ARIEX: Automated ranking of information extractors

Patricia Jiménez [a,*], Rafael Corchuelo [a], Hassan A. Sleiman [b]

[a] *Universidad de Sevilla, ETSI Informática, Avda. de la Reina Mercedes, s/n. Sevilla E-41012, Spain*
[b] *Commissariat à l'Énergie Atomique et aux Énergies Alternatives LIST, LADIS, Digiteo Labs Saclay, 91191 CEDEX, Gif Sur Yvette, France*

Keywords:

Web documents
Information extraction
Ranking method
Automatisation

A B S T R A C T

Information extractors are used to transform the user-friendly information in a web document into structured information that can be used to feed a knowledge-based system. Researchers are interested in ranking them to find out which one performs the best. Unfortunately, many rankings in the literature are deficient. There are a number of formal methods to rank information extractors, but they also have many problems and have not reached widespread popularity. In this article, we present ARIEX, which is an automated method to rank web information extraction proposals. It does not have any of the problems that we have identified in the literature. Our proposal shall definitely help authors make sure that they have advanced the state of the art not only conceptually, but from an empirical point of view; it shall also help practitioners make informed decisions on which proposal is the most adequate for a particular problem.

1. Introduction

A web information extractor works on user-friendly web documents that have been typically gathered using a crawler [1]. They analyse the documents and extract the information that they provide in a structured format that can be used to feed knowledge-based systems. The information is commonly structured into attributes and records, to which we collectively refer to as slots.

Fig. 1 illustrates what web information extraction is about. It shows an excerpt of a sample web document that provides a listing of records regarding phones. The document is rendered in a friendly format that a person can easily understand. The problem is that the information in this document is not structured, which means that it is not easy to use it in an automated process. Information extractors are devised to help in this task, since they can transform the web document on the left into the structured information on the right. Formally speaking, an information extractor can be modelled as a function that maps DOM nodes or text fragments onto slots that assign a meaning to them, e.g., *Phone, model, seller*, or *price*. The definition is simple because the problem is simple to formulate; what makes it a challenging research field is that devising a machine learner that can learn such a mapping as effectively and efficiently as possible is not trivial at all. This has made it quite an active research field; for instance, as of the time of writing this article, our library reports on roughly 4 190 proposals that have been published in the last decade.

The existing proposals can be classified as rule-based, which require a rule set that specifies how to extract the information of interest, or heuristic-based, which have built-in extraction rules that are based on heuristics. Depending on the kind of document on which they work, they can be further classified as free-text or semi-structured. In the literature, there are many proposals to learn rule sets. Some of them are supervised, that is, they require the user to provide an annotated learning set from which rules that map the information of interest onto appropriate user-defined slots are learnt automatically; contrarily, others are unsupervised, which means that they can learn the rule sets from learning sets that are not annotated, but require the user to interpret them and handcraft mappings that assign the information extracted by each rule onto the appropriate user-defined slot. Many proposals are closed, chiefly in the field of semi-structured information extractors, which means that they are intended to work with documents from a given source; a few ones, chiefly in the field of free-text information extractors, are open, which means that they are intended to work with documents on a given topic, independently from the site from which they are downloaded.

Currently, there are many proposals on web information extraction in the literature. Laender et al. [2], Chang et al. [3], Kushmerick and Thomas [4], Turmo et al. [5], Sarawagi [6], Sleiman and Corchuelo [7], and Ferrara et al. [8] have published some comprehensive surveys on this topic. Unfortunately, heuristic-based proposals have not been surveyed so far; the reader might be interested in consulting references [9–16] for further information. Etzioni et al. [17] provided additional details on open information extractors.

* Corresponding author. Tel.: +34 954552770.
E-mail addresses: patriciajimenez@us.es (P. Jiménez), corchu@us.es (R. Corchuelo), hassan.sleiman@cea.fr (H.A. Sleiman).

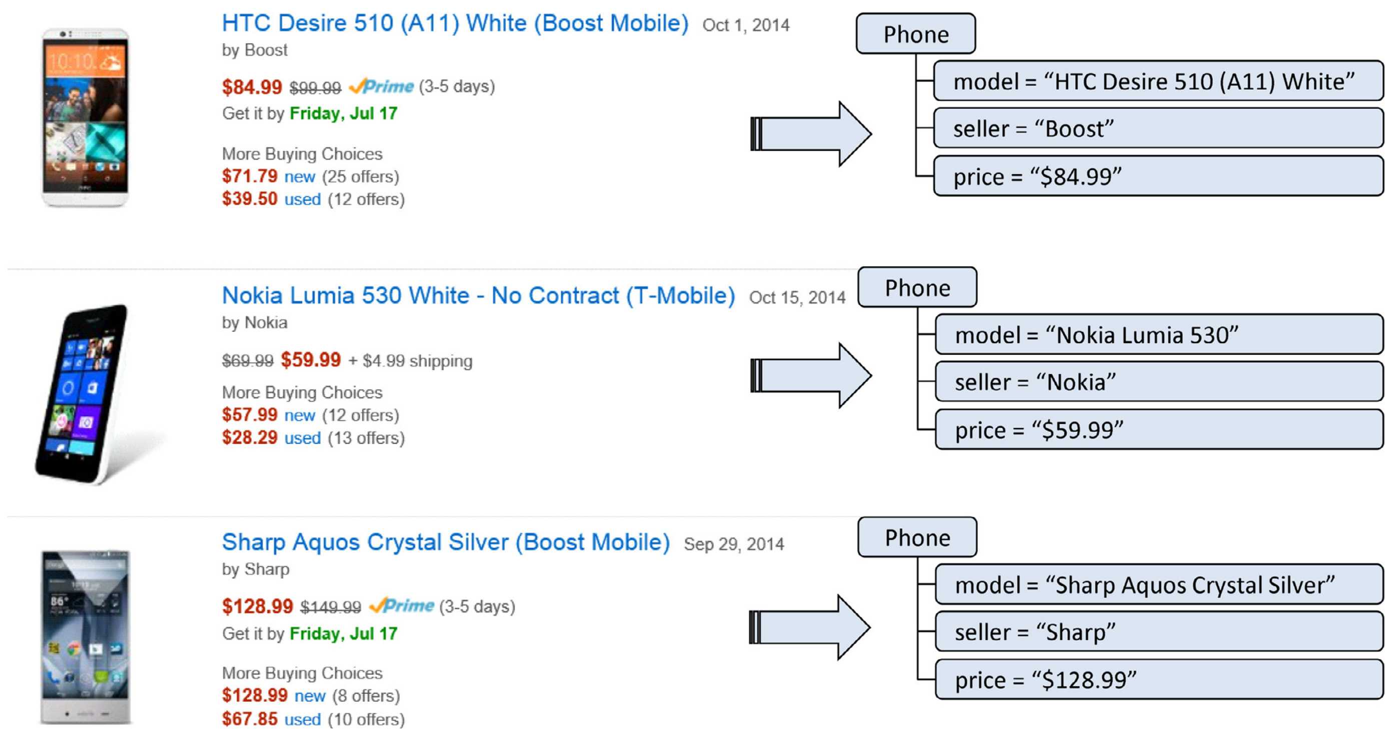


Fig. 1. An illustration of information extraction.

The authors of new proposals must obviously compare them to others so that they can prove that they have introduced conceptual innovations that advance the state of the art. But this is not enough: it is also necessary to rank them regarding their performance; in other words, it is necessary to evaluate them regarding some effectiveness and efficiency measures and then compare the results so as to compute a ranking in which the best-performing proposals are at the top. Practitioners are obviously very interested in such rankings since they lay the foundation to make informed decisions regarding which proposal should be used to solve a given problem.

In our opinion, a good ranking method must have the following key features: it must be automated, so that researchers can bias the conclusions as little as possible, open, so that it can easily accommodate new performance measures, and agnostic, so that it can be applied to as many different kinds of proposals as possible. Furthermore, it must also address the following key questions: how to set up the experimental environment, how to create appropriate evaluation splits, how to compute the experimental data, how to cook them (regarding how to purge them, compute derived measures, or normalise them), how to compute the rankings, and how to report on the results.

We have surveyed many proposals on web information extraction that use an informal method to rank them [2–8]. Unfortunately, our conclusion is that they provide a foundation and some guidelines, but do not have the key features or address the key questions that we have identified above regarding a good ranking method. The informal methods were not intended to be reused, but to help the authors of a proposal support the idea that it outperforms others; as a conclusion, they are not automated, open, or agnostic, but ad-hoc; furthermore, they do not usually disclose many important details regarding the experimental environment; it is not commonly clear how the evaluation splits are created; neither is it clear how the matchings required to compute most effectiveness measures are counted; the experimental data are not cooked; and the resulting rankings are not generally statistically sound. As a conclusion, the stringency level varies from paper to paper, which makes the results available in the literature difficult to reuse when a new proposal needs to be compared to them. Unfortunately, there are very few formal methods in the literature [18–23]. They are generally supported by software tools that aid in computing the experimental data, but they are not actually automated; neither are they open, since they commit to a particular set of performance measures and everything in the method revolves around them; they all originated in a community that was interested in supervised free-text proposals, so they have not paid attention to other kinds of proposals; they report on several alternatives to create evaluation splits, but do not assess the pros and cons or commit to a specific method; they gather experimental data and compute precision- and recall-related measures, but it is not clear how they compute the matches; they do not provide a method to cook the experimental data; and the resulting rankings must be handcrafted, although they pay attention to ensuring that the results are statistically sound.

In this article, we report on ARIEX (Automated Ranking of Information EXtractors), which is a method to evaluate, compare, and then rank web information extraction proposals. It overcomes the problems that we have found in the literature since it reduces the bias that a researcher can introduce in the results because it is automated; it can easily accommodate new performance measures as they are devised and proven to be adequate in our context because it is open; it does not commit to a particular kind of extractor, but has been designed to rank as many proposals as possible because it is agnostic; it provides a clear guideline regarding how the experimental environment must be set up, with a special emphasis on selecting the most appropriate set of performance measures so that the conclusions are global and unbiased; it provides a method to compute as many evaluation splits as possible out of the datasets available; it provides a method to compute the experimental data that takes into account how matchings are computed and does not neglect unsupervised or heuristic-based proposals; it provides a new statistically-sound method to purge the experimental data, it also takes into account derived measures, and provides a normalisation method that makes it open; and it provides a statistically sound method to compute per-measure rankings and then combine them all taking into

```
Step 1: Set up the experimental environment
        - Describe the hardware and the software
        - $proposals$ = select some proposals
        - $datasets$ = select some datasets
        - $measures$ = select some performance measures
        - Set parameters:
            $\alpha$ = select a statistical significance level
            $\gamma$ = select the number of repetitions to compute evaluation splits
            $\omega$ = select the weights of performance measures
            $\eta$ = select a purging measure from $measures$
Step 2: Create evaluation splits
        - $splits$ = $computeEvaluationSplits(datasets)$
Step 3: Compute raw experimental data
        - $rawData$ = $runExperiments(proposals, measures, splits)$
Step 4: Cook the experimental data
        - $purgedData$ = $purgeData(rawData)$
        - $derivedData$ = $computeDerivedData(purgedData)$
        - $normalisedData$ = $normaliseData(purgedData \cup derivedData)$
Step 5: Compute rankings
        - $localRankings$ = $computeLocalRankings(normalisedData)$
        - $globalRanking$ = $computeGlobalRanking(normalisedData)$
Step 6: Produce a report
```

Fig. 2. Steps of our method.

account both a researcher's preferences and the deviations of the performance measures.

Our aim is to reach a widespread agreement so that ARIEX is used in new papers and articles on web information extraction. Authors shall benefit from a method that allows them to prove that they have advanced the state of the art with a proposal that not only introduces a novel approach to the problem, but beats others in the literature from an empirical point of view; practitioners shall be able to make informed decisions regarding which of the proposals in the literature is the most appropriate for a particular problem.

The rest of the article is organised as follows: Section 2 describes the details of ARIEX, Section 3 presents a case study that illustrates how it works in practice, Section 4 reports on the related work, and Section 5 summarises our conclusions. Appendix A provides an insight into the performance measures that we discuss in the article.

2. Description of our method

ARIEX consists of the steps that are summarised in Fig. 2. The first step consists in setting up the experimental environment. The second step consists in computing a number of evaluation splits, that is, pairs of learning and testing sets. The third step consists in running the selected proposals on the previous evaluation splits to gather raw experimental data, that is, the values of the selected performance measures as they are computed on the available evaluation splits. The forth step consists in cooking the raw experimental data as follows: first, they are purged, then derived measures are computed, and, finally, the purged data and the derived data are normalised. The fifth step consists in computing a local ranking per performance measure and then a global ranking. The last step consists in producing a report that summarises the study.

In the following subsections, we provide additional details on each step.

2.1. Step 1: Set up the experimental environment

The experimental environment consists of the hardware and the software used to evaluate a number of proposals plus some data that must be provided by a researcher, namely: the proposals to be ranked, the datasets on which they must be evaluated, the performance measures to compare them, and the values of the parameters. Typically, the researcher is an author who has devised a new web information extraction proposal and wishes to rank it with respect to others in the literature, or a practitioner who needs to extract information from a web site and has to make an informed decision regarding which of the proposals in the literature is the most appropriate.

We provide additional details in the following subsections.

Describe the hardware and the software. Our suggestion is that the researcher should describe the hardware and the software that she or he used to perform her or his experiments. We do not think that it is necessary to describe them thoroughly, because it is very unlikely that other researchers can reproduce exactly the same environment, but it commonly helps have an overall idea of the experimental conditions. Clearly, running an experimentation on a mid-class computer is not the same as running it on a super-computing facility, and this should be made explicit so that the efficiency results can be assessed properly.

Select some proposals. We suggest that the researcher should select the most closely-related proposals (where closeness is measured in terms of conceptual similarity) and some state-of-the-art ones (where state-of-the-art is measured in terms of how recent, sound, and/or well-performing they are). The most closely related proposals must be selected because, otherwise, we cannot prove that the conceptual innovations in a new proposal are worth from a practical point of view; but not only must a new proposal beat the most closely-related ones, but also others that are conceptually different but have proven to achieve good performance.

Select some datasets. Having standard dataset repositories is very important since they allow to compare different proposals on a corpora that can be carefully selected so that the documents are representative enough of both the regularities and irregularities with which web information extractors have to deal. In other words, such repositories should provide controlled, well-documented datasets that put an emphasis on the many difficult cases with which an information extractor has to deal. This makes it impossible that a proposal be evaluated only on datasets on which it works very well and, thus, reduces the chances to bias the results.

The list of public repositories available includes RISE, TBDW, and TIPSTER[1]; furthermore, Freitag [24] compiled the Seminar Announcement collection, Califf and Mooney [25] compiled the Job Posting collection, and Reuters made available the Reuters-2157 collection on company acquisitions[2]; other authors have assembled their own public repositories [10,26–28] and some conferences have also published some repositories, chiefly the MUC conferences.

Note that not every repository provides adequate datasets for every proposal. For instance, there are repositories that focus on free-text documents and others that focus on semi-structured documents; there are repositories that focus on single-record documents and others that focus on multi-record documents; furthermore, there are many repositories in which each dataset was gathered from a single site and others in which some datasets were gathered from different sites and are then appropriate for open information extractors only. A researcher must select the repositories and the datasets that are adequate for the proposals that she or he wishes to rank.

Regarding the number of datasets, there is not a standard in the literature. Our suggestion is that there should be at least 20–30 datasets available and that each one should provide at least 20–30 documents; these figures are generally considered large enough to draw statistically solid conclusions [29]. ARIEX requires the datasets to provide the same number of documents, so that it can create appropriate evaluation splits. In cases in which the datasets available do not provide the same number of documents, our suggestion is to discard some documents randomly or to split them into several smaller datasets. The reason why we do not think that the datasets should provide more than 20–30 documents is that they all have to be annotated for evaluation purposes, which is a time-consuming and error-prone task; neither think we that a proposal that requires a large number of documents to achieve good results is useful from a practical point of view.

Select some performance measures. Performance measures can be classified into effectiveness and efficiency measures. The former focus on assessing the ability of a proposal to discern well between the information to be extracted and the information to be ignored, whereas the latter focus on the computing resources that it requires to do so.

