN2010-21744, TIN2010-09809-E, TIN2010-10811-E, TIN2010-09988-E, and TIN2011-15497-E).

References

[1] Manuel Álvarez, Alberto Pan, Juan Raposo, Fernando Bellas, Fidel Cacheda, Extracting lists of data records from semi-structured web pages, Data Knowl. Eng. 64 (2) (2008) 491–509, http://dx.doi.org/10.1016/j.datak.2007.10.002.

[2] Arvind Arasu, Hector Garcia-Molina, Extracting structured data from web pages, in: SIGMOD Conference, 2003, pp. 337–348. http://dx.doi.org/10.1145/872757.872799.

[3] Chia-Hui Chang, Shao-Chen Lui, IEPAD: information extraction based on pattern discovery, in: WWW, 2001, pp. 681–688. http://dx.doi.org/10.1145/371920.372182.

[4] Chia-Hui Chang, Mohammed Kayed, Moheb R. Girgis, Khaled F. Shaalan, A survey of web information extraction systems, IEEE Trans. Knowl. Data Eng. 18 (10) (2006) 1411–1428, http://dx.doi.org/10.1109/TKDE.2006.152.
[5] Boris Chidlovskii, Bruno Roustant, Marc Brette, Documentum ECI self-repairing wrappers: performance analysis, in: SIGMOD Conference, pp. 708–717, 2006. http://dx.doi.org/10.1145/1142473.1142555.
[6] Valter Crescenzi, Giansalvatore Mecca, Grammars have exceptions, Inf. Syst. 23 (8) (1998) 539–565, http://dx.doi.org/10.1016/S0306-4379(98)00028-3.
[7] Valter Crescenzi, Giansalvatore Mecca, Automatic information extraction from large websites, J. ACM 51 (5) (2004) 731–779, http://dx.doi.org/10.1145/1017460.1017462.
[8] Hazem Elmeleegy, Jayant Madhavan, Alon Y. Halevy, Harvesting relational tables from lists on the Web, PVLDB 2 (1) (2009) 1078–1089, URL ⟨http://www.vldb.org/pvldb/2/vldb09-325.pdf⟩.
[9] Rahul Gupta, Sunita Sarawagi, Answering table augmentation queries from unstructured lists on the Web, PVLDB 2 (1) (2009) 289–300, URL ⟨http://www.vldb.org/pvldb/2/vldb09-652.pdf⟩.
[10] Chun-Nan Hsu, Ming-Tzung Dung, Generating finite-state transducers for semi-structured data extraction from the web, Inf. Syst. 23 (8) (1998) 521–538, http://dx.doi.org/10.1016/S0306-4379(98)00027-1.
[11] Mohammed Kayed, Chia-Hui Chang, FiVaTech: page-level web data extraction from template pages, IEEE Trans. Knowl. Data Eng. 22 (2) (2010) 249–263, http://dx.doi.org/10.1109/TKDE.2009.82.
[12] N. Kushmerick, Regression testing for wrapper maintenance, in: National Conference on Artificial Intelligence, 1999, pp. 74–79.
[13] N. Kushmerick, Wrapper verification, World Wide Web 3 (2) (2000) 79–94, http://dx.doi.org/10.1023/A:1019229612909.
[14] Nicholas Kushmerick, Daniel S. Weld, Robert B. Doorenbos, Wrapper induction for information extraction, in: IJCAI (1), 1997, pp. 729–737. ⟨ftp://ftp.cs.washington.edu/pub/ai/kushmerick-ijcai97.ps.Z⟩.
[15] K. Lerman, S. Minton, C.A. Knoblock, Wrapper maintenance: a machine learning approach, J. Artif. Intell. Res. 18 (2003) 149–181.
[16] Kristina Lerman, Steven Minton, Craig A. Knoblock, Wrapper maintenance: a machine learning approach, J. Artif. Intell. Res. (JAIR) 18 (2003) 149–181, http://dx.doi.org/10.1613/jair.1145.
[17] Bing Liu, Yanhong Zhai, NET: a system for extracting web data from flat and nested data records, in: WISE, 2005, pp. 487–495. http://dx.doi.org/10.1007/11581062_39.
[18] R. McCann, B.K. AlShebli, Q. Le, H. Nguyen, L. Vu, A. Doan, Mapping maintenance for data integration systems, in: International Conference on Very Large Data Bases, 2005, pp. 1018–1030.
