TAPON: a two-phase machine learning approach for semantic labelling

Daniel Ayala[a,*], Inma Hernández[a], David Ruiz[a], Miguel Toro[a]

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

Abstract

Through semantic labelling we enrich structured information from sources such as HTML pages, tables, or JSON files, with labels to integrate it into a local ontology. This process involves measuring some features of the information and then finding the classes that best describe it. The problem with current techniques is that they do not model relationships between classes. Their features fall short when some classes have very similar structures or textual formats. In order to deal with this problem, we have devised TAPON: a new semantic labelling technique that computes novel features that take into account the relationships. TAPON computes these features by means of a two-phase approach. In the first phase, we compute simple features and obtain a preliminary set of labels (hints). In the second phase, we inject our novel features and obtain a refined set of labels. Our experimental results show that our technique, thanks to our rich feature catalogue and novel modelling, achieves higher accuracy than other state-of-the-art techniques.

Keywords: Semantic labelling, Information Integration, Machine Learning

1. Introduction

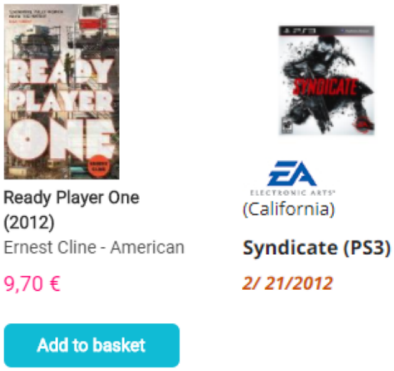
Semantic labelling consists in endowing information from structured data sources (such as processed HTML pages [27], JSON, CSV or XML files) with labels that denote the semantic class of each element. A semantic class represents a known information type that can be used to integrate heterogeneous information into a local schema. Being able to identify which class some information belongs to has a variety of uses, such as ontology matching in extensional techniques [6], information fusion [30], information verification [13], information extraction [12], or wrapper repair [4]. Some possible sources of semantic classes are: datatype and object properties from an OWL ontology (which is the most prominent source in the related work), database table and attribute names, and HTML table headers.

In order to exemplify semantic labelling and introduce the concepts related to information structures to which we refer in this paper, Figure 1 shows very similar information in three possible source formats: an HTML document, a table, and RDF data, corresponding to information about a book and a videogame respectively[1]. In Figures 2(a) and 2(b) we have represented this information in generic tree-like structures, in which there are structural elements we call records (e.g., videogames, books, or authors) as well as textual elements that we call attributes (e.g., names, titles, or prices). Records correspond to the intermediary nodes of the tree and have no textual value, while attributes correspond to the leaf nodes of the tree and have a literal attached to them (which may be numeric). Additionally, Figure 2(c) depicts information whose semantic class is unknown. The purpose of a semantic labelling technique would be to label this information with known semantic classes (which are, in this toy example, restricted to those in the figures). For example, such technique could label attribute `$C2` as a date after observing that it has two slash characters, a feature that is typical of dates. The input can vary depending on the technique used to label: it can be individual instances, datasets, or groups of the former.

Semantic labelling techniques apply models that, in one way or another, measure the value of some

*Corresponding author
Email addresses: dayala1@us.es (Daniel Ayala), inmahernandez@us.es (Inma Hernández), druiz@us.es (David Ruiz), migueltoro@us.es (Miguel Toro)

[1] schema = prefix for http://schema.org



(a) HTML elements.

Title	Publisher	Headquarters	Date	Platform
NHL 13	Electronic Arts	California	9/11/2012	PS3
Syndicate	Electronic Arts	California	2/21/2012	PS3
...	...	...	...	...

Title	Author	Nationality	Year	Price
Milk and Honey	Rupi Kaur	Canadian	2016	13,99€
Ready Player One	Ernest Cline	American	2012	9,70€
...	...	...	...	...

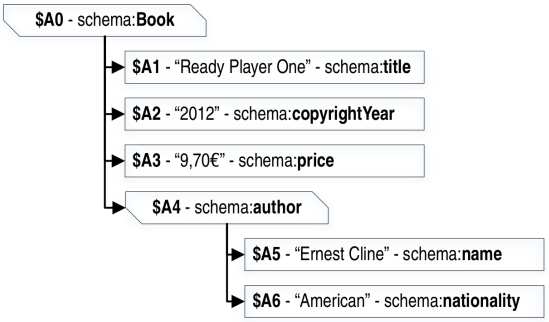
(b) Tables.

```
<book1> <type> <Book> .
<book1> <hasTitle> "Ready Player One" .
<book1> <hasAuthor> <author1> .
<author1> <type> <Author> .
<author1> <hasName> "Ernest Cline" .
<author1> <hasNationality> "American" .
<book1> <hasYear> "2012" .
<book1> <hasPrice> "9,70 €" .
```

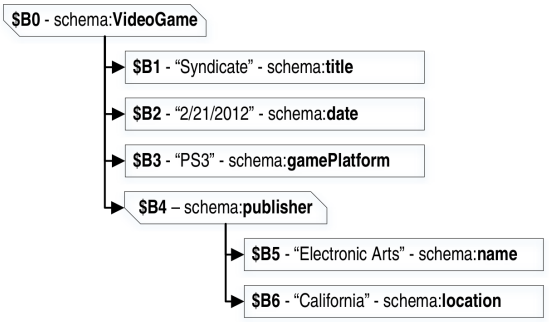
```
<game1> <type> <Videogame> .
<game1> <hasTitle> "Syndicate" .
<game1> <hasPublisher> <publisher1> .
<pub1> <type> <Publisher> .
<pub1> <hasName> "Electronic Arts" .
<pub1> <hasLocation> "California" .
<game1> <hasDate> "2/21/2012" .
<game1> <hasPlatform> "PS3" .
```

(c) RDF triples.

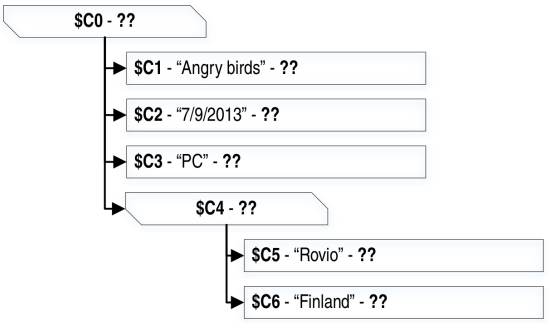
Figure 1: Similar structured information in different source formats.