We suggest that the researcher should select a number of effectiveness measures that must fulfil the following requirements: (a) they must be global, that is, they must provide an overview of how a proposal behaves regarding every kind of error it can make; (b) they must not be biased in the presence of unbalanced datasets, which are natural in web information extraction; (c) and they must be extensible from a per-slot level to a per-extractor level in an unbiased manner. Regarding the efficiency measures, our only requirement is that they must be stable, that is, they must not vary significantly when a proposal is applied multiple times to the same evaluation split. Furthermore, the set of selected performance measures must fulfil the following requirements: (a) they must be orthogonal, that is, they must focus on assessing different complementary aspects of performance in an attempt to provide as a wide view of a proposal as possible; (b) and there must not be many measures, since, otherwise, their individual contribution to the global ranking blurs easily.

We propose to use three types of effectiveness measures in ARIEX, namely: (a) error-related measures, which must assess the errors that a proposal makes, that is, the slot instances that are not correctly extracted or discarded; (b) generalisation ability, which must assess the ability of a proposal to work well with as few documents as possible; (c) and failure-related measures, which must assess the mistakes that are not due to a proposal itself, but its available implementation. Regarding efficiency, our proposal is to use both time- and memory-related measures.

We have surveyed the literature, and we have found that there are a variety of performance measures that we have carefully analysed. Our conclusion is that the following ones fulfil our requirements and are then very appropriate in the context of ARIEX: the area under the ROC curve (*AUC-ROC*) as the error-related measure, the performance knee (*PK*) as the generalisation measure, the failure ratio (*FR*) as the failure-related measure, plus learning time (*LT*) and extraction time (*ET*) as time-related measures, and learning memory (*LM*) and extraction memory (*EM*) as memory-related measures. Note that our proposal clearly deviates from the literature, where precision- and recall-related measures are the standard, but we have proved that ours are more adequate in our context. In Appendix A, we justify our decision and provide enough details regarding each of the previous measures.

Set parameters. Regarding the parameters of ARIEX, the researcher is requested to set α, which is the significance level at which statistical tests are performed, γ, which is the number of repetitions performed to create evaluation splits, ω, which is a vector with the weights of the performance measures according to the researcher's preferences, and η, which is a performance measure that we wish to use to purge our experimental data.

In the literature, α is typically set to 0.05, which provides 95% statistical confidence. We suggest setting γ to a value in range 10–20 which generally leads to a sufficiently large number of sets of evaluation splits. The weights of the performance measures are completely up to the researcher. Regarding η, our suggestion is to use the area under the ROC curve, since our survey of the literature reveals that this is the most appropriate measure in our context, cf. Appendix A.

2.2. Step 2: Create evaluation splits

Before using the selected datasets, we need to split them into evaluation splits, that is, pairs of learning and testing sets. Note that learning sets do not actually make sense for proposals that are based on heuristics, so we have to create evaluation splits that are specifically tailored to these proposals.

Our proposal is to use the sub-sampling method that is presented in Fig. 3. It takes a collection of datasets as input and returns a set of evaluation splits, i.e., a set of tuples of the form (ls, ts) in which ls denotes a learning set and ts denotes a testing set. For every dataset, the method first sets h to half its size and then repeats the following steps γ times: it first creates two reservoirs of documents called r_1, which stores a random half of the documents in the corresponding dataset, and r_2, which stores the other half. Then, it iterates h times and updates a learning set that is initialised to an empty set and a testing set that is initialised to the documents in the second reservoir. In each iteration, a random document from the first reservoir is selected and used to grow the previous learning set, as long as it has not been selected previously; furthermore, a document is removed from the previous testing set. The result that is returned by the method is updated with two tuples in each iteration: the first one is of the form (ls_s, r_2), and it is intended to be used as an evaluation set for rule-based proposals; note that the learning set grows in each iteration, but the testing set remains the same so that it is easy to evaluate how growing the learning set has an impact on the effectiveness of a proposal. The second one is of the form $(\emptyset, ts_{h+1-s})$; note that these evaluation splits are appropriate for heuristic-based proposals because they do

[1] These repositories are available at http://www.isi.edu/info-agents/RISE, http://daisen.cc.kyushu-u.ac.jp/TBDW/, and http://www.itl.nist.gov/iaui/894.02/related_projects/tipster, respectively.

[2] The Reuters collection is available at http://www.daviddlewis.com/resources/testcollections/reuters21578.

```
method $computeEvaluationSplits$($datasets$)
    $result$ = $\emptyset$
    for each dataset $ds$ in $datasets$ do
        $h$ = $\lfloor \frac{|ds|}{2} \rfloor$
        repeat $\gamma$ times
            $r_1$ = select $h$ documents from $ds$
            $r_2$ = select $h$ documents from $ds \setminus r_1$
            $ls_0$ = $\emptyset$
            $ts_{h+1}$ = $r_2$
            $d_2$ = $null$
            for $s$ = 1 until $h$ do
                $d_1$ = select one document from $r_1 \setminus ls_{s-1}$
                $ls_s$ = $ls_{s-1} \cup \{d_1\}$
                $ts_{h+1-s}$ = $ts_{h+2-s} \setminus \{d_2\}$
                $d_2$ = select one document from $ts_{h+1-s}$
                $result$ = $result \cup \{(ls_s, r_2), (\emptyset, ts_{h+1-s})\}$
            end
        end
    end
return $result$
```

Fig. 3. Method to compute evaluation splits.

```
method $runExperiments$($proposals$, $measures$, $splits$)
    $result$ = $\emptyset$
    for each proposal $p$ in $proposals$ do
        for each evaluation split $s$ in $splits$ do
            if $s$ is appropriate for $p$ then
                $m$ = select non-derived measures from $measures$
                $d$ = execute $p$ on $s$ and compute measures $m$
                let $result(p, s)$ = $d$
            end
        end
    end
return $result$
```

Fig. 4. Method to compute raw experimental data.

not have a learning phase and consequently do not require a learning set.

2.3. Step 3: Compute raw experimental data

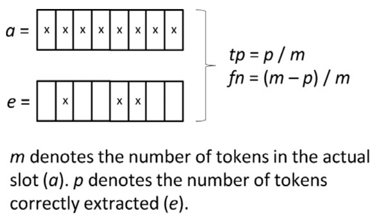
Computing the raw experimental data consists in running every proposal on the appropriate evaluation splits, depending on whether they are rule-based or heuristic-based, and then collecting the non-derived performance measures that the researcher has selected.

Fig. 4 shows the method that we propose to use. It works on a collection of proposals, a collection of measures, and a collection of evaluation splits; it computes a map called *result* in which pairs (p, s) of proposals and evaluation splits are associated with maps d that associate every performance measure with the value that was computed regarding proposal p on evaluation split s. In other words, the experimental data can be interpreted as a map from pairs of proposals and evaluation splits onto the corresponding values of the measures. Note that not every evaluation split is appropriate to run every proposal: rule-based proposals can work on evaluation splits that provide both a learning and a testing set, whereas heuristic-based proposals must be run on evaluation splits that provide a testing set only.

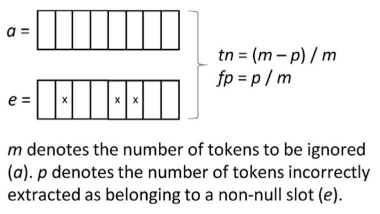
Our experience proves that there are cases in which it is not possible to compute the performance measures regarding a given proposal on a given evaluation split because there is a bug in the implementation or it simply takes too long or consumes too much memory and cannot be executed. Note that such failures must be recorded as missing values; such values shall later be used to compute failure-related derived measures.

The method to gather the experimental data is straightforward, but computing confusion matrices or dealing with unsupervised and heuristic-based proposals is, however, a little more involved. In the following subsections, we provide additional details.

Computing confusion matrices. The documents in the input datasets must be annotated; that is, a person must have labelled every piece of information to be extracted with a user-defined slot; the information that is not to be extracted is assumed to be implicitly labelled as belonging to a pre-defined null slot. Given a testing set, it is not difficult to compute the exact matchings, that is, the pieces of information


a =
e =
tp = p / m
fn = (m – p) / m
m denotes the number of tokens in the actual slot (a). p denotes the number of tokens correctly extracted (e).

(a) Positive matching.


a =
e =
tn = (m – p) / m
fp = p / m
m denotes the number of tokens to be ignored (a). p denotes the number of tokens incorrectly extracted as belonging to a non-null slot (e).

(b) Negative matching.

Fig. 5. Cases when computing matchings.

that are correctly or incorrectly extracted as belonging to a given slot. The problem is how to compute inexact matchings. Such matchings are common, for instance, with proposals that work on DOM trees since DOM nodes are very likely not to be perfectly aligned with the information to be extracted.

To introduce our proposal, we assume that $S = \{s_1, s_2, \ldots, s_n\}$ denotes the set of slots on which we are working. Given a testing set, we need to compute n confusion matrices of the form $C_i = (tp_i, tn_i, fp_i, fn_i)$, where *i* ranges in the set of slots *S*, tp_i denotes the number of true positives for slot *i*, tn_i denotes the number of true negatives for slot *i*, fp_i denotes the number of false positives for slot *i*, and fn_i denotes the number of false negatives for slot *i*. To compute these matrices, it is necessary to analyse each slot in isolation and represent the documents as sequences of slot instances; note that given a slot *s*, its instances lead to positive matchings and the instances of the other slots, including the null slot, lead to negative matchings. Fig. 5 illustrates both cases and how we propose to deal with them, namely:

Case 1: positive matching. In this case, the document provides an actual instance of slot *s* that has *m* tokens, *p* of which are extracted as belonging to that slot, whereas the remaining $m - p$ tokens are extracted as belonging to another slot or not extracted at all. In such a case, our proposal is to count the ratio of tokens that are correctly extracted as the number of true positives in this matching, that is, $tp = p/m$; similarly, the ratio of tokens that are not correctly extracted as belonging to slot *s* must be computed as false negatives, that is, $fn = (m - p)/m$. Note that if $m = p$, then it means that every token in the actual instance of the slot has been extracted correctly, in which case there is an exact matching that contributes with one true positive and zero false negatives, as expected. Note that if $p > m$, then the remaining $p - m$ tokens shall be counted as false positives in the following case.

Case 2: negative matching. In this case, the document does not provide an instance of slot *s*, that is, the slot at a given position is a slot different from *s*, possibly the null slot. Again, we can assume that the slot has *m* actual tokens, that *p* such tokens are extracted as belonging to slot *s*, and that $m - p$ tokens are not extracted as belonging to slot *s*. In this case, our proposal is to count $(m - p)/m$ true negatives, since this is the ratio of tokens that have not been extracted as belonging to slot *s*, and to count p/m false positives, since this is the ratio of tokens that have been incorrectly extracted as belonging to slot *s*. Note that if $m = p$, then every token has been incorrectly extracted as belonging to slot *s*, in which case it contributes to the count with one false positive and zero true negatives, as expected. If $p > m$, then the corresponding $p - m$ tokens are computed as true positives in the previous case.

Dealing with unsupervised and heuristic-based proposals. In the case of supervised proposals, it is not difficult to compute matchings because they learn extraction rules that are specifically tailored to extracting information as belonging to one of the slots that the user has

```
method $mapSlots(actual, extracted)$
    $result = \emptyset$
    for each slot $e$ in $extracted$ do
        $m = -\infty$
        for each slot $a$ in $actual$ do
            $n$ = compute $\eta$ on $a$ and $e$
            if $n > m$ then
                $m = n$
                $s$ = slot of $a$
            end
        end
        let $result(e) = s$
    end
return $result$
```

Fig. 6. Method to map extracted slots onto actual slots.

defined in the input datasets. In the case of unsupervised or heuristic-based proposals, the problem is complicated by the fact that they ignore the annotations in the input datasets since they were devised to learn to extract as much information as possible from them, which is assigned to computer-generated slots. It is the user who must analyse the resulting computer-generated slots and map them onto user-defined slots, that is, she or he must assign a meaning to them.

Since we are interested in an automated method, we have to perform the previous mapping automatically. Recall that we require the researcher who uses ARIEX to decide on a so-called purging measure to which we refer to as η. Such measure is an effectiveness measure that is expected to provide a good overview of how good a proposal is and then helps purge the experimental data. We can use it to deal with unsupervised and heuristic-based proposals as shown in the method in Fig. 6. This method gets a collection of actual slots, that is, the pieces of information and their corresponding labels as the user has provided them in a testing set, and a collection of extracted slots, that is, the pieces of information that an information extractor has returned when it was run on that testing set. The method then iterates over every pair of actual and extracted slot and computes the purging measure η on them. It returns a map called *result* in which each extracted slot is associated with the label of the actual slot with which the purging measure achieves its maximum value. In our experience, this mapping is as effective as a handcrafted-mapping, but it is completely automated.

2.4. Step 4: Cook the experimental data

Cooking the experimental data consists in removing some of them so that the remaining ones can be used to compute the resulting rankings. First, the data must be purged, then derived measures must be computed, and, finally, the experimental data must be normalised. We provide additional details in the following subsections.

```
method $purgeData$($data$)
    $proposals$ = get proposals in $data$
    $result$ = $data$
    for each proposal $p$ in $proposals$ do
        $pk$ = $computePerformanceKnee(data, p, \eta)$
        if $p$ is rule-based then
            $dataToRemove$ = $\{(p, s, d) \mid \exists l, t \bullet s = (l, t) \wedge (p, s, d) \in data \wedge |l| \neq pk\}$
        else
            $dataToRemove$ = $\{(p, s, d) \mid \exists t \bullet s = (\emptyset, t) \wedge (p, s, d) \in data \wedge |t| \neq pk\}$
        end
        $result$ = $result \setminus dataToRemove$

        $t$ = compute per-extractor $\eta$ for $p$ using $result$
        if $t \leq$ minimum acceptable value of $\eta$ then
            $dataToRemove$ = $\{(p, s, d) \mid (p, s, d) \in result\}$
            $result$ = $result \setminus dataToRemove$
        end
    end
return $result$
```

Fig. 7. Method to purge experimental data.

```
method $computeDerivedData$($data$)
    $proposals$ = get proposals in $data$
    $measures$ = get measures in $data$
    $result$ = $\emptyset$
    for each proposal $p$ in $proposals$ do
        $m$ = select derived measures from $measures$
        $d$ = compute measures $m$ from $p$ and $data$
        let $result(p, null)$ = $d$
    end
return $result$
```

Fig. 8. Method to compute derived data.

Purging data. In the previous step, we have computed many experimental data. To perform as a fair comparison as possible, we have to compare the experimental data that corresponds to the best-performing evaluation splits. Thus, we have to remove the other splits as well as those that correspond to proposals that perform very bad. Recall that we require the researcher to set a purging measure η, which refers to the measure that ARIEX uses to decide which data must be removed.

Fig. 7 shows the method that we have designed to purge the experimental data. It takes some experimental data as input and returns a subset of them. It iterates over the set of proposals in the experimental data and proceeds in two steps, namely: first, it removes the data that does not correspond to the best-performing evaluation splits, and then removes all of the data that correspond to the proposals that perform very bad according to the selected purging measure.

The complex part of the first step is to compute the set of evaluation splits on which a proposal performs the best. We use the purging measure to compute a performance knee, that is an inflection point above which a proposal does not improve as the size of the evaluation splits increases. Computing a performance knee is not straightforward; we provide additional details on the method that we have devised in Appendix A. Once the performance knee is computed, the data that do not correspond to evaluation splits whose size is equal to the performance knee can be purged. Note that computing the size of an evaluation split depends on the proposal: if it is rule-based, then the size of the evaluation split is computed as the size of the corresponding learning set; if it is heuristic-based, then it is computed as the size of the corresponding testing set. Note, too, that ARIEX only keeps γ evaluation splits for a proposal that is not purged, all of which are the size of the best-performing evaluation splits found for that proposal.

The second step removes the data that correspond to proposals that are very bad. These are the proposals that do not achieve a value for the purging measure above a minimum acceptable value when it is computed on a per-extractor level. Recall that our suggestion is to use the area under the ROC curve as the purging measure; it is well-known that proposals whose area under the ROC curve is below 0.50 perform worse than a random guess [30], which we consider bad enough to discard them.

Computing derived data. The data that we have got from the experimentation are computed on a per-evaluation-split basis. That is, a proposal is run on an evaluation split and the corresponding performance measures are computed. There are some measures that cannot be computed that way, but must be derived from the experimental data once it is purged.

