On Learning Web Information Extraction Rules with TANGO

Patricia Jiménez*

Universidad de Sevilla, ETSI Informática.
Avda. de la Reina Mercedes, s/n. Sevilla E-41012, Spain.

Rafael Corchuelo

Universidad de Sevilla, ETSI Informática.
Avda. de la Reina Mercedes, s/n. Sevilla E-41012, Spain.

Abstract

The research on Enterprise Systems Integration focuses on proposals to support business processes by re-using existing systems. Wrappers help re-use web applications that provide a user interface only. They emulate a human user who interacts with them and extracts the information of interest in a structured format. In this article, we present TANGO, which is our proposal to learn rules to extract information from semi-structured web documents with high precision and recall, which is a must in the context of Enterprise Systems Integration. It relies on an open catalogue of features that helps map the input documents into a knowledge base in which every DOM node is represented by means of HTML, DOM, CSS, relational, and user-defined features. Then a procedure with many variation points is used to learn extraction rules from that knowledge base; the variation points include heuristics that range from how to select a condition to how to simplify the resulting rules. We also provide a systematic method to help re-configure our proposal. Our exhaustive experimentation proves that it beats others regarding effectiveness and is efficient enough for practical purposes. Our proposal was devised to be as configurable as possible, which helps adapt it to particular web sites and evolve it when necessary.

Keywords: Web information extraction; semi-structured documents; open catalogues of features; learning rules; variation points; configuration method.

*Corresponding author
Email addresses: patriciajimenez@us.es (Patricia Jiménez), corchu@us.es (Rafael Corchuelo)

1. Introduction

Enterprise Systems Integration is an engineering discipline that deals with technologies and methods that help engineers devise solutions to support business processes by re-using existing enterprise systems [12]. The problem is challenging insofar web applications are concerned because they are typically designed to be used by people, not by software agents. In many cases, they have to be integrated by emulating the interactions of a human user and then extracting the information of interest from the resulting web documents. Such interactions are performed by means of so-called wrappers, which are typically composed of the following modules: a form filler, which maps queries onto search forms, a navigator, which navigates from the search forms to the appropriate documents, an information extractor, which extracts the information of interest from the previous documents, and an information verifier, which checks the information extracted. Enterprise Systems Integration requires wrappers that can achieve high precision and recall, chiefly when the business processes that they support are mission-critical; the time to learn the rules or to execute them is not an issue as long as it falls within reasonable limits.

Our research focuses on information extractors. The literature provides some proposals to extract information from unstructured documents [33], that is, documents that are written in natural language, and others to deal with semi-structured documents [6], that is, HTML-based documents in which the information is rendered in tables, lists, forms, and similar regular formats. They can be further classified into heuristic-based information extractors or rule-based information extractors. The former are unsupervised proposals that rely on general heuristics that have proven to work well in many cases; the latter require specific-purpose rules that are learnt from a learning set supervisedly or unsupervisedly, depending on whether the user has to annotate the learning sets with the information to be extracted or not. A number of authors have worked on orthogonal topics, e.g., finding the regions of a document that provide interesting information [27], finding named entities from a small sample [14, 35], or re-learning the rules with as little effort as possible [36]. Lately, the problem of extracting information from unknown sites is getting much attention, but the only conclusive results are regarding unstructured documents [10, 22].

We focus on supervised rule-based proposals to learn rules to extract information from semi-structured web documents. We have surveyed many proposals in this field [3–5, 7, 8, 11, 13, 16–18, 20, 21, 23, 28, 30, 32]. Our conclusion is that the authors have tried to make them very general so that they can be applied to documents from as many web sites as possible. The problem is that most of them are monolithic solutions, that is, there is a tight inter-dependency between the features that are computed from the input documents and the learning procedure, which was not designed to be re-configured. This makes it difficult to evolve them independently so that they can be adapted to particular web sites or to keep with the evolution of HTML. It is not surprising then that web information extraction is an active research field in which new proposals sprout out continuously; unfortunately, they tend to fade away quickly because

they are not flexible enough.

In our opinion, the research in this field must focus on flexible proposals, which implies that they have to use an open catalogue of features, a learning procedure with variation points, and there must be a method to help re-configure them so that they can be adapted when necessary. The catalogue being open means that the learning procedure does not depend on any particular features, which implies that they can be included, removed, or replaced very easily in order to experiment with them. This allows to adapt a proposal to a particular web site or to changes in the way that HTML is used. A learning procedure with variation points relies on a number of heuristics whose implementation is plugged into the procedure; this makes it very flexible and facilitates adapting the proposal when necessary or trying new alternatives if one thinks that they can result in better performance. Note that adapting a proposal requires to make multi-criteria decisions, which is not intuitive at all; therefore, it is also necessary to have a method to help re-configure it when necessary.

None of the proposals that we have surveyed fulfils the previous requirements. A few authors have explored the idea of using an open catalogue of features [3, 11, 13, 18], but they did not identify the variation points of their learning procedures; consequently, none of them produced a method to help re-configure their proposals. In this article, we present TANGO, which is a rule learner that specialises in web information extraction; it relies on an open catalogue of features and a learning procedure in which we have identified eight variation points that implement eleven heuristics; furthermore, we provide a method to help re-configure it. Our detailed conceptual comparison reveals that our proposal clearly diverges from the existing ones and our exhaustive experimental study confirms that it is more effective than other state-of-the-art proposals and that it is efficient enough for practical purposes. In our experiments, we have been able to learn very good rules from as few as six randomly-selected documents.

The rest of the article is organised as follows: Section 2 presents a detailed description of our proposal; Section 3 reports on how we have configured it; Section 4 presents the results of our experimental analysis; Section 5 analyses the related work; finally, Section 6 concludes our work.

2. Description of our proposal

In this section, we describe our proposal. We first present some definitions and then delve into our catalogue of features, our learning procedure, our variation points, and our configuration method.

2.1. Definitions

Definition 1 (Documents). *We use the standard notation to represent variables, sets, and logical formulae. We denote sequences as $\langle x_1, x_2, \ldots, x_n \rangle$; given a sequence s, we denote its length as $|s|$; given two sequences s_1 and s_2, we denote their concatenation as $s_1 \oplus s_2$.*

Definition 2 (Documents). *A document is a character string that adheres to the HTML syntax and can then be represented as the root node of the corresponding DOM tree [15, 34].*

Definition 3 (Features). *Features are functions that map nodes onto values or other nodes. The former are referred to as attributive features, and they can be based on HTML attributes [15], DOM attributes [34], CSS attributes [2], or user-defined functions; the latter are relational features and they build on the usual relationships amongst the nodes of a DOM tree, e.g., parents or children.*

Definition 4 (Slots and annotations). *A slot is a label that provides a meaning to the information in a node. An annotation is a function that maps a subset of nodes onto a set of slots. We assume that the slots may be organised hierarchically so that there is a first-level slot that contains some nested slots. (Note that it is common to use term slot to refer to both a label and the nodes that are extracted with that label.)*

Definition 5 (Datasets). *A dataset is a ground first-order representation of an annotation and the instantiation of a catalogue of features on a set of documents. The datasets that are used to learn rules are referred to as learning sets and the datasets that are used to test rules are referred to as testing sets.*

Definition 6 (Rules and conditions). *A rule consists of a collection of conditions that characterise the nodes that provide the information to be extracted. We represent them as* $\langle h, b_1, b_2, \ldots, b_n \rangle$ $(n \geq 0)$*, where* h *denotes the head, which is a slot instantiator, and* b_i $(i = 1 \,..\, n)$ *denotes the body, which consists of feature instantiators, comparators, and/or further slot instantiators. A slot instantiator is a condition of the form* $s(N)$*, where* s *denotes a slot and* N *is a variable that can be bound to every node in the input documents. A feature instantiator is a condition that binds the value of a feature on a node to a constant or a variable; feature instantiators can be negated, in which case the condition is satisfied if the corresponding feature cannot be instantiated. A comparator is a condition that compares a variable to another variable or a constant using the usual relational operators. A condition is said to be determinate in the context of a rule if it is a feature instantiator, it can be instantiated exactly once on every positive example that is matched by the rule, at most once on every negative example, and does not return the same value on every example.*

Definition 7 (Scores and gains). *We require a rule scorer to assess how good a rule is. Intuitively, it must return high scores for rules that are close to be a solution and low scores for the others. Since our proposal learns rules by adding conditions incrementally, it is also necessary to compute the gain that adding a specific condition achieves. If* r *represents the current rule and* r' *represents the rule that results from adding condition* c *to* r*, that is,* $r' = r \oplus \langle c \rangle$*, then we compute the gain of condition* c *as* $p' \, (s' - s)$*, where* p' *denotes the number of positive examples matched by rule* r'*,* s' *is the score of rule* r'*, and* s *is the score of rule* r*. Realise that we weight the difference of scores with the number*

firstToken	countOfBlanks	width	isSibling
lastToken	countOfUppercaseBigrams	backgroundColour	left
parent	height	isLowercase	isURL
lastSibling	isUppercase	beginsWithPunctuation	countOfUppercaseTokens
xPos	isCurrency	isCapitalised	isEmail
yPos	countOfAlphaNum	fontSize	lineHeight
firstSibling	countOfLowercaseTrigrams	isDate	isBlank
tag	countOfChildren	isISBN	endsWithParenthesis
children	countOfSiblings	beginsWithParenthesis	hasCurrencySymbol
lastToken	isNumber	display	verticalAlign
penultimateToken	hasBracketedNumber	fontWeight	textAlign
siblingIndex	countOfLowercaseTokens	hasBracketedAlphaNum	borderBottomWidth
lastToken	countOfLetters	isYear	borderLeftWidth
beginsWithNumber	countOfCapitals	countOfLowercaseBigrams	hasQuestionMark
nextSibling	countOfDigits	marginBottom	isPhone
len	countOfUppercaseTrigrams	borderRightColour	borderBottomColour
isTextNode	textDecoration	endsWithPunctuation	countOfBigrams
depth	hasNotBlanks	isAlphaNum	countOfTrigrams
countOfTokens	countOfIntegers	isNumber	ancestor

Table 1: Partial catalogue of features. (Sorted by empirical frequency.)

of positive examples matched by r', which helps make a difference that rewards the conditions that match more positive examples.

2.2. Catalogue of features

TANGO works on an open catalogue of features, which means that the learning procedure does not require it to provide any specific features. On the contrary, the user can try different features or ways to compute them in order to configure our proposal so that it performs as well as possible.

By default, we provide a catalogue that has proven to work well in our experiments. It provides many features that compute the attributes that are defined in the HTML, DOM, and CSS recommendations [2, 15, 34]. It also provides a number user-defined features.

The HTML and the CSS features are attributive because they map the DOM nodes onto values that represent their HTML and CSS attributes; the DOM features can be either attributive, e.g., the depth of a node, or relational, e.g., the parent of a node. The user-defined features are typically used to implement simple semantic checks, e.g., whether a node contains a number or not.

Table 1 provides an excerpt of our current catalogue. The actual catalogue consists of roughly two hundred features, of which we have shown the most used according to our experimentation.

2.3. Learning procedure

The learning procedure works on a learning set that consists of a ground first-order representation of the input documents and their annotation. It then uses a top-down algorithm that learns a rule set for each slot. It starts with an overly-general rule that matches every node in the learning set and then extends it by adding conditions that constraint the subset of nodes that it matches; when

```
 1: method TANGO(documents, annotation)
 2:     – Step 1: initialisation.
 3:     result = ∅
 4:     dataset = create a dataset from documents and annotation
 5:     – Step 2: learn a rule set for every slot.
 6:     for each different slot in annotation do
 7:         learningSet = create a learning set for slot from dataset
 8:         learningSet = preProcessLearningSet(learningSet, slot)
 9:         ruleset = learnRuleSet(learningSet, slot)
10:         result = result ∪ {ruleSet}
11:     end
12: return result
```

Figure 1: Main procedure.

a rule that matches positive examples only is found, it is considered a solution; the positive examples that it matches are then removed from the learning set and the procedure is re-started until no positive example remains to be matched or it is not possible to find a rule, which is very unlikely in practice. Our proposal also manages a set of savepoints to which it can backtrack if the current search path is not good enough.

2.3.1. Main procedure

Figure 1 shows the main procedure, which works on a set of documents and an annotation; it returns a set of rule sets, each of which is specifically tailored to extracting information that belongs to a given slot.

The first step consists in initialising the result to an empty set and then creating a dataset from the input documents and their annotation. Basically, we have to loop through a user-provided catalogue of features and try to instantiate them on every node of the input documents.

The second step iterates through the set of slots used in the annotation of the input documents. For each slot, it first creates a learning set from the previous dataset, pre-processes it, and then invokes the procedure to learn a rule set; the result is stored in the result variable, which is returned when the loop finishes. Creating the learning set amounts to creating a new dataset in which the positive examples are the nodes that belong to the slot that is being analysed and the negative examples are the remaining nodes. In order to reduce the computational effort, the learning sets that correspond to nested slots have information about the nodes in the enclosing slots only. The resulting learning set must be simplified using a variation point called preProcessLearningSet so that learning is as efficient as possible.

Example 1. *Figure 2 shows a sample web document with a listing of phone codes; the countries for which the system does not have a code are starred. Table 2 shows a partial instantiation of our catalogue of features and Table 3 shows the corresponding annotation.*

	Attributive features								Relational features	
	HTML		DOM		Rendering		User-defined			
node	tag	style	depth	children	yPos	xPos	len	isNumber	parent	left
n_1	html		1	2	0	0	18	false		
n_2	head		2	1	0	0	5	false	n_1	
n_3	title		3	1	0	0	5	false	n_2	
n_4			4	0	0	0	5	false	n_3	
n_5	body		2	3	0	0	13	false	n_1	n_2
n_6	h1		3	1	0	0	2	false	n_5	
n_7			4	0	0	0	2	false	n_6	
n_8	ul	a	3	3	16	0	9	false	n_5	n_6
n_9	li		4	3	16	0	3	false	n_8	
n_{10}	span		5	1	16	0	1	false	n_9	
n_{11}			6	0	16	0	1	false	n_{10}	
n_{12}			5	0	16	32	1	false	n_9	n_{10}
n_{13}	span		5	1	16	36	1	true	n_9	n_{12}
n_{14}			6	0	16	36	1	true	n_{13}	
n_{15}	li		4	3	32	0	3	false	n_8	n_9
n_{16}			5	0	32	0	1	false	n_{15}	
n_{17}			5	0	32	20	1	false	n_{15}	n_{16}
n_{18}			5	0	32	40	1	false	n_{15}	n_{17}
n_{19}	li		4	3	48	0	3	false	n_8	n_{15}
n_{20}	span		5	1	48	0	1	false	n_{19}	
n_{21}			6	0	48	0	1	false	n_{20}	
n_{22}			5	0	48	32	1	false	n_{19}	n_{20}
n_{23}	span		5	1	48	36	1	true	n_{19}	n_{22}
n_{24}			6	0	48	36	1	true	n_{23}	
n_{25}	ul	b	3	1	64	0	2	false	n_5	n_8
n_{26}	li		4	2	64	0	2	false	n_{25}	
n_{27}	span		5	1	64	0	1	false	n_{26}	
n_{28}			6	0	64	0	1	false	n_{27}	
n_{29}	span		5	1	64	36	1	true	n_{26}	n_{27}
n_{30}			6	0	64	36	1	true	n_{29}	

Table 2: Sample feature instantiation.

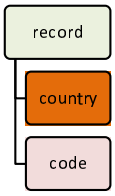
Structure of slots	Annotation		Annotation		Annotation	
	node	slot	node	slot	node	slot
	n_1		n_{11}	country	n_{21}	country
	n_2		n_{12}		n_{22}	
	n_3		n_{13}		n_{23}	
	n_4		n_{14}	code	n_{24}	code
	n_5		n_{15}	record	n_{25}	
	n_6		n_{16}		n_{26}	
	n_7		n_{17}	country	n_{27}	
	n_8		n_{18}		n_{28}	
	n_9	record	n_{19}	record	n_{29}	
	n_{10}		n_{20}		n_{30}	

Table 3: Sample annotation.

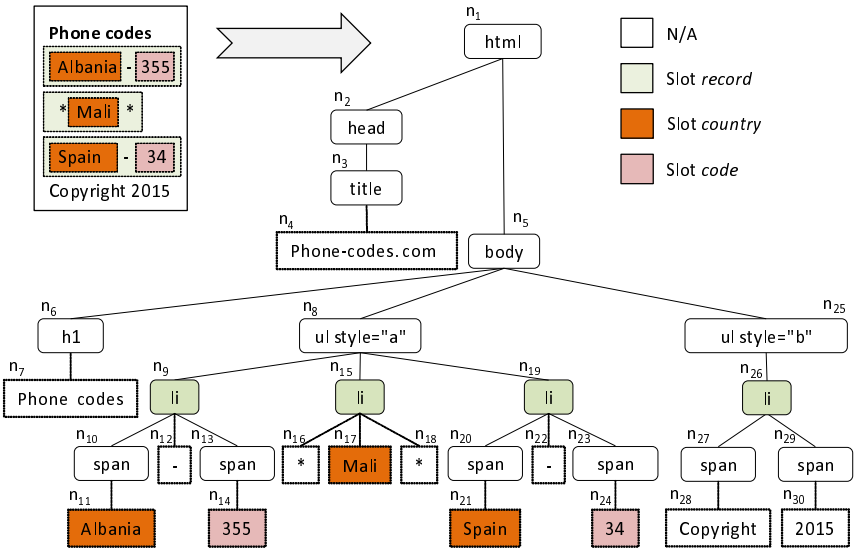


Figure 2: Sample web document.