(a) Book record.



(b) VideoGame record.



(c) Unlabelled record.

Figure 2: The information from several sources, as records and attributes.

features of the information in order to find which are the semantic classes that best describe it. Some techniques define a feature catalogue with some aspects of the information that can be measured to model that information, such as the number of uppercase letters or the cosine similarity between two pieces of text [13, 14, 15, 18, 20, 30]. Other techniques do not define this catalogue explicitly [16, 32, 19, 24, 23], but ultimately rely on some kind of measure. For example, Ramnandan et al. [24] perform Lucene queries, which are based on TF-IDF scores, and Ritze et al. [25] use queries to knowledge bases to obtain candidate classes, the queries being ultimately based on TF-IDF scores or similar measures.

The main problem that we have observed in these techniques is that the features they take into account are mostly, if not exclusively, based on the textual value of attributes (which may be numeric or not). These features, such as the occurrence of a string that matches a regular expression or the value of a numeric value, can be enough to discriminate certain cases, such as dates, which have a quite distinct textual format, or month numbers, which follow a clear numerical distribution. However, they may not be enough to distinguish highly similar classes, or even to assign labels to records. For example, in Figure 2(c), a semantic labelling technique that was only based on text-based features would not be able to label records `$C0` and `$C4` because they are structural elements without

textual features, and it may have problems discerning whether $C1 is a name or a title. Even if the technique computed some simple structural features from the instances, such as their depth in the dataset, it would be impossible to distinguish between videogames and books, or authors and publishers, since they are structurally identical.

To deal with this problem, we have devised TAPON, a new two-phase, supervised, machine-learning semantic labelling technique. Our main contribution is the application of novel features in two phases that go beyond simple text-based and structural features. In the first phase (the hint-free phase), we compute simple features that can be computed on attributes or the structure of the records, such as the letter density or the number of attributes they have. These features are used to obtain a set of temporary labels we call hints. In the second phase (the hint-based phase), the presence of hints allows us to compute, in addition to the former features, novel features that model the relationships between classes. After computing these features, we refine the labels with the added information. Note that even though our technique is supervised, the increase in availability of open data sources makes it trivial to obtain large amounts of labelled information.

Figure 3 represents the workflow of our technique. Observe how, in Figure 2(c), we can easily label $C4 as a publisher by measuring its distance to the nearest instance labelled as a gamePlatform, which is small in the case of publishers and infinite in the case of authors (since there is no instance labelled as such anywhere near authors). Feature "minimum distance to class C" is a hint-based feature, which measures the distances to instances of every known class, some of them useful to the classifier.

Thanks to our two-phase approach, TAPON achieves better labelling precision in the cases in which the aforementioned relationships between classes help classify problematic classes. When that is not the case, precision does not decrease.

In order to empirically prove the effectiveness of our technique, we have carried out experiments in which TAPON was used to label a number of real-world datasets, comparing its performance with the performance of labelling techniques based on the techniques by Kushmerick [14], Ramnandan et al. [24], Soru and Ngomo [30], Pham et al. [23], and Neville and Jensen [21]. Our experiments focus on an application scenario in which datasets are labelled individually and have arbitrary, unknown structures. These situations are common, chiefly in web environments [5]. The results of our experiments show that TAPON is able to achieve better performance than other techniques in the scenario that we have described, with the iterative application of our novel features playing an important role. We train our technique with data from linked open data repositories, which provide a large amount of labelled information. However, TAPON is source-independent and takes the generic structures we have described as input.

The rest of the article is organised as follows: Section 2 contains a detailed description of TAPON; Section 3 describes the relevant techniques we have identified in the literature, as well as their problems; Section 4 presents our experimental results regarding both labelling accuracy and training time of TAPON; finally, Section 5 summarises our work and contributions to the state-of-the-art.

2. Our proposal

TAPON builds on a two-phase, machine-learning technique. A classifier is learnt using our large feature catalogue and then applied to label unlabelled instances. It first learns a classification model in which no features take the relationships between classes into account; the labels obtained by this model are then used as hints in a second phase, in which a new model is enriched with additional features that are based on the hints. Our hypothesis is that the hints and the information related to relationships between classes (included through the additional features) help create an improved model that can be used to provide more accurate labels. Our experimental results prove this hypothesis to be true, as we illustrate in Section 4.

Next, we first introduce some preliminaries, then a specification of the circumstances under which our technique can be applied, then a detailed description of our technique, and finally an application example shown in Figure 4. This example is also used to exemplify concepts in the preliminaries.

2.1. Preliminaries

Dataset: a structure with data arbitrarily organised as a set of records or attributes with explicit relationships. We denote the set of all possible datasets by $\mathcal{D}$. Records have other

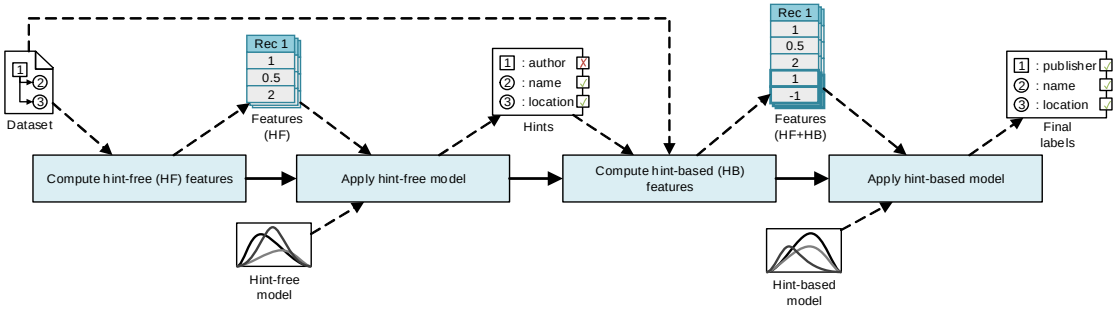


Figure 3: Workflow of TAPON.

records or attributes. Attributes have a textual value. Both records and attributes are instances of a class (in some contexts there may be more than one) and can be endowed with a label that denotes it but can be erroneous. If a label is temporary in the labelling process and is still subject to changes, it is considered a hint.