Fig. 8 presents the method that we propose to compute the derived measures. It works on some purged experimental data and

```
method $normaliseData(data)$
    $proposals$ = get proposals in $data$
    $measures$ = get measures in $data$
    $result = \emptyset$
    for each proposal $p$ in $proposals$ do
        for each measure $m$ in $measures$ do
            $W = \{w \mid \exists s, d \cdot (p, s, d) \in data \wedge w = d(m)\}$
            $(a, b) = (\min W, \max W)$
            if $m$ must be maximised then
                $W' = \{w' \mid \exists s, d \cdot (p, s, d) \in data \wedge w' = (d(m) - a) \text{ div } (b - a)\}$
            else
                $W' = \{w' \mid \exists s, d \cdot (p, s, d) \in data \wedge w' = 1.00 - (d(m) - a) \text{ div } (b - a)\}$
            end
            let $result(p, m) = W'$
        end
    end
return $result$
```

Fig. 9. Method to normalise experimental data.

returns a map in which every pair of the form (*p, null*) is associated with another map *d; p* is a proposal and *null* denotes that the evaluation was not performed on a particular evaluation split, but derived from the existing ones; *d* is a map in which every derived measure is associated with its corresponding value.

Previously, we mentioned that our proposal is to compute the performance knee (*PK*) and the failure ratio (*FR*) as derived measures. They both can be easily computed on the purged data, namely: computing the performance knee is straightforward since we actually computed it in the previous step and discarded the evaluation splits with different sizes, so we only have to see what the size of the remaining evaluation splits is; computing the failure ratio amounts to counting the number of missing values in the input data and calculating the ratio to the total number of values.

Normalising the experimental data. Unfortunately, the performance measures do not range within the same intervals and their goodness are different, namely: the area under the ROC curve (*AUC-ROC*) ranges between 0.00 and 1.00 and the greater the better (recall that proposals whose *AUC-ROC* is equal or less than 0.50 are discarded when the experimental data are purged); the performance knee (*PK*) ranges between 1 and an arbitrarily large number and the smaller the better; the failure ratio (*FR*) ranges between 0.00 and 1.00 and the smaller the better; the learning time (*LT*) and the extraction time (*ET*) range between 0.00 CPU seconds and an arbitrarily large number and the smaller the better; finally, the learning memory (*LM*) and the extraction memory (*EM*) range between 0.00 GiB and an arbitrarily large number and the smaller the better.

Fig. 9 presents the method that we propose to normalise the experimental data within range 0.00..1.00, so that the lower bound corresponds to bad values and the upper bound corresponds to good values. That transformation can be performed easily since it amounts to translating the range of each performance measure and then computing its complement if that measure needs to be minimised. Note that this method works on the purged experimental data, which is a map in which each pair of proposal *p* and evaluation split *s* is associated with a map *d* that associates every performance measure with the value that was computed regarding proposal *p* on evaluation split *s*. (Recall that the evaluation split may be null in the case of derived measures.) The method to normalise the data transforms them into a new map in which each pair of proposal *p* and measure *m* is associated with the set *W′* of normalised values of that measure regarding that proposal. (In the pseudo-code, *x* div *y* equals x/y if $y \neq 0.00$; otherwise, it equals 1.00.)

2.5. Step 5: Compute rankings

This step of ARIEX consists in computing the final results, which consists of a number of local rankings and a global ranking. In the following subsections, we provide additional details on the methods that we propose.

Computing local rankings. Fig. 10 presents our method to compute the local rankings. It works on the normalised experimental data and returns a map in which each measure is associated with an ordered collection of proposals. It compares every pair of proposals regarding every measure using Wilcoxon's Rank-Sum test [29]. Note that only pairs of different proposals are compared and that the order in which the comparison is performed is irrelevant; in the pseudo-code, we assume that $\prec$ denotes an arbitrary ordering of the proposals, e.g., the lexicographic ordering. The test returns a p-value that, according to Bonferroni's correction, has to be compared to the statistical significance level α set by the researcher divided by the number of comparisons to be performed, which is $(k^2 - k)/2$, where k denotes the number of proposals to be compared. In the case of derived measures, the experimental data provides only a value; in such cases Wilcoxon's Rank-Sum trivially returns 0.00 if the measures to be compared have different values, and 1.00 if the values are the same. Both sets H_0 and H_1 store triplets of the form (m, p_1, p_2); the triplets in H_0 indicate the pairs of proposals for which the ranking data does not provide enough evidence to conclude that they behave differently regarding the performance measure; the triplets in H_1 indicate the pairs of proposals for which the ranking data provides enough evidence to consider that they behave differently regarding the performance measure.

Unfortunately, the previous procedure does not necessarily result in a total pre-order. Generally speaking, such situations occur when there is a minimal sequence of proposals $\langle p_1, p_2, \ldots, p_n\rangle$ such that Wilcoxon's Rank-Sum test does not find enough evidence to conclude that p_i behaves differently from p_{i+1} for every $i = 1 \ldots n - 1$, but it finds enough evidence to conclude that p_1 behaves differently from p_n. Our proposal to transform such chains into total pre-orders is to break them assuming that p_j does not behave like p_{j+1}, where p_j and p_{j+1} $(1 \leq j < n)$ denote the pair of proposals for which Wilcoxon's

```
method $computeLocalRankings(data)$
    $proposals$ = get proposals in $data$
    $measures$ = get measures in $data$
    $result$ = $\emptyset$
    for each measure $m$ in $measures$ do
        $H_0$ = $\emptyset$
        $H_1$ = $\emptyset$
        for each pair of proposals $p_1, p_2$ in $proposals$ such that $p_1 \prec p_2$ do
            $p\text{-}value$ = $Wilcoxon\text{-}Rank\text{-}Sum\text{-}Test(data(p_1, m), data(p_2, m))$
            $k$ = $|proposals|$
            $n$ = $(k^2 - k)/2$
            if $p\text{-}value \geq \alpha/n$ then
                $H_0$ = $H_0 \cup \{(m, p_1, p_2)\}$
            else
                $H_1$ = $H_1 \cup \{(m, p_1, p_2)\}$
            end
        end
        $H$ = transform $H_0$ and $H_1$ into a total pre-order
        $result$ = $result \cup H$
    end
return $result$
```

Fig. 10. Method to compute local rankings.

Rank-Sum test returns the smallest p-value above the significance level α; in other words, we suggest selecting the couple of proposals for which the experimental data provides more evidence that they behave differently. There is obviously a chance to make a mistake, but it is the only way to transform the results of Wilcoxon's Rank-Sum test into a total pre-order. Note that the decision might be taken arbitrarily at any other point in the chain and the results would be the same: there is only a chance to make a mistake at the point where the chain is arbitrarily broken.

Computing a global ranking. To compute a global ranking, we need to combine the experimental data to produce a single scalar value per proposal. Recall that ARIEX requires the researcher to provide a vector ω that assigns each performance measure a weight according to her or his preferences. That weight is used as follows to compute a rank for each proposal p:

$$K^p = \sum_{m \in measures} \omega_m K^p_m$$

where ω_m denotes the weight that a researcher has assigned to measure m and K^p_m is a rank regarding that measure, where m ranges in the set of measures selected by the researcher and p ranges in the set of proposals. (Obviously, the sum of the weights must equal 1.00.) We propose to compute K^p_m as follows:

$$K^p_m = \frac{\text{mdr}^p_m}{\max_{p \in proposals} \text{mdr}^p_m}$$

where mdr^p_m represents a mean-to-deviation ratio that is computed as follows:

$$\text{mdr}^p_m = \begin{cases} \dfrac{(\mu^p_m)^2}{\sigma^p_m} & \text{if } \sigma^p_m \neq 0.00 \\ \mu^p_m & \text{otherwise} \end{cases}$$

where μ^p_m denotes the mean value of measure m regarding proposal p, and σ^p_m its standard deviation. Note that this ratio maps every measure onto a value that weights its mean value with the inverse coefficient of deviation (μ^p_m/σ^p_m) as long as the standard deviation is not zero; intuitively, the smallest the coefficient of deviation with respect to the mean value, the better that measure because it is more stable. The cases in which the standard deviation is zero typically correspond to derived measures, for which only one value is collected; in such cases, the mean-to-deviation ratio is trivially defined as the mean value.

Fig. 11 presents the method that we propose to compute the global ranking. It iterates twice over the set of pairs of proposals and measures. In the first iteration, it computes a map called *mdr* that maps every pair of proposal p and measure m onto its corresponding mean-to-deviation ratio. In the second iteration, it computes the resulting ranks and stores them in map *result*.

2.6. Step 6: Produce a report

The last step of ARIEX consists in producing a report in which the results of the previous steps are summarised and commented by the researcher. Below, we present a suggestion regarding how to organise it.

Abstract. As usual, the abstract must provide a short overview of the report and highlight the original findings.

Experimental environment. The goal of this section is to provide an overview of the experimental environment. Our suggestion is to organise it as follows:

Hardware and software. Regarding the hardware, we suggest that the researcher should report on the processors, the motherboard, the memory, the persistent storage, and whether it was virtual or bare metal. Regarding the software, we suggest that the researcher should report on the operating system, the virtual machines, and the libraries used, if applicable. She or he should also report on the changes that were conducted to customise the default configurations, if any.

```
method $computeGlobalRanking$($data$)
    $proposals$ = get proposals in $data$
    $measures$ = get measures in $data$
    for each proposal $p$ in $proposals$ do
        for each measure $m$ in $measures$ do
            $W = \bigcup\{V \mid (p, m, V) \in data\}$
            $(\mu, \sigma) = (\text{mean}\, W, \text{stdev}\, W)$
            if $\sigma \neq 0.00$ then
                let $mdr(p, m) = \mu^2/\sigma$
            else
                let $mdr(p, m) = \mu$
            end
        end
    end
    for each proposal $p$ in $proposals$ do
        let $result(p) = 0$
        for each measure $m$ in $measures$ do
            $a = \max_{q \in proposals} mdr(q, m)$
            $K = mdr(p, m)/a$
            let $result(p) = result(p) + \omega_m K$
        end
    end
return $result$
```

Fig. 11. Method to compute a global ranking.

Proposals. For each proposal, the report should list its name, key references to the literature, a classification (that is, whether it is rule-based or heuristic-based, supervised or unsupervised in the case of rule-based proposals, free-text or semi-structured, and open or closed), the implementation used in the experiments, and some comments that may help the reader understand key facts regarding it.

Performance measures. We suggest that the researcher should organise the measures in categories since our experience proves that this usually helps understand their importance better. For each measure, the report should list its name, whether it is derived or not, its definition, the interval in which it ranges, its goodness, and its weight (ω). The report should also make it clear what the selected purging measure is (η) and provide a justification regarding their weights. (Please, recall that we have made some suggestions regarding the most appropriate measures, but our method is open to accommodate new measures as they are proven to be adequate in our context.)

Datasets. For each dataset, the report should list its name and version, the web site from which it was downloaded, the number of documents that it provides, and how large they are in average. We suggest that the datasets should be grouped in categories according to their topic and that the researcher should list the slots that were extracted in each category.

Statistics. The researcher must report on the significance level that she or he selected (α) and the number of repetitions set in the method to compute the evaluation splits (γ).

Experimental data. This section must report on the experimental data that was computed from the experiments. Our suggestion is to organise it as follows:

Non-derived measures. Note that the amount of data regarding these measures is typically huge. Including them in the report makes little sense and would be of little interest, since they are too many data for a person to understand them. It is, however, interesting to try to learn from these data how a proposal behaves in practice regarding the non-derived performance measures. Our suggestion is to provide charts and tables regarding the mean values of the measures, which may provide a rough intuition regarding how a proposal behaves, and then use the least squares regression method to compute the approximation that maximises the determination coefficient R^2. The researcher should comment on how the conceptual innovations in each proposal are reflected on the results. She or he should, however, avoid comparing the results to each other, since they just provide a rough approximation to how each proposal behaves. Note that the points in the charts and tables are averaged from many data, and such values do not take the distribution of values into account; as a conclusion, comparing them might lead to wrong conclusions that cannot be supported from a statistical point of view.

Derived measures. We suggest that the report should present them in a table or a chart and comment on their values from a conceptual point of view. Our proposal is to compute the performance knee and the failure ratio as derived measures. The researcher should reflect on the experimental results and try to discern the conceptual reason why a proposal has a lower performance knee than the others. Furthermore, she or he should also reflect on the reasons why the failure ratio of a proposal is not zero; it is very important to discern if the failures were due to an intrinsic feature of a proposal or a bug in its implementation.

Purged proposals. If a proposal was removed because it did not achieve a value above the minimum allowable threshold for the purging measure, then the researcher should comment on the conceptual reasons why that happened.

Rankings. This section must report on the rankings computed by ARIEX. Our suggestion is to organise it as follows:

Local rankings. We suggest that the report should present them in a table in which the empirical rankings should be listed, and then the p-values computed by Wilcoxon's Rank-Sum test on every pair of proposals regarding every performance measure; the table should also report on the statistical ranking computed using the method that we have proposed.

Global ranking. We suggest that the report should present the global ranking in a table and a chart. The researcher should comment on the results and provide a conceptual explanation.

Conclusions. The report should include a section in which the researcher summarises her or his conclusions from conducting the experimental study, evaluating, and comparing the proposals that she or he selected.

Bibliography. The report should include references to the literature where further information on the proposals, the datasets, or other key issues can be found.

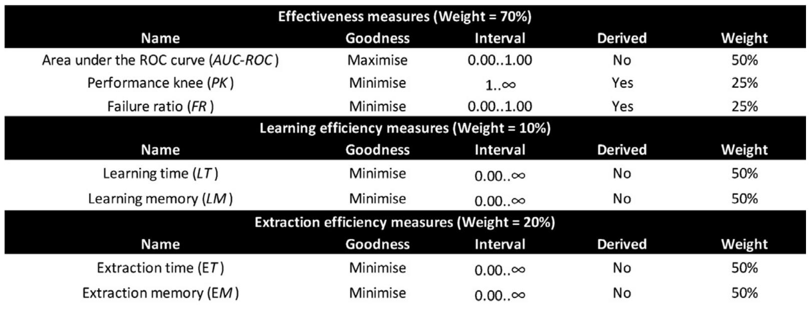
3. A case study

In this section, we present one of the many case studies that we have conducted to polish our proposal. Below, we present the corresponding report.

Table 1
Proposals analysed in our case study.

Name	References	Classification	Implementation
P0	-	Rule-based (supervised), semi-structured, closed	Java 7
P1	-	Heuristic-based, semi-structured, closed	Java 7
P2	-	Rule-based (unsupervised), semi-structured, closed	Java 7
P3	-	Rule-based (supervised), semi-structured, closed	Java 7
P4	-	Rule-based (supervised), semi-structured, closed	Java 7

Table 2
Performance measures used in our case study.



Effectiveness measures (Weight = 70%)				
Name	**Goodness**	**Interval**	**Derived**	**Weight**
Area under the ROC curve (*AUC-ROC*)	Maximise	0.00..1.00	No	50%
Performance knee (*PK*)	Minimise	1..∞	Yes	25%
Failure ratio (*FR*)	Minimise	0.00..1.00	Yes	25%
Learning efficiency measures (Weight = 10%)				
Name	**Goodness**	**Interval**	**Derived**	**Weight**
Learning time (*LT*)	Minimise	0.00..∞	No	50%
Learning memory (*LM*)	Minimise	0.00..∞	No	50%
Extraction efficiency measures (Weight = 20%)				
Name	**Goodness**	**Interval**	**Derived**	**Weight**
Extraction time (*ET*)	Minimise	0.00..∞	No	50%
Extraction memory (*EM*)	Minimise	0.00..∞	No	50%

AUC-ROC is the purging measure.

Abstract. In this case study, we have evaluated, compared, and ranked five proposals to which we refer to as P0, P1, P2, P3, and P4. We keep them anonymous because it is not our intention to contribute with a ranking of a few existing proposals, but to illustrate how ARIEX works in practice so that it can serve as a guideline for other researchers. Our study clearly reveals that analysing the experimental data intuitively can very easily lead to wrong conclusions that are not supported from a statistical point of view.

Experimental environment. Next, we report on the experimental environment that we used in our study.