Regarding slot record, TANGO first creates a learning set in which the nodes that belong to that slot are positive examples and the others are negative ones; the instantiation of the features remains the same. Regarding slots country and code, which are nested within the previous slot, TANGO creates similar learning sets, the difference being that features are instantiated only on the nodes that belong to a sub-tree that is known to be a record. Since this example is very simple, it is not actually necessary to perform any pre-processing.

From those learning sets, TANGO learns the following rule sets:

$$\{\langle record(N_0), tag(N_0, \text{'li'}), parent(N_0, N_1), style(N_1, \text{'a'})\rangle\},$$
$$\{\langle country(N_0), xPos(N_0, A_1), A_1 \leq 20, \neg left(\_, N_0)\rangle,$$
$$\langle country(N_0), xPos(N_0, 20)\rangle\},$$
$$\{\langle code(N_0), \neg tag(N_0, \_), isNumber(N_0)\rangle\}$$

2.3.2. Learning a rule set

Figure 3 presents the procedure to learn a rule set, which works on a learning set and a slot; it returns a set of rules that are specifically tailored to extracting information that belongs to that slot.

As usual, the first step is an initialisation step that simply sets the resulting rule set to the empty set.

The second step is a loop that iterates until no new rule is found or no positive example remains in the learning set. In each iteration, the procedure to learn a rule is invoked using the current learning set and the input slot as parameters. If this procedure returns *null*, then it means that it has not been able to find a rule that matches the positive examples in the learning set; otherwise, it

```
 1: method learnRuleSet(learningSet, slot)
 2:     – Step 1: initialisation.
 3:     ruleSet = ∅
 4:     – Step 2: learn rules.
 5:     repeat
 6:         rule = learnRule(learningSet, slot)
 7:         if rule ≠ null then
 8:             ruleSet = ruleSet ∪ {rule}
 9:             learningSet = learningSet \ (positive examples matched by rule)
10:         end
11:     until rule = null ∨ there are no positive examples in learningSet
12:     – Step 3: post-process the rule set.
13:     ruleSet = postProcessRuleSet(ruleSet)
14: return ruleSet
```

Figure 3: Procedure to learn a rule set.

returns a rule that matches some positive examples and no negative one, that is, a solution. If a rule is returned, then the resulting rule set is updated, the learning set is subtracted the positive examples that are matched by that rule, and the procedure is re-started if the learning set is not empty.

The third step relies on variation point postProcessRuleSet, which simplifies the resulting rule set so that it can be executed as efficiently as possible.

Example 2. *Let us focus on slot country. In the first iteration of Step 2, it learns a rule that matches countries with a phone code, namely:*

$$\langle country(N_0), depth(N_0, A_2), children(N_0, A_3), yPos(N_0, A_4), xPos(N_0, A_5), \\ A_5 \leq 20, \neg left(N_4, N_0)\rangle$$

The procedure then removes the positive examples matched by the previous rule from the learning set and invokes the procedure to learn a rule again. Now, it returns a rule that matches starred countries, namely:

$$\langle country(N_0), depth(N_0, A_2), children(N_0, A_3), yPos(N_0, A_4), xPos(N_0, A_5), \\ A_5 = 20\rangle$$

Since the previous rules match every positive example, they are then post-processed, which results in the following simplified rule set:

$$\{\langle country(N_0), xPos(N_0, A_1), A_1 \leq 20, \neg left(\_, N_0)\rangle, \\ \langle country(N_0), xPos(N_0, 20)\rangle\}$$

2.3.3. Learning a rule

Figure 4 presents the procedure to learn a rule, which works on a learning set and a slot. It returns a solution if possible, that is, a rule that matches

```
 1: method learnRule(learningSet, slot)
 2:     – Step 1: initialisation.
 3:     savepoints = ∅
 4:     var = generate a fresh variable
 5:     rule = ⟨slot(var)⟩
 6:     score = RULESCORER(rule, learningSet)
 7:     – Step 2: extend the current rule.
 8:     repeat
 9:         – Step 2.1: compute and select candidates.
10:         candidates = computeCandidates(rule, score, learningSet)
11:         (bestCandidates, saveCandidates) =
12:                 SELECTCANDIDATES(rule, score, candidates, learningSet)
13:         – Step 2.2: update savepoints and current rule.
14:         savepoints = updateSavepoints(savepoints, rule, saveCandidates, learningSet)
15:         rule = rule ⊕ ⟨conditions in bestCandidates⟩
16:         score = RULESCORER(rule, learningSet)
17:         – Step 2.3: check for a replacement.
18:         if bestCandidates = ∅ ∨ ISTOOCOMPLEX(rule, score, learningSet) then
19:             (rule, savepoints) = findBestSavepoint(savepoints)
20:         end
21:     until rule = null ∨ isSolution(rule, learningSet)
22:     – Step 3: check for better savepoints.
23:     if rule ≠ null then
24:         rule = findBetterSavepoint(savepoints, rule, score, learningSet)
25:     end
26: return rule
```

Figure 4: Procedure to learn a rule.

at least a positive example, but no negative one; in cases in which a solution cannot be found, it returns a *null* value.

The first step consists in initialising a set of savepoints and the rule that is going to be learnt. The savepoints are initialised to an empty set; during the learning process, this set stores some promising rules that might be used to backtrack if the current search path is not good enough. The rule is initialised to $\langle slot(var) \rangle$, where *slot* denotes the slot for which the procedure is learning a rule and *var* denotes a fresh variable; in our examples we use N_0 to denote that variable. This rule trivially matches every example in the learning set because no conditions have been added to its body yet. Note that we need to compute a score to assess how good the rule is; since this computation can be accomplished in a variety of different ways, we have implemented this procedure as a variation point called RULESCORER.

The second step is a loop that extends the initial rule with new conditions and updates the savepoints. It consists of three sub-steps. The first one computes a set of candidates, which are tuples of the form (c, g, d), where c represents a condition that might possibly be added to the current rule, g the gain that it would achieve, and d is a Boolean that indicates if c is determinate. It then selects a subset of them to extend the current rule and another subset to update the savepoints, which is implemented using a variation point called SELECTCANDIDATES because there are several alternatives available. The second sub-step first calls a procedure to update the savepoints, then extends the current rule, and finally re-computes its score. The third sub-step checks for a replacement of the current rule, which is a savepoint to which the procedure can backtrack in cases in which no candidate is selected to extend the current rule or cases in which it is too complex. Note that the learning process might explore arbitrarily complex rules, which does not make sense in practice. This calls for a mechanism to check whether a rule is complex enough not to explore it. We have implemented it using a variation point called ISTOOCOMPLEX because there are several choices available.

The third step attempts to substitute the rule that has been learnt in the previous step by a better savepoint. To understand this intricacy, assume that $r_1 = \langle h, c_1 \rangle$ is the current rule and that it can be extended as $r_2 = \langle h, c_1, c_2 \rangle$ or $r_3 = \langle h, c_1, c_3 \rangle$. Assume, too, that r_2 matches 10 positive examples and no negative one, whereas r_3 matches 30 positive examples and one negative example. Even though rule r_2 is a solution, rule r_3 might be considered to provide more gain because it matches more positive examples and thus might lead to a smaller rule set. In such cases, the search should stick with r_3 and r_2 should be kept as a savepoint. The problem is that rule r_3 might lead to a solution that matches less than 10 positive examples in the following iterations; in such cases, it does not make sense to return rule r_3, but rule r_2.

Example 3. *Let us focus on slot country. Regarding the variation points, we use Information Content as the heuristic to implement the rule scorer; to extend the current rule, we select the candidate that achieves the maximum gain only if no candidate has at least 80% the maximum possible gain and there are no*

determinate candidates; to create new savepoints, we select the best candidate that results in a solution, if any, or the candidates whose gain is at least 80% the gain of the best condition that has been added to the rule; furthermore, we only keep two savepoints in this example.

Our procedure starts working on the following initial rule:

$$\langle country(N_0) \rangle$$

It then generates a number of candidates that can possibly be used to extend it, namely:

$$\{(\neg tag(N_0, A_1), 1.59, false), (depth(N_0, A_2), 0.00, true),\\ (children(N_0, A_3), 0.00, true), (yPos(N_0, A_4), 0.00, true),\\ (xPos(N_0, A_5), 0.00, true)\}$$

Since the current rule scores at -2.12*, the maximum gain that a condition can achieve on it is* 6.35*, which happens when that condition preserves the positive examples matched by the current rule but does not match any negative examples. None of the previous candidates achieves at least 80% the maximum gain, so we have to extend the rule using the candidates that are determinate conditions, namely:*

$$\langle country(N_0), depth(N_0, A_2), children(N_0, A_3), yPos(N_0, A_4), xPos(N_0, A_5) \rangle$$

Since the rule was extended using determinate conditions and the result is not a solution, then none of the previous conditions can be used to update the set of savepoints, which remains empty. Branching the current rule results in the following candidates:

$$\{(\neg tag(N_0, A_1), 1.59, false), (\neg isNumber(N_0), 1.59, false),\\ (parent(N_0, N_1), 2.23, false), (A_2 \neq 5, 2.23, false),\\ (A_2 > 5, 2.23, false), (A_2 = 6, 2.23, false), (A_2 \geq 6, 2.23, false),\\ (A_5 \leq 20, 3.35, false), (A_5 < 32, 3.35, false)\}$$

None of them achieves the minimum gain required to be selected, but there are not any determinate conditions, which means that we have to select the one that provides the maximum gain, that is, $A_5 \leq 20$ *or* $A_5 < 32$*. Since there is a tie, we break it arbitrarily and select the first condition, namely:*

$$\langle country(N_0), depth(N_0, A_2), children(N_0, A_3), yPos(N_0, A_4), xPos(N_0, A_5),\\ A_5 \leq 20 \rangle$$

This rule scores at -1.32 *and the maximum possible gain that a condition may achieve on it is* 2.64*, which implies that a condition must achieve a gain of at least* 2.11 *so that it can be selected to create a savepoint. This implies that any of the candidates might be selected to update the savepoints, except for* $\neg tag(N_0, A_1)$ *and* $\neg isNumber(N_0)$*. Since we are keeping only two savepoints, the set gets updated as follows:*

```
1: method computeCandidates(rule, score, learningSet)
2:     – Step 1: branch the rule.
3:     conditions = BRANCH(rule, learningSet)
4:     – Step 2: bound candidate conditions.
5:     candidates = ∅
6:     stop = false
7:     for each condition in conditions while ¬stop do
8:         newRule = rule ⊕ ⟨condition⟩
9:         newScore = RULESCORER(newRule, learningSet)
10:        candidate = BOUND(rule, score, newRule, newScore)
11:        if candidate ≠ null then
12:            candidates = candidates ∪ {candidate}
13:            stop = ISPROMISINGCANDIDATE(rule, score, newRule, newScore)
14:        end
15:    end
16: return candidates
```

Figure 5: Procedure to compute candidates.

$$\{(\langle country(N_0), depth(N_0, A_2), children(N_0, A_3), yPos(N_0, A_4), xPos(N_0, A_5), parent(N_0, N_1)\rangle, -1.00),$$
$$(\langle country(N_0), depth(N_0, A_2), children(N_0, A_3), yPos(N_0, A_4), xPos(N_0, A_5), A_5 < 32\rangle, -1.32)\}$$

The savepoints are represented as tuples of the form (r, s), where r denotes a rule and s denotes its corresponding score. Branching the current rule results in the following candidates:

$$\{(\neg tag(N_0, A_1), 1.47, false), (\neg parent(N_2, N_0), 1.47, false),$$
$$(\neg left(N_4, N_0), 2.64, false)\}$$

Condition $\neg left(N_4, N_0)$ has the maximum possible gain, which is 2.64. That means that adding this condition to the current rule results in a solution, namely:

$$\langle country(N_0), depth(N_0, A_2), children(N_0, A_3), yPos(N_0, A_4), xPos(N_0, A_5), A_5 \leq 20, \neg left(N_4, N_0)\rangle$$

2.3.4. Computing candidates

Figure 5 presents the procedure to compute the candidates that can possibly be used to extend a rule or to create new savepoints. It works on the current rule, its score, and a learning set; it returns a set of candidates. The candidates are represented as tuples of the form (c, g, d), where c denotes a condition, g the gain that is achieved when condition c is added to the input rule, and d is a Boolean value that indicates if c is determinate.

The first step branches the input rule, which consists in generating a sequence of conditions that can be used to extend it. There can be many approaches to generating such conditions, which justifies using a variation point called BRANCH.

The second step is a loop that bounds the conditions that have been generated in the previous step. Bounding a condition means that we assess its gain and decide on whether it is bad enough to prune it. Since there are several choices, we have implemented the bounding procedure using a variation point called BOUND. The search can be stopped at any moment, when a promising condition is found, that is, when a condition is considered so good that it is not necessary to continue exploring the others. Since there are also several alternatives to implement this stopping criterion and it is not clear which one is the best one, we have implemented it using a variation point called ISPROMISINGCANDIDATE.

Example 4. *Assume that we have to branch and then bound the following initial rule:*

$\langle country(N_0) \rangle$

A simple approach to branching consists in generating every possible feature instantiator regarding node N_0, which would result in the following candidates, where the "?" means that the gain cannot be computed because the corresponding score is indeterminate:

$$
\begin{array}{ll}
\{(tag(N_0, A_1), ?, false), & (\neg tag(N_0, A_1), 1.59, false) \\
\ (depth(N_0, A_2), 0.00, true), & (\neg depth(N_0, A_2), ?, false) \\
\ (children(N_0, A_3), 0.00, true), & (\neg children(N_0, A_3), ?, false) \\
\ (yPos(N_0, A_4), 0.00, true), & (\neg yPos(N_0, A_4), ?, false) \\
\ (xPos(N_0, A_5), 0.00, true), & (\neg xPos(N_0, A_5), ?, false) \\
\ (len(N_0, A_6), 0.00, false), & (\neg len(N_0, A_6), ?, false) \\
\ (isNumber(N_0), ?, false), & (\neg isNumber(N_0), 1.59, false) \\
\ (parent(N_0, N_1), 2.23, false), & (\neg parent(N_0, N_1), -1.05, false) \\
\ (parent(N_2, N_0), ?, false), & (\neg parent(N_2, N_0), 1.59, false) \\
\ (left(N_0, N_3), -0.47, false), & (\neg left(N_0, N_3), 0.62, false) \\
\ (left(N_4, N_0), -0.47, false), & (\neg left(N_4, N_0), 0.62, false)\}
\end{array}
$$

To bound the candidates, we first examine condition $tag(N_0, A_1)$, which is pruned because its gain is indeterminate; we then examine $\neg tag(N_0, A_1)$, whose gain is 1.59*; this is the best condition found so far, so we set the pruning threshold to* 80% *that gain, that is,* 1.27*; this means that the following conditions shall be pruned unless they can achieve this minimum gain or they are determinate. Then, condition $depth(N_0, A_2)$ is examined; note that it does not achieve any gain at all, but it is determinate; this means that we have already found a condition that can expand the search space and possibly lead to a better rule. The maximum gain that a condition can achieve on the current rule is* 6.35*, so we can set the new pruning threshold to* 80% *this maximum, that is,* 5.08 *instead of* 1.27*.*

2.3.5. Managing savepoints

We have devised three procedures that help manage the set of savepoints, namely: *updateSavepoints*, to update them, *findBestSavepoint*, to find the best

```
1: method updateSavepoints(savepoints, rule, candidates, learningSet)
2:     for each candidate (c, s, d) in candidates do
3:         – Step 1: extend the current rule.
4:         newRule = rule ⊕ ⟨c⟩
5:         newScore = ruleScorer(newRule, learningSet)
6:         – Step 2: find a savepoint to replace.
7:         (r, s) = (null, −∞)
8:         if isSolution(newRule, learningSet) then
9:             (r, s) = find a savepoint (r, s) in savepoints such that
10:                        isSolution(r, learningSet)
11:        end
12:        if r = null ∧ |savepoints| = k then
13:            (r, s) = find savepoint (r, s) in savepoints such that
14:                        s is the minimum score
15:        end
16:        – Step 3: update the set of savepoints.
17:        if r ≠ null ∧ newScore > s then
18:            savepoints = savepoints \ {(r, s)} ∪ {(newRule, newScore)}
19:        elsif r = null then
20:            savepoints = savepoints ∪ {(newRule, newScore)}
21:        end
22:    end
23: return savepoints
```

Figure 6: Procedure to update the savepoints.

```
1: method findBestSavepoint(savepoints)
2:     (r, s) = select a savepoint (r, s) with maximum score from savepoints
3:     sp = savepoints \ {(r, s)}
4: return (r, sp)
```

Figure 7: Procedure to find the best savepoint.

```
1: method findBetterSavepoint(savepoints, rule, score, learningSet)
2:     (r, s) = select a savepoint (r, s) from savepoints such that
3:                 isSolution(r, learningSet)
4:     if r ≠ null ∧ s > score then
5:         rule = r
6:     end
7: return rule
```

Figure 8: Procedure to find a better savepoint.

one, if any, and *findBetterSavepoint*, to find one that is better than a given solution, if any. Next, we provide additional explanations on each procedure.