EXAMPLE: The dataset in Figure 4 contains a schema:VideoGame record. This record has four attributes and another record. These belong to classes schema:title, schema:date, schema:gamePlatform, and schema:Publisher, which has a schema:name and a schema:location. Some labels are erroneous: the title attribute has been labelled as a "schema:name", the schema:publisher record has been labelled as an "schema:author", and the schema:VideoGame record has been labelled as a "schema:Book". These labels have been given in an intermediary step of the process and are temporary. Therefore, they are considered hints.

Features: functions that take as input an instance and output a numeric value. They include any measurement that can be taken from the dataset and is considered to be relevant. Features can be applicable to records, attributes, or both. We denote the set of all possible features by $\mathcal{F}$. If a feature requires the presence of hints in order to compute its values, we call it a hint-based feature. Otherwise we call it a hint-free feature. We denote the set of all possible hint-free features by $\mathcal{F}_{HF}$, and that of hint-based features by $\mathcal{F}_{HB}$.

EXAMPLE: Features can be as simple as computing the number of characters from an attribute, more elaborate, such as the number of attributes of a record, or even include complex statistical computations such as the average number of letters among attributes within a 3 steps distance of a record with a digit density greater than 5%. The aforementioned features (number of characters, etc.) do not require the presence of a hint to be computed, and thus are hint-free features. An example of a hint-based feature would be the number of instances with the label schema:gamePlatform contained by a record. We can not compute this feature unless instances are already labelled with a hint that identifies some of them as a schema:gamePlatform.

Model: a feature-based characterisation of classes learnt from the instances in a set of labelled datasets, which can be applied to any instance in order to infer its class. The representation is based on a set of features that must be computed when the model is trained or applied. We denote the set of all possible models by $\mathcal{M}$ and their training by means of function $learnModel : \mathbb{P}(\mathcal{D}) \times \mathbb{P}(\mathcal{F}) \rightarrow \mathcal{M}$. When a model is applied to a dataset, it predicts the class of each instance. We denote by means of function $applyModel : \mathcal{M} \times \mathbb{P}(\mathcal{D}) \times \mathbb{P}(\mathcal{F}) \rightarrow \mathbb{P}(\mathcal{D})$ the application of a model to a set of datasets using a certain set of features, returning the same set of datasets with the new labels. Note that the set of features used to learn and apply the model must be the same.

EXAMPLE: The models that are learnt in our experiments are one-vs-all random-forest classifiers that use the feature catalogue in Tables 1 and 2.

2.2. Application

With regards to the application of TAPON, we make the following assumptions: information to be labelled is structured; the input during each application is a single dataset; datasets to be labelled may contain any number of arbitrarily structured records and attributes; all records and attributes in an input dataset must be labelled; datasets must be labelled in an individual basis, without assuming that there are several datasets with grouped instances that are known to share the same class and can be labelled at the same time; finally, only records and attributes are labelled, not relationships. While TAPON supports hierarchical structures, this does not limit its application to deep or complex structures, since it would be trivial to represent a tuple with a flat structure that only has attributes at the same level.

A good example of this scenario is unsupervised information extraction [3, 11? , 28], which consists in extracting useful structured information from Web pages through heuristics instead of learning rules from labelled examples. This allows such extractors to obtain information in web-scale, with a single pass, and without human interaction. In this context, semantic labelling would be needed to give semantics to the information, which could range from generic nouns in triples to structured information in a JSON file. The datasets obtained from these extractors must be labelled individually, since they have variable structures. These structures are very common in the Web, where there are fields of variable length, or optional fields [5, 28]. For example, one dataset could contain a record with three attributes, while another one could contain a record with two attributes and another record. There is no known correspondence between the classes of the instances, so they can't be grouped and each dataset must be labelled individually.

In a similar vein, TAPON can also be used to perform information verification [13, 15, 18], which consists in verifying that the extraction rules applied to HTML pages keep working properly (for example, by making sure that the instances of a class keep a certain textual format). This verification is performed in an individual basis, and both records and attributes are subject to it. TAPON can be trained from labelled examples that are known to be correct and then applied to extracted information to verify the labels. For example, let's suppose that we have created a web wrapper that periodically extracts information from a page about an author to monitor new publications. The wrapper uses selectors and x-paths to extract information about every publication, such as publication b (bold elements inside publications) to obtain the year of publications. However, after some time, the format of the page changes, the title is now the element in bold, and consequently the titles are extracted as years. If we trained TAPON from a first set of correctly extracted datasets, we could apply it to new extractions and confirm that data is labelled correctly. In the former example, the elements assumed to be years would be labelled as titles, detecting the error.

TAPON can also be used to perform ontology matching at an extensional level [6] by using the resulting labels to measure similarity between classes. Let's suppose we want to match classes in ontology A, which include "author" and "publication title", with classes in ontology B, which includes "name" and "title". We train TAPON using datasets from A, and then apply it to label datasets from B. Then we observe that, most times, "name" attributes are labelled as "author", and "title" attributes as "publication title". This means that, from the point of view of the model, they are highly similar and they are a potential match. We can customise the features used by the classifier in order to change how similarity is measured. Semantic labelling somehow overlaps with schema matching, since both approaches deal with information integration. Schema matching consists in identifying the relationships that exist between elements in two schemas [17]. Therefore, it inherently involves analysing features from those schemas, such as attribute names and types, explicit data constraints, or structural properties of the schemas. While in some cases schema matching techniques may use statistics collected from data instances as features to assess how similar two classes in a schema are, the focus of these proposals is not to select the best features to characterise each class, but to identify the relationship between classes . By contrast, in semantic labelling scenarios, the schemas behind the data are not always available, which means that proposals in this area focus on selecting the best features from data instances to characterise each class, disregarding the schema. Techniques to semantically label pieces of data have a variety of applications, one of which could be schema matching, i.e., a schema matching proposal could rely on a semantic labelling technique to characterise each class in a pair of schemas and infer that there is a relationship between two

of those classes.