Hardware and software. The experiments were run on a virtual computer that was equipped with four Intel Xeon E7-4807 cores that ran at 1.87 GHz, had 4 GiB of RAM, and 16 GiB of persistent storage. The motherboard was a Supermicro X8QB6. All of the proposals were run using the Oracle Java Development Kit 1.7.9_02. The operating system was Microsoft Windows 7 Pro 64-bit. The regular expression engine required by some proposals was provided by GNU RegEx 1.1.4. Some proposals required the input documents to be cleaned before running on them; we used JTidy R938 and JSoup 1.8.2. No changes to the default configuration of the hardware or the software were made.

Proposals. Table 1 summarises the proposals that we have studied. They all work on semi-structured documents and are closed, but differ significantly regarding the techniques on which they rely, namely: P0 refers to a very simple baseline that uses rules of the form $L - R$, where L and R are 5-token disjunctive patterns that match the left and the right of the information that has been annotated in the learning sets; P1 is a heuristic-based proposal that compares a number of documents to find the differences amongst them, which are returned as the extracted information; P2 is a rule-based, unsupervised proposal that also finds differences amongst a number of documents and generalises them into a regular expression with variables that capture the differences; P3 is a hybrid proposal that first learns the structure of the information using an automata and then learns transition conditions using a standard machine-learning technique; and P4 is a proposal that learns DOM-based extraction rules using a propositional inductive logic technique.

Performance measures. Table 2 summarises the performance measures that we have used. We have adhered to the suggestions that we have made in the previous section; additional details are provided in Appendix A. Note that we have grouped the measures into effectiveness measures, whose weight is 70%, learning efficiency measures, whose weight is 10%, and extraction efficiency measures, whose weight is 20%. These figures reflect our opinion that it is very important that a proposal be very effective and efficient at extracting information, but the time to learn a rule set, if applicable, is not a concern nowadays; however, it must not be completely neglected because, other things equal, the learning efficiency can make a difference between two proposals.

Datasets. Table 3 summarises the datasets that we have used. We selected 40 datasets from Sleiman and Corchuelo's repository (version 1.0) [31]. They were grouped in the following categories: books, movies, cars, events, doctors, jobs, realty, and sport players. From each dataset, we randomly selected a subset of 30 documents and we extracted records of the form $Record\{attribute_1, attribute_2, \ldots, attribute_n\}$.

Statistics. We set the significance level to $\alpha = 0.05$ and the number of repetitions in the method to compute the evaluation splits to $\gamma = 10$.

Experimental data. Next, we report on the measures that we have collected.

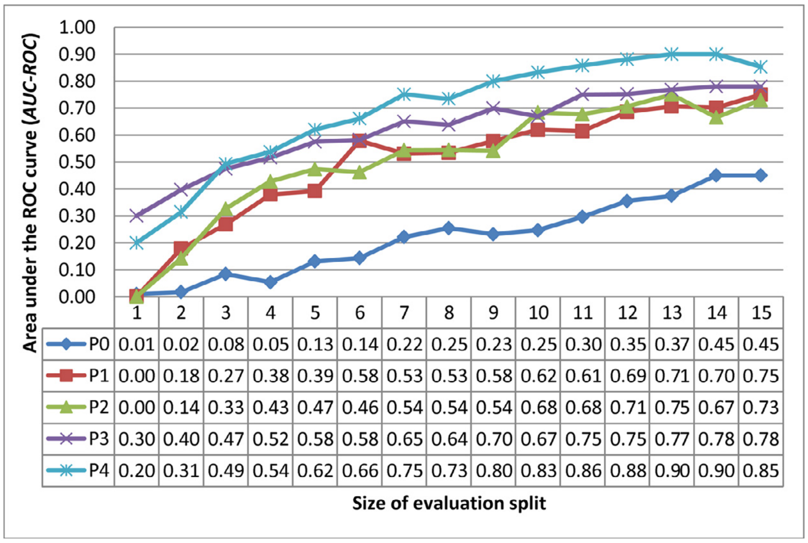
Non-derived measures. Tables 4–6 report on the mean values that we gathered regarding the non-derived measures (before purging them) and Table 7 presents the best approximations that we have found.

Regarding the area under the ROC curve, note that all of the proposals seem to behave logarithmically with respect to the size of the evaluation splits (S), except for P0, which seems to behave linearly. Regarding the proposals that learn rules, the logarithmic behaviour was expected because the larger an evaluation split, the more learning documents are available,

Table 3
Datasets used in our case study.

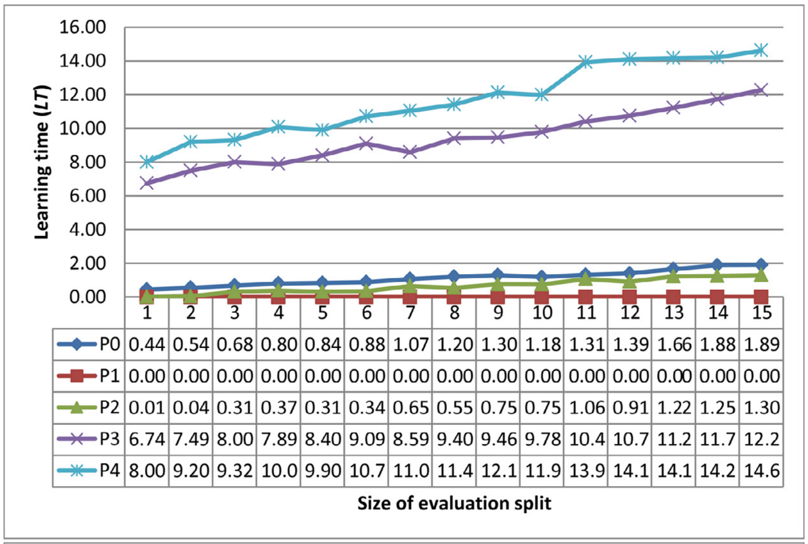
Category	Site	Slots	Size (KiB)
Books	Abe Books	Book{title, author, price, isbn, year}	37.65
	Awesome Books		20.15
	Better World Books		125.23
	Many Books		26.84
	Waterstones		79.68
Movies	IMDB	Movie{title, director, actor, year, runtime}	97.35
	Disney Movies		47.26
	Albania Movies		5.7
	All Movies		33.79
	Soul Films		28.48
Cars	Auto Trader	Car{model, colour, engine, price, transmission}	183.51
	Car Max		67.26
	Car Zone		71.05
	Classic Cars for Sale		76.02
	Internet Autoguide		154.22
Events	Linked In	Event{date, place, title, url}	9.89
	All Conferences		17.83
	Mbendi		6.95
	Net Lib		2.13
	RD Learning		4.23
Doctors	Web MD	Doctor{name, address, phone, specialty}	59.23
	Ame. Medical Assoc.		24.87
	Dentists		11.92
	Dr. Score		23.78
	Steady Health		81.39
Jobs	Insight into Diversity	Job{company, location, category}	30.36
	4 Jobs		79.76
	6 Figure Jobs		72.79
	Career Builder		54.17
	Job of Mine		23.9
Real Estate	Yahoo!	Property{address, beds, baths, size, price}	93.94
	Haart		89.64
	Homes		59.32
	Remax		69.98
	Trulia		175.39
Sport Players	Player Profiles	Player{name, birth, hight, weight, club}	20.89
	UEFA		63.42
	ATP World Tour		135.55
	NFL		94.92
	Soccer Base		85.02

Table 4
Non-derived effectiveness measures computed in our case study.

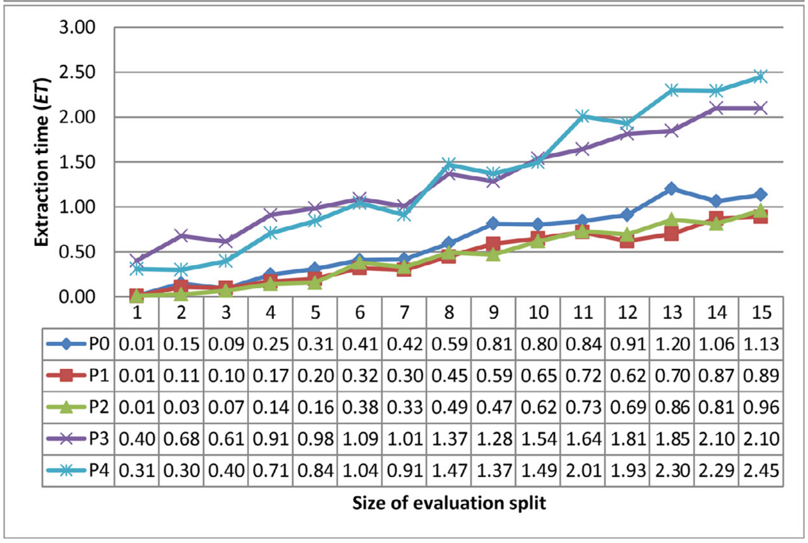


	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P0	0.01	0.02	0.08	0.05	0.13	0.14	0.22	0.25	0.23	0.25	0.30	0.35	0.37	0.45	0.45
P1	0.00	0.18	0.27	0.38	0.39	0.58	0.53	0.53	0.58	0.62	0.61	0.69	0.71	0.70	0.75
P2	0.00	0.14	0.33	0.43	0.47	0.46	0.54	0.54	0.54	0.68	0.68	0.71	0.75	0.67	0.73
P3	0.30	0.40	0.47	0.52	0.58	0.58	0.65	0.64	0.70	0.67	0.75	0.75	0.77	0.78	0.78
P4	0.20	0.31	0.49	0.54	0.62	0.66	0.75	0.73	0.80	0.83	0.86	0.88	0.90	0.90	0.85

Table 5
Non-derived time-related measures computed in our case study.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P0	0.44	0.54	0.68	0.80	0.84	0.88	1.07	1.20	1.30	1.18	1.31	1.39	1.66	1.88	1.89
P1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
P2	0.01	0.04	0.31	0.37	0.31	0.34	0.65	0.55	0.75	0.75	1.06	0.91	1.22	1.25	1.30
P3	6.74	7.49	8.00	7.89	8.40	9.09	8.59	9.40	9.46	9.78	10.4	10.7	11.2	11.7	12.2
P4	8.00	9.20	9.32	10.0	9.90	10.7	11.0	11.4	12.1	11.9	13.9	14.1	14.1	14.2	14.6



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P0	0.01	0.15	0.09	0.25	0.31	0.41	0.42	0.59	0.81	0.80	0.84	0.91	1.20	1.06	1.13
P1	0.01	0.11	0.10	0.17	0.20	0.32	0.30	0.45	0.59	0.65	0.72	0.62	0.70	0.87	0.89
P2	0.01	0.03	0.07	0.14	0.16	0.38	0.33	0.49	0.47	0.62	0.73	0.69	0.86	0.81	0.96
P3	0.40	0.68	0.61	0.91	0.98	1.09	1.01	1.37	1.28	1.54	1.64	1.81	1.85	2.10	2.10
P4	0.31	0.30	0.40	0.71	0.84	1.04	0.91	1.47	1.37	1.49	2.01	1.93	2.30	2.29	2.45

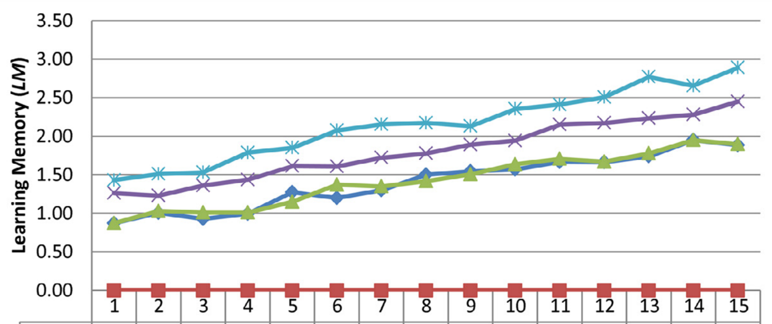
which increases the chances to learn extraction rules that are more general and effective. The behaviour of P0 is linear because the technique on which it relies does not actually attempt to learn a rule set that can generalise the features of the information to be extracted; the more learning documents, the more patterns are available, but the technique is far too naive and hardly can extract information from documents that are very similar to the learning documents; as a conclusion, it is not surprising that it behaves linearly, with a very small slope. Before concluding, we would like to highlight that proposal P0 cannot achieve a value for *AUC-ROC* greater than 0.50, which means that it behaves worse than a random guess and can then be removed from our study.

Regarding the learning time and the extraction time, all of the proposals seem to behave linearly, with small slopes; this confirms that they are very scalable and are then appropriate to deal with large datasets. Furthermore, the learning times range from a few milliseconds to quarter a minute and the extraction times range from a few milliseconds to a few seconds, which is reasonable in this context. Note that proposal P1 is based on heuristics, so it does not have a learning phase; thus the learning times are 0.00 seconds in every case.

Regarding the learning and extraction memory, they also seem to require an amount of memory that evolves linearly as the size of the evaluation splits increases; this is again a good piece of news since it confirms that all of the proposals are scalable in practice. Note that it is commonly required a little more memory to learn a rule set than to apply it, but the overall memory footprint seems very small in every case. Again, the learning memory required by P1 is 0.00 GiB in every case because it is a heuristic-based proposal.

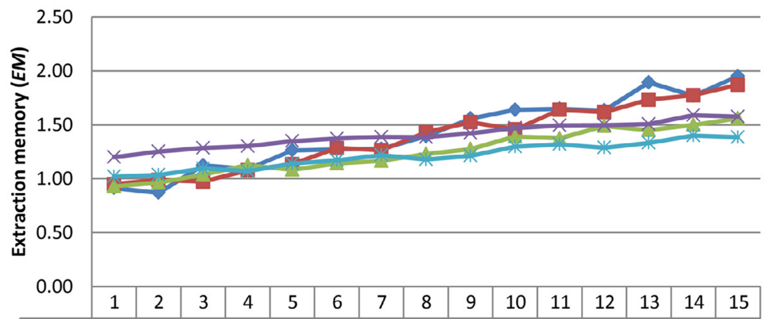
- **Derived measures.** Table 8 reports on the derived measures that we have computed.
  - Regarding the performance knee, note that both proposals P0 and P1 seem to require 15 documents so that they are able to achieve their best performance, which seems to be a clear indication that they might improve a little more if more documents were available in the evaluation splits; note, however, that 15 documents can be considered a large number, chiefly because it is necessary to annotate all of the information to be extracted so that the effectiveness measures can be computed. P2 seems to achieve its best performance with 12 documents, P3 with 11 documents, and

Table 6
Non-derived memory-related measures computed in our case study.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
◆ P0	0.87	1.01	0.93	0.99	1.27	1.20	1.30	1.50	1.54	1.57	1.67	1.66	1.74	1.95	1.88
■ P1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
▲ P2	0.87	1.03	1.01	1.01	1.15	1.37	1.35	1.42	1.51	1.63	1.71	1.67	1.78	1.95	1.90
× P3	1.26	1.23	1.36	1.44	1.61	1.61	1.72	1.78	1.89	1.94	2.15	2.17	2.23	2.28	2.45
✳ P4	1.43	1.51	1.53	1.79	1.85	2.08	2.15	2.17	2.13	2.35	2.41	2.51	2.77	2.66	2.89

Size of evaluation split



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
◆ P0	0.91	0.87	1.12	1.09	1.26	1.27	1.28	1.39	1.56	1.64	1.65	1.63	1.89	1.77	1.95
■ P1	0.95	0.99	0.97	1.08	1.14	1.28	1.27	1.43	1.52	1.46	1.64	1.62	1.73	1.77	1.87
▲ P2	0.93	0.96	1.04	1.13	1.08	1.14	1.17	1.23	1.28	1.39	1.38	1.49	1.45	1.50	1.56
× P3	1.20	1.25	1.28	1.30	1.35	1.37	1.39	1.39	1.42	1.47	1.49	1.50	1.51	1.59	1.58
✳ P4	1.02	1.04	1.09	1.07	1.14	1.17	1.22	1.18	1.21	1.30	1.32	1.29	1.33	1.40	1.39

Size of evaluation split

P4 with only 9 documents. Recall that P0 does not actually attempt to generalise rule sets, but uses prefixes and suffixes verbatim; thus, the more documents available, the more chances that the extraction rule set captures enough sequences of tokens so that the technique can extract correct information from the testing sets. On the contrary, P1 is a heuristic-based proposal that finds differences amongst documents, so the more documents, the more variability and the easier to infer which information has to be extracted. Proposal P2 is similar in spirit to P1, since it also compares differences amongst documents, so it also requires a relatively high number of documents to achieve its best performance. Proposals P3 and P4, which are based on standard machine-learning techniques seem to be the best at producing general-enough rules from as few as 11 or 9 documents, respectively.