Figure 6 shows the procedure to update the savepoints. It works on the current set of savepoints, the current rule (before it is extended with the best candidates), the set of candidates that have been selected to update the savepoints, and the learning set. It returns an updated set of savepoints that fulfils two properties, namely: a) it has at most k savepoints, where k is a user-defined parameter; b) and there is at most one savepoint that is a solution (note that if TANGO backtracks to such a savepoint, then it returns it immediately; thus, it suffices to keep one savepoint that is a solution).

The procedure iterates over the set of candidates and proceeds in three steps. In the first step, it creates a new rule by adding the condition in the current candidate and computes its score. In the second step, it searches for a savepoint to replace, namely: if the new rule is a solution, then it searches for the only savepoint that is a solution, if any; if it is not a solution, then it retrieves the savepoint with the minimum score if the set of savepoints is full (since, otherwise, there is no need to replace any savepoint). We assume that if the search for a savepoint fails, then the rule returned is *null* and the corresponding score is $-\infty$. The third step updates the savepoints as follows: if there is a savepoint to replace and its score is smaller than the score of the new rule, then it is replaced; otherwise, if there is not a savepoint to replace, the new rule is added to the set of savepoints. This guarantees that there are no more than k savepoints, of which only one can be a solution.

Figure 7 shows the procedure to find the best savepoint. It works on the current set of savepoints and returns a tuple of the form (r, sp), where r denotes the rule associated with the savepoint that has the maximum score or *null* if no such savepoint exists, and sp denotes the updated set of savepoints.

Figure 8 shows the procedure to find a better savepoint, as long as it is a solution. It works on the current set of savepoints, a rule that is a solution,

its corresponding score, and the learning set. It first searches for the savepoint that is a solution; if it exists, then it replaces the current rule by the rule that is associated with that savepoint if it scores better; otherwise, it returns the input rule.

2.4. Variation points

Our learning procedure relies on a number of variation points, each of which groups one or more heuristics that can be implemented using several alternatives. We describe them in the following subsections.

2.4.1. Variation point VP1: selectCandidates

The goal of this variation point is twofold: select the most promising candidates to expand the current rule and some of the remaining ones to create savepoints. This should help TANGO learn very effective extraction rules whose efficiency does not degrade significantly when they are executed.

Heuristic H1: select best candidates. We have considered the following alternatives: A0) Select the candidate that provides the maximum gain to extend the current rule and then select the following $k = 20$ candidates to create savepoints. A1) Select the candidate with the maximum gain as long as it is at least 80% the maximum gain that a condition can achieve on the current rule; else, select every determinate condition; if no such condition is a candidate, then select the one with the highest gain. To create the savepoints, we select the candidate with the highest gain out of the candidates that result in a solution, if any; if it corresponds to a non-determinate condition, then we also select the candidates whose gain is at least 80% the gain of that condition; otherwise, no more candidates are selected.

The first alternative is a simple approach that has been used many times in the literature. Although it is very simple and might work well in some cases, our intuition was that there would be cases in which the candidates selected would lead to local maxima and would not help learn good rules. The rationale behind the second alternative is that the candidate with the maximum gain can be added to the current rule as long as its gain is high enough with regard to the maximum gain that a condition may achieve; otherwise, it is better to add determinate conditions because they help expand the search space; if no determinate conditions exists, then we have to resort to the condition that provides the maximum gain, like in Alternative A0. Note that there might be still cases in which we need to perform backtracking. Thus, we think that it makes more sense to select a candidate that results in a solution, unless there is another candidate that results in more gain, in which case the former might be kept as a savepoint, if possible. Now, if the candidates selected to extend the current rule are not determinate, we then select the candidates that achieve a gain that is high enough in comparison with that candidate; otherwise, if the selected candidates were determinate, no more candidates are selected to create savepoints. The rationale behind this idea is that if determinate conditions are selected, it then means that there were no conditions whose gain exceeded

80% the maximum gain since, otherwise, they would have been selected instead of determinate conditions. Thus, in this case, we are not interested in using conditions that do not provide enough gain to create savepoints. If a non-determinate condition is selected to expand the rule, we can be less demanding regarding the candidates that should be selected to create new savepoints so that we shall only select the k best candidates as long as their gain exceeds 80% the gain of the non-determinate condition selected to extend the current rule. The reason why we selected $k = 20$ candidates and 80% the maximum gain is purely experimental, but we cannot report on all of our results due to space constraints; thus, we decided to report on the best ones only.

2.4.2. Variation point VP2: PREPROCESSLEARNINGSET

This variation point deals with simplifying a learning set so that the learning process is more efficient; the simplification might also have a positive impact on the effectiveness and the efficiency of the resulting rules.

Heuristic H2: reduce negative examples. We have considered the following alternatives: A0) Work with the whole learning set. A1) Select a subset of negative examples that are in the neighbourhood of every positive example. A2) Select a subset of negative examples that are in the neighbourhood of every positive example plus a random subset of the remaining negative examples. A3) Select a subset with the most similar negative examples that correspond to every positive example. A5) Select a subset with the most similar negative examples that correspond to every positive example plus a random subset of the remaining negative examples.

The first alternative is a simple approach in which every negative example is considered in the learning process. Our intuition was that this would not be efficient because there are typically many negative examples. Our hypothesis was that it would be possible to discard many such negative examples from the learning set without a negative impact on the effectiveness of the resulting rules, as long as the negative examples that are kept are still representative of the whole set of negative examples. The other alternatives were intended to find that subset. The rationale behind Alternative A1 is to discard the negative examples that are not in the neighbourhood of the nodes that correspond to the positive examples. By neighbourhood, we refer to the nodes that can be reached within a given radius by applying relational features transitively. We experimented with radius $r = 10$ when computing the neighbourhood of a given positive example. Alternative A2 is based on A1, but it includes a set of negative examples that are selected randomly; we set the radius to compute the neighbours to $r = 10$ and the percentage of nodes selected randomly to $p = 10\%$. Alternative A3 searches for the most similar negative examples and discards the remaining ones. We measured the similarity between any two nodes using the well-known Euclidean distance on the attributive features; in the case of non-numeric features, we computed the difference between two different values as 1.00 and the difference between equal values as 0.00. We experimented with the $k = 50$ most similar negative examples to each positive example. Alternative A4 explores the k most

similar negative examples for each positive example but it includes a small percentage of negative examples that are selected randomly; we selected the $k = 50$ most similar negative examples to each positive example and $p = 50\%$ of the remaining negative examples. We experimented with many different radiuses and percentages, but we cannot report on them all due to space constraints; this is the reason why we report on the best values only.

Heuristic H3: binarise features. We have considered the following alternatives: A0) Work with the original features. A1) Binarise them.

When TANGO selects a feature instantiator to extend the current rule, it makes a blind decision: thus far, that instantiator is the best condition that can be added to the rule, but the value of the feature is not constrained at all; if necessary, it can be constrained later using a comparator. In other words, constraining the value of a feature is a two-step procedure. Binarising features is a process by means of which a single step suffices to instantiate a feature and constraint its value. Obviously, only attributive features can be binarised because they are the only ones that provide values that can be constrained by means of comparators. The binarisation process works as follows: a discrete feature f that ranges over the set of values $\{v_1, v_2, \ldots, v_n\}$ is transformed into a collection of new features of the form $f_{v_1}, f_{v_2}, \ldots, f_{v_n}$; simply put, $f_{v_i}(N)$ is satisfied as long as $f(N, A) \wedge A = v_i$ is satisfied, where N ranges over the set of nodes, A ranges over the set of values of feature f, and i ranges in interval $1 \,..\, n$. A numeric feature f that ranges over the set of values $\{v_1, v_2, \ldots, v_n\}$ is transformed into a collection of new features of the form $f^{\theta}_{v_1}, f^{\theta}_{v_2}, \ldots, f^{\theta}_{v_n}$, where θ represents a comparison operator; simply put, $f^{\theta}_{v_i}(N)$ is satisfied as long as $f(N, A) \wedge A\, \theta\, v_i$ is satisfied, where N denotes a variable that ranges over the set of nodes, A is a variable that ranges over the set of values of feature f, θ is a comparison operator, and i ranges in interval $1 \,..\, n$.

2.4.3. Variation point VP3: isPromisingCandidate

This variation point deals with determining if a candidate is good enough to stop the search for new candidates. The goal is to configure TANGO so that it can learn rules more efficiently without degrading their effectiveness or increasing the time required to execute them.

Heuristic H4: check promising candidates. We have considered the following alternatives: A0) No condition is considered promising enough to stop the search unless it leads to a rule that matches all of the positive examples matched by the current rule and discards all of the negative examples that it matches. A1) Stop the search when the gain achieved by a condition is at least 80% the maximum possible gain on the current rule. A2) Like Alternative A1, but we also require the resulting rule to be a solution.

Alternative A0 is the safest one, since almost every condition is analysed and transformed into a candidate to extend the current rule or to create a savepoint. The only case in which not all of the candidates are evaluated is when the best solution is found. The rationale behind Alternative A1 is that Heuristic H1

selects a candidate whose gain is at least 80% the maximum possible gain on the current rule; thus, we might stop the search for candidates when such a candidate is found; the only problem would be that if that candidate proved not to be good enough in the forthcoming iterations, then we would miss some better candidates that would have been evaluated later in the same iteration. Alternative A2 is a bit more demanding since it requires the candidate on which the search stops to result in a solution.

2.4.4. Variation point VP4: RULESCORER

This variation point deals with assessing rules. The goal is to configure TANGO so that it learns extraction rules that are more effective and efficient.

Heuristic H5: compute scores. We have considered the following alternatives, where *tp*, *tn*, *fp*, and *fn* denote the components of a confusion matrix and $N = tp+fp+tn+fn$: A0) Information Content, which computes the score as $\log \frac{tp}{tp+fp}$. A1) Accuracy-based Information Content, which computes it as $\log \frac{tp+tn}{tp+fp+fn+tn}$. A2) Satisfaction, which computes it as $\frac{\frac{tp}{tp+fp} - \frac{tp+fn}{N}}{1-\frac{tp+fn}{N}}$. A3) Laplace Estimate, which computes it as $\frac{tp+1}{tp+fp+2}$. A4) Piatetski-Shapiro's measure, which computes it as $\frac{tp\ tn-fp\ fn}{N^2}$.

Alternative A0 computes the score as the logarithm of precision, which penalises very much conditions that reduce the number of positive examples matched by the current rule. Alternative A1 is similar, but it computes the logarithm of accuracy, which takes both the number of false negatives and true negatives into account. Intuitively, the higher the number of true positive and true negatives, the better; contrarily, the higher the number of false positive and false negatives, the worse. Satisfaction reaches its maximum when precision is close to 1.00 and decreases steadily as the number of positive examples matched decreases. Alternative A3 penalises rules that match few positive examples; if a rule does not match any examples, then the result is 0.50, which is as effective as a random guess; contrarily, if it matches many examples, it tends to the precision. Finally, Alternative A4 tends to give higher scores when there are more true positives and true negatives than false negatives and false positives, respectively.

2.4.5. Variation point VP5: BOUND

This variation point deals with determining whether a condition deserves to be selected as a candidate to extend the current rule or to create a savepoint. The goal is to configure TANGO so that it learns extraction rules more efficiently without degrading their effectiveness or increasing their execution time.

Heuristic H6: prune candidates. We have considered the following alternatives: A0) Do not prune at all. A1) Prune every condition that does not result in a gain that is at least 80% the gain of the best condition found so far, unless it is a determinate condition; if a determinate condition is found, then the pruning

threshold is changed to 80% the maximum gain that a condition can achieve on the current rule.

Alternative A0 is the safest one because it does not prune any of the conditions that result from branching the current rule; that is, the search is exhaustive. The second alternative prunes the conditions that do not achieve a high enough gain with regard to the best condition found so far; intuitively, one might think that if the first condition achieves a given gain, then we might trivially discard every condition that does not exceed that gain, but we guessed that this would be too stringent, not to mention that we need to keep a few ones to create savepoints. Note that determinate conditions are never pruned because they help expand the search space and avoid local maxima. The key is that when such a condition is found, we know that there is at least a condition that can be used to extend the current rule. Actually, Heuristic H1 is not going to select any candidates that do not achieve 80% the maximum gain once a determinate condition is found, so we can make our heuristic much more stringent safely, that is, we can prune every non-determinate condition that cannot achieve the minimum gain.

2.4.6. Variation point VP6: postProcessRuleSet

This variation point attempts to simplify a rule set so that it is more efficient.

Heuristic H7: post-process rule sets. We have considered the following alternatives: A0) The rule set is not simplified. A1) The rules are simplified by removing useless conditions, subsumed rules, and folding constants.

Simplifying a rule set might help execute the rules more efficiently. We guessed that it would be common to find useless conditions in the rules learnt by TANGO. This happens when determinate conditions are added to a rule; such conditions typically provide little or no gain and they are added to escape local maxima; it is in the next iteration that they are expected to introduce new comparators or further feature instantiators, but there are cases in which they are neglected forever because there are other features that result in conditions that provide more gain. We also guessed that some rules might subsume other rules, that is, the examples that they match are a subset of the examples that another rule matches; in such cases, the former rule can be discarded. This happens because every rule is learnt independently from the others. Finally, we also thought that folding constants might help make the rules a little more efficient and easier to understand. Typically, TANGO learns many pairs of conditions of the form $f(N, A) \wedge A = v$, where f denotes an attributive feature, N a variable that can be bound to a node, A a variable that is bound to the value of feature f on node N, and v is one of the values in the range of that feature; such pairs of conditions may be easily simplified as $f(N, v)$ if variable A is not used anywhere else.

2.4.7. Variation point VP7: branch

This variation point deals with computing the set of conditions that can possibly be added to the current rule. The goal is to configure TANGO so that

it can learn extraction rules more efficiently without degrading their effectiveness or increasing the time required to execute them.

Heuristic H8: allow recursion. We have considered the following alternatives: A0) No recursion is allowed. A1) Rules are allowed to be recursive.

Alternative A0 is very simple, as usual. Alternative A1 is a little more involved because we have to make sure that making a rule recursive does not result in infinite recursion. Such a recursion occurs when a recursive condition includes a variable that is instantiated with the same example that is going to be extracted. The simplest solution to solve this problem is to determine if there exists a complete order amongst the variable used in the recursive condition and the variable in the head of the rule, both of which must be bound to nodes. If such an order exists, then it means that the variable in the recursive condition and the variable in the head cannot be instantiated with the same examples, which guarantees that the recursion is safe; if no such order exists, then the recursion is unsafe and must be avoided. This check can be implemented very efficiently using Ajwani et al.'s algorithm [1], for instance.

Heuristic H9: sort conditions. We have considered the following alternatives: A0) Conditions are generated in a random order. A1) Comparators are generated first, then slot instantiators (if recursion is allowed), and then feature instantiators in random order. A2) Comparators are generated first, then slot instantiators (if recursion is allowed), and then the feature instantiators are generated using an empirical frequency-based order.

Alternative A0 is the simplest one, as usual. Alternative A1 simply makes it explicit that we guessed that comparators would typically result in higher gains than feature instantiators. Alternative A2 relies on an order that we have computed empirically, cf. Table 1. Note that we have performed hundreds of experiments and that we have used TANGO to learn thousands of rules. What we have done is to compute the frequency with which each feature was used in a rule; we guessed that generating the feature instantiators using that empirical order might help learn rules more efficiently.

Heuristic H10: consider input/output modes. We have considered the following alternatives: A0) Do not take input/output modes into account. A1) Take them into account.

Alternative A0 is the simplest one and it helps explore as many feature instantiators as possible. Alternative A1 restricts the relational feature instantiators so that the first parameter is a bound variable and the second one is an unbound variable. Note that such a restriction may have a subtle implication regarding the catalogue of features and some relational features. For instance, recall that our running example includes relational features *parent* and *left*. If input/output modes are not taken into account, then $parent(N, M)$ helps navigate from a node to its parent if N is bound and M is unbound or from a node to its children if N is unbound and M is bound; similarly, $left(N, M)$ helps navigate from a node to its left sibling or from a node to its right sibling. Note

that this is not possible if input/output modes are taken into account; in such cases, features *right* and *child* must be provided explicitly in the catalogue of features or, otherwise, TANGO will not be able to find the children of a node or its right sibling.

2.4.8. Variation point VP8: isTooComplex

This variation point deals with preventing TANGO from learning very specific rules, which is expected to have a positive impact on efficiency.

Heuristic H11: check complexity of rules. We have considered the following alternatives: A0) Do not use any complexity criteria. A1) Use the Minimum Description Length principle [26].

The idea behind the Minimum Description Length principle is that a rule that requires more bits to be encoded than to encode the examples that it matches is too complex to be explored. The number of bits to encode a rule is computed as the sum of the bits required to encode every condition in its body, which is computed as the number of bits required to encode the features in the catalogue, plus the number of combinations that would result from combining their parameters, the built-in comparators, and the target slot instantiator (if recursion is allowed).

2.5. Configuration method

Our proposal relies on an open catalogue of features and many variation points. In order to configure it, it is necessary to use a sound method that allows to explore the different alternatives systematically, so that we can make informed decisions. Our configuration method is purely experimental. We suggest that a repository of datasets from different web sites should be assembled. In order for the conclusions to be statistically solid, the repository must provide a sufficiently large number of documents from different web sites on different domains.

The proposal should be run on this repository and the usual effectiveness and efficiency measures should be computed, namely: precision (P), recall (R), the F_1 score (F_1), learning time (LT), and extraction time (ET). Let M denote the previous set of performance measures. We assume that the experimenter provides a map β that assigns a weight in range $[0.00\,..\,1.00]$ to every measure in M. Obviously, the weights must sum up to 1.00 so that they are consistent.