Regarding the format of inputs, TAPON takes generic JSON datasets as inputs. This way it can be applied to data from any source after converting it. The conversion is trivial: in tables or CSV files each row would be a record and each column an attribute. In RDF data, URI resources would be records and literals would be attributes.

2.3. Description

TAPON is based on the concatenation of two classification models that use different features sets in their corresponding phases. The idea behind the concatenation of models is that the application of the first model provides additional information that is used by the second model, in a similar way to iterative techniques such as PageRank [22]. The first model uses the hint-free features in Table 1, and it is applied to unlabelled instances to infer their class. Since hints are not needed in this phase, we refer to it as the hint-free phase. The second model uses additional hint-based features shown in Table 2 (in addition to those used by the first model). These take the output of the first model into account as a set of hints, and can potentially help distinguish classes with large similarity. We refer to this phase as the hint-based phase. Note that hint-based features always have a class as a parameter, which is automatically instantiated for every class in the training datasets.

Regarding models, our technique uses one-vs-all classifiers where the aggregation of the binary classifiers is done by means of an additional multiclass classifier that uses the outputs of the binary classifiers as features. Each binary classifier returns the probability of an instance belonging to its class. The multiclass classifier combines their outputs to infer a final label. This way, we may have a schema:name classifier, a schema:date classifier, and a schema:price classifier, among others. These would return the probability of the instance belonging to each class, which would be used by the multiclass classifier to infer which one is the most accurate label for the instance. This approach is more sophisticated than taking the class with the highest probability, and allows using classifiers with binary outputs (1 or 0 for each class, instead of a probability).

The results from binary classifiers endow each class with a probability, which is useful if we want to enrich labels with a probability. This probability can help assess the reliability of the assigned labels. In cases in which the instance to be labelled belongs to a class that was not present in the training datasets, this probability might be low, showing that the label might not be accurate enough. In this scenario, the user could set a threshold to ignore labels with a probability below it. We consider the problem of labelling information that does not belong to a training class to be beyond the scope of this article. Nevertheless, we ensure that our technique is still able to label that information, and that a probability can be provided to help discern good labels from poor labels.

There is a downside to our one-vs-all approach: the creation of a binary classifier per class may be detrimental when there is a significantly large number of different classes (such as several hundreds of classes). This is a limitation of TAPON, and in these cases, we recommend using lighter classification techniques or using other labelling techniques conjointly with ours. However, this case is not typical in the scenario we focus on, since data is usually integrated into a local model focused on a domain with a few dozens of classes.

The learning process by which the two models are obtained is also based on the addition of features. It can be summarised in the following steps in Process 1: create the hint-free model using hint-free features (step 1); apply it to obtain hints (step 2); create the hint-based model, adding hint-based features (step 3). Note that hint-based feature values are computed from the hints obtained by the first model, and not from the training labels that are known to be true. This way, we avoid situations in which the hint-based model does not work properly

Process 1 Learning models.

1: **Input**
2: $D : \mathbb{P}(\mathcal{D})$ –Set of labelled datasets
3: $F_{HF} : \mathbb{P}(\mathcal{F}_{HF})$ –Set of hint-free features
4: $F_{HB} : \mathbb{P}(\mathcal{F}_{HB})$ –Set of hint-based features
5: **Output**
6: $m_{HF} : \mathcal{M}$ –Hint-free model
7: $m_{HB} : \mathcal{M}$ –Hint-based model
8:
9: – Step 1: hint-free model creation
10: $m_{HF} \leftarrow learnModel(D, F_{HF})$
11: – Step 2: hints assignation
12: $D \leftarrow applyModel(m_{HF}, D, F_{HF})$
13: – Step 3: hint-based model creation
14: $m_{HB} \leftarrow learnModel(D, F_{HF} \cup F_{HB})$