- Regarding the failure ratio, note that proposals P0, P1, and P2 have failed on some datasets. P0 failed in 13.00% of the evaluation splits; after working this issue out, we found that the problem was the library that it uses to implement regular expressions, which uses a backtracking parser that fails to match some regular expressions of the form $\alpha|\alpha\beta$. P1 and P2 failed in 6.00% of the evaluation splits because they cannot work on a single document, so there were many evaluation splits on which they could not work.

Rankings. Next, we report on the local rankings and the global ranking that we have computed.

Local rankings. Table 9 reports on the local rankings that we have computed. The first column reports on the empirical ranks, as they are computed from the empirical data after purging and normalising them. They are computed very straightforwardly since they only require to average the values of the corresponding measures regarding every proposal. Then come the p-values that were computed using Wilcoxon's Rank-Sum test. Since we set the confidence level to its standard value $\alpha = 0.05$ and we have to compare 4 proposals, that means that we have to perform 6 comparisons on the same data. In other words, the decision boundary for the test is $\alpha/6 = 8.33\,10^{-3}$. In the table, we have highlighted the p-values that are below this decision boundary and thus indicate that there is enough evidence in the experimental data to conclude that the difference in rank amongst two given proposals is statistically significant. The last column reports on the resulting statistical rank. Note

Table 7
Best approximations of non-derived measures computed in our case study.

Area under the ROC curve (*AUC-ROC*)		
Proposal	Approximation	R^2
P0	0.03 *S* − 0.04	0.97
P1	0.27 ln *S*	0.97
P2	0.27 ln *S*	0.97
P3	0.19 ln *S* + 0.27	0.98
P4	0.28 ln *S* + 0.17 0	0.98

Learning time (*LT*)		
Proposal	Approximation	R^2
P0	0.10 *S* + 0.35	0.96
P1	0.00	1.00
P2	0.09 *S* − 0.09	0.95
P3	0.36 *S* + 6.56	0.97
P4	0.47 *S* + 7.87	0.97

Learning memory (*LM*)		
Proposal	Approximation	R^2
P0	0.08 *S* + 0.79	0.96
P1	0.00	1.00
P2	0.08 *S* + 0.81	0.97
P3	0.09 *S* + 1.11	0.99
P4	0.10 *S* + 1.34	0.97

Extraction time (*ET*)		
Proposal	Approximation	R^2
P0	0.09 *S* − 0.10	0.97
P1	0.06 *S* − 0.07	0.96
P2	0.07 *S* − 0.12	0.97
P3	0.12 *S* + 0.34	0.98
P4	0.16 *S*	0.97

Extraction memory (*EM*)		
Proposal	Approximation	R^2
P0	0.07 *S* + 0.82	0.96
P1	0.07 *S* + 0.83	0.98
P2	0.05 *S* + 0.89	0.98
P3	0.03 *S* + 1.20	0.98
P4	0.03 *S* + 0.99	0.96

S represents the size of the evaluation splits.

that Wilcoxon's Rank-Sum test does not lead to a total pre-order in the case of the area under the ROC curve: according to the test, there is not enough evidence to conclude that P1 behaves differently from P2; neither is there enough evidence to conclude that P2 behaves differently from P3; but there is enough evidence to conclude that P1 behaves differently from P3. To transform this into a total pre-order, we decided to break the chain at the comparison between P2 and P3, since this is the comparison for which Wilcoxon's Rank-Sum test returns the smallest p-value above the significance level, that is, the couple of proposals for which the data provides more evidence that they behave differently. The result is that we rank P3 and P4 in a group and P1 and P2 in a different group.

Global ranking. Table 10 reports on the global ranking that we have computed. According to the weights that we have assigned to the performance measures, the best-performing proposal is P3, which is closely followed by P4, and then come P2 and P1. This result is not surprising at all because the local rankings make a clear difference between P3 and P4 and the other proposals regarding effectiveness. Although there is a clear difference regarding learning time, too, they all range within a few seconds, which does not make an actual difference according to our preferences. There is also a difference regarding the extraction time, but not large enough to compensate for the superior effectiveness of P3 and P4.

Conclusions. In this case study, we have evaluated and compared five proposals in the literature.

P0 was a simple baseline, and it did not prove to be competitive enough with regard to the other proposals. It was removed from the comparison because it could not achieve an area under the ROC curve better than 0.50, which means that it performs worse than a random guess.

The best performing proposal was P3, which is a hybrid attempt to leverage standard machine-learning techniques that has proven to learn rule sets that are very effective and efficient. This proves that the idea of using these kind of techniques is very promising in the context of semi-structured documents. Such approaches have not been paid much attention in the literature, which means that it is worth exploring them. P4, which is based on inductive logic programming, ranks very close to P3, which is also an indication that trying to leverage standard machine-learning techniques is a good idea.

P1 and P2 rank at the bottom. None of them requires the user to provide an annotated learning set and they do their best at finding the differences amongst the documents on which they work, which is very likely a super-set of the information to be extracted. However, their inability to take the user knowledge into account has a clear impact on their performance.

Bibliography. This section is intentionally blank in this case study because we decided to keep the proposals anonymous.

4. Related work

In the following subsections, we first summarise the proposals that we have found in the literature; we then discuss on their key features and how they address the key questions that we have identified regarding a good ranking method; in every case, we make a point of highlighting how ARIEX advances the state of the art.

4.1. Review of the literature

We have found many informal methods in the general literature on information extraction, plus a few formal ones. In this section, we summarise our key findings regarding them.

The literature on information extractors. We have surveyed many proposals on web information extraction [2–8]. Our conclusion is that they provide a foundation and some guidelines to evaluate and compare information extraction proposals, but not formal methods that have the key features or address the key questions that we have identified. Obviously, their focus was not to provide such a method, but to support the idea that the new proposals that they introduced were better than others in the literature.

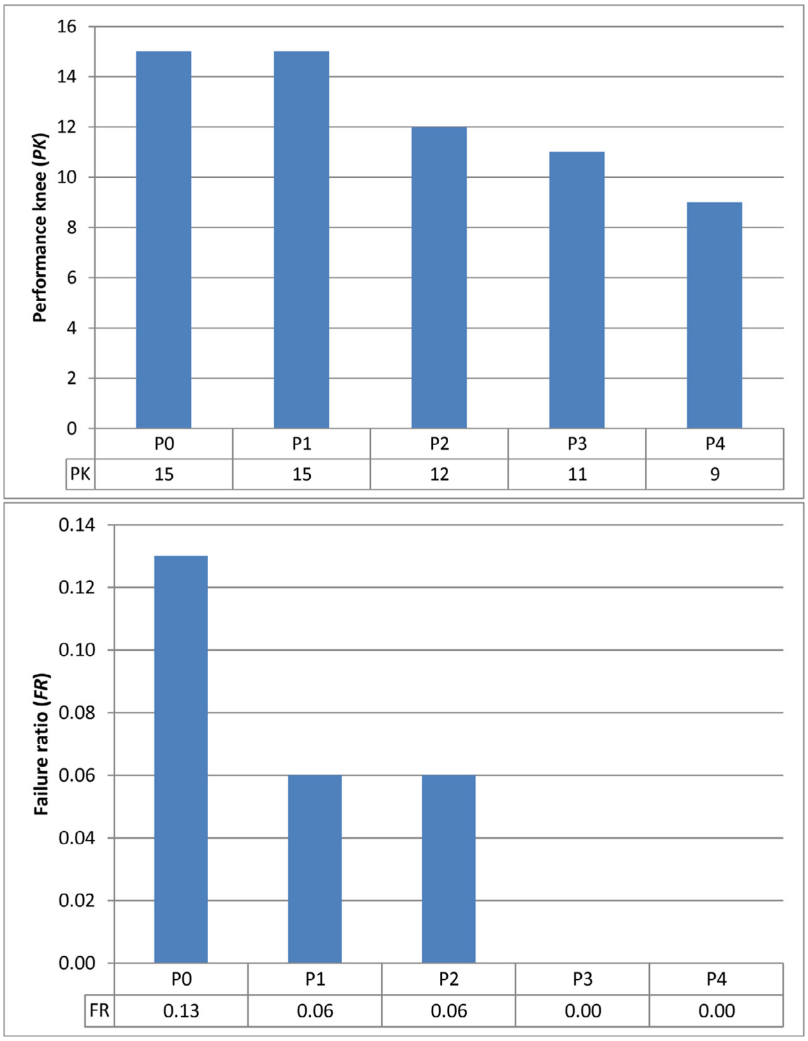
There are a few proposals that are a little surprising because they do not report on any experimental results [32–34] or report on very few [35,36], which does not contribute at all to drawing solid conclusions. Most of the remaining proposals provide enough empirical results, which helps support the conclusions better, but the methods used to evaluate and compare them were not solid enough.

Regarding the experimental environment, only a few proposals have paid attention to describing the hardware and the software used in the evaluation process [11,31,37]. Regarding comparing the results, it was surprising that many proposals were not compared to others at all, but to some variations of themselves that resulted from changing the values of their configuration parameters [4,15,36,38–49]. It was also surprising that not many proposals were evaluated on at least 20 datasets, which is the minimum that we recommend [11,13,26,31,39,41,44,48,50–52]; a few other proposals were evaluated on 10–20 datasets, which still amounts to a significant number of experiments [38,53]; the others were evaluated on less than 10 datasets, which we do not think is acceptable to draw conclusions.

Regarding effectiveness, all of the proposals report on precision, recall, and the F_1 score. Only a few report on the learning curves as a means to assess their ability to learn good extraction rules from as few documents as possible [13,25,44,54,55]. A few ones reported on the minimum number of documents that they require to learn effective extraction rules [14,15,48,52]. Unfortunately, very few proposals report on efficiency measures [11,15,31,38,40,53,56,57].

It is not commonly clear how the evaluation splits were created, since the procedure to create them is not mentioned at all; in some cases, it is unclear if the evaluation sets were different from the learning sets. According to the few cases in which this information

Table 8
Derived measures computed in our case study.



	P0	P1	P2	P3	P4
PK	15	15	12	11	9

	P0	P1	P2	P3	P4
FR	0.13	0.06	0.06	0.00	0.00

is provided, it seems that the favourite method is 10-fold cross validation [25,54,55,57] or a variant [13,45,57]. A few authors also used repeated random splits in which the documents available were randomly selected as learning or evaluation documents multiple times. In many cases, the partition was 50%–50% [11,24,58–60], but there are cases in which the learning set was smaller than the evaluation set [38,39,46,52,53] and vice versa [44,47,54,55,61]. Summing up, most proposals used testing sets that were not larger than the corresponding learning sets. A few proposals used a single random split, that is, they did not repeat the procedure multiple times [11,39,46,47,53,61]. Many proposals that work on free-text documents used the official testing sets that were released at the MUC conferences [42,43,55,62].

Regarding how the experimental data are used, it seems that every of the previous proposals analyses the data themselves, without cooking them. Furthermore, the results are analysed from a statistical point of view in very few cases [31,37] which makes it difficult to assess if the differences found amongst a number of proposals are statistically significant or not. Finally, no global rankings are computed; the proposals are compared according to different measures in isolation, but no attempt is made to compute a global ranking.

The literature on formal methods. We have also surveyed the few existing formal ranking methods. Lehnert and Sundheim [18], Chinchor et al. [19], and Hirschman [20] range amongst the first authors who worked on this topic. They worked in the context of the well-known MUC conferences, whose focus was on extracting information from free-text documents; they published a number of datasets so that the proposals that were presented at these conferences could be evaluated using a semi-automatic software tool that computed precision, recall, over-generation, and fallout. They proposed to analyse these measures in isolation with the help of tables and charts; the only exception was recall and precision, which could be analysed together since they can be easily combined thanks to the well-known F_1 score. They proposed to use an approximate randomisation method to find groups of proposals that rank equally or differently according to the values of the performance measures; this method is not intended to produce a ranking automatically, but to help a researcher handcraft a set of per-measure rankings by analysing the corresponding tables and charts.

Lavelli et al. [21] criticised the previous work and highlighted some common mistakes that authors make when they evaluate and compare their proposals. Their work was extended and updated by Ireson et al. [22] and Lavelli et al. [23], who reported on the conclusions from The Pascal Network of Excellence. They provided a repository that was composed of 1 100 annotated free-text documents that were intended to evaluate different systems as homogeneously as possible. They also explored some new ideas regarding experimentation, namely: studying how brittle a proposal is by

Table 9
Local rankings computed in our case study.

				Area under the ROC curve (*AUC-ROC*)				
Empirical ranking				Wilcoxon's Rank-Sum			Local ranking	
Proposal	Rank		P1	P2	P3	P4	Proposal	Rank
P1	2	P1	-	0.32	1.30E-03	1.00E-03	P1	2
P2	3	P2	-	-	0.27	0.25	P2	2
P3	2	P3	-	-	-	0.12	P3	1
P4	1	P4	-	-	-	-	P4	1

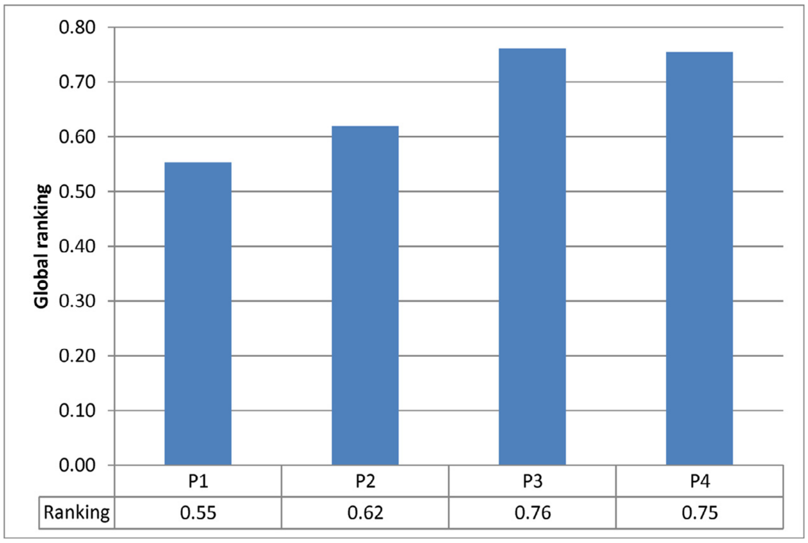
				Learning time (*LT*)				
Empirical ranking				Wilcoxon's Rank-Sum			Local ranking	
Proposal	Rank		P1	P2	P3	P4	Proposal	Rank
P1	2	P1	-	0.85	1.12E-03	1.10E-03	P1	1
P2	1	P2	-	-	9.12E-04	8.70E-04	P2	2
P3	3	P3	-	-	-	2.30E-03	P3	3
P4	4	P4	-	-	-	-	P4	4

				Learning memory (*LM*)				
Empirical ranking				Wilcoxon's Rank-Sum			Local ranking	
Proposal	Rank		P1	P2	P3	P4	Proposal	Rank
P1	1	P1	-	0.65	0.56	0.62	P1	1
P2	2	P2	-	-	0.47	0.34	P2	1
P3	4	P3	-	-	-	0.59	P3	1
P4	3	P4	-	-	-	-	P4	1

				Extraction time (*ET*)				
Empirical ranking				Wilcoxon's Rank-Sum			Local ranking	
Proposal	Rank		P1	P2	P3	P4	Proposal	Rank
P1	2	P1	-	0.78	2.30E-03	1.20E-03	P1	1
P2	1	P2	-	-	9.00E-04	7.80E-04	P2	1
P3	4	P3	-	-	-	0.68	P3	2
P4	3	P4	-	-	-	-	P4	2

				Extraction memory (*EM*)				
Empirical ranking				Wilcoxon's Rank-Sum			Local ranking	
Proposal	Rank		P1	P2	P3	P4	Proposal	Rank
P1	2	P1	-	0.56	0.56	0.78	P1	1
P2	2	P2	-	-	0.48	0.56	P2	1
P3	2	P3	-	-	-	0.69	P3	1
P4	1	P4	-	-	-	-	P4	1

Table 10
Global ranking computed in our case study.