[19] Ion Muslea, Steven Minton, Craig A. Knoblock, Hierarchical wrapper induction for semistructured information sources, Auton. Agents Multi-Agent Syst. 4 (1/2) (2001) 93–114, http://dx.doi.org/10.1023/A:1010022931168.
[20] Juan Raposo, Alberto Pan, Manuel Álvarez, Justo Hidalgo, Automatically maintaining wrappers for semi-structured web sources, Data Knowl. Eng. 61 (2) (2007) 331–358, http://dx.doi.org/10.1016/j.datak.2006.06.006.
[21] Arnaud Sahuguet, Fabien Azavant, Building intelligent web applications using lightweight wrappers, Data Knowl. Eng. 36 (3) (2001) 283–316, http://dx.doi.org/10.1016/S0169-023X(00)00051-3.
[22] Sunita Sarawagi, Information extraction, Found. Trends Databases 1 (3) (2007) 261–377, http://dx.doi.org/10.1561/1900000003.
[23] David J. Sheskin, Handbook of Parametric and Non-Parametric Statistical Procedures, 5 edition, Chapman and Hall/CRC, London, UK, 2011.
[24] Kai Simon, Georg Lausen, ViPER: augmenting automatic information extraction with visual perceptions, in: CIKM, 2005, pp. 381–388. http://dx.doi.org/10.1145/1099554.1099672.
[25] H.A. Sleiman, R. Corchuelo, An architecture for web information agents, in: ISDA, vol. 11, 2011, pp. 18–23. http://dx.doi.org/10.1109/ISDA.2011.6121624.
[26] Hassan A. Sleiman, Rafael Corchuelo, An unsupervised technique to extract information from semi-structured web pages, in: WISE, 2012, pp. 631–637. http://dx.doi.org/10.1007/978-3-642-35063-4_46.
[27] Hassan A. Sleiman, Rafael Corchuelo, A survey on region extractors from web documents, IEEE Trans. Knowl. Data Eng. 25 (9) (2012) 1960–1981. http://dx.doi.org/10.1109/TKDE.2012.135.
[28] Hassan A. Sleiman, Rafael Corchuelo, Tex: an efficient and effective unsupervised web information extractor, Knowl.-Based Syst. 39 (2013) 109–123, http://dx.doi.org/10.1016/j.knosys.2012.10.009.
[29] Stephen Soderland, Learning information extraction rules for semi-structured and free text, Mach. Learn. 34 (1–3) (1999) 233–272, http://dx.doi.org/10.1023/A:1007562322031.
[30] Cui Tao, David W. Embley, Automatic hidden-web table interpretation conceptualization and semantic annotation, Data Knowl. Eng. 68 (7) (2009) 683–703, http://dx.doi.org/10.1016/j.datak.2009.02.010.
[31] Jordi Turmo, Alicia Ageno, Neus Català, Adaptive information extraction, ACM Comput. Surv. 38 (2) (2006), http://dx.doi.org/10.1145/1132956.1132957.

Hassan A. Sleiman received a PhD degreefrom the University of Sevilla in 2012, where he is currently a lecturer with the Department of Computer Languages and Systems. Previously, he worked as a software engineer for companies such as Dynagent and set up a spin-off called Indevia. His is currently researching on web data extraction as a means to populate large datasets in the Web of Data.

Rafael Corchuelo is a Reader of Software Engineering who is with the Department of Computer Languages and Systems of the University of Seville, Spain. He earned a PhD degree from this University and leads its research group on distributed systems (TDG) since 1997. His research interests focus on Enterprise Application Integration and Enterprise Information Integration. Previously, he produced some results on multiparty interaction and fairness issues.