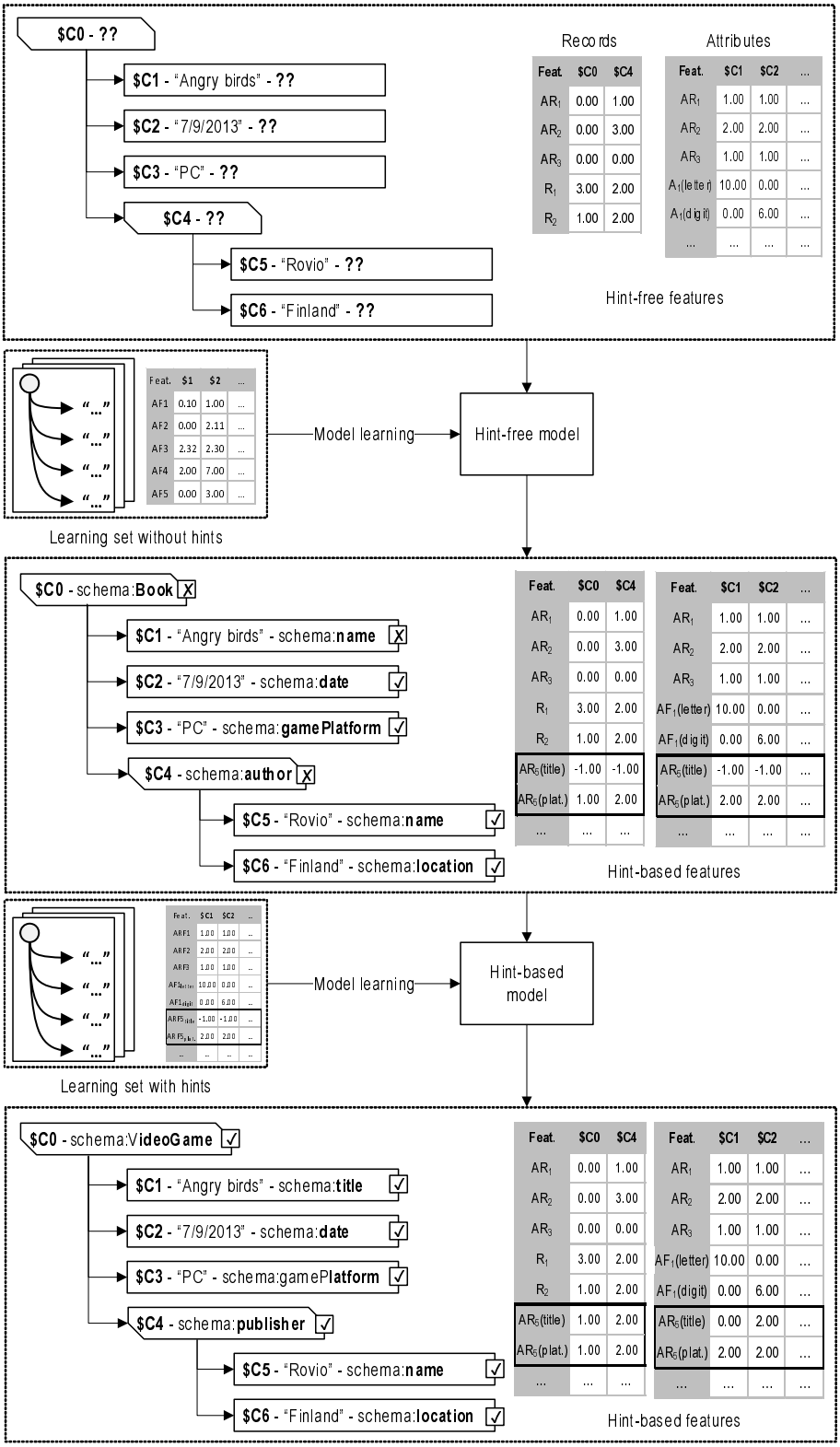


Figure 4: Hint-free and hint-based phases. Hint-based features have been highlighted.

ID	Feature	Description	Scope
$A_1(S)$	Number of occ. of symbol type S	The number of occurrences in the attribute of symbols of type S (letters, numbers, punctuation, symbols, separators, other). The considered types can be customised.	A
$A_2(T)$	Number of occ. of token type T	The number of occurrences in the attribute of token of type T (words starting with a lowercase letter, words starting with an uppercase letter followed by a non-separator character, uppercase words, numeric strings, HTML tags). The considered types can be customized.	A
$A_3(S)$	Density of smbol type C	The density in the attribute of symbols of type S. The density is computed as the number of occurrences of a character type divided by the total number of symbols in the attribute.	A
$A_4(T)$	Density of token type T	The density in the attribute of token of type T. The density is computed as described in AF3	A
$A_5(C)$	Average shared prefix length for class C	Average length of the shared prefix between the text of the attribute and a set of stored examples of class C. The shared prefix is the set of characters that two attributes have in common in the beginning. If the attributes start with a different character, the length is 0.	A
$A_6(C)$	Average shared suffix length for class C	Average length of the shared prefix between the text of the attribute and a set of stored examples of class C. The shared suffix is the set of characters that two attributes have in common in the end. If the attributes end with a different character, the length is 0.	A
$A_7(C)$	Average edit distance to class C	Average Jaro edit distance between the attribute and a set of stored examples of class C.	A
A_8	Numeric Value	The numeric value of the text of the attribute if it matches a number pattern. -1.0 otherwise	A
$A_9(C)$	Index score of class C	The score obtained by the document that represents class C when querying an index by using the attribute as query. This technique is described in [18].	A
R_1	Number of attributes	The number of attributes of the record.	R
R_2	Number of records	The number of records of the record.	R
AR_1	Depth	The depth in the dataset of the instance.	A & R
AR_2	Number of same level attributes	The number of attributes at the same structural level as the instance.	A & R
AR_3	Number of same level attributes	The number of records at the same structural level as the instance.	A & R

Table 1: Hint-free features. A = Attributes. R = Records.

ID	Feature	Description	Scope
$R_3(C)$	Number of instances of class C in the record	The number of instances labelled with class C contained in the record.	R
$R_4(C)$	Density of instances of class C in the record	The density of instances labelled with class C contained in the record. The density is computed as the number of instances labelled with class C divided by the total number of instances.	R
$AR_4(C)$	Density of same level instances of class C	The density of instances labelled with class C among the instances at the same structural level	A & R
$AR_5(C)$	Minimum distance to class C	The minimum distance in the dataset from the instance to an instance labelled with class C. Distance is meaured as the minimum number of edges that must be traversed to get to an instance.	A & R

Table 2: Hint-based features. A = Attributes. R = Records.

because the hints predicted by the hint-free model differ too much from the actual classes that were used to train the hint-based model. Thus, the hint-based model learns to infer labels that are based on the apparent class of the instances rather than on the actual ones, which are not available when it is applied in a real world scenario.

Once both models have been created, they are concatenated in the semantic labelling process as shown in Process 2. During the hint-free phase, the output of the hint-free model is used as a set of hints (step 1); during the hint-based phase, the hints are taken into account by the hint-based model, whose output is the final label (step 2).

Each model can be implemented using any existing classification technique, including well-known classification techniques such as ensemble classifiers, random forests, logistic regression or neural networks.

Since record and attribute classes are disjoint, and a class can not be shared by both records and attributes, we create separate models for records and attributes, using different feature sets (though they may share features, like feature AR_1 in Table 1, which can be applied to both attributes and records). Record models are learnt using only feature vectors computed from record instances, whereas attribute models are learnt only from fea-

Process 2 Semantic labelling.

Input
  $D : \mathbb{P}(\mathcal{D})$ –Set of unlabelled datasets
  $m_{HF} : \mathcal{M}$ –Hint-free model
  $m_{HB} : \mathcal{M}$ –Hint-based model
  $F_{HF} : \mathbb{P}(\mathcal{F}_{HF})$ –Set of hint-free features
  $F_{HB} : \mathbb{P}(\mathcal{F}_{HB})$ –Set of hint-based features
Output
  $D : \mathbb{P}(\mathcal{D})$ –Set of labelled datasets

  – Step 1: hint-free model application
  $D \leftarrow applyModel(m_{HF}, D, F_{HF})$
  – Step 2: hint-based model application
  $D \leftarrow applyModel(m_{HB}, D, F_{HF} \cup F_{HB})$

ture vectors computed from attribute instances. Note that this does not prevent record feature vectors from containing features that are based on attributes and vice versa. For example, feature RF3 in Table 2 is only applied to records, but it measures the number of instances in a record that belong to each class, including attribute ones.