	P1	P2	P3	P4
Ranking	0.55	0.62	0.76	0.75

using test documents that are sampled from a time frame that is different from the time frame used to collect the learning documents; studying the impact of 4-fold cross validation on the performance results; studying the learning curve, that is, how a proposal behaves as new documents are available in the learning sets; analysing active learning strategies, that is, studying how adding new documents to a learning set using a given heuristic may have an impact on the learning curve; and studying how enriching a learning set with data that comes from unannotated documents may have an impact on the results. The proposals were evaluated on the basis of precision, recall, and the F_1 score using a version of the software tool used in the MUC conferences. They proposed to use the same approximate randomisation method as in the MUC conferences to help produce the resulting rankings; they also proposed to use the bootstrap method to compute confidence intervals for every performance measure in an attempt not to draw conclusions from their raw mean values, but to take the effects of randomness into account. Note that the statistical tests are not intended to produce a ranking automatically, but to help a researcher handcraft a number of per-measure rankings.

4.2. *Key features*

The papers that introduce a new web information extractor do not provide enough details regarding whether the informal methods that they use are automated or not. We implicitly assume that the authors had some automated support to run their experiments and to compute the performance measures, but we do not think that the methods can be automated because they mostly rely on a researcher commenting on the experimental data and producing the resulting rankings. Neither are they open, since most of them compute precision, recall, and the F_1 score, or agnostic, since they were devised in the context of a specific proposal. Obviously, the authors' goal was not to devise a ranking method, just to evaluate their proposals and to provide some evidence that they could beat others in the literature.

The formal methods that we have surveyed are also supported by tools to compute the performance measures. However, they cannot be considered automated methods since they just provide some guidelines to help a researcher elaborate on the results of the experiments. They propose to use some statistical tests that help compute the rankings, but they still require a researcher to interpret some tables and charts. Neither are these methods open, since they commit to using a number of performance measures and provide specific guidelines to produce per-measure rankings. Furthermore, all of the methods focus on ranking supervised free-text proposals, which leaves out many other proposals in the literature.

The conclusion seems to be that the available informal or formal methods in the literature can be considered guidelines that are intended to help researchers produce per-measure rankings. We have managed to devise a method that is automated, which reduces the bias introduced by the researcher, open, since it can accommodate new performance measures as they are published and proven to be appropriate in our context, and agnostic, since it can be applied to any kind of proposal.

4.3. *Setting up the experimental environment*

The papers on web information extraction do not generally put an emphasis on describing the hardware or the software. They generally provide a list with the datasets that were used, but few other details are presented. They commonly commit to precision, recall, and the F_1 scores as effectiveness measures; unfortunately, almost none of them reports on efficiency measures.

The specific ranking methods do not put an emphasis on describing the hardware or the software. They all are accompanied with collections of datasets that were devised by a community of researchers. They focus almost exclusively on precision- and recall-based measures; efficiency measures are not taken into account.

The common theme in the literature seems to be that describing the hardware and the software is usually paid very little attention and that the performance measures used just provide a partial view of how good an information extractor is. In ARIEX, we did not forget about describing them.

Regarding the performance measures, it is surprising that most of the rankings in the literature focus exclusively on effectiveness measures, and do not provide a clue on efficiency measures. Our method proposes to use both kinds of measures since, otherwise, the resulting rankings would not provide an overall picture of how a proposal performs. We propose to use a small set of orthogonal measures, that is, measures that focus on different performance issues and are complementary to each other. In our survey of the literature, we have also found that precision, recall, and the F_1 score are the most common measures, but, unfortunately, it has also revealed that they are not the most appropriate in our context. The reason is that precision and, therefore, the F_1 score are skewed by unbalanced datasets, which are very common in our context.

In ARIEX, we have carefully studied the effectiveness measures in the literature, and we have selected the most appropriate in our context, cf. Appendix A. Note, however, that the method itself is not bound with these particular measures; it is open to accommodate new measures that might appear in the literature and prove to be appropriate in our context.

4.4. *Creating evaluation splits*

Generally speaking, it is not clear at all how the evaluation splits were created in the papers on web information extraction. In the few cases in which that information is provided, the favourite re-sampling method was k fold-cross validation; a few authors also tried n-repeated random sub-sampling and others borrowed the evaluation splits from other proposals in the literature. Note that reporting on how the evaluation splits are created is of uttermost importance since, otherwise, it is not clear if the results are biased or over-fitted.

Regarding the formal ranking methods, the proposals that originated in the context of the MUC conferences did not report on a method to create evaluation splits. They provided a single evaluation split that was handcrafted by the MUC community, which was obviously not enough to draw solid conclusions. To overcome this problem, they proposed to use the well-known bootstrap method in order to compute statistically accurate estimators of the performance measures. The other proposals simply comment on the methods that have been used by other authors and suggest that it is important to mention both the proportions between learning and testing sets and the re-sampling method used to split the datasets. They also encourage researchers to use existing learning and testing sets when using standard datasets and also recommend using as many evaluation splits as possible in order to achieve as much statistical significance as possible. Unfortunately, they do not commit to a specific method or analyse the pros and cons of the existing proposals.

The standard to create the evaluation splits seems to be the k-fold cross validation method, where k is typically set to 10. We think that this is not appropriate in the field of web information extraction because the learning sets are $k-1$ times larger than the testing sets. It is necessary that the testing sets are sufficiently large so as to reduce the possibilities that a proposal seems to work well or bad by chance. The problem is that one must provide 100 annotated documents to have 10 test documents, which is a time-consuming and error-prone task. Furthermore, all of the evaluation splits are the same size, which makes it difficult to analyse how the number of documents may have an impact on the performance of a proposal. Last, but not least, a dataset may include an outlier document, that is, a document that deviates largely from the others and makes it difficult to extract information from it; unfortunately, that document shall be in $k-1$ learning sets, that is, shall have a negative impact on every experiment, but one. Furthermore, the method promotes evaluating all of the available proposals on the same evaluation splits, which we think provides a biased view of their performance since some proposals need evaluation splits with only a few learning documents and others need evaluation splits that have a larger number of learning documents to achieve good results. Note that evaluating proposals on different splits does not make the comparison unfair if the appropriate statistical methods are used; however, the datasets on which the splits are computed must be the same because, otherwise, the features of the documents would be different and the comparison would not make sense. Note too, that this method is not appropriate for heuristic-based proposals since it uses as many documents as possible to produce learning sets that these proposals do not require.

Neither think we that *n*-repeated random sub-sampling is appropriate for our purpose; at least, as it has been instantiated in the literature. The reason is that authors have used this method to create evaluation splits in which, typically, most documents are used for learning purposes, which we have justified is not appropriate in our context. Furthermore, all of the evaluation splits are the same size since the only purpose is to make the choice of documents as random as possible, which we have also justified is not appropriate in our context.

In ARIEX, we propose a sub-sampling method that overcomes the problems that we have identified. It splits the datasets into a number of evaluation splits so that there is a set of evaluation splits for each possible size of the learning or the testing sets, as long as the size of the learning set never exceeds the size of the testing set. Thus, ARIEX ensures that the learning sets are not larger than the testing sets; it can generate a reasonable number of evaluation splits from relatively small datasets; furthermore, the existence of outlier documents does not have a negative impact on every evaluation split, which helps identify them. We run the proposals in their best experimental conditions, which involves computing the size of the evaluation splits on which they perform the best. Thus, every proposal is likely to use a different evaluation split on the same dataset and the comparison will be still fair. Heuristic-based proposals can use testing sets that are larger than in the case of *k*-fold cross validation and the availability of both learning and testing sets of different sizes allows to compare several proposals regarding their best-performing evaluation splits.

4.5. Computing raw experimental data

The papers on web information extraction do not generally put an emphasis on explaining how the experimental data are computed. The specific ranking methods rely on software tools that set a standard format for the data and allow to compute the performance measures automatically, but the user is allowed to make some corrections interactively.

The effectiveness measures in the literature [63] are commonly computed from confusion matrices that record the number of true positives, true negatives, false positives, and false negatives that are computed in a testing set regarding each slot; such per-slot measures must later be combined into per-extractor measures. Confusion matrices are well-known in the literature, but there are some issues that make computing them difficult in our context.

The first issue is regarding how matchings are computed. A correct matching happens when a piece of information that actually belongs to a slot is extracted as belonging to that slot. Intuitively, correct matchings should be exact, but this interpretation is very restrictive in practice. It is common that an information extractor produces inexact matchings, that is, that it extracts a part of the information of interest or some spurious information; this is particularly true in the case of extractors that work on DOM trees, since the text contained in a DOM node is not usually aligned with the information that is expected to be extracted. The papers that introduce new information extraction proposals do not report on how matchings are dealt with. Regarding the formal methods, the idea of inexact matching was first introduced by Chinchor et al. [19]; later, Lavelli et al. [23] emphasised that the way that matchings are computed may have an impact on the results of an evaluation and then on the final ranking, but they did not elaborate more on this issue. Chinchor et al. [19] used a simple approach in which they compute confusion matrices using exact matchings, but record the number of partial and incorrect matchings; then, they compute their effectiveness measures using customised formulae. Their approach is very simple, because they count a partial matching as half a true positive and an incorrect matching as one false positive and one false negative. Unfortunately, they reported on problems to compute true negatives in the case of multi-valued slots; consequently, they had trouble to compute the effectiveness measures that depend on this count. They had to resort to an interactive post-processing phase in which a user could mend the measures that were computed automatically, thus increasing the chances to introduce a bias in the results. We think that how matchings are computed must be taken into account when computing confusion matrices, since, otherwise, they do not actually reflect the effectiveness of an information extractor, but just provide an approximation. In ARIEX, we have devised a method that basically takes into account the ratio of tokens that must have been extracted or discarded with regard to the tokens that have been extracted; it has proven to be both simple and effective at dealing with the problem of inexact matchings.

The second issue is regarding how effectiveness measures that are computed on a per-slot basis are generalised to per-extractor measures. In the literature, effectiveness measures generalised by using macro-, micro-, or weighted averages. Unfortunately, the authors have not commonly paid attention to the problem that micro- and weighted averages are skewed in the context of unbalanced datasets, which makes them of little interest in our context [64]. In ARIEX, we recommend using macro-averages because they are known not to be skewed in our context; furthermore, we suggest using the area under the ROC curve as an effectiveness measure and we have found an efficient means to extend it to a per-extractor level [65].

The third issue is regarding how to compute confusion matrices when evaluating an unsupervised or a heuristic-based proposal. If a proposal is supervised, then it is trained to return pieces of text as belonging to a specific user-defined slot; contrarily, if a proposal is unsupervised, then it learns to extract as much information as possible, which is automatically assigned to computer-generated slots; heuristic-based proposals do not learn extraction rules, but assign the information that they extract directly to computer-generated slots. The problem is how to map the computer-generated slots onto the appropriate user-defined slots so that confusion matrices can be computed. In the papers in which an unsupervised or a heuristic-based proposal has been evaluated, the authors have handcrafted these mappings, but this is a time consuming task, not to mention error-prone. In ARIEX, we provide a specific automated method to compute confusion matrices for unsupervised and heuristic-based proposals.

4.6. Cooking the experimental data

Unfortunately, both the papers on web information extraction and the formal methods that we have found in the literature use the experimental data as they are gathered from running the experiments. However, we support the idea that the experimentation would be less biased and more stringent if each proposal was compared in its best experimental conditions. Otherwise, the rankings might be biased because a proposal might seem to perform better than another, but the latter might perform better if different evaluation splits were chosen. Furthermore, there are derived performance measures that cannot be computed on a per-evaluation-split basis, but have to be computed from the raw experimental data. It is important that the data be normalised, since, otherwise, the differences in range or deviation might have an impact on the resulting rankings. Furthermore, unless the experimental data are normalised, it is very difficult that a ranking method can work with arbitrary sets of performance measures.

In ARIEX, we purge the raw experimental data so as to remove the data that correspond to very bad proposals according to a given purging measure, and to remove the evaluation splits that do not correspond to the best performing-splits for each proposal. Simply put, we choose the smallest evaluation splits on which a proposal achieves its best effectiveness results. We look for those smallest evaluation splits because the less documents a set has, the less annotation effort is required (in supervised proposals) and the faster it is expected to work (generally speaking). This is the reason why we call them the best-performing splits. We also take into account that there can be performance measures that cannot be computed on a per-evaluation-set basis, but are derived from other measures that are computed on that basis. Finally, we normalise the data so that all of the measures range within the same interval and the interpretation of this interval is homogeneous.

4.7. Computing rankings

The informal ranking methods simply average the experimental data and use the results to rank the proposals. Since they usually focus on precision and recall, the final ranking can be computed in terms of the F_1 score, which combines them both. The problem with such rankings is that they do not take the deviations into account, so it is not clear whether the results are skewed by the data distribution; that is, it is not clear if the differences in the mean measures are significant from a statistical point of view. There are only a few papers that perform a statistical analysis. Furthermore, the rankings regarding each measure are produced and studied in isolation; that is, no attempt to derive a global ranking is made.

All of the formal ranking methods use statistical procedures to make sure that the resulting rankings are as sound as possible. The problem is that they rely on techniques that are computationally intensive and outdated; furthermore, there are many cases in which they do not lead to a total ranking, but the problem has not been studied further because these methods are not intended to be automated, but require a person to interpret the results and draw conclusions. Furthermore, none of the methods provide a means to compute a global ranking from the experimental data.

ARIEX also computes local rankings on a per-measure basis and it also makes sure that the results are statistically sound. The difference is that we rely on Wilcoxon's Rank-Sum test, which is efficient and there are two versions that are specifically adapted to small and large datasets. This is a non-parametric test because it does not assume that the experimental data have a pre-defined distribution and it works on non-paired samples, so that it can be applied to the experimental data computed from the best-performing evaluation splits of each proposal. Furthermore, we propose a method to compute a global ranking that relies on all of the performance measures, instead of studying them in isolation. It is novel in that we take both the researcher's preferences and a combination of the means and the deviations of the performance measures into account.

4.8. Reporting on the results

The informal methods that we have surveyed typically report on the experimental data and then provide some conceptual explanations. The formal methods provide some intuitive guidelines, but they do not make a proposal regarding how to write a report.

In ARIEX, we have carefully studied how to organise such a report. Our emphasis was on organising it as effectively as possible and on providing the information that researchers need to understand how a proposal compares to others, without providing spurious information or information that is of little interest for practical purposes.

5. Conclusions

The literature provides many proposals on web information extraction. We have surveyed many rule-based or heuristic-based proposals, supervised or unsupervised rule-based proposals, proposals that work on free-text or semi-structured documents, and proposals that are open or closed. Our conclusion is that most of them use ad-hoc ranking methods that are not sufficiently specified and that, in some cases, have important deficiencies. There are a few formal methods to rank information extractors, but they also have a number of problems that have hindered their applicability in practice. As a conclusion, the existing web information extraction proposals have been ranked using quite heterogeneous methods, which makes comparing the results that have been published in the literature impossible.

In this article, we introduce ARIEX, which is a method to rank web information extractors that overcomes the deficiencies that we have found in the literature. It is automated, so that researchers can bias the conclusions as little as possible, open, so that it can easily accommodate new performance measures, and agnostic, so that it can be applied to as many different kinds of proposals as possible. Furthermore, it addresses the following questions: how to set up the experimental environment, how to create appropriate evaluation splits, how to compute the experimental data, how to cook them, how to compute the rankings, and how to report on the results.

We have also analysed the performance measures that have typically been used in the literature and we have concluded that they are skewed in contexts in which the datasets are unbalanced, which are common in our context. We have made a recommendation regarding a set of performance measures that are appropriate in this context. They take into account both the effectiveness and the efficiency of a proposal, it is small so that a researcher can easily decide on the weight of each measure, and the measures themselves are orthogonal, so that they provide a good overview of how a proposal performs. We have also supported the idea that each proposal must be compared regarding its best experimental conditions and we have reported on a method to find them that is based on computing a so-called performance knee.