Now, assume that we are dealing with a performance measure m, that we have gathered a set of values W regarding it, that a denotes the minimum value in set W, and that b denotes the maximum value. If m has to be maximised, then we define its set of normalised values as $W' = \{w' \mid \exists w \cdot w \in W \wedge w' = (w - a) \operatorname{div} (b - a)\}$; otherwise, we define its set of normalised values as $W' = \{w' \mid \exists w \cdot w \in W \wedge w' = 1.00 - (w - a) \operatorname{div} (b - a)\}$. ($x \operatorname{div} y$ equals x/y if $y \neq 0.00$; otherwise, it equals 1.00.) The values in W' range in interval $[0.00\,..\,1.00]$, so that the closer a value to the lower bound the worse and the closer to the upper bound the better. Let M' denote a set of new measures that are in one-to-one correspondence with the measures in M, but are normalised according to the previous procedure.

Let p denote a configuration of our system in which we have made some decisions regarding the alternative to implement a heuristic in a variation point. Our proposal is to compute its rank as follows:

$$K^p = \sum_{m' \in M'} \beta(m)\, K^p_{m'}$$
$$K^p_{m'} = \frac{\mathrm{MDR}^p_{m'}}{\mathrm{MDR}^{max}_{m'}}$$

where $\mathrm{MDR}^p_{m'}$ denotes the mean-to-deviation ratio of alternative p with regard to normalised performance measure m' and $\mathrm{MDR}^{max}_{m'}$ denotes the maximum mean-to-deviation ratio of performance measure m' across all of the alternatives. This ratio is defined as follows:

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

where m denotes an arbitrary performance measure, μ^p_m denotes its mean value regarding alternative p, and σ^p_m its standard deviation regarding alternative p. Note that this ratio maps every measure onto a value that weights its mean value with the inverse coefficient of variation (μ^p_m/σ^p_m) as long as the standard deviation is not zero; intuitively, the smallest the coefficient of variation with respect to the mean value, the better that measure because it is more stable. If the standard deviation is zero, the mean-to-deviation ratio is then trivially defined as the mean value.

We would also like to mention that our experimental analysis has revealed that some alternatives fail when they are applied to some datasets. Sometimes, the reason is that they consume too much memory; sometimes, they cannot learn a rule in a reasonable time; sometimes, the dataset has some characteristics that make it impossible to execute the learner on it. That means that we also need to compute a failure ratio for every alternative under consideration, which is defined as follows:

$$FR = \frac{F}{D},$$

where F denotes the number of datasets on which an alternative did not work, and D denotes the number of datasets on which the alternative was run. Intuitively, the closer to 0.00 the better and the closer to 1.00 the worse. We obviously, are not willing to accept an alternative whose failure ratio is different from 0.00, since that means that it is not generally applicable.

3. Configuring our proposal

In this section, we report on how we have configured our proposal. Our experimentation environment was set up as follows:

Category	Site	Slots	Number of documents	Number of HTML Errors	Number of positive examples	Number of negative examples	Annotation effort
Jobs	Insight into Diversity	Job{company, location, category}	30	67.07	30.00	12 905.50	00:45:00
	4 Jobs	Job{company, location, category}	30	110.47	30.00	9 657.75	00:50:00
	6 Figure Jobs	Job{company, location, category}	30	169.37	30.00	18 687.25	01:15:00
	Career Builder	Job{company, location, category}	30	93.97	30.00	10 055.00	00:50:00
	Job of Mine	Job{company, location, category}	30	41.03	30.00	7 210.75	00:50:00
		Summary	**30.00**	**96.38**	**30.00**	**11 703.25**	**00:54:00**
Cars	Auto Trader	Car{color, doors, engine, mileage, model, price, transmission, type}	30	273.23	30.00	12 462.67	02:13:00
	Car Max	Car{color, mileage, model, price, transmission, year, type}	30	191.47	30.00	9 222.63	01:57:00
	Car Zone	Car{color, doors, engine, location, make, mileage, model, price, transmission, year, type}	30	118.80	30.00	7 211.25	02:17:00
	Classic Cars for Sale	Car{color, location, make, model, price, transmission, year, type}	30	25.03	28.90	10 678.50	02:37:00
	Internet Autoguide	Car{color, doors, engine, location, mileage, price, transmission, type}	30	163.90	30.00	9 656.78	02:05:00
		Summary	**30.00**	**154.49**	**29.78**	**9 846.36**	**02:13:48**
Real Estate	Haart	Property{address, bedrooms, price}	30	40.00	9.00	9 662.00	01:00:00
	Homes	Property{address, bedrooms, bathrooms, size, price}	30	99.93	30.00	5 974.33	01:34:00
	Remax	Property{address, bedrooms, bathrooms, size, price}	30	75.70	30.00	12 097.50	01:50:00
	Trulia	Property{address, bedrooms, bathrooms, size, price}	30	312.73	30.00	27 320.00	02:10:00
		Summary	**30.00**	**132.09**	**24.75**	**13 763.46**	**01:38:30**
Doctors	Web MD	Doctor{name, address, phone, specialty}	30	24.10	30.00	12 872.00	02:35:00
	Ame. Medical Assoc.	Doctor{name, address, phone, specialty}	30	36.00	30.00	8 682.00	03:40:00
	Dentists	Doctor{name, address, phone, fax, specialty}	30	103.27	28.00	2 752.50	02:19:00
	Dr. Score	Doctor{name, address, phone, specialty}	30	33.07	27.14	5 679.00	02:12:00
	Steady Health	Doctor{name, address, specialty}	30	24.00	30.00	24 811.00	03:12:00
		Summary	**30.00**	**44.09**	**29.03**	**10 959.30**	**02:47:36**
Events	Linked In	Event{date, place, title, url}	30	23.67	29.00	9 077.80	02:03:00
	All Conferences	Event{date, place, title, url}	30	30.47	30.00	12 739.40	03:45:00
	Mbendi	Event{date, place, title, url}	30	27.00	30.00	6 120.00	01:18:00
	RD Learning	Event{date, place, title, url}	30	14.00	30.00	3 388.80	01:24:00
		Summary	**30.00**	**23.79**	**29.75**	**7 831.50**	**02:07:30**
Films	Albania Movies	Film{title, director, actor, year, runtime}	30	20.90	23.50	6 698.33	01:23:00
	All Movies	Film{title, director, actor, year, runtime}	30	32.33	78.33	36 428.33	01:46:00
	Disney Movies	Film{title, actor, year, runtime}	30	59.40	30.00	6 627.00	01:02:00
	IMDB	Film{title, director, actor, year, runtime}	30	12.00	40.33	18 023.67	01:18:00
	Soul Films	Film{title, director, actor, year}	30	66.13	65.40	17 985.00	00:54:00
		Summary	**30.00**	**38.15**	**47.51**	**17 152.47**	**01:16:36**
Books	Abe Books	Book{title, author, price, isbn}	30	58.73	35.60	8 370.40	01:47:00
	Awesome Books	Book{title, author, price, isbn, year}	30	43.27	37.17	6 612.83	01:57:00
	Better World Books	Book{title, author, price}	30	46.00	34.50	19 716.50	01:49:00
	Many Books	Book{title, author, year}	30	130.00	30.50	13 515.25	01:54:00
	Waterstones	Book{title, author, price}	30	129.10	31.50	13 856.00	02:17:00
		Summary	**30.00**	**81.42**	**33.85**	**12 414.20**	**01:56:48**
Players	Player Profiles	Player{name, birth, hight, weight, club}	30	35.07	30.00	30 840.00	02:10:00
	UEFA	Player{name, birth, country, position}	30	31.80	30.00	8 956.60	02:22:00
	ATP World Tour	Player{name, birth, age, hight, weigth, country}	30	92.03	30.00	15 284.57	01:57:00
	NFL	Player{name, birth, hight, weigth, age, college}	30	84.16	30.00	12 654.71	02:35:00
	Soccer Base	Player{name, birth, age, hight, weigth, country, club}	30	156.37	30.00	73 712.78	03:19:00
		Summary	**30.00**	**79.89**	**30.00**	**28 289.73**	**02:28:36**

Table 4: Description of our datasets (Part 1).

Category	Site	Slots	Number of documents	Number of HTML Errors	Number of positive examples	Number of negative examples	Annotation effort
ExAlg	Amazon Cars	Car{make, model, price}	21	20.00	54.00	3 179.25	00:40:00
	UEFA Players	Player{name, country}	20	10.40	180.33	7 010.67	01:10:00
	Amazon Pop Artists	Artist{name}	19	35.00	2 805.00	18 322.00	01:15:00
	UEFA Teams	Team{association, country, fifa-affiliation, founded, general-secretary, president, press-officer, team, uefa-affiliation}	20	32.80	19.90	5 412.80	00:45:00
	Aus Open Players	Player{name, birth-date, birth-place, country, height, money, weight}	29	66.73	29.00	44 437.13	03:30:00
	E-Bay Bids	Bid{price, bids, location}	50	18.12	30.00	33 393.00	01:30:00
	Baseball League	Player{name, position, team}	9	26.00	550.40	9 544.80	00:50:00
	Netflix Films	Film{title, director, length, year}	50	125.86	30.00	30 733.00	01:00:00
	RPM Find Packages	Package{name, description, operating-system}	20	9.90	2 562.00	22 359.00	02:30:00
		Summary	**26.44**	**38.31**	**695.63**	**19 376.85**	**01:27:47**
RISE	Bigbook	Business{name, city, phone, street}	235	20.61	566.00	16 825.20	01:55:00
	IAF	Finder{name, email, organisation, service-provider}	252	13.20	64.33	8 791.17	02:45:00
	Okra	Citizen{name, email}	10	15.42	380.00	14 086.00	00:40:00
	LA Weekly	Restaurant{ name, address, phone}	28	4.93	126.25	3 508.75	01:10:00
	Zagat	Restaurant{name, address, type}	91	31.92	32.75	4 894.25	01:30:00
		Summary	**123.20**	**17.22**	**233.87**	**9 621.07**	**01:36:00**

Table 5: Description of our datasets (Part 2).

Relative weights of performance measures: we set them to $\beta(F_1) = 0.50$, $\beta(LT) = 0.30$, and $\beta(ET) = 0.20$. These figures reflect that effectiveness is very important, since the goal is to achieve rules that are very precise and have high recall, and the learning time is a little more important than the extraction time in this particular context.

Experimentation datasets: Tables 4 and 5 describe our experimentation datasets. They provide many actual web documents from a variety of domains, including the semi-structured datasets from the classical ExAlg and RISE repositories. The last column reports on the time that we spent at annotating them completely, which is less than two hours in average. Note that we had to annotate all of the documents in order to compute the effectiveness results. In a production setting, it suffices to annotate a few documents that provide at least an example of every possible formatting; in other words, the annotation effort is not a problem.

Hardware and software configuration: we performed our experiments on a virtual computer that was equipped with four Intel Xeon E7 4807 cores that run at 1.87 GHz, had 64 GiB of RAM, and 2 TiB of storage. The operating system was Windows 7 Pro 64-bit and we used the following software packages: Oracle's Java Development Kit 1.7.9_02, JSoup 1.7.1, PhantonJS 2.1.1, and SWI-Prolog 5.4.7. No changes were made to the default hardware or software configurations.

Tables 6 and 7 summarise the results of our analysis. We report on the mean and the standard deviation of each performance measure, plus the mean-to-deviation ratio (MDR), the failure ratio (FR), and the rank (K) of every alternative. All of the results were computed using a learning set that was

	Heuristic	Alternative		P	R	F1	LT	ET	FR	K
VP1: select candidates	H1: select best candidates	A0	Mean	0.77	0.74	0.74	178.22	2.49		
			Std. Dev.	0.37	0.36	0.36	619.34	3.89	0.18	0.14
			MDR	1.62	1.49	1.53	51.28	1.60		
		A1	Mean	0.98	0.96	0.97	38.88	4.47		
			Std. Dev.	0.02	0.05	0.04	63.79	5.53	0.00	0.66
			MDR	39.76	16.88	24.31	23.69	3.61		
VP2: pre-process learning set	H2: reduce negative examples	A0	Mean	0.98	0.96	0.97	38.88	4.47		
			Std. Dev.	0.02	0.05	0.04	63.79	5.53	0.00	0.50
			MDR	39.76	16.88	24.31	23.69	3.61		
		A1	Mean	0.96	0.97	0.95	29.62	4.19		
			Std. Dev.	0.09	0.05	0.08	59.36	5.41	0.00	0.36
			MDR	10.42	20.45	11.27	14.78	3.25		
		A2	Mean	0.97	0.96	0.96	30.02	4.23		
			Std. Dev.	0.05	0.06	0.05	59.66	5.43	0.00	0.51
			MDR	20.15	16.85	18.83	15.10	3.30		
		A3	Mean	0.86	0.94	0.86	28.82	3.90		
			Std. Dev.	0.19	0.15	0.19	73.73	5.27	0.02	0.28
			MDR	3.94	5.85	3.98	11.26	2.89		
		A4	Mean	0.97	0.96	0.96	56.42	4.05		
			Std. Dev.	0.06	0.06	0.06	169.82	5.34	0.00	0.42
			MDR	16.72	15.08	15.97	18.75	3.07		
	H3: binarise features	A0	Mean	0.97	0.96	0.96	30.02	4.23		
			Std. Dev.	0.05	0.06	0.05	59.66	5.43	0.00	0.96
			MDR	20.15	16.85	18.83	15.10	3.30		
		A1	Mean	0.94	0.92	0.93	617.77	169.23		
			Std. Dev.	0.19	0.20	0.20	1839.87	639.22	0.02	0.12
			MDR	4.57	4.20	4.36	207.43	44.80		
VP3: is promising candidate	H4: select best candidates	A0	Mean	0.97	0.96	0.96	30.02	4.23		
			Std. Dev.	0.05	0.06	0.05	59.66	5.43	0.00	0.59
			MDR	20.15	16.85	18.83	15.10	3.30		
		A1	Mean	0.94	0.94	0.93	32.13	5.77		
			Std. Dev.	0.08	0.09	0.08	64.10	7.34	0.00	0.31
			MDR	11.02	10.15	10.75	16.10	4.54		
		A2	Mean	0.90	0.90	0.89	27.04	5.38		
			Std. Dev.	0.18	0.16	0.17	41.96	6.47	0.00	0.12
			MDR	4.56	5.00	4.57	17.43	4.48		
VP4: rule scorer	H5: compute scores	A0	Mean	0.97	0.96	0.96	30.02	4.23		
			Std. Dev.	0.05	0.06	0.05	59.66	5.43	0.00	0.68
			MDR	20.15	16.85	18.83	15.10	3.30		
		A1	Mean	0.96	0.96	0.95	20.40	5.27		
			Std. Dev.	0.05	0.06	0.06	46.14	6.48	0.00	0.64
			MDR	19.05	14.45	15.70	9.02	4.28		
		A2	Mean	0.96	0.96	0.95	45.66	6.47		
			Std. Dev.	0.06	0.06	0.07	88.39	8.35	0.00	0.37
			MDR	15.31	14.31	13.62	23.58	5.02		
		A3	Mean	0.96	0.95	0.95	43.69	6.22		
			Std. Dev.	0.06	0.06	0.06	86.80	7.47	0.00	0.42
			MDR	15.78	14.44	14.98	21.99	5.19		
		A4	Mean	0.67	0.66	0.66	22.07	4.23		
			Std. Dev.	0.40	0.39	0.39	45.89	6.51	0.00	0.29
			MDR	1.13	1.12	1.12	10.62	2.74		

Table 6: Results of our configuration analysis. (Part 1)

composed of six documents that we selected randomly; the remaining ones were used for testing purposes.

3.1. Variation point VP1: selectCandidates

Heuristic H1: select best candidates. Our conclusion regarding effectiveness is that Alternative A1 produces better results since precision, recall, and the F_1 score increase considerably with regard to Alternative A0. In average, the precision of Alternative A1 is 0.21 ± 0.39 higher, its recall is 0.23 ± 0.42 higher, and its F_1 score is 0.23 ± 0.40 higher than the corresponding measures regarding Alternative A0. Furthermore, the standard deviation of every effectiveness measure in Alternative A1 is smaller, which means that it is generally more stable than Alternative A0, that is, it does not generally produce rules whose effectiveness largely deviates from the average.