After the hint-based phase, we can use the resulting labels as hints again, which may possibly alter some of the hint-based features values. Consequently, we might apply the hint-based model again to refine the set of labels by repeating the hint-based phase. This results in an iterative process that continues until the set of labels is stable (or a maximum number of iterations is reached). However, our experiments show that repeating the hint-based phase does not improve the results in a significant way; actually, with some classification configurations, they get worse. This is probably caused by the fact that, after the second phase, hints differ too much from those in the training examples of the second model, which correspond to the labels after one iteration, not two. Some alternatives, such as training the second model with the actual classes as hints, or training further models, could make further iterations more useful, but the thorough study of how each of these alternatives behave is beyond the scope of this paper.

Lastly, while we have devised a complete, independent semantic labelling technique, its hint-free phase could as well be replaced by any other existing labelling technique, as long as it provides a label that can be used to compute hint-based features in the hint-based phase. Therefore, our technique is able to integrate state-of-the-art labelling techniques, such as Ramnandan et al. [24]'s or Neumaier et al. [20]'s.

2.4. Application example

To illustrate our technique, we use as an application example a scenario in which there are two data sources, related to the domains of videogames and books. For the sake of simplicity, the only existing classes are those in Figure 2. Videogames have a title, a date, a platform and a publisher with a name and a location. Books have a title, a year, a price, and one or more authors with a name and a nationality. Some of the classes, such as the classes schema:schema:VideoGame and schema:Book, are used in structurally identical records.

In order to create the models in Figure 4, first, we create the hint-free model by learning from the feature vectors of the instances in a learning set. We only compute the features in Table 1. Afterwards, we apply these models to label the instances in the learning set (the same instances that were used to learn the hint-free model), endowing them with a hint that allows us to compute the additional features in Table 2 and create the hint-based models.

Then, we apply the hint-free model in the hint-free phase to label the instances of a dataset that contains one schema:VideoGame record. We initially assign label ?? to every instance to emphasise that their classes are unknown. Since we have applied the hint-free models, we only compute the hint-free features, which may have paremeters that are instanced several times, as is the case with A_1. After this phase, the instances are given a label. The labels schema:Book, schema:Name, and schema:author are incorrect, since the actual classes of $C0, $C1, and $C4 are schema:VideoGame, schema:title, and schema:publisher. The incorrect label of $C0 is due to the similarity between classes schema:VideoGame and schema:Book, and the same happens with $C1 (similarity between classes schema:title and schema:name, since they have a similar format) and $C4 (similarity between classes schema:publisher and schema:author, since they are structurally identical). Note that these labels are not the final labels; they are mere hints that are corrected in the next phase.

Thanks to the presence of hints, we can compute additional hint-based features and apply the hint-based models in the hint-based phase to correct the former hints. The hint-based models are more complex, and take into account features that help tell the schema:VideoGame and schema:Book classes

apart, as well as the schema:name and schema:title classes, and the schema:publisher and schema:author classes. For example, the distance to an attribute of class schema:gamePlatform can help correct all incorrect labels, since in the case of books, such distance is infinite. This reduces similarity between classes and increases the quality of the models, that are now able to correctly label all instances. Note that we create instances for each known class. Some of them will be useful, such as the distance to a platform, while some may not add useful information.

3. Related work

Related work includes semantic labelling techniques, as well as other techniques that were not devised with semantic labelling in mind, but that share the same principles as semantic labelling ones, i.e., creating models that provide a probabilistic view of the information. In the particular case of semantic labelling, this probabilistic view is used for classification. Proposals from other areas use it for different purposes, but the models they create are useful and relevant to semantic labelling to the point where some of the techniques can be used for semantic labelling by merely feeding the features they describe to a classifier. For this reason, we study them along with semantic labelling techniques, and refer to them, overall, as information modelling techniques.

After surveying the literature, our conclusion is that the existing techniques for information modelling can be broadly classified into hard [26] and soft [13, 14, 18, 16, 32, 19, 24, 23, 20] techniques. The former learn a structural model for a dataset, that is, a set of properties including their type (integer, date, string, etc.) and multiplicities (1, 0..1, 0..*, etc.). Soft modelling techniques go a step further since they aim to project the datasets onto a feature space which allows to infer a probabilistic representation of the information.

Using these features, it is possible to model differences between classes that would go unnoticed using hard techniques, since features can cover a wide range of aspects that are not usually taken into account by hard modelling, such as the proportion of punctuation characters in a piece of text. For example, the textual value of the price attribute in Figure 2(a), "$9.70", is a string, and thus could be considered a title by a hard model, although it is a price; and a book with 400 authors would not break the cardinality restriction 1..*, so its structure would be considered correct, although it is almost certainly incorrect.

We focus our study on soft techniques, since they are able to model the subtle differences between a wide variety of classes, which is key to perform web scale information modelling and labelling, in which there are many similar information classes that are difficult to tell apart. We have identified three groups of related techniques: techniques used for information verification, i.e., learning a model used to verify labelled datasets by identifying those datasets that contain errors [13, 14, 15, 18], techniques used for semantic labelling, that is, labelling information with known classes so that it can be integrated into a known schema [16, 32, 19, 24, 23, 20], and techniques developed in the linked data context [30], focusing on ontology integration and linking tasks.

Regarding information verification, Kushmerick [13, 14] devised a probabilistic technique that models features as Gaussian random variables. Lerman et al. [15] create a vector with the average of several features computed from the instances of a correct dataset. The vector is later compared to the vector computed from new datasets using the χ^2 goodness-of-fit test. McCann et al. [18] improved on Kushmerick's technique by normalising probabilities, assigning weights to features, and adding noise using hand-crafted perturbations. The goal of the previous techniques is to raise an alarm when a new, unverified dataset deviates significantly from the information used to learn the model. The verification only relies on text-based features and the number of tuples in the dataset, and the relationships between classes are not considered. Only Kushmerick [14] can deal with any arbitrarily structured information, whereas the others deal with tuples only. Their models do not take records into account.