Summing up, we think that we have contributed to the state of the art with a solid method to evaluate, compare, and rank information extraction proposals. It was our experience regarding devising new information extractors that motivated us to work on it [7,31,37,66–68]. Unfortunately, that experience also revealed that there are two obstacles in practice: the lack of public datasets and the unavailability of public implementations. Previously, we have listed the repositories of which we are aware. They provide a large collection of datasets on very different topics and a sufficiently large number of documents; the problem is that we found that most of them are outdated, that is, they provide web documents that are not representative enough of the kind of web documents that typical web sites provide nowadays. Our experience also proves that there are very few proposals with a public implementation; contacting the authors does not usually help since it is very unlikely to get an answer from them; furthermore, most of the implementations one can find are research prototypes that are very difficult to set up and get running; implementing other proposals is not commonly sensible because their details are subtle and sometimes not described in complete detail, not to mention the effort that such a re-implementation requires. Unfortunately, there is little we can do regarding the previous problems, except for pushing other authors to contribute to the research community with their datasets and implementations. We think that the method that we present in this article shall definitely help in this task since it provides a solid guideline to evaluate, compare, and rank different proposals; we expect that researchers are not so reluctant to publish their datasets and implementations if they know that they are going to be compared using a solid method that guarantees that the results cannot be biased and are then not subject to (mis)interpretations. We think that this is very appealing for both the authors of new information extractors, who can use ARIEX to systematically prove that their proposals outperform others in the literature, and practitioners, who can use it to make informed decisions regarding which of the proposals in the literature is the most appropriate to tackle a particular information extraction problem.

In a world in which the information that is available on the Web is increasingly feeding knowledge-based systems, we think that a systematic method to rank information extractors is a must and that our contribution to the field is solid enough to be used in forthcoming research studies.

Acknowledgements

We are grateful to Dr. José C. Riquelme, from the University of Sevilla, Spain, for the many fruitful discussions regarding evaluating proposals and comparing experimental results. We would also like to thank Dr. Francisco Herrera, from the University of Granada, for sharing his statistic analysis software with the research community. Our work was supported by FEDER funds that were awarded by the European Commission, the Spanish Government, and the Andalusian government through contracts TIN2007-64119, P07-TIC-2602, P08-TIC-4100, TIN2008-04718-E, TIN2010-21744, TIN2010-09809-E, TIN2010-10811-E, TIN2010-09988-E, TIN2011-15497-E, and TIN2013-40848-R. The work by Patricia Jiménez was also partially supported by the University of Southern California during a research visit that she paid to the Information Sciences Institute.

Appendix A. Performance measures

We have surveyed the literature, and we have found many effectiveness measures [63,69]; unfortunately, we have not found so many efficiency measures. In this appendix, we report on them and justify the measures that we suggest using in ARIEX, as long as no other measures are published and proved to be more appropriate. We first report on effectiveness measures, then on efficiency measures, and, finally, we provide a justification regarding some implementation-related measures that are commonly neglected in the literature.

A.1. Effectiveness measures

Our proposal regarding effectiveness measures is to classify them into error-related measures, generalisation-related measures, and failure-related measures. Next, we summarise our findings regarding them.

Error-related measure. We have surveyed the literature, and we have found many error-related measures [63,69], cf. Table A.1. Some of them are partial because they focus on either how good a proposal is at extracting or ignoring slots; the others are global because they were designed to report on both abilities at the same time.

The partial measures can be further classified as follows: a) measures that assess the error type I (aka false alarms), that is, the number of pieces of information that are incorrectly extracted as belonging to a given slot or its complement; these measures include precision (P), the false positive rate (FPR), and the true negative rate (TNR); b) and measures that assess the error type II (aka misses), that is, the number of pieces of information that are not extracted as belonging to a given slot or its complement; these measures include recall (R), the false negative rate (FNR), and the negative predictive value (NPV).

The global measures that we have found are the following: the F_1 score (F_1), accuracy (Acc), the Matthews correlation coefficient (MCC), the area under the PR curve ($AUC\text{-}PR$), and the area under the ROC curve ($AUC\text{-}ROC$).

Generalisation-related measures. Regarding generalisation measures, our survey of the literature suggests that so-called learning curves should be used. Such curves display how the performance of a supervised proposal evolves as the learning set is grown from a relatively small set of documents up to an arbitrarily large set. There is typically a size of the learning set at which the performance achieves its maximum value and becomes stable; that size is a knee in the learning curve, that is, an inflection point that can be compared to others in order to assess how good a proposal is at generalising good extraction rules from a small set of input documents. Our proposal is to use this performance knee (PK) as a measure to assess the effort required to assemble the set of documents from which a proposal must learn an extraction rule set. Although the idea is conceptually simple, we have found two important problems, which we have addressed in our method.

The first problem is regarding heuristic-based proposals. They do not have a learning phase, so we can select the minimum number of documents that allows a proposal to work at its maximum performance as its corresponding performance knee.

The second problem is that we have not found any results regarding how to compute the performance knee; the literature suggests that the learning curves be compared intuitively, which is not appropriate to devise an automated method. We have devised a method to compute the exact performance knee, which is presented in Fig. A.1. It gets some raw experimental data, a proposal, and a measure as input, and it returns the corresponding performance knee. The idea is to map the problem onto a statistical problem as follows: given a proposal, we create n new variables $X_1, X_2, \ldots, X_n$, where each X_i ranges over the values of the selected measure when it is computed on the evaluation splits of size i, where i ranges from 1 to n. Realise that these variables can be viewed as experimental samples of some unknown random variables. Note that prior to initialising variables X_i, our method needs to compute the set of evaluation split sizes in the experimental data, to which we refer to as T; n simply denotes the maximum evaluation split size. How the size is computed depends on whether the proposal being analysed is rule-based or heuristic-based: in the former case, it is computed as the size of the learning sets; in the latter case, it is computed as the size of the testing sets. After computing the X_i variables, we have to find a permutation r that ranks them according to their average value, in increasing order; in cases in which there are ties, we suggest that they should be broken by putting the variable that corresponds to the smallest evaluation split first. We then can apply the well-known Wilcoxon's Rank-Sum test [29] to find the first variable that is statistically indistinguishable from $X_{r(n)}$; in other words, to find the variable that corresponds to the smallest evaluation split on which the input proposal achieves a performance regarding the input measure that is statistically indistinguishable from the maximum. Given two samples of two random variables, this test computes a p-value that must be compared to $\alpha/(n-1)$, where α denotes the statistical significance level set by the researcher as a parameter of ARIEX; note that we cannot compare it to α since we need to perform several tests on the same data, so it is necessary to apply Bonferroni's correction [29]; when the p-value is equal to or greater than the $(n-1)$-th part of the significance level, the variables are indistinguishable from a statistical point of view; if all of the variables are indistinguishable from $X_{r(n)}$, that means that we are in an exceptional case in which a technique performs the same in every situation, which is, obviously, not expected to be very frequent in practice.

Before concluding, we would like to emphasise that the method that we have proposed is generic, since it can compute the performance knee of an arbitrary measure and proposal. However, our study of the literature proves that the only measure that seems appropriate in our context is the area under the ROC curve. We have, however, decided to propose a generic method since many authors are working on new performance measures and ARIEX is open to accommodate them as they are devised and proved to be appropriate in our context.

Failure-related measures. Regarding the failure-related measures, our survey of the literature reveals that no-one has paid attention to them. Authors have basically ignored that the implementations are far from perfect and, thus, may fail, which we think is very important from a practitioner's point of view.

Table A1
Common error related measures.

Name	Aliases	Definition	Range	Goodness
Error Type I measures (Partial)				
Precision (*P*)	Positive Predictive Value (*PPV*)	*tp* / (*tp* + *fp*)	0.00..1.00	Maximise
False Positive Rate (*FPR*)	Negative Error (*NE*), Fallout (*FO*)	*fp* / (*fp* + *tn*)	0.00..1.00	Minimise
True Negative Rate (*TNR*)	Specificity (*SPC*), Negative Accuracy (*NA*)	*tn* / (*tn* + *fp*)	0.00..1.00	Maximise
Error Type II measures (Partial)				
Name	**Aliases**	**Definition**	**Range**	**Goodness**
Recall (*R*)	Sensitivity (*Sens*), True Positive Rate (*TPR*), Hit Rate (*HR*)	*tp* / (*tp* + *fn*)	0.00..1.00	Maximise
False Negative Rate (*FNR*)	Positive Error (*PE*)	*fn* / (*fn* + *tp*)	0.00..1.00	Minimise
Negative Predictive Value (*NPV*)		*tn* / (*tn* + *fn*)	0.00.1.00	Maximise
Global measures				
Name		**Definition**	**Range**	**Goodness**
F_1 score (F_1)		2 tp^2 / ((*fn* + *tp*) (*fp* + *tp*) (*tp* / (*tp* + *fn*) + *tp* / (*tp* + *fp*)))	0.00..1.00	Maximise
Accuracy (*Acc*)		(*tp* + *tn*) / (*tp* + *tn* + *fp* + *fn*)	0.00..1.00	Maximise
Matthews Correlation Coefficient (*MCC*)		(tp tn - fp fn) / sqrt((*tp* + *fp*) (*tp* + *fn*) (*tn* + *fp*) (*tn* + *fn*))	-1.00..1.00	Maximise
Area under the PR curve (*AUC-PR*)		0.5 (*tp* / (*tp* + *fn*) -1) (*tp* / (*tp* + *fp*) - 1) - *tp* (*tp* / (*tp* + *fp*) - 1) / (*tp* + *fn*) + tp^2 / (2 (*tp* + *fn*) (*tp* + *fp*))		
Area under the ROC curve (*AUC-ROC*)		0.5 (1 + *tp* / (*tp* + *fn*) - *tn* / (*tn* + *fp*))	0.00..1.00	Maximise

The definitions refer to a confusion matrix *C* = (*tp*, *tn*, *fp*, *fn*).

```
method $computePerformanceKnee$($data$, $proposal$, $measure$)
    if $proposal$ is rule-based then
        $T$ = $\{s \mid \exists l, t, d \bullet (proposal, (l, t), d) \in data \wedge s = |l|\}$
        $n$ = $\max T$
        for each $i \in [1 \,..\, n]$ do
            let $X_i$ = $\{v \mid \exists l, t, d \bullet (proposal, (l, t), d) \in data \wedge |l| = i \wedge v = d(measure)\}$
        end
    else
        $T$ = $\{s \mid \exists l, t, d \bullet (proposal, (l, t), d) \in data \wedge s = |t|\}$
        $n$ = $\max T$
        for each $i \in [1 \,..\, n]$ do
            let $X_i$ = $\{v \mid \exists l, t, d \bullet (proposal, (l, t), d) \in data \wedge |t| = i \wedge v = d(measure)\}$
        end
    end
    let $r : \{1, 2, \ldots n\} \mapsto \{1, 2, \ldots, n\}$ be a permutation such that
        $\overline{X_{r(i)}} \leq \overline{X_{r(i+1)}}$ for every $i \in [1 \,..\, n-1]$
    let $result$ be the smallest $r(i)$ such that
        $Wilcoxon\text{-}Rank\text{-}Sum\text{-}Test(X_{r(i)}, X_{r(n)}) \geq \alpha/(n-1)$
return $result$
```

Fig. A.1. Method to compute the performance knee.

Our proposal is to use a measure called failure ratio (*FR*), which amounts to the ratio of evaluation splits on which the available implementation of a proposal failed with regard to the total number of evaluation splits on which it was run. This measure can be trivially computed from raw experimental data, so we do not provide any additional details here.

A.2. *Efficiency measures*

Unfortunately, efficiency measures have not been paid much attention in the literature. Almost no author reports on them, but we think that they are very important to provide an actual overall picture of how a proposal performs in practice. They are of uttermost importance to practitioners who have to make a decision regarding which the most appropriate proposal is regarding a particular problem.

We suggest using the following ones: learning time (*LT*) and learning memory (*LM*), which refer to the time taken and the memory required to learn a rule set, respectively; and extraction time (*ET*) and extraction memory (*EM*), which refer to the time taken and the memory required to extract information from a document. If a proposal is based on heuristics, then its learning time and its learning memory can be trivially set to zero, since it does not learn any rules, but extracts information directly from a dataset.

Regarding the learning time and the extraction time, it is worth mentioning that it is common to distinguish between computer and user time. The former refers to the time that the CPU or the IO devices are allocated to running a process, whereas the latter refers to the total time that elapses since a process is started until it finishes, which includes the time that the computer is running other processes. Computer times tend to be quite stable, i.e., when an algorithm is repeatedly executed on the same input they do not vary largely; contrarily, user times are not so stable because they depend on many other processes that can run concurrently on the same machine. As a conclusion, our proposal is to measure computer times only.

A.3. A note on global error-related measures

Previously, we have reported on the error-related measures that we have found in the literature. Obviously, we recommend using the global ones since they are the only that report on how good a proposal is at both extracting the information in which we are interested and ignoring the rest. The standard is to use the F_1 score, which combines precision and recall. Unfortunately, our study reveals that this measure is not appropriate in the context of ARIEX.

The reason is that the F_1 score is skewed in the context of unbalanced datasets. A dataset is said to be unbalanced when the number of instances of a slot deviates from the number of instances of the remaining slots. In our context, the datasets are naturally unbalanced because the amount of information to be ignored in a web document typically exceeds the amount of information to be extracted; furthermore, some slots are optional and some others are multi-valued, which also contributes to making the datasets naturally unbalanced.

To understand the reason why using the F_1 score in the context of unbalanced datasets is problematic, we use the following example: assume that a proposal is evaluated on a dataset that has 15 documents that provide a total of 15 instances of a given slot; assume, too, that the resulting confusion matrix is $(tp_1, tn_1, fp_1, fn_1) = (15, 2371, 98, 0)$, which implies that precision is 0.13, recall is 1.00, and the F_1 score is 0.23. In other words, it does not seem to be a good proposal because precision is very low, but note that this proposal is actually very good because it makes very few mistakes. Assume that another proposal is evaluated on a dataset that provides 15 documents, but only 13 instances of the slot being considered; assume, too, that the corresponding confusion matrix is $(tp_2, tn_2, fp_2, fn_2) = (13, 960, 40, 0)$. That is, its precision is 0.25, its recall is 1.00, and its F_1 score is 0.39. Neither seems this proposal to be excellent, but a little better than the previous one. Note however, that a deeper analysis can easily reveal that both proposals behave very similarly because they successfully extract every instance of the slot being considered and roughly 4% of the examples to be ignored are mistakenly extracted as belonging to that slot. In other words, they behave very similarly regarding their ability to extract or ignore information. The problem is that the F_1 score provides a distorted view of these proposals because they have been evaluated on testing sets that have different skews.

A good global error-related measure must depend only on the proposal being evaluated, not on the dataset used to evaluate it being balanced or unbalanced. That is, it should be possible to maximise the measure by improving the techniques that lie at the core of a proposal, not by changing the proportion of information to be ignored in a dataset.

To find out which of the global error-related measures that we have presented before is not skewed in the presence of unbalanced datasets, we have re-written their formulations in terms of the following measures: the true positive rate ($TPR = tp/(tp + fn)$), which measures the proportion of instances of a slot that are extracted as belonging to that slot with regard to the total number of actual instances of that slot, the false positive rate ($FPR = fp/(fp + tn)$), which measures the proportion of information that is extracted as belonging to a given slot with regard to the information that must be ignored, and the skew of the dataset ($S = (tn + fp)/(tp + fn)$), which measures the proportion of information to be ignored with regard to the information to be extracted as belonging to a given slot. Note that we have selected the true positive rate and the false positive rate because these measures provide a clear picture of how good a proposal is at extracting or ignoring information and they have been proven not to be skewed in the context of unbalanced datasets [30]. Next, we present the results of re-writing the global error-related measures in terms of the previous measures:

$$F_1 = \frac{2\ TPR}{FPR\ S + TPR + 1}$$

$$Acc = -\frac{(FPR - 1)\ S - TPR}{S + 1}$$

$$MCC = \frac{\alpha\ (FPR - TPR)\ \sqrt{\beta + TPR}}{(FPR^2 - FPR)\ S^2 - \beta + ((2\ FPR - 1)\ S - 1)\ TPR + TPR^2}$$

$$\text{where } \alpha = \sqrt{-FPR\ S + S - TPR + 1}\ \sqrt{S}$$

$$\text{and } \beta = FPR\ S$$

$$AUC\text{-}PR = \frac{FPR\ S\ TPR + FPR\ S + TPR^2}{2\ (FPR\ S + TPR)}$$

$$AUC\text{-}ROC = \frac{1}{2}\ (1 + TPR - FPR)$$

Realise that the area under the ROC curve is the only measure that does not depend on S when it is re-written, which analytically proves that it is the only measure that is not skewed in the context of unbalanced datasets. In other words, it is the only that we can recommend in the context of ARIEX. Regarding our previous examples, the area under the ROC curve is 0.98 in both cases, which reflects that the corresponding proposals were not that bad and that the dataset being unbalanced does not have an impact on the results.