	Heuristic	Alternative		P	R	F1	LT	ET	FR	K
VP5: bound	H6: prune candidates	A0	Mean	0.97	0.96	0.96	30.02	4.23		
			Std. Dev.	0.05	0.06	0.05	59.66	5.43	0.00	0.55
			MDR	20.15	16.85	18.83	15.10	3.30		
		A1	Mean	0.97	0.96	0.96	22.17	5.97		
			Std. Dev.	0.05	0.05	0.05	40.12	7.19	0.00	0.56
			MDR	19.28	19.67	19.59	12.25	4.96		
VP6: post-process rule set	H7: post-process rule sets	A0	Mean	0.97	0.96	0.96	22.17	5.97		
			Std. Dev.	0.05	0.05	0.05	40.12	7.19	0.00	0.69
			MDR	19.28	19.67	19.59	12.25	4.96		
		A1	Mean	0.96	0.97	0.96	47.10	3.92		
			Std. Dev.	0.08	0.05	0.07	67.01	4.85	0.00	0.42
			MDR	11.80	17.65	13.56	33.10	3.17		
	H8: allow recursion	A0	Mean	0.97	0.96	0.96	22.17	5.97		
			Std. Dev.	0.05	0.05	0.05	40.12	7.19	0.00	0.65
			MDR	19.28	19.67	19.59	12.25	4.96		
		A1	Mean	0.96	0.95	0.95	42.60	5.85		
			Std. Dev.	0.05	0.07	0.06	75.03	6.88	0.00	0.37
			MDR	17.44	12.35	14.62	24.19	4.98		
VP7: branch	H9: sort candidates	A0	Mean	0.97	0.96	0.96	22.17	5.97		
			Std. Dev.	0.05	0.05	0.05	40.12	7.19	0.00	0.50
			MDR	19.28	19.67	19.59	12.25	4.96		
		A1	Mean	0.97	0.96	0.96	18.62	5.89		
			Std. Dev.	0.08	0.06	0.07	28.79	6.92	0.00	0.33
			MDR	12.19	14.23	12.84	12.04	5.01		
		A2	Mean	0.96	0.96	0.95	16.30	3.69		
			Std. Dev.	0.08	0.05	0.08	39.63	4.83	0.00	0.52
			MDR	10.84	16.89	11.61	6.71	2.83		
VP8: is too complex	H10: consider input/output modes	A0	Mean	0.96	0.96	0.95	16.30	3.69		
			Std. Dev.	0.08	0.05	0.08	39.63	4.83	0.00	0.63
			MDR	10.84	16.89	11.61	6.71	2.83		
		A1	Mean	0.96	0.96	0.95	14.22	5.44		
			Std. Dev.	0.08	0.06	0.08	23.35	7.40	0.00	0.50
			MDR	10.83	16.09	11.51	8.66	4.00		
	H11: check complexity	A0	Mean	0.96	0.96	0.95	16.30	3.69		
			Std. Dev.	0.08	0.05	0.08	39.63	4.83	0.00	0.59
			MDR	10.84	16.89	11.61	6.71	2.83		
		A1	Mean	0.96	0.96	0.95	23.16	3.70		
			Std. Dev.	0.08	0.06	0.08	55.68	4.83	0.00	0.50
			MDR	10.84	16.68	11.57	9.63	2.84		

Table 7: Results of our configuration analysis. (Part 2)

Our conclusion regarding efficiency is that Alternative A1 seems to be faster when learning rules since it is 139.34 ± 683.13 minutes faster than Alternative A0, which is a significant improvement; however, Alternative A1 is slower when executing the rules that it learns, but we must take into account that there are many datasets in which Alternative A0 could not learn any rules and that, sometimes, the resulting rule sets were unable to match all of the positive examples; this is very likely the reason why the rules produced by Alternative A0 are 1.98 ± 9.42 minutes faster. Backtracking was not performed in many cases; then, our experiments cannot support the idea that Alternative A1 is better or worse than Alternative A0 regarding the selection of candidates to create savepoints.

The rank of Alternative A1 is 0.66, which is much better than the rank of Alternative A0, which is 0.14. Note also that the failure ratio of Alternative A1 is exactly zero, which means that it was able to learn rules for every dataset, whereas Alternative A0 was not. Therefore, our conclusion is that Alternative A1 is the best one.

3.2. Variation point VP2: PREPROCESSLEARNINGSET

Heuristic H2: reduce negative examples. Our conclusion is that the best alternative is A2. Selecting the closest neighbours in a radius has proved to help TANGO learn rules that discern well amongst positive and negative examples that are very near in the DOM tree. Furthermore, selecting a small percentage of the remaining negative examples helps it produce rules that are general enough to make a difference amongst the positive examples and others that are very far away in the DOM tree. This alternative is a bit worse regarding precision and the F_1 score than the baseline; it behaves similarly in terms of recall and extraction time, but improves very much in terms of learning time since it is 8.86 ± 123.45 minutes faster. Alternative A4 got a rank that is close to the rank of the baseline but it is still a bit worse. Neither did Alternatives A1 nor A3 produce better results than the baseline.

Heuristic H3: binarise features. To our surprise, the results prove that the best choice is Alternative A0, that is, not binarising the features. Regarding effectiveness, the precision of the baseline is 0.03 ± 0.24 higher, its recall is 0.04 ± 0.26 higher, and its F_1 score is 0.03 ± 0.26 higher. Regarding efficiency, the differences are more significant: Alternative A1 is $587.76 \pm 1\,889.52$ minutes slower with regard to Alternative A0 when learning rules and 165.00 ± 644.64 minutes slower when the rules are executed. We found several explanations for that behaviour, namely: a) the number of features to be considered grows dramatically, which leads to extremely large learning sets that are very costly to process; b) when Alternative A0 is used, TANGO typically selects several determinate conditions in the first iteration, that is, several feature instantiators, and the corresponding feature values are typically constrained in the forthcoming iterations by means of comparators. This means that the number of comparators that are explored and evaluated depends on the number of feature instantiators that were added to the rule in the previous iteration. When binarisation is used, a feature instantiator both instantiates a feature and constrains its values

at the same time, which means that many features that are not promising at all must be considered in each iteration. Consequently, the number of candidates to explore and evaluate in each iteration grows significantly when binarisation is used. Furthermore, the standard deviation of the performance measures is smaller regarding Alternative A0, which means that it is generally more stable than Alternative A1. Note, too, that Alternative A1 was unable to find a set of rules in a few cases. Thus, our conclusion is that the best alternative is to use the features as they are provided.

3.3. Variation point VP3: IsPromisingCandidate

Heuristic H4: check promising candidates. Our results suggest that the best alternative to implement this heuristic is the default one. Alternative A1 has 0.03 ± 0.13 less precision, 0.03 ± 0.14 less recall, and 0.03 ± 0.13 less F_1 score than Alternative A0; furthermore, its learning time is 2.11 ± 123.76 minutes worse and the extraction time is 1.54 ± 12.77 minutes worse. We found out that the first candidate that exceeds the selected threshold is not typically the best one, and that it is common that some candidates that are explored later provide more gain and result in better rules; unfortunately, this alternative prevents TANGO from finding them. In other words, the number of candidates that are explored in each iteration is smaller, but the total number of iterations increases; this contributes to increasing the learning time and producing more specific rules that are not likely to work well with new unseen documents. Neither did Alternative A2 perform better: it was able to learn solutions faster than Alternative A0, exactly 2.98 ± 101.62 minutes faster; unfortunately, the resulting rule sets were larger since the individual rules learnt were more specific, which worsened the extraction time by 1.15 ± 11.89 minutes. Notice, too, that Alternative A0 is the most stable one since it achieves the lowest deviations. As a conclusion, Alternative A0 is the best one to implement this heuristic.

We think that the more stringent the criterion to select a promising candidate, the better the effectiveness. However, we did not manage to find a criterion that could improve on the baseline because if it is very stringent, then the behaviour of TANGO is similar to the baseline, i.e., it tends to perform an exhaustive search.

3.4. Variation point VP4: ruleScorer

Heuristic H5: compute scores. Regarding effectiveness, none of the alternatives that we have analysed can beat the baseline, but some of them achieve results that are very similar. This means that using the logarithm of the precision of a rule contributes positively to the overall performance of our system. Accuracy-based Information Content is the one that achieved the best results below the baseline, but the baseline provides 0.01 ± 0.10 more precision and 0.01 ± 0.11 more F_1 score. Regarding efficiency, Alternative A1 proved to learn 9.61 ± 105.80 minutes faster, but it was 1.03 ± 11.91 minutes slower when executing the rules. Laplace ranks at the third place, which makes sense, since it is also based on precision, like Information Content. However, one can easily realise the advantage

of using the logarithmic function, chiefly in both learning and extraction times. Satisfaction is not bad regarding effectiveness, but it is similar to the baseline regarding effectiveness and it is 15.64 ± 105.54 minutes slower regarding learning time and 2.24 ± 11.94 minutes slower regarding extraction time. Finally, Piatetski Shapiro's measure seems to be the worst one, since it achieved good results regarding efficiency by sacrificing the effectiveness considerably. Our conclusion is that every alternative, but the last one, can achieve good results regarding effectiveness.

However, the differences in efficiency are more remarkable. According to our intuition and to the K rank, it makes sense to select Information Content as our rule scorer since it seems to be the best alternative, followed by Accuracy-based Information Content. There is not a clear reason why some rule scorers performed better than others, but, in most cases, they all were able to guide the search properly. That is, the system was able to find perfect rules that matched the whole set of positive examples in the learning sets. However, it seems that the choice of some conditions during the learning phase had an impact on the testing phase and some of the candidates selected by some alternatives were not general enough to extract the information from the testing sets. This caused a penalty to precision and/or recall, which made a difference in some datasets because the system was unable to extract all of the positive examples or it extracted some more negative examples.

3.5. Variation point VP5: BOUND

Heuristic H6: prune candidates. Regarding effectiveness, there is a tie, because there are no clear differences between precision, recall, or the F_1 score between alternatives A0 and A1. However, Alternative A1 is 7.85 ± 99.77 minutes faster than Alternative A0 when learning rules, which is an important difference. Regarding the extraction time, the baseline is 1.74 ± 12.62 minutes faster. As the learning time that results from using Alternative A1 is much better, it ranks at the top and we can then select it as the best one. It makes sense that Alternative A1 is as effective as the baseline because it just prunes candidates that are not going to be selected to expand the rule according to Heuristic H1, so that good candidates are still kept; however, it avoids considering a number of candidates that are not promising enough so that it reduces the learning time.

3.6. Variation point VP6: POSTPROCESSRULESET

Heuristic H7: post-process rule sets. Again, the differences in effectiveness are not very significant. In some cases performing post processing led to better results regarding precision, recall, or the F_1 score, but in other cases, it resulted in worse results; thus, we conclude that both alternatives behave similarly regarding effectiveness. We studied this issue and we found out that the problem was that removing some conditions or rules from a rule set may not have any impact when the rules are executed on the learning set, but may have an impact when they are executed on a testing set and result in different precisions or recalls. Finding useless conditions or subsumed rules also has an impact on efficiency:

we have found that the learning time is 24.93 ± 107.13 minutes slower with Alternative A1. Contrarily, the time to execute the rules is, as expected, better, since it is 2.05 ± 12.04 minutes faster than the baseline. However, this difference is not enough when compared to the time spent in the learning process. What makes the learning process slower in Alternative A1 is the evaluation process once the rules are learnt. That is, analysing every single condition in the context of a rule to check if removing it can have a negative impact on their effectiveness. Furthermore, analysing every single rule in a rule set to find out subsumed rules also has a negative impact on effectiveness. Thus, our conclusion is that post-processing the resulting rule sets is not really worth.

3.7. Variation point VP7: BRANCH

Heuristic H8: allow recursion. The differences in effectiveness are not very significant. What makes a big difference is the learning time, which is 20.43 ± 115.14 minutes slower in Alternative A1. This had a very negative impact on the computation of the rank and made us select the baseline as the best alternative in this heuristic. The reason why Alternative A1 took longer during the learning phase is because it has to compute if there is an order between any two variables that can be instantiated with nodes, which took very long and the rules did not improve because there was not a single case in which the slot instantiator was included in the body of the rule, so there were not any improvements regarding making rules more simpler and/or general.

Heuristic H9: sort conditions. Note that it is Alternative A2 the one that performs the best according to our rank, which was not surprising. The baseline seems to perform a little better than Alternative A2 regarding effectiveness but the differences are negligible. Alternative A0 results in a precision that is 0.01 ± 0.13 higher, a recall that is 0.00 ± 0.10 higher, and an F_1 score that is 0.01 ± 0.12 higher than the corresponding ones in Alternative A2. However, both alternatives A1 and A2 are faster than the baseline when learning rules, namely: Alternative A1 is 3.54 ± 68.91 minutes faster and Alternative A2 is 5.86 ± 79.74 minutes faster. However, regarding the extraction time, Alternative A2 beats both Alternatives A0 and A1 since it is 2.28 ± 12.02 minutes faster. The improvement in both learning and extraction times led us to select Alternative A2 as the best one. It makes sense that sorting the features according to their empirical frequencies results in better timings since the features that have proven to work better at making a difference amongst the positive and the negative examples are prioritised and this helps find the best conditions faster.

Heuristic H10: consider input/output modes. Again, the differences in effectiveness are not very significant. Regarding the efficiency, it was expected that Alternative A1 reduced the learning time since there are some conditions that are not generated during the branching procedure; our experimental results confirm this idea because Alternative A1 is 2.08 ± 62.98 minutes faster than Alternative A0. Unfortunately, the extraction time worsened because it was 1.74 ± 2.57 minutes slower in Alternative A1. Thus, the only improvement on

learning time of Alternative A1 regarding A0 is not enough to select it. As a conclusion, we keep Alternative A0 as the best one.

3.8. Variation point VP8: isTooComplex

Heuristic H11: check complexity of rules. There are no differences in effectiveness since the rules learnt are exactly the same. It seems that the rules never become very complex since, in most of the cases, just one rule was enough to match the whole set of positive examples in the learning set. Consequently, there are not any differences regarding extraction time. However, computing the bits to encode each condition that is added to a rule and encoding the examples that it matches makes the learning process a bit more inefficient. This is why the learning time in Alternative A1 is 6.85 ± 95.31 minutes slower. As a conclusion, we prefer to keep Alternative A0 as the best one.

4. Experimental analysis

In this section, we report on the results of our experimental analysis and provide an insight into the difficult cases that we have found. We used the same experimental environment as we used to configure our proposal. We found an implementation for SoftMealy [17] and Wien [21], which are classical proposals, and RoadRunner [9], FiVaTech [19], and Trinity [29], which are recent proposals; we also experimented with an approach that is based on Aleph [31]. We used Iman-Davenport's test to find out if there are statistically significant differences in the empirical ranks and then Hommel's test to compare the best ranked proposal to the others. We used Kendall's Tau test to check if there is a significant correlation between the number of errors in the input documents and the effectiveness of our proposal. The statistical tests were performed at the standard significance level $\alpha = 0.05$.

4.1. Effectiveness analysis

Table 8 reports on the raw effectiveness data that we got from our experimentation. The first two lines also provide a summary of the results in terms of mean value and standard deviation. Table 9 summarises the results of our statistical analysis.

Regarding precision, TANGO seems to be the best proposal; furthermore, it is the most stable because its standard deviation is the smallest one. The other proposals can also achieve good results regarding precision, but their deviation with respect to the mean is larger. Note, however, that some other proposals can achieve results that are very good, too, chiefly Aleph. Iman-Davenport's test returns a p-value that is nearly zero, which is a strong indication that there are differences in rank amongst the proposals that we have compared. Since TANGO ranks the first regarding precision, we now have to compare it to the others using Hommel's test. It confirms that the differences in rank amongst TANGO and Trinity, SoftMealy, FivaTech, Wien, and RoadRunner, are statistically significant because it returns adjusted p-values that are very small with