Regarding semantic labelling, Limaye et al. [16] use objective function maximisation to find the label assignment that is more consistent with the information in a dataset. Venetis et al. [32] use Bayesian classification to find the most likely label for a group of attributes using a database of samples. Mulwad et al. [19] use an incremental message propagation algorithm to label groups of different attributes with a record class and each individual attribute with a named entity using Wikitology as an external knowledge base. Ritze et al. [25] use an iterative process to label record named entities

in HTML tables and their attributes by mapping them to DBpedia types and properties. In a similar vein, Zhang [33] iteratively labels records that correspond to named entities and their attributes from HTML tables while using information from the HTML context of the tables to create a richer representation of cells and columns. Ramnandan et al. [24] use distribution equivalence tests when numeric values are detected, as well as a Lucene index when non-numeric ones are detected: they store attributes as examples and label new groups of attributes by using them to query the index (this technique was later used in [31]). Pham et al. [23] expand on this idea by enlarging the feature catalogue, introducing similarity measures. Neumaier et al. [20] focus exclusively on numerical values and compute feature vectors from sets of numbers. The majority of these techniques do not define a set of features, but rely on queries to knowledge bases. The relationships between classes are not modelled. Only Ramnandan et al. [24], Pham et al. [23] and Neumaier et al. [20] deal with any arbitrarily structured information. They do not label records beyond the first one (the one that would represent the entire row in a table), only attributes, since they assume that records always correspond to named entities, while they can be structural elements that group other instances, such as dates or addresses. Limaye et al. [16], Venetis et al. [32], Mulwad et al. [19],Ritze et al. [25], and Zhang [33] are limited to named entities, e.g., people or places. Finally, since all semantic labelling techniques label groups of instances, they assume that information is grouped into bags of examples that are known to share the same class.

Regarding linked data, Soru and Ngomo [30] survey several classification techniques which are applied to feature vectors representing pairs of records. Each feature vector contains several features related to cosine distance and edit distance. Though these techniques can be applied to information with any structure, the models they create are limited to text-based features, such as cosine distance, and therefore do not model the relationships between classes.

Finally, Neville and Jensen [21] introduce the idea of iteratively applying a classification model that, in each step, computes features based on the labels assigned in the former step, thus refining the labels in each iteration. This technique is similar to ours, since it performs iterative labelling by computing features that change after each iteration. However, there are substantial differences: TAPON has a two-phase application, as opposed to Neville and Jensen, that only has one which would correspond to the hint-based phase; TAPON is trained with hint-based features that have been computed from inferred labels instead of actual classes, since during application actual classes are not available; finally, TAPON focuses on semantic labelling in a multi-domain context. Overall, TAPON focuses on the specific task of semantic labelling of records and attributes, which led us to devise a more optimised technique and feature catalogue.

4. Experimental analysis

Our experiments consist of performing semantic labelling on both records and attributes in a multi-source scenario with real-world information. They aim to prove that, in the application scenario we focus on, TAPON is able to model and label classes with higher accuracy than other state-of-the-art techniques, achieving improved F_1 score thanks to our large feature catalogue with more complex features, that is significantly enriched by the addition of hint-based features that reflect the relationship between classes of instances. The different techniques are applied to a testing dataset in order to compare the inferred labels to the actual ones.

4.1. Setup

We have used Spark [7]'s random forest classifiers implementation for the creation of binary classifiers and its multilayer perceptron classifiers implementation for the creation of multiclass classifiers, since other configurations yield worse results in the informal tests that we have performed, and Soru and Ngomo [30] suggest that these classifiers yield good results in data linking tasks, which are similar in nature to semantic labelling. Our configuration of the random forest classifiers include the creation of 100 trees with a maximum depth of 40, Gini as the impurity measure, and a minimum information gain of 0.01. Our configuration of the perceptron classifiers include performing 500 iterations with no hidden layers. The other parameters were set to their defaults.

We have used datasets from 10 different linked data sources obtained from RKB explorer [8], an infrastructure for linked data providers whose collection of datasets have already been used for testing elsewhere [29, 1, 10]. Note that we have used

linked data datasets for convenience, since they are a good source of labelled, structured information; however, the datasets were transformed into the generic structures we have described. Our datasets contain a total of 3155 records belonging to 24 different classes, and a total of 7736 attributes belonging to 53 different classes with 3483 unique textual values. Classes have, on average, 141.44 instances, with a standard deviation of 162.20. Records have, on average, 0.72 records and 2.45 attributes. 29% of attributes have a numeric value; datasets have, on average, 2.73 levels of depth (2 levels corresponding to a flat record), and a maximum of 4 levels.

The specific datasets that were used are: DBLP articles, NFS awards, EPSRC grants, Telegraphis countries, Restaurants, European Patent Office classifications, Edubase schools, DigitalEconomy profiles, Dev8d programmes, and Courseware books. They have been selected because of their varied nature so that they cover a wide range of cases and labelling them is challenging. They have classes with small and large numbers of instances (though most classes have between 40 and 100 instances), both numerical and non-numerical values among attributes, both flat records and deeper structures, and groups of classes with high similarity that make it harder to classify them. We have joined these datasets into a single testing scenario in order to increase the difficulty of labelling each instance as much as possible. Separating them would greatly lower the number of instances and thus make classification much easier.

Note that we can not give a good measure of the magnitude of this similarity in an unsupervised way, since it is precisely through modelling techniques that we can compute how difficult it is to tell apart certain classes by measuring how well said techniques perform. The overall results obtained by different techniques in our experiments are not close to perfection and thus prove that creating a model is not, in this case, a trivial task. For example, class http://xmlns.com/foaf/0.1/givenName is often confused with other classes that have a similar format such as http://xmlns.com/foaf/0.1/familyName. Among the techniques we have implemented, which are later described in detail, Soru labels 10.00% of them correctly with class http://xmlns.com/foaf/0.1/givenName, Kush 0.00% of them, Ram 0.17% of them, Pham 0.09% of them, Neville 35.11% of them, and even our technique only labels 15.89% of them correctly.

We have divided the selected datasets into 10 folds, and applied leave-p-out validation [2] by using 8 folds for training and 2 folds for testing in each experiment, resulting in a total of 45 experiments. The datasets create an experimentation scenario with the following remarkable properties: it consists of both attributes and records, labelled with a class; information is arbitrarily structured, with tuples and other deeper structures; and datasets are not aligned, which means they must be labelled individually.