A.4. A note on computing per-extractor measures

To compute a ranking, we need to compute the performance measures on a per-extractor level. It is very easy to compute per-extractor efficiency measures because we just need to measure the time that elapses since an experiment starts running until it finishes or to probe the maximum amount of memory requested. Regarding effectiveness measures, the problem is a little more involved because we can compute per-slot measures and we then have to combine them in a manner that makes sense in the context of unbalanced datasets.

Generally speaking, the problem has been addressed using macro-, micro-, or weighted averages. Macro averages are calculated by computing the effectiveness measures in a per-slot basis and then computing their unweighted averages; micro averages are computed from a global confusion matrix that is, in turn, computed by adding the confusion matrices that correspond to each slot; weighted averages are computed like macro averages, but the measures are weighted by the number of actual instances of each slot. Our general recommendation is to use macro averages because micro- and weighted averages have been proven to be skewed in the context of unbalanced datasets [64].

We have also found some specific research results regarding the area under the ROC curve, which is the most appropriate effectiveness measure that we have found so far [30,65,70]. The only one that seems both effective and computationally tractable is the one by Hand and Till [65], which computes it as follows:

$$AUC\text{-}ROC = \frac{\sum_{i,j \in S, i \prec j} AUC\text{-}ROC_{i,j}}{(|S|^2 - |S|)/2}$$

where $AUC\text{-}ROC_{i,j}$ refers to a new pairwise measure that combines every two slots, $\prec$ denotes an arbitrary ordering of the slots, e.g., a lexicographic ordering, and S denotes the set of slots to be extracted.

Note that there are $(|S|^2 - |S|)/2$ pairs of slots if their order is not taken into account. Simply put, the proposal amounts to macro averaging the pairwise area under the ROC curve, which has proven to work very well.

A.5. A note on implementation-related measures

The failure ratio, the timings, and the amounts of memory are related to a particular implementation of a proposal. Some researchers might argue that they are not appropriate as performance measures because they might lead to a distorted view of a proposal. The reason is that they depend on a programmer's ability to produce efficient code, on the implementation language, on the hardware and the software used to run the experiments. In other words, an intrinsically very efficient proposal might seem worse than another one because it was not well implemented or because the experimentation environment was not configured properly. However, we think that the failure ratio, the timings, and the amounts of memory are the only way for a practitioner to have a good overall picture of how a proposal performs in practice.

Some researchers might argue that we should evaluate the efficiency of a proposal building on its theoretical time or space complexity, but we do not think that such an approach is realistic because only a few authors have characterised the theoretical complexity of their proposals; furthermore, many of them have characterised an upper bound to the actual theoretical complexity to prove that their proposals are computationally tractable, not their actual complexity; even worse: even if we knew the exact theoretical complexity of every proposal, the relationships amongst most theoretical complexity classes are still open problems in computer science [71].

Thus our conclusion is that the failure ratio, the timings, and the amounts of memory that we propose to compute are very appropriate from a practitioner's point of view. ARIEX is flexible enough to allow the researcher who is using it to decide on the weight of every measure, which may range from 0.00% to 100.00%. That is, the final ranking might forget about the failure ratio, the timings, and the amounts of memory if the researcher does not find them appropriate; furthermore, ARIEX is open to accommodate new performance measures if they are proven to be adequate in our context.

References

[1] I. Hernández, C.R. Rivero, D. Ruiz, R. Corchuelo, CALA: an unsupervised URL-based web page classification system, Knowl.Based Syst. 57 (2014) 168–180.
[2] A.H.F. Laender, B.A. Ribeiro-Neto, A.S. da Silva, J.S. Teixeira, A brief survey of web data extraction tools, SIGMOD Record 31 (2) (2002) 84–93.
[3] C.-H. Chang, M. Kayed, M.R. Girgis, K.F. Shaalan, A survey of web information extraction systems, IEEE Trans. Knowl. Data Eng. 18 (10) (2006) 1411–1428.
[4] N. Kushmerick, B. Thomas, Adaptive information extraction: core technologies for information agents, in: AgentLink, 2003, pp. 79–103.
[5] J. Turmo, A. Ageno, N. Català, Adaptive information extraction, ACM Comput. Surv. 38 (2) (2006).
[6] S. Sarawagi, Information extraction, Found. Trends Databases 1 (3) (2008) 261–377.
[7] H.A. Sleiman, R. Corchuelo, A survey on region extractors from web documents, IEEE Trans. Knowl. Data Eng. 25 (9) (2013) 1960–1981.
[8] E. Ferrara, P.D. Meo, G. Fiumara, R. Baumgartner, Web data extraction, applications and techniques: a survey, Knowl.Based Syst. 70 (2014) 301–323.
[9] N. Papadakis, D. Skoutas, K. Raftopoulos, T.A. Varvarigou, Stavies: a system for information extraction from unknown web data sources through automatic web wrapper generation using clustering techniques, IEEE Trans. Knowl. Data Eng. 17 (12) (2005) 1638–1652.
[10] M. Álvarez, A. Pan, J. Raposo, F. Bellas, F. Cacheda, Extracting lists of data records from semi-structured web pages, Data Knowl. Eng. 64 (2) (2008) 491–509.
[11] K. Simon, G. Lausen, ViPER: augmenting automatic information extraction with visual perceptions, in: Proceedings of the CIKM, 2005, pp. 381–388.
[12] L. Li, Y. Liu, A. Obregon, M. Weatherston, Visual segmentation-based data record extraction from web documents, in: Proceedings of the IRI, 2007, pp. 502–507.
[13] W. Su, J. Wang, F.H. Lochovsky, ODE: ontology-assisted data extraction, ACM Trans. Database Syst. 34 (2) (2009).
[14] Y.K. Shen, D.R. Karger, U-REST: an unsupervised record extraction system, in: Proceedings of the WWW, 2007, pp. 1347–1348.
[15] H. Elmeleegy, J. Madhavan, A.Y. Halevy, Harvesting relational tables from lists on the Web, VLDB J. 20 (2) (2011) 209–226.
[16] R. Gupta, S. Sarawagi, Answering table augmentation queries from unstructured lists on the Web, PVLDB 2 (1) (2009) 289–300.
[17] O. Etzioni, A. Fader, J. Christensen, S. Soderland, Mausam, Open information extraction: the second generation, in: Proceedings of the IJCAI, 2011, pp. 3–10.
[18] W.G. Lehnert, B. Sundheim, A performance evaluation of text-analysis technologies, AI Mag. 12 (3) (1991) 81–94.
[19] N. Chinchor, L. Hirschman, D.D. Lewis, Evaluating message understanding systems: an analysis of the third message understanding conference, Comput. Linguist. 19 (3) (1993) 409–449.
[20] L. Hirschman, The evolution of evaluation: lessons from the Message Understanding Conferences, Comput. Speech Lang. 12 (4) (1998) 281–305.
[21] A. Lavelli, M.E. Califf, F. Ciravegna, D. Freitag, C. Giuliano, N. Kushmerick, L. Romano, A critical survey of the methodology for IE evaluation, in: Proceedings of the LREC, 2004, pp. 1655–1658.
[22] N. Ireson, F. Ciravegna, M.E. Califf, D. Freitag, N. Kushmerick, A. Lavelli, Evaluating machine learning for information extraction, in: Proceedings of the International Conference on Machine Learning, 2005, pp. 345–352.
[23] A. Lavelli, M.E. Califf, F. Ciravegna, D. Freitag, N. Kushmerick, C. Giuliano, L. Romano, N. Ireson, Evaluation of machine learning-based information extraction algorithms: criticisms and recommendations, Lang. Res. Eval. 42 (4) (2008) 361–393.
[24] D. Freitag, Machine learning for information extraction in informal domains, Mach. Learn. 39 (2/3) (2000) 169–202.
[25] M.E. Califf, R.J. Mooney, Bottom-up relational learning of pattern matching rules for information extraction, J. Mach. Learn. Res. 4 (2003) 177–210.
[26] A. Arasu, H. Garcia-Molina, Extracting structured data from web pages, in: Proceedings of the SIGMOD Conference, 2003, pp. 337–348.
[27] R. Krishnamurthy, Y. Li, S. Raghavan, F. Reiss, S. Vaithyanathan, H. Zhu, SystemT: a system for declarative information extraction, SIGMOD Record 37 (4) (2008) 7–13.
[28] F.M. Suchanek, M. Sozio, G. Weikum, SOFIE: a self-organizing framework for information extraction, in: Proceedings of the WWW, 2009, pp. 631–640.
[29] D.J. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, fifth, Chapman and Hall/CRC, 2012.
[30] T. Fawcett, An introduction to ROC analysis, Patt. Recognit. Lett. 27 (8) (2006) 861–874.
[31] H.A. Sleiman, R. Corchuelo, Trinity: on using trinary trees for unsupervised web data extraction, IEEE Trans. Knowl. Data Eng. 26 (6) (2014) 1544–1556.
[32] A. Sahuguet, F. Azavant, Building intelligent web applications using lightweight wrappers, Data Knowl. Eng. 36 (3) (2001) 283–316.
[33] R. Baumgartner, O. Frölich, G. Gottlob, The Lixto systems applications in business intelligence and the semantic Web, in: Proceedings of the ESWC, 2007, pp. 16–26.
[34] J. Raposo, A. Pan, M. Álvarez, J. Hidalgo, Ángel Viña, The Wargo system: semi-automatic wrapper generation in presence of complex data access modes, in: Proceedings of the DEXA Workshops, 2002, pp. 313–320.
[35] J.-T. Kim, D.I. Moldovan, Acquisition of linguistic patterns for knowledge-based information extraction, IEEE Trans. Knowl. Data Eng. 7 (5) (1995) 713–724.
[36] F. Gomez, C. Segami, Semantic interpretation and knowledge extraction, Knowl.Based Syst. 20 (1) (2007) 51–60.
[37] H.A. Sleiman, R. Corchuelo, TEX: an efficient and effective unsupervised web information extractor, Knowl.Based Syst. 39 (2013) 109–123.
[38] B. Chidlovskii, Wrapping web information providers by transducer induction, in: Proceedings of the ECML, 2001, pp. 61–72.
[39] P. Gulhane, A. Madaan, R.R. Mehta, J. Ramamirtham, R. Rastogi, S. Satpal, S.H. Sengamedu, A. Tengli, C. Tiwari, Web-scale information extraction with Vertex, in: Proceedings of the ICDE, 2011, pp. 1209–1220.
[40] L. Liu, C. Pu, W. Han, XWRAP: an XML-enabled wrapper construction system for web information sources, in: Proceedings of the ICDE, 2000, pp. 611–621.
[41] M. Álvarez, A. Pan, J. Raposo, F. Bellas, F. Cacheda, Finding and extracting data records from web pages, Signal Process. Syst. 59 (1) (2010) 123–137.
[42] S. Miller, H. Fox, L.A. Ramshaw, R.M. Weischedel, A novel use of statistical parsing to extract information from text, in: Proceedings of the ANLP, 2000, pp. 226–233.
[43] E. Riloff, Automatically generating extraction patterns from untagged text, in: AAAI/IAAI, vol. 2, 1996, pp. 1044–1049.
[44] C.-N. Hsu, M.-T. Dung, Generating finite-state transducers for semi-structured data extraction from the Web, Inf. Syst. 23 (8) (1998) 521–538.
[45] M. Skounakis, M. Craven, S. Ray, Hierarchical hidden Markov models for information extraction, in: Proceedings of the IJCAI, 2003, pp. 427–433.
[46] C. Bădică, A. Bădică, E. Popescu, A. Abraham, L-Wrappers: concepts, properties and construction, Soft Comput. 11 (8) (2007) 753–772.
[47] K. Seymore, A. McCallum, R. Rosenfeld, Learning hidden Markov model structure for information extraction, in: Proceedings of the AAAI, 1999, pp. 37–42.
[48] A.W. Hogue, D.R. Karger, Thresher: automating the unwrapping of semantic content from the World Wide Web, in: Proceedings of the WWW, 2005, pp. 86–95.
[49] C. Zhang, W. Xu, Z. Ma, S. Gao, Q. Li, J. Guo, Construction of semantic bootstrapping models for relation extraction, Knowl.Based Syst. 83 (2015) 128–137.
[50] B. Liu, Y. Zhai, NET: a system for extracting web data from flat and nested data records, in: Proceedings of the WISE, 2005, pp. 487–495.
[51] J. Park, D. Barbosa, Adaptive record extraction from web pages, in: Proceedings of the WWW, 2007, pp. 1335–1336.
[52] I. Muslea, S. Minton, C.A. Knoblock, Hierarchical wrapper induction for semistructured information sources, Auton. Agents Multi-Agent Syst. 4 (1/2) (2001) 93–114.
[53] U. Irmak, T. Suel, Interactive wrapper generation with minimal user effort, in: Proceedings of the WWW, 2006, pp. 553–563.
[54] L. Peshkin, A. Pfeffer, Bayesian information extraction network, in: Proceedings of the IJCAI, 2003, pp. 421–426.

[55] S. Soderland, Learning information extraction rules for semi-structured and free text, Mach. Learn. 34 (1–3) (1999) 233–272.
[56] M. Kayed, C.-H. Chang, FiVaTech: page-level web data extraction from template pages, IEEE Trans. Knowl. Data Eng. 22 (2) (2010) 249–263.
[57] R. Kosala, H. Blockeel, M. Bruynooghe, J.V. den Bussche, Information extraction from structured documents using *k*-testable tree automaton inference, Data Knowl. Eng. 58 (2) (2006) 129–158.
[58] H.L. Chieu, H.T. Ng, A maximum entropy approach to information extraction from semi-structured and free text, in: Proceedings of the AAAI/IAAI, 2002, pp. 786–791.
[59] A. Finn, N. Kushmerick, Information extraction by convergent boundary classification, in: Proceedings of the AAAI, 2004, pp. 1–6.
[60] D. Roth, W. tau Yih, Relational learning via propositional algorithms: an information extraction case study, in: Proceedings of the IJCAI, 2001, pp. 1257–1263.
[61] C. Cox, J. Nicolson, J.R. Finkel, C. Manning, P. Langley, Template sampling for leveraging domain knowledge in information extraction, in: Proceedings of the PASCAL Challenges Workshop, 2005, pp. 1–4.
[62] H.L. Chieu, H.T. Ng, Y.K. Lee, Closing the gap: learning-based information extraction rivaling knowledge-engineering methods, in: Proceedings of the ACL, 2003, pp. 216–223.
[63] M. Sokolova, G. Lapalme, A systematic analysis of performance measures for classification tasks, Inf. Process. Manage. 45 (4) (2009) 427–437.
[64] G. Forman, A pitfall and solution in multi-class feature selection for text classification, in: Proceedings of the ICML, 2004, pp. 38–46.
[65] D.J. Hand, R.J. Till, A simple generalisation of the area under the ROC curve for multiple class classification problems, Mach. Learn. 45 (2) (2001) 171–186.
[66] P. Jiménez, H.A. Sleiman, R. Corchuelo, Feeding software agents with web information, in: Proceedings of the PAAMS, 2015, pp. 135–142.
[67] H.A. Sleiman, R. Corchuelo, A class of neural-network-based transducers for web information extraction, Neurocomputing 135 (2014) 61–68.
[68] H.A. Sleiman, R. Corchuelo, A reference architecture to devise web information extractors, in: Proceedings of the CAiSE (Workshops), 2012, pp. 235–248.
[69] C. Ferri, J. Hernández-Orallo, R. Modroiu, An experimental comparison of performance measures for classification, Patt. Recognit. Lett. 30 (1) (2009) 27–38.
[70] T. Landgrebe, R.P.W. Duin, Approximating the multiclass ROC by pairwise analysis, Patt. Recognit. Lett. 28 (13) (2007) 1747–1758.
[71] W.I. Gasarch, Classifying problems into complexity classes, Adv. Comput. 95 (2015) 239–292.