Dataset	SoftMealy			Wien			RoadRunner			FivaTech			Trinity			Aleph			TANGO		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
Mean	0.70	0.61	0.63	0.59	0.68	0.58	0.52	0.69	0.53	0.62	0.82	0.68	0.80	0.90	0.84	0.91	0.90	0.90	0.96	0.96	0.95
Std. Deviation	0.17	0.33	0.31	0.22	0.33	0.27	0.29	0.37	0.31	0.25	0.19	0.24	0.12	0.10	0.09	0.10	0.11	0.10	0.08	0.05	0.08
Insight into Diversity	0.45	0.45	0.45	0.76	0.97	0.85	0.42	0.48	0.45	0.98	0.67	0.80	0.63	1.00	0.77	0.98	0.92	0.95	0.97	0.92	0.94
4 Jobs	0.42	0.15	0.22	0.86	0.85	0.85	0.10	1.00	0.18	0.87	0.60	0.71	0.82	0.90	0.86	0.82	0.77	0.80	0.97	0.78	0.84
6 Figure Jobs	0.53	1.00	0.69	0.21	1.00	0.35	0.23	1.00	0.38	0.93	0.92	0.93	0.70	0.95	0.81	0.86	0.88	0.87	0.82	0.93	0.83
Career Builder	0.70	0.09	0.16	0.48	1.00	0.65	0.02	0.07	0.03	0.59	1.00	0.74	0.85	0.92	0.88	1.00	0.92	0.96	0.93	0.89	0.90
Job of Mine	0.46	0.05	0.10	0.34	0.42	0.38	0.61	0.66	0.63	0.53	0.52	0.53	0.67	0.99	0.80	0.89	0.99	0.94	0.99	0.96	0.97
Auto Trader	0.75	1.00	0.86	0.64	0.00	0.00	-	-	-	-	-	-	0.81	0.81	0.81	0.90	0.91	0.91	0.96	0.96	0.96
Car Max	0.78	0.80	0.79	0.76	0.78	0.77	0.76	0.95	0.84	0.32	0.82	0.46	0.83	0.80	0.81	0.80	0.81	0.81	1.00	1.00	1.00
Car Zone	0.67	0.02	0.04	0.73	0.77	0.75	0.56	1.00	0.72	0.83	0.99	0.90	0.91	0.91	0.91	0.89	0.83	0.86	0.95	0.97	0.95
Classic Cars for Sale	0.86	0.89	0.88	0.10	1.00	0.19	0.36	0.46	0.40	-	-	-	0.61	0.84	0.71	0.92	0.83	0.87	0.91	0.97	0.92
Internet Autoguide	0.46	0.43	0.44	0.17	0.02	0.04	0.90	0.99	0.94	0.85	0.93	0.89	0.76	0.99	0.86	0.83	0.88	0.85	0.92	0.95	0.94
Haart	0.79	0.90	0.84	0.67	0.68	0.68	0.56	0.78	0.65	0.67	0.95	0.78	0.88	0.95	0.91	0.90	1.00	0.95	1.00	0.99	1.00
Homes	0.77	0.72	0.74	0.82	0.88	0.85	0.99	0.95	0.97	-	-	-	0.96	0.98	0.97	0.83	0.81	0.82	1.00	1.00	1.00
Remax	0.59	0.79	0.68	0.80	1.00	0.89	0.26	0.05	0.08	0.70	0.71	0.71	0.47	0.95	0.63	0.99	0.98	0.98	0.99	0.98	0.99
Trulia	0.78	0.85	0.81	0.76	0.87	0.81	0.21	0.04	0.06	-	-	-	0.46	0.99	0.63	0.93	0.93	0.93	0.98	0.95	0.96
Web MD	0.76	0.40	0.52	0.59	0.57	0.58	0.22	0.04	0.07	0.54	0.95	0.69	0.96	0.94	0.95	0.93	0.94	0.93	0.79	0.93	0.82
Ame. Medical Assoc.	0.53	0.31	0.39	0.55	0.56	0.55	-	-	-	0.11	0.19	0.14	0.73	0.93	0.82	0.80	0.79	0.80	0.99	0.94	0.97
Dentists	0.56	0.60	0.58	0.88	0.99	0.93	0.84	1.00	0.92	0.40	0.95	0.56	0.86	0.99	0.92	1.00	0.89	0.94	1.00	0.92	0.95
Dr. Score	0.73	0.80	0.77	0.71	0.78	0.74	0.65	0.98	0.78	0.67	0.95	0.79	0.72	0.95	0.82	0.61	0.64	0.62	0.92	0.84	0.85
Steady Health	0.56	0.19	0.28	0.62	0.66	0.64	0.81	0.99	0.89	0.75	1.00	0.86	0.79	0.94	0.86	1.00	0.89	0.94	1.00	1.00	1.00
Linked In	0.78	0.53	0.63	0.56	0.20	0.29	0.38	0.49	0.43	0.78	0.87	0.83	0.89	0.86	0.87	0.92	1.00	0.96	1.00	0.98	0.99
All Conferences	0.96	0.17	0.28	0.78	0.35	0.48	0.61	1.00	0.76	0.71	0.80	0.75	0.97	0.96	0.96	0.99	0.95	0.97	0.94	0.99	0.96
Mbendi	1.00	0.55	0.71	0.66	0.40	0.50	0.62	0.82	0.71	0.61	0.99	0.76	0.81	0.97	0.88	0.96	0.96	0.96	0.90	1.00	0.93
RD Learning	0.34	0.33	0.33	0.35	1.00	0.52	0.73	1.00	0.85	0.86	0.74	0.80	0.75	0.94	0.83	0.98	1.00	0.99	0.99	1.00	0.99
Albania Movies	1.00	0.37	0.54	0.72	1.00	0.83	0.48	0.77	0.59	0.75	0.73	0.74	0.81	0.76	0.78	0.92	0.94	0.93	1.00	0.97	0.98
All Movies	0.78	0.20	0.32	0.02	0.04	0.03	0.23	1.00	0.38	0.62	0.66	0.64	0.92	0.81	0.86	0.91	0.80	0.85	0.92	0.95	0.92
Disney Movies	0.93	0.92	0.92	0.60	1.00	0.75	0.41	1.00	0.58	0.68	0.58	0.62	0.78	0.76	0.77	0.97	0.96	0.96	0.79	0.99	0.84
IMDB	0.69	0.78	0.74	0.19	0.30	0.23	0.20	1.00	0.33	0.68	0.73	0.70	0.80	0.81	0.80	0.93	0.93	0.93	1.00	0.99	1.00
Soul Films	0.90	1.00	0.95	0.72	1.00	0.83	0.49	0.45	0.47	0.41	0.96	0.58	0.86	0.91	0.88	0.95	0.92	0.94	1.00	0.94	0.97
Abe Books	0.63	1.00	0.77	0.50	0.09	0.15	0.60	0.53	0.56	0.75	1.00	0.86	0.90	0.96	0.93	0.94	0.92	0.93	1.00	0.94	0.97
Awesome Books	0.85	0.37	0.52	0.77	0.20	0.31	0.70	0.43	0.54	0.80	0.96	0.87	0.91	0.86	0.88	0.99	0.91	0.95	0.99	1.00	0.99
Better World Books	0.78	0.96	0.86	0.37	0.34	0.36	-	-	-	0.91	0.93	0.92	0.71	0.70	0.70	0.95	0.95	0.95	1.00	1.00	1.00
Many Books	0.70	1.00	0.83	0.01	0.22	0.02	0.88	1.00	0.94	0.54	1.00	0.70	0.77	0.90	0.83	0.99	0.99	0.99	0.97	0.98	0.97
Waterstones	1.00	0.92	0.96	0.68	0.67	0.68	0.71	0.77	0.74	0.73	0.86	0.79	0.87	0.89	0.88	0.96	1.00	0.98	1.00	1.00	1.00
Player Profiles	0.66	0.06	0.11	0.90	0.99	0.94	-	-	-	0.08	0.93	0.14	0.81	1.00	0.89	0.97	0.95	0.96	0.96	0.94	0.95
UEFA	0.75	1.00	0.86	0.83	0.94	0.88	0.86	0.91	0.88	-	-	-	0.97	0.92	0.94	1.00	1.00	1.00	1.00	1.00	1.00
ATP World Tour	0.78	1.00	0.88	0.48	0.67	0.56	0.72	0.89	0.80	0.91	1.00	0.95	0.79	0.90	0.84	0.93	0.97	0.95	0.98	0.97	0.98
NFL	0.56	1.00	0.71	0.62	1.00	0.77	0.83	0.92	0.87	0.40	0.78	0.53	0.72	1.00	0.84	0.76	0.65	0.70	1.00	1.00	1.00
Soccer Base	0.74	0.87	0.80	0.64	1.00	0.78	0.66	0.95	0.77	-	-	-	0.96	0.97	0.97	0.89	0.90	0.89	0.98	1.00	0.99
Amazon Cars	0.76	0.91	0.83	0.72	0.95	0.82	1.00	0.10	0.18	0.37	0.63	0.47	0.63	0.65	0.64	0.97	1.00	0.99	1.00	1.00	1.00
UEFA Players	0.80	0.87	0.83	0.73	0.48	0.58	0.81	0.96	0.88	0.65	1.00	0.79	0.74	0.83	0.78	1.00	0.99	0.99	0.50	1.00	0.60
Amazon Pop Artists	0.88	0.68	0.77	0.74	1.00	0.85	0.23	0.07	0.10	0.94	1.00	0.97	0.86	1.00	0.92	-	-	-	1.00	1.00	1.00
UEFA Teams	0.80	0.84	0.82	0.38	0.75	0.51	0.86	1.00	0.93	0.87	0.91	0.89	0.91	0.92	0.91	0.54	0.50	0.52	1.00	1.00	1.00
Aus Open Players	0.40	0.22	0.29	0.47	0.25	0.33	0.61	1.00	0.76	0.04	0.77	0.08	0.70	0.94	0.81	1.00	0.93	0.96	1.00	0.96	0.98
E-Bay Bids	0.63	0.07	0.13	0.65	0.09	0.15	0.70	0.79	0.74	0.68	0.99	0.81	0.70	0.96	0.81	0.64	0.66	0.65	0.83	0.72	0.70
Major League Baseball	0.87	0.37	0.52	0.47	0.28	0.35	0.19	1.00	0.32	0.73	1.00	0.84	0.75	0.48	0.58	1.00	0.98	0.99	0.98	1.00	0.99
Netflix Films	0.65	0.78	0.71	0.79	0.97	0.87	0.54	0.74	0.63	0.77	0.73	0.74	0.79	1.00	0.89	0.99	0.93	0.96	1.00	0.95	0.97
RPM Find Packages	0.71	0.04	0.08	0.84	0.94	0.88	0.01	0.07	0.02	0.02	0.63	0.04	0.75	1.00	0.86	1.00	0.93	0.96	1.00	0.99	1.00
Bigbook	0.79	0.75	0.77	0.58	0.91	0.70	0.29	0.03	0.05	-	-	-	0.87	0.92	0.89	0.80	0.80	0.80	1.00	1.00	1.00
IAF	0.28	0.41	0.34	0.74	1.00	0.85	0.87	0.08	0.15	0.25	0.67	0.37	0.60	1.00	0.75	0.92	0.87	0.90	0.94	0.99	0.96
Okra	0.66	1.00	0.80	0.36	0.63	0.46	0.01	0.03	0.02	0.31	0.33	0.32	0.98	0.78	0.87	0.96	0.95	0.95	1.00	0.99	1.00
LA Weekly	0.44	0.56	1.30	0.63	0.79	0.70	0.06	1.00	0.11	0.62	0.47	0.54	0.77	0.88	0.82	0.99	0.83	0.91	0.99	0.97	0.98
Zagat	0.60	0.62	1.50	0.53	1.00	0.70	0.24	1.00	0.39	0.87	0.94	0.90	0.95	0.85	0.90	1.00	0.97	0.98	1.00	1.00	1.00

Table 8: Effectiveness results.

Precision

Sample ranking		Iman-Davenport's	Hommel's adjusted p-values						Statistical ranking	
Technique	Rank	p-value		Aleph	Trinity	SoftMealy	FiVaTech	Wien	RoadRunner	Technique	Rank
TANGO	1.60									TANGO	1
Aleph	2.33									Aleph	1
Trinity	3.54									Trinity	2
SoftMealy	4.53	5.76E-46	TANGO	8.45E-02	9.10E-06	1.33E-11	1.02E-15	2.34E-18	4.21E-20	SoftMealy	2
FiVaTech	5.07									FiVaTech	2
Wien	5.38									Wien	2
RoadRunner	5.57									RoadRunner	2

Recall

Sample ranking		Iman-Davenport's	Hommel's adjusted p-values						Statistical ranking	
Technique	Rank	p-value		Trinity	Aleph	RoadRunner	FiVaTech	Wien	SoftMealy	Technique	Rank
TANGO	2.52									TANGO	1
Trinity	3.43									Trinity	2
Aleph	3.91									Aleph	2
RoadRunner	4.21	6.42E-09	TANGO	3.11E-02	2.00E-03	1.94E-04	9.71E-05	1.31E-05	6.16E-09	RoadRunner	2
FiVaTech	4.31									FiVaTech	2
Wien	4.51									Wien	2
SoftMealy	5.11									SoftMealy	2

F_1 score

Sample ranking		Iman-Davenport's	Hommel's adjusted p-values						Statistical ranking	
Technique	Rank	p-value		Aleph	Trinity	FiVaTech	SoftMealy	Wien	RoadRunner	Technique	Rank
TANGO	1.53									TANGO	1
Aleph	2.56									Aleph	2
Trinity	3.45									Trinity	2
FiVaTech	4.81	7.33E-42	TANGO	1.52E-02	1.13E-05	3.00E-14	2.15E-15	1.26E-18	8.35E-19	SoftMealy	2
SoftMealy	4.96									FiVaTech	2
Wien	5.34									Wien	2
RoadRunner	5.36									RoadRunner	2

Table 9: Statistical ranks regarding effectiveness.

regard to the standard significance level. In the case of Aleph, the statistical test did not find any significant differences since the corresponding adjusted p-value is greater than the standard significance level. In other words, our experimental data provides enough evidence to reject the hypothesis that TANGO behaves similarly to Trinity, SoftMealy, FivaTech, Wien, and RoadRunner, regarding precision; that is, it supports the idea that TANGO can learn rules that are more precise than the other proposals, but we cannot reject the hypothesis that TANGO behaves similarly to Aleph.

Regarding recall, TANGO seems to be the best proposal; furthermore, it is the most stable because its deviation is the smallest one. Note, however, that the other proposals can achieve results that are very good, too, chiefly Trinity and Aleph. Iman-Davenport's test returns a p-value that is very close to zero, which is a strong indication that there are differences in rank amongst the proposals that we have compared. Hommel's test confirms that the differences in rank between TANGO and the other proposals are statistically significant at the standard significance level. As a conclusion, the experimental data provides enough evidence to reject the hypothesis that TANGO behaves similarly to the other proposals regarding recall; that is, it supports the idea that TANGO ranks at the first position.

Regarding the F_1 score, TANGO is the best one and the most stable. Trinity and Aleph are also very stable, but their results regarding F_1 are a bit poorer. Iman-Davenport's test returns a p-value that is nearly zero, which strongly supports the hypothesis that there are statistically significant differences in rank. Hommel's test returns adjusted p-values that are clearly smaller than the significance level in every case, which supports the hypothesis that the differences in rank between TANGO and every other proposal are statistically significant, too; that is, we can safely assume that it ranks the first.

Since TANGO works on the tree representation of the input documents, we need to parse them and correct their HTML errors. It was then necessary to perform a statistical analysis to find out if our experiments provide enough evidence to conclude that the presence of errors in the input documents has an impact on the effectiveness of our proposal. Kendall's Tau test returned $\tau = -0.10$ with p-value 0.59. Since τ is very close to zero and that the p-value is clearly greater than the standard significance level, then the experimental data does not provide enough evidence to reject the hypothesis that the correlation is zero. In other words, our experiments do not provide any evidences that the effectiveness of our proposal may be biased by the errors in the HTML code of the input documents.

Our conclusions are that TANGO outperforms the other proposals regarding effectiveness and that it is the proposal whose results are more stable. The statistical tests that we have performed have found enough evidence in our experimental data to support the hypothesis that the differences in the empirical rank amongst TANGO and the other proposals are significant at the standard significance level, except for the case of precision, in which case the experimental data does not provide enough evidence to conclude that TANGO and Aleph perform differently. Note, too, that proposals like RoadRunner, FiVaTech, and

Aleph cannot deal with all of our datasets; such situations are indicated with a dash in Table 8. The reason is that they took more than 1 CPU day to learn a rule or that they raised an exception; in both cases, we could not compute any effectiveness measures for the corresponding datasets.

4.2. Efficiency analysis

Table 10 reports on the raw efficiency data that we got from our experimentation. The first two lines also provide a summary of the results in terms of mean value and standard deviation. Table 11 summarises the results of our statistical test.

Regarding learning times, it seems that Trinity is the proposal that takes less time to learn a rule set. In most cases, it does not take more than a tenth of a second. It is followed by RoadRunner, SoftMealy, and Wien, whose learning times are very similar; then come Aleph and FivaTech; finally, TANGO ranks at the last position. Iman-Davenport's test returns a p-value that is very close to zero, which clearly supports the hypothesis that there are differences in rank amongst these proposals. Hommel's test also returns adjusted p-values that are very small with respect to the significance level, which also reveals that the experimental data provides enough evidence to support the hypothesis that Trinity performs better than the other proposals.

Regarding extraction times, Wien, SoftMealy, Aleph, and TANGO seem to be the proposals that have the worst performance; RoadRunner and Trinity seem to be very similar; finally, FivaTech seems to be in the middle and its extraction time is still competitive. The timings regarding TANGO are the worst, since applying the rules learnt takes roughly 221.68 seconds in average; neither is its standard deviation small, which means that the results are not as stable as we wished. Iman-Davenport's test returns a p-value that is nearly zero, which clearly indicates that there are statistically significant differences in the empirical rank. Hommel's test returns adjusted p-values that are not smaller than the standard significance level regarding the comparisons of Trinity, RoadRunner, and FivaTech. Therefore, we cannot reject the hypothesis that they behave statistically similarly regarding extraction times, that is, we have to assume that they all rank at the first position. The test, however, finds enough evidence to reject the hypothesis that the previous proposals and the others behave similarly regarding the extraction time.

Note that we report on the efficiency figures regarding a research prototype of TANGO. It was implemented in Java and relies on a free Prolog engine to implement rules; we think that an ad-hoc implementation would be more efficient because it would avoid wasting resources at exchanging data with the Prolog engine and going through its compilation and unification processes. Recall that we focus on web information extraction problems in the context of Enterprise Systems Integration, where the challenge is to learn rules with very high precision and recall, and our experimentation confirms that we have succeeded regarding this issue. Thus, we think that the penalty to learn more effective rules is not a serious drawback.

Dataset	SoftMealy		Wien		RoadRunner		FivaTech		Trinity		Aleph		TANGO	
	LT	ET	LT	ET	LT	ET	LT	ET	LT	ET	LT	ET	LT	ET
Mean	5.00	35.51	5.28	8.58	5.24	0.36	64.25	0.44	0.10	0.34	27.61	53.06	978.19	221.68
Std. Deviation	4.33	38.35	3.88	9.78	6.20	0.48	104.17	0.58	0.14	0.47	36.73	47.34	2377.61	289.77
Insight into Diversity	4.90	11.67	3.20	4.99	2.41	1.00	9.35	0.08	0.05	1.00	13.40	50.57	1855.92	229.80
4 Jobs	4.07	36.60	6.09	6.57	1.40	1.00	8.32	0.07	0.04	0.01	17.08	37.44	289.08	104.22
6 Figure Jobs	7.65	18.51	7.44	8.32	13.83	1.00	53.21	0.24	0.02	0.01	12.20	71.17	2342.82	285.24
Career Builder	5.20	35.44	5.62	5.39	5.75	0.01	136.03	0.22	0.03	1.00	16.24	34.12	259.20	101.10
Job of Mine	3.63	15.61	2.86	3.32	1.49	0.01	30.33	0.14	0.03	0.01	15.42	30.23	246.48	83.76
Auto Trader	6.78	72.39	6.07	9.73	-	-	-	-	0.12	0.02	18.40	41.81	811.68	267.00
Car Max	9.72	22.91	3.50	5.34	13.27	0.01	19.24	0.14	0.14	0.01	15.54	32.91	426.96	175.02
Car Zone	3.43	28.88	5.76	3.32	2.72	1.00	198.79	0.41	0.02	1.00	15.23	30.35	455.22	241.74
Classic Cars for Sale	13.74	78.55	11.89	7.19	28.40	0.01	-	-	0.13	0.01	22.01	129.59	411.66	417.42
Internet Autoguide	4.33	68.24	4.01	6.26	4.42	1.00	71.50	1.00	0.05	1.00	15.73	31.38	475.80	178.68
Haart	4.03	69.80	3.27	4.82	3.12	0.01	9.40	0.06	0.02	1.00	17.36	41.82	98.64	103.38
Homes	4.36	45.57	4.80	8.18	2.43	0.01	-	-	0.11	0.01	13.79	23.64	84.72	92.04
Remax	3.09	17.80	8.30	5.17	7.85	0.01	47.73	0.07	0.22	0.01	16.40	36.77	94.44	146.04
Trulia	11.87	196.63	12.94	25.37	19.97	1.00	-	-	0.48	1.00	18.95	119.56	4933.20	609.42
Web MD	4.54	20.85	10.47	10.17	14.49	0.01	7.64	1.00	0.01	0.01	18.24	46.42	241.62	154.26
Ame. Medical Assoc.	4.16	7.30	3.68	7.39	-	-	1.53	0.20	0.03	1.00	16.05	33.56	207.90	104.34
Dentists	1.56	5.67	1.36	1.28	0.39	0.01	4.86	0.05	0.01	1.00	12.78	9.40	29.82	30.54
Dr. Score	2.81	17.51	2.07	2.51	1.01	1.00	32.29	1.00	0.01	0.01	13.89	17.72	390.90	97.26
Steady Health	4.85	40.00	5.95	6.78	7.61	0.02	5.71	0.08	0.30	0.01	34.05	97.53	1010.58	312.30
Linked In	6.06	29.18	1.87	3.34	1.66	0.01	34.56	2.35	0.02	0.01	16.15	26.32	189.78	84.96
All Conferences	6.34	43.01	3.68	3.32	1.88	0.01	18.54	1.00	0.05	0.01	18.49	36.57	289.32	129.60
Mbendi	1.64	3.98	2.08	1.28	0.75	1.00	0.97	0.02	0.00	0.01	13.82	19.06	165.60	58.62
RD Learning	2.24	4.96	1.44	1.36	0.33	0.01	3.51	0.01	0.01	0.01	12.43	11.31	172.62	35.88
Albania Movies	2.08	3.45	1.36	1.16	0.91	0.01	1.88	0.01	0.01	1.00	15.19	67.41	203.34	106.32
All Movies	11.54	38.87	3.23	4.11	1.90	0.01	6.23	1.00	0.27	0.01	125.19	37.21	777.78	499.14
Disney Movies	4.35	31.14	2.00	2.67	1.99	0.01	121.28	0.05	0.67	0.01	74.71	25.91	374.70	107.40
IMDB	19.89	63.25	19.47	11.47	9.92	0.01	32.13	2.32	0.24	1.00	111.09	68.47	2263.44	383.88
Soul Films	6.68	26.09	4.03	9.37	1.91	0.01	10.64	0.03	0.02	0.01	27.51	122.65	2422.62	243.18
Abe Books	9.95	18.71	10.74	10.27	3.29	0.01	9.78	0.09	0.02	0.01	15.06	32.12	206.46	112.62
Awesome Books	2.53	11.49	3.25	6.28	1.55	0.01	3.67	0.10	0.02	0.01	14.92	25.81	136.26	104.88
Better World Books	19.55	28.83	7.93	11.89	-	-	47.54	0.27	0.10	0.01	12.21	73.74	168.48	193.62
Many Books	7.39	21.03	2.82	5.62	1.31	0.01	82.71	0.09	0.13	0.01	10.08	49.78	518.10	154.98
Waterstones	5.00	53.46	5.58	6.03	3.47	1.00	29.37	1.38	0.04	0.01	11.45	49.49	419.64	143.76
Player Profiles	2.92	17.43	5.83	3.43	-	-	8.06	1.00	0.06	0.16	19.90	103.81	515.76	439.32
UEFA	4.00	23.08	2.69	3.97	6.89	0.02	-	-	0.03	0.01	18.20	31.93	57.30	102.72
ATP World Tour	9.31	125.86	11.81	12.76	8.87	0.03	50.01	0.58	0.38	0.02	19.60	56.60	403.68	276.00
NFL	6.34	27.66	9.69	6.64	19.88	0.02	72.49	1.00	0.08	0.01	16.55	43.50	122.88	197.04
Soccer Base	8.07	33.30	12.95	7.56	10.06	1.00	-	-	0.34	1.00	51.36	277.67	16444.44	1962.54
Amazon Cars	0.68	7.02	8.41	5.66	0.80	1.00	2.55	1.00	0.01	1.00	13.39	11.00	22.62	25.92
UEFA Players	1.43	11.64	3.79	2.58	0.41	0.01	12.58	0.03	0.01	0.01	16.93	18.68	537.36	29.22
Amazon Pop Artists	3.61	16.20	7.96	10.22	1.03	1.00	107.04	0.08	0.01	0.01	-	-	202.68	153.96
UEFA Teams	0.49	3.33	0.80	2.28	0.47	0.01	0.54	0.01	0.02	1.00	11.74	15.23	170.76	89.64
Aus Open Players	0.81	15.27	5.26	16.96	3.12	1.00	80.70	0.18	0.14	1.00	21.67	147.52	2613.30	782.70
E-Bay Bids	1.14	11.11	5.59	13.65	2.11	0.01	397.98	0.34	0.25	0.06	22.60	119.96	590.46	336.84
Major League Baseball	1.33	13.67	4.39	4.28	1.75	0.01	184.47	1.00	0.02	0.01	21.28	13.79	1188.78	62.22
Netflix Films	2.26	23.88	4.85	31.55	4.73	0.01	399.01	0.53	0.08	0.01	17.60	105.26	941.46	348.18
RPM Find Packages	0.83	18.23	1.39	10.42	0.75	1.00	28.59	0.06	0.02	1.00	14.93	68.16	216.60	183.78
Bigbook	1.61	114.85	1.78	62.29	14.27	1.00	-	-	0.06	1.00	17.81	48.88	372.84	168.36
IAF	1.77	9.23	0.90	1.67	0.44	0.01	7.00	0.04	0.07	0.01	68.21	17.15	2855.22	123.18
Okra	0.78	150.01	1.13	24.48	10.39	1.00	425.36	0.26	0.06	0.01	228.41	28.76	545.28	81.12
LA Weekly	1.30	23.45	0.57	1.89	0.48	0.01	2.69	0.03	0.01	0.01	13.27	16.60	54.36	26.34
Zagat	1.50	14.11	5.83	13.75	3.85	1.00	73.51	0.04	0.07	1.00	13.54	19.61	35.28	45.90

Table 10: Efficiency results.

Learning time

Sample ranking		Iman-Davenport's	Hommel's adjusted p-values						Statistical ranking	
Technique	Rank	p-value		RoadRunner	SoftMealy	Wien	Aleph	FivaTech	TANGO	Technique	Rank
Trinity	1.00	4.78E-111	Trinity	2.35E-06	1.19E-06	7.08E-08	2.79E-23	3.38E-24	4.37E-43	Trinity	1
RoadRunner	3.00									RoadRunner	2
SoftMealy	3.12									SoftMealy	2
Wien	3.37									Wien	2
Aleph	5.27									Aleph	2
FivaTech	5.37									FivaTech	2
TANGO	6.88									TANGO	2

Extraction time

Sample ranking		Iman-Davenport's	Hommel's adjusted p-values						Statistical ranking	
Technique	Rank	p-value		RoadRunner	FiVaTech	Wien	SoftMealy	Aleph	TANGO	Technique	Rank
Trinity	1.67	6.25E-159	Trinity	6.50E-01	1.25E-01	3.21E-08	1.82E-16	7.69E-21	5.56E-35	Trinity	1
RoadRunner	1.87									RoadRunner	1
FivaTech	2.46									FivaTech	1
Wien	4.10									Wien	2
SoftMealy	5.23									SoftMealy	2
Aleph	5.71									Aleph	2
TANGO	6.96									TANGO	2

Table 11: Statistical ranks regarding efficiency.

4.3. An insight into the difficult cases

Although effectiveness is generally close to 100%, there are some datasets in which our proposal could not reach at least 90% precision and/or recall, which we consider the minimum sensible threshold nowadays. We have gone through these cases and we have found that the reason is not related to the structure of the documents, but to the six documents that were randomly selected for the learning set and/or the alternatives that we selected in our default configuration.

Precision is poor regarding datasets 6 Figure Jobs and UEFA Players. The problem was due to the fact that we selected Alternative A2 to implement Heuristic H9, namely: sort conditions according to our empirical order. This alternative led to a tie amongst several feature instantiators when extending a partially constructed rule, which was broken arbitrarily. This resulted in adding a feature instantiator that was more general than it would have been using Alternative A1 or breaking the tie differently; the problem with that feature instantiator is that it matched some negative examples, which obviously had a negative impact on precision. There was also a problem with precision in dataset E-Bay Bids, namely: we could learn rules with precision 1.00 when we explored Alternative A1 in Heuristic H1 to select the best candidate conditions to extend the current rule and to create savepoints; the precision remained 1.00 when we explored Alternative A2 in Heuristic H2 to reduce the number of negative examples. Unfortunately, Alternative A1 in Heuristic H6 removed some slot instantiators that were very specific. As a consequence, the resulting rule was more general than expected, which contributed to increasing recall from 0.72 to 0.86, but reduced precision from 1.00 to 0.87. The problem regarding precision in the Web MD and the Disney Movies datasets was the same as in the E-Bay Bids dataset: recall was improved at the expense of extracting some false positives, which had a negative impact on precision.

Recall is poor regarding datasets 4 Jobs, Career Builder, and Dr. Scores; the

reason is that the documents in the learning set did not account for enough variability regarding a few slots; as a result, the rules that TANGO learnt were not appropriate for some testing documents, which had a negative impact on recall. In the case of Career Builder, there was an additional problem with the implementation of Heuristic H9: there was a tie when selecting the feature instantiator to extend a rule, and it was broken in favour of a candidate that resulted in a rule that was more specific than it should have been: precision improved from 0.92 to 0.93, but recall worsened from 0.91 to 0.89. In the case of dataset E-Bay Bids, the recall is poor solely due to the six random documents that were selected for the learning set, which did not account for enough variability.

We also went through the cases in which our proposal took more than 100 minutes to learn a rule set. Our first intuition was that there might have been some errors in the annotations. Our experience proves that such errors have an important negative impact on efficiency because TANGO typically explores the search space exhaustively before concluding that no rule can match both the correct and the incorrect positive examples and discard both the correct and the incorrect negative examples. After checking our datasets, we did not find any annotation errors. The problem was simpler: for instance, in datasets Auto Trader and Soccer Base, we had to learn rules for roughly ten different slots, whereas in dataset Amazon Pop Artists there are only two slots, which clearly justifies the extra learning time; in datasets like Trulia, IAF, IMDB, and Soul Films the problem was not with the number of slots, but with the many different formats in which they are rendered, which complicated the search for a rule set that takes them all into account.

5. Related work

In the introduction, we provided a short survey of the many existing proposals to extract information from web documents. In this section, we focus on the proposals that are most closely-related to ours, namely: SRV [13], Irmak and Suel's proposal, L-Wrappers [3], and Fernández-Villamor et al.' proposal. In the following subsections, we compare them to TANGO along the following dimensions: their catalogues of features, their learning procedures, their variation points, and their configuration method; we conclude with some additional miscellaneous comparisons.

5.1. Catalogue of features

For a machine learner to work, it has to be fed with a representation of the input documents that maps them onto a number of features from which it is possible to discern the difference between the information to be extracted and the information to be ignored. It does not make sense to believe that a single catalogue of features can deal with every web site or that it shall be appropriate forever. For instance, there are web sites that use intricate formats to make it difficult for software agents to extract information from them; furthermore, ten years ago it was usual to use HTML tables to render information, which

is nowadays a deprecated technique. As a conclusion, it is expected that the catalogue of features needs to be replaced or at least adapted from time to time. That it is why it is of uttermost importance that it is open and that the proposal is not bound with the specific features that it provides.

SRV relies on a limited catalogue of features that includes some HTML features and a few user-defined ones, but no DOM or CSS features; it also includes relational features to navigate from one token to the next or the previous one, or to the first token of the next column, the previous column, the next row, the previous row, or the header when dealing with tables. SRV features can only be computed on tokens, not on sequences of tokens or nodes. Irmak and Suel's proposal builds on a catalogue that includes a subset of HTML and DOM features, plus some user-defined features; unfortunately, the authors did not provide many details on them. L-Wrappers relies on a unique attributive feature to map nodes onto their corresponding tags and four relational features: next sibling, parent, first child, and last child. As a result, the conditions in the rules basically attempt to classify nodes by means of their tags and the tags of their neighbours. Fernández-Villamor et al.' proposal considers a feature that classifies HTML tags into links, images, and other tags, a subset of DOM features regarding bounding boxes, widths, heights, font size, font weight, and font family (which are narrowed to serif, sans-serif, and monospace families), plus a relational feature that allows to fetch the parent of a node.

TANGO relies on an extensive catalogue of attributive features that includes every HTML, DOM, and CSS feature defined by the W3C recommendations plus user-defined features; the catalogue of relational features includes features to fetch the parent of a node, its ancestors, its children, and siblings. The features are computed on nodes, which means that some of them work on their tokens. The catalogue has been designed so that it can be easily replaced, since there is nothing in our proposal that is bound with the specific features provided by the catalogue. The catalogues of features that are provided by other proposals are very limited. In the case of SRV, it is open and can be replaced because there is nothing in the proposals that is specific to the features in the catalogue; in the case of Irmak and Suel's proposal, the catalogue seems to be open, too, but the authors did not provide many details; in the case of L-Wrappers or Fernández-Villamor et al.' proposal the catalogue cannot be considered open because there are subtle inter-dependencies with the learning procedure.

5.2. Learning procedure

Learning procedures can be top-down or bottom-up. In the former case, the search for rules starts with overly-general rules that match every example in the learning set; it then adds conditions that constraint the examples that are matched, and the process continues until a rule that matches at least a positive example and no negative example is found. In the latter case, the search starts with overly-specific rules that match a single positive example; it then generalises or drops some conditions so that the resulting rules match as many positive examples as possible, and the process continues until no further generalisation is possible. In practice, both approaches have proven to work

well, even though the bottom-up approach has got some criticism regarding information extraction [13]. TANGO is a top-down proposal, so we restrict our attention to SRV and L-Wrappers, which are also top-down.

A difference with regard to SRV and L-Wrappers is that TANGO is intended to learn extraction rules for slots that are structured hierarchically. That is, TANGO can deal with data models in which the information to be extracted is represented by means of records that are composed of attributes or further nested records. In other words, TANGO first learns rules to extract first-level slots and then creates specific learning sets to learn additional rules to extract their nested records or attributes. This approach has proven to work very well in practice because it reduces the size of the learning sets significantly. Furthermore, it is a sensible approach to work with documents that have listings of records, since, otherwise, it would not be easy to identify which slots are nested into which other slots.

There are many additional differences between TANGO and SRV, namely: a) SRV's learning process requires a specific-purpose procedure for each type of condition. Furthermore, it also requires a specific-purpose procedure to generate the first condition in the body of a rule. Such first condition is of the form $some(T, L, F, V)$, where T is a variable that can be bound to a token inside a positive example, L denotes a sequence of relational features that allow to navigate from that token to a neighbour, F denotes a feature, and V a value for that feature. In other words, these conditions are intended to check that a token has a given value for a feature. Unfortunately, if more features of that node have to be constrained, the token must be re-bound. This means that tokens whose features help discern well amongst positive and negative examples need to be rebound several times; an additional intricate implication of re-bounding is that tokens that belong to negative examples and tokens that belong to positive examples cannot be compared regarding their relative positions. This might have a negative impact on efficiency because this requires to search the whole condition space several times, which also includes exploring and evaluating the same conditions several times. TANGO does not require specific-purpose procedures to generate different types of conditions; it relies on a variation point called BRANCH that generates every condition that might possibly be added to a rule; furthermore, TANGO does not consider every possible condition as a candidate, but implements a heuristic to bound the conditions and generate a subset of candidates. Neither does TANGO suffer from the re-binding problem in SRV since a relational feature instantiator can bind any node to a variable, which allows to analyse as many attributive features as necessary in the forthcoming steps. b) SRV has many problems to compute the negative examples. Such examples include every subsequence of tokens in the input documents that is not explicitly annotated as a positive example; the problem is that a document with n tokens has $O(n^2)$ possible subsequences of tokens, which are typically too many to be computed explicitly. As a consequence, SRV has to introduce a hard bias regarding the size in tokens of the negative examples that are considered. In TANGO, computing the negative examples is as easy as fetching the set of nodes that are not explicitly labelled with a user-defined slot. c) SRV

does not take into account any heuristic to select the best candidate conditions to be added to a rule; it just computes their gains and selects the condition that provides more gain. Contrarily, TANGO relies on a variation point since it is not clear which the best heuristic can be. In our experiments, we have proven that the heuristic that we propose is very effective but it can be replaced very easily. d) SRV stops searching for a rule when it finds a solution, independently from how complex it is. TANGO includes a variation point that allows to stop exploring a rule when it becomes too complex. Basically, this prevents TANGO from learning very specific rules that work well on the learning set but do not generalise well in a production setting. SRV only includes a simple heuristic to prevent learning too specific rules: it discards conditions that result in rules that match less than five positive examples, which can be problematic when dealing with detail documents that report on a single item since they typically provide only one positive example of each slot (or a few ones in the case of multi-valued slots, e.g., the authors of a book). e) SRV cannot backtrack from bad decisions. This implies that it has to return the first rule that it finds, even in cases in which there are some candidates that might result in a rule that matches more positive examples. Since the search process is blind and there is not a guarantee that a rule that currently matches more positive examples can actually lead to a solution, SRV has to select the first solution that it finds. Contrarily, TANGO implements a savepoint mechanism that allows it to explore promising rules and backtrack if they are finally found not to be good enough. f) SRV did not take into account that preprocessing the learning set might have an impact on the efficiency, whereas TANGO has proven that reducing the negative examples is appropriate. g) As far as we know, SRV does not post-process the rule sets that it learns, whereas TANGO has proven that post-processing them may result in simpler rules, although it increases the learning time.

There are also many differences with regard to L-Wrappers. Note that this proposal basically consists in mapping the input documents onto a knowledge base and then using the FOIL system [24] to learn extraction rules. FOIL is a general-purpose inductive logic system and it was not tailored to the problem of information extraction; unfortunately, it did not prove to be efficient enough as it was used in L-Wrappers. The authors mentioned that their approach is infeasible in practice when a record has more than two attributes (records are flat tuples in this proposal). Due to this problem, they had to design a complementary approach that learns to extract pairs of attributes and then merges the results into a single rule. The main problem is regarding the exponential explosion of negative examples, which was estimated in the order of $O(n^k)$ for a document with n nodes and records with k attributes. Negative examples are computed by the FOIL system using the Closed World assumption. Unfortunately, this is inefficient because FOIL has to examine every possible instantiation of every possible feature on every possible node; in practice, the authors had to reduce the number of negative examples to roughly 0.10% for their approach to be manageable; it is not clear whether that reduction works well in a general setting because the proposal was evaluated on very few datasets. Merging can alleviate the problem, but does not solve it because it requires to

compute a rule for each of the pairs of attributes in a record. This approach may be problematic insofar missing or permuted attributes and different formattings increase the number of pairs significantly. TANGO learns rules to extract the positive examples independently from each other, which is more efficient and resilient to missing and permuted attributes or alternating formats.

5.3. Variation points

The variation points of a proposal identify the procedures for which different alternatives exist. A priori, it is not possible to make a decision regarding which the best implementation is because it depends on a variety of factors. So it is necessary to identify them, to identify some alternatives, and to have a method to make a decision regarding which the best configuration is.

Unfortunately, none of the proposals that we have surveyed rely on variation points. The authors devised a number of algorithms that were configured to perform as well as possible, but it is not clear at all which one has to be replaced if it is necessary to re-configure the proposals. That is the common theme behind every proposal that we have surveyed, and we think that this is one of the reasons why they tend to fade away quickly and are replaced by new proposals that sprout out continuously.

In TANGO, we have carefully identified a number of variation points, namely: how to pre-process a learning set, how to post-process a rule set, how to select the candidates to extend a rule or to create savepoints, how to score a rule, how to check if a rule is too complex, how to branch a rule into a number of candidate conditions, how to bound candidate conditions, and how to check if a candidate is promising; we have also identified eleven heuristics and many alternatives to implement them. The result is a very flexible learning system.

5.4. Configuration method

Identifying variation points is not enough: it is also necessary to have a method to select the best alternatives to implement the heuristics that they encapsulate. Ours is the only proposal that provides such a method; since the authors of the other proposals did not identify any variation points, it is not surprising at all that they did not work on a configuration method.

Developing a configuration method is not a trivial task, since it has to combine both effectiveness and efficiency measures into a single rank that must take both the mean value and the deviation of each performance measure into account. We have devised a rank that can combine any performance measure and allows the experimenter to set their relative weights to reflect which measures he or she thinks are more important than the others.

5.5. Miscellaneous

Before concluding our insight into the related work, we would like to explore a few more dimensions. They highlight important differences that are not clearly aligned with the previous comparison dimensions, but make our proposal different from the others.

Rules. All of the related proposals learn Horn-like rules, but they differentiate regarding their expressiveness levels.

SRV's rules rely on the following kinds of conditions: checking the length of a token, checking that a token has a given value for a feature, checking that every token in a positive example has a given value for a feature, checking the position of a token, and checking the distance between two tokens. The conditions cannot be negated and slot instantiators are not allowed, which means that the rules cannot be recursive. Irmak and Suel's proposal rely on conditions that can be applied to either element nodes, e.g., checking that the tag is a given one or checking that an attribute has a given value, or text nodes, e.g., checking that it matches a given regular expression or checking that it is the i-th child; it is not clear if their proposal can deal with negated conditions or recursion; neither is it clear if inequalities are allowed. L-Wrappers' rules rely on two types of conditions only: checking whether a node has a given tag and fetching a neighbour; the authors researched regarding using negated conditions and came to the conclusion that they were not useful with their catalogue of features because they only helped identify nodes without a left sibling, i.e., the first child, or nodes without a right sibling, i.e., the last child; furthermore, they did not explore recursion. Fernández-Villamor et al.' rules rely on two kinds of conditions: comparators to constraint the tag, the width, the height, the font size, the font weight, or the font family, and parent instantiators.

The main difference is that TANGO's rules rely on slot instantiators (which allow for recursive rules), feature instantiators (which allow to instantiate any feature in the catalogue, if possible), and comparators (which help constraint the values of attributive features); furthermore, the conditions can be negated.

Evaluation. Unfortunately, none of the most closely-related proposals were evaluated on a sufficiently large number of datasets; neither were they compared using statistically-sound methods.

SRV was evaluated on three datasets and it was empirically compared to two naive baselines by the same author. In L-Wrappers, the authors focused on evaluating their proposal on a single dataset on which it worked reasonably well, but they did not conduct an exhaustive experimentation or an empirical comparison with other proposals in the literature. Irmak and Suel focused on evaluating their proposal on fourteen datasets; four of them had been used to evaluate previous traditional information extractors with which this proposal was compared; the others were gathered from more up-to-date web sites, but they did not conduct an exhaustive experimentation or an empirical comparison with other proposals in the literature; furthermore, the way that they computed the effectiveness of their proposal was not the standard one because they used a so-called verification set that was used to request feedback from the user and correct the extraction rules. Finally, Fernández-Villamor et al. reported on an experimentation with three datasets; no empirical comparison with other proposals was provided.

Contrarily, TANGO was evaluated on 52 datasets and it was empirically compared to six other state-of-the-art proposals; our conclusions were supported by

means of statistically-sound methods that proved that the differences amongst TANGO and the other proposals are statistically significant.

6. Conclusions and future work

In this article, we present TANGO, which is a proposal to learn web information extraction rules for semi-structured web documents. We work within the context of Enterprise Systems Integration, which is a field in which precision and recall are of uttermost importance. TANGO originates from a proposal that we presented elsewhere [25], in the context of a conference on business information systems in which the participants were interested in our solution to help develop software agents that can extract information from web documents to feed automated business processes. There we presented some preliminary ideas on instantiating the general inductive logic programming paradigm to deal with learning web information extraction rules and we got valuable feedback that guided us to devise a new version of our proposal in which the driver was flexibility. The version of TANGO that we present in this article relies on an open catalogue of features, on a learning procedure in which we have identified a variety of variation points, and has a companion method that helps re-configure it when necessary. This makes our proposal clearly deviate from others in the literature, which are monolithic. We have performed an exhaustive experimentation and we have analysed the results using statistically-sound methods; our conclusion is that TANGO can learn rules that are more precise and have higher recall than other state-of-the-art proposals and that it is efficient enough to be used in real-world scenarios.

Our current version of TANGO learns extraction rules for a slot independently from the others. A reviewer suggested an interesting future research direction: it might be a good idea to characterise a slot with regard to the others. The key is to set an order amongst them, so that the extraction rules are learnt according to that order; the rule regarding the first slot must be learnt in isolation, but the succeeding ones might then use the previous ones, which would be re-used as regular Boolean feature instantiators. This might result in a rule like "a node provides an author name if it has some particular features and its parent provides a book title". Intuitively, re-using rules that have been learnt previously might help make our proposal more effective and/or efficient, but there are a couple of issues that require further research, namely: the false positives and false negatives that are produced by a previous rule introduce noise, which may have a negative impact on both effectiveness and efficiency; furthermore, the presence of slot permutations and optional slots may also have unexpected results.

It would also be interesting to do some research regarding how to break ties. Our insight into the difficult cases that we have found in our experimental analysis has revealed that many of them were due to bad decisions regarding candidate conditions that resulted in exactly the same gains. Although it has not been a serious problem, we think that it might be interesting to try to devise a heuristic to break such ties.

Acknowledgements

Our work was supported by the FEDER funds that were awarded by the European Commission and the Spanish and the Andalusian R&D&I programmes with grants TIN2007-64119, P07-TIC-2602, P08-TIC-4100, TIN2008-04718-E, TIN2010-21744, TIN2010-09809-E, TIN2010-10811-E, TIN2010-09988-E, TIN-2011-15497-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.

References

[1] D. Ajwani, A. Cosgaya-Lozano, and N. Zeh. A topological sorting algorithm for large graphs. *ACM Journal of Experimental Algorithmics*, 17(1), 2011. doi: 10.1145/2133803.2330083.

[2] B. Bos, T. Çelik, I. Hickson, and H. W. Lie. Cascading style sheets specification. Technical report, W3C, 2014. URL `http://www.w3.org/TR/2011/REC-CSS2-20110607/`.

[3] C. Bădică, A. Bădică, E. Popescu, and A. Abraham. L-Wrappers: concepts, properties and construction. *Soft Comput.*, 11(8):753–772, 2007. doi: 10.1007/s00500-006-0118-y.

[4] M. E. Califf and R. J. Mooney. Bottom-up relational learning of pattern matching rules for information extraction. *Journal of Machine Learning Research*, 4:177–210, 2003. URL `http://www.jmlr.org/papers/v4/califf03a.html`.

[5] C.-H. Chang and S.-C. Kuo. OLERA: semisupervised web-data extraction with visual support. *IEEE Intelligent Systems*, 19(6):56–64, 2004. doi: 10.1109/MIS.2004.71.

[6] C.-H. Chang, M. Kayed, M. R. Girgis, and K. F. Shaalan. A survey of web information extraction systems. *IEEE Trans. Knowl. Data Eng.*, 18(10): 1411–1428, 2006. doi: 10.1109/TKDE.2006.152.

[7] B. Chidlovskii. Wrapping web information providers by transducer induction. In *ECML*, pages 61–72, 2001. doi: 10.1007/3-540-44795-4_6.

[8] W. W. Cohen, M. Hurst, and L. S. Jensen. A learning system for wrapping tables and lists in HTML documents. In *WWW*, pages 232–241, 2002. doi: 10.1145/511446.511477.

[9] V. Crescenzi and P. Merialdo. Wrapper inference for ambiguous web pages. *Applied Artificial Intelligence*, 22(1&2):21–52, 2008. doi: 10.1080/08839510701853093.

[10] O. Etzioni, A. Fader, J. Christensen, S. Soderland, and Mausam. Open information extraction: the second generation. In *IJCAI*, pages 3–10, 2011. URL `http://ijcai.org/papers11/Papers/IJCAI11-012.pdf`.

[11] J. I. Fernández-Villamor, C. Ángel Iglesias, and M. Garijo. First-order logic rule induction for information extraction in web resources. *International Journal on Artificial Intelligence Tools*, 21(6), 2012. doi: 10.1142/S0218213012500327.

[12] D. R. Ferreira. *Enterprise Systems Integration: a Process-Oriented Approach*. Springer, 2013. doi: 10.1007/978-3-642-40796-3.

[13] D. Freitag. Machine learning for information extraction in informal domains. *Machine Learning*, 39(2/3):169–202, 2000. doi: 10.1023/A:1007601113994.

[14] Y. He and D. Xin. SEISA: set expansion by iterative similarity aggregation. In *WWW*, pages 427–436, 2011. doi: 10.1145/1963405.1963467.

[15] I. Hickson, R. Berjon, S. Faulkner, T. Leithead, E. D. Navara, E. O'Connor, and S. Pfeiffer. HTML 5: a vocabulary and associated APIs for HTML and XHTML. Technical report, W3C, 2014. URL `http://www.w3.org/TR/2014/REC-html5-20141028/`.

[16] A. W. Hogue and D. R. Karger. Thresher: automating the unwrapping of semantic content from the world wide web. In *WWW*, pages 86–95, 2005. doi: 10.1145/1060745.1060762.

[17] C.-N. Hsu and M.-T. Dung. Generating finite-state transducers for semi-structured data extraction from the Web. *Inf. Syst.*, 23(8):521–538, 1998. doi: 10.1016/S0306-4379(98)00027-1.

[18] U. Irmak and T. Suel. Interactive wrapper generation with minimal user effort. In *WWW*, pages 553–563, 2006. doi: 10.1145/1135777.1135859.

[19] M. Kayed and C.-H. Chang. FiVaTech: page-level web data extraction from template pages. *IEEE Trans. Knowl. Data Eng.*, 22(2):249–263, 2010. doi: 10.1109/TKDE.2009.82.

[20] R. Kosala, H. Blockeel, M. Bruynooghe, and J. V. den Bussche. Information extraction from structured documents using k-testable tree automaton inference. *Data Knowl. Eng.*, 58(2):129–158, 2006. doi: 10.1016/j.datak.2005.05.002.

[21] N. Kushmerick, D. S. Weld, and R. B. Doorenbos. Wrapper induction for information extraction. In *IJCAI (1)*, pages 729–737, 1997. URL `http://homes.cs.washington.edu/ weld/papers/kushmerick-ijcai97.pdf`.

[22] T. M. Mitchell, W. W. Cohen, E. R. Hruschka, P. P. Talukdar, J. Betteridge, A. Carlson, B. D. Mishra, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. A. Platanios, A. Ritter, M. Samadi, B. Settles, R. C. Wang, D. T. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. Never-ending learning. In *AAAI*, pages 2302–2310, 2015. doi: 10.3233/978-1-61499-098-7-5.

[23] I. Muslea, S. Minton, and C. A. Knoblock. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems*, 4(1/2):93–114, 2001. doi: 10.1023/A:1010022931168.

[24] J. R. Quinlan and M. Cameron-Jones. Induction of logic programs: FOIL and related systems. *New Generation Comput.*, 13(3&4):287–312, 1995. doi: 10.1007/BF03037228.

[25] A. M. R. Quintero, P. Jiménez, and R. Corchuelo. A novel approach to web information extraction. In *BIS*, pages 152–161, 2015. doi: 10.1007/978-3-319-19027-3_13.

[26] J. Rissanen. The Minimum Description Length principle. In *Encyclopedia of Machine Learning*, pages 666–668. Springer, 2010. doi: 10.1007/978-0-387-30164-8_540.

[27] H. A. Sleiman and R. Corchuelo. A survey on region extractors from web documents. *IEEE Trans. Knowl. Data Eng.*, 25(9):1960–1981, 2013. doi: 10.1109/TKDE.2012.135.

[28] H. A. Sleiman and R. Corchuelo. A class of neural-network-based transducers for web information extraction. *Neurocomputing*, 135:61–68, 2014. doi: 10.1016/j.neucom.2013.05.057.

[29] H. A. Sleiman and R. Corchuelo. Trinity: on using trinary trees for unsupervised web data extraction. *IEEE Trans. Knowl. Data Eng.*, 26(6): 1544–1556, 2014. URL `10.1109/TKDE.2013.161`.

[30] S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3):233–272, 1999. doi: 10.1023/A:1007562322031.

[31] A. Srinivasan. The Aleph manual. Technical report, University of Oxford, 2004. URL `http://www.cs.ox.ac.uk/activities/machlearn/Aleph`.

[32] W. Su, J. Wang, and F. H. Lochovsky. ODE: ontology-assisted data extraction. *ACM Trans. Database Syst.*, 34(2), 2009. doi: 10.1145/1538909.1538914.

[33] J. Turmo, A. Ageno, and N. Català. Adaptive information extraction. *ACM Computing Surveys*, 38(2), 2006. doi: 10.1145/1132956.1132957.

[34] A. van Kesteren, A. Gregor, A. Russell, and R. Berjon. Document Object Model 4. Technical report, W3C, 2014. URL http://www.w3.org/TR/2014/WD-dom-20140710/.

[35] R. C. Wang and W. W. Cohen. Iterative set expansion of named entities using the web. In *ICDM*, pages 1091–1096, 2008. doi: 10.1109/ICDM.2008.145.

[36] Y. Xia, Y. Yang, S. Zhang, and H. Yu. Automatic wrapper generation and maintenance. In *PACLIC*, pages 90–99, 2011. URL http://www.aclweb.org/anthology/Y11-1010.