We measure the classification accuracy (rate of instances that were correctly labelled), as well as the macro-averaged precision, recall and F_1 score, computed as the average of the precision, recall and F_1 of each class, respectively. We consider the accuracy to be the most appropriate measure, since it has a clear interpretation in multi-class classification. Other measures have been included because they are the scientific standard, but are not always a clear indicator of labelling quality. The following situation exemplifies this: suppose there are 20 possible labels. If a classifier always outputs the same label, its macro-precision (macro-averaged precision) will be 0.95, since in 19 out of 20 classes, there are no false positives. We do not include the micro-averaged measurements (micro precision, recall, and F_1) because they have the same value as the accuracy.

Since Kush, Soru, Ram, and Pham are unable to label records, we only use them to label attributes, and measure their indicators by only taking attributes into account (e.g. the accuracy of Pham is measured as the rate of attributes that were correctly labelled). Ram, and Pham are fed individual attributes, and any feature that requires a group of attributes is computed from a group that contains a single one.

We performed our experiments on a computer with an Intel Xeon E7-4807 that ran at 1.87 GHz, had 16 GB of RAM, Windows 7 Pro 64-bit, Java 1.7, and Spark 2.10 for Java. No changes were made to their default configurations.

Our datasets, measures and source code have been made available online[2].

4.2. Improvement of the hint-based phase

We compare the results obtained by TAPON without including hint-based features and performing only the first phase (TAPON-HF), and TAPON

[2] At http://www.tdg-seville.info/dayala/TAPON

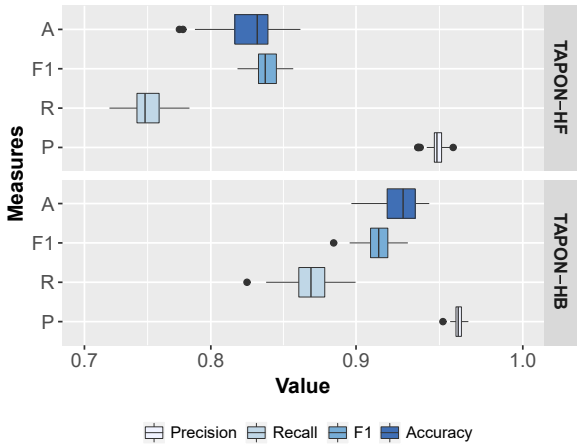


Figure 5: Boxplot with results obtained by TAPON.

Technique	Precision	Recall	F1	Accuracy
TAPON-HB	[0.96, 0.96]	[0.87, 0.87]	[0.91, 0.92]	[0.92, 0.93]
TAPON-HF	[0.95, 0.95]	[0.75, 0.76]	[0.84, 0.84]	[0.82, 0.83]

Table 3: Results summary displaying the 95% confidence interval (TAPON).

including hint-based features and performing two phases (TAPON-HB). This experiment aims to prove the significant improvement of adding hint-based features when using the same classification technique.

Figure 5 shows the results obtained by both techniques. Table 3 shows a numerical summary. The accuracy when adding hint-based feaures is significantly better, with an improvement of around 10%. Even without using hint-based features, the accuracy is high, thanks to the large feature catalogue.

In order to prove that hint-based features are actually used by the model, we have studied their use by the binary classifiers. We have taken the top 5 features of every binary classifier in the hint-based model, sorted by their importance across all trees as suggested by Hastie et al. [9], and computed the fraction of hint-based features among the total. 40.75% of features were hint-based, which clearly shows their usefulness.

4.3. Comparison of TAPON and other methods

We compare the results obtained by the following techniques: TAPON including hints-based features (TAPON-HB); Ramnandan et al. [24]'s technique (Ram); Pham et al. [23]'s technique excluding its histogram similarity feature, which can not be computed properly from a single instance (Pham); and a version of our one-vs-all classification that implements Neville and Jensen [21]'s technique by using a single model and, during training, computing hint-based features using the actual class of the instances (Neville). This experiment aims to prove that our technique achieves better results than other techniques that represent a different approach in the application scenario we have described. Other techniques in the related work have not been included because they are not applicable in our scenario. For example, table labelling techniques can only be applied to named entities and Neumaier et al. [20]'s technique can only label numerical attributes.

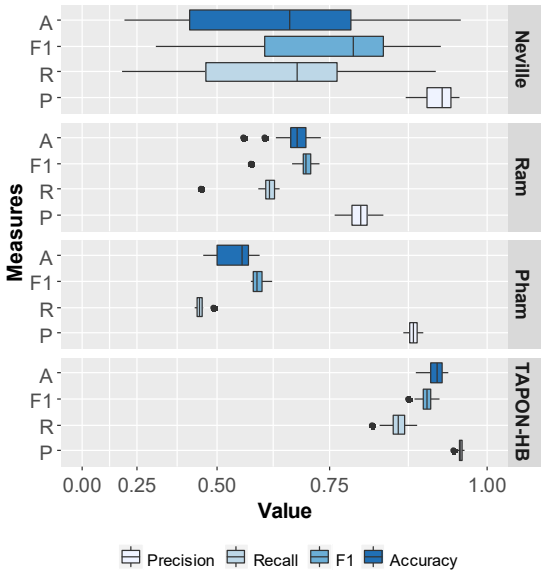


Figure 6: Boxplot with results obtained by different methods.

Technique	Precision	Recall	F1	Accuracy
TAPON-HB	[0.96, 0.96]	[0.87, 0.87]	[0.91, 0.92]	[0.92, 0.93]
Ram	[0.80, 0.81]	[0.62, 0.63]	[0.70, 0.71]	[0.68, 0.70]
Neville	[0.93, 0.94]	[0.55, 0.68]	[0.66, 0.77]	[0.54, 0.68]
Pham	[0.87, 0.91]	[0.42, 0.49]	[0.45, 0.52]	[0.51, 0.58]

Table 4: Results summary displaying the 95% confidence interval (methods).

Figure 6 shows the results obtained by the aforementioned techniques. Table 4 shows a numerical summary. Our technique achieves better accuracy than the other techniques that we have applied in our experiments.

Neville seems to have a large instability. We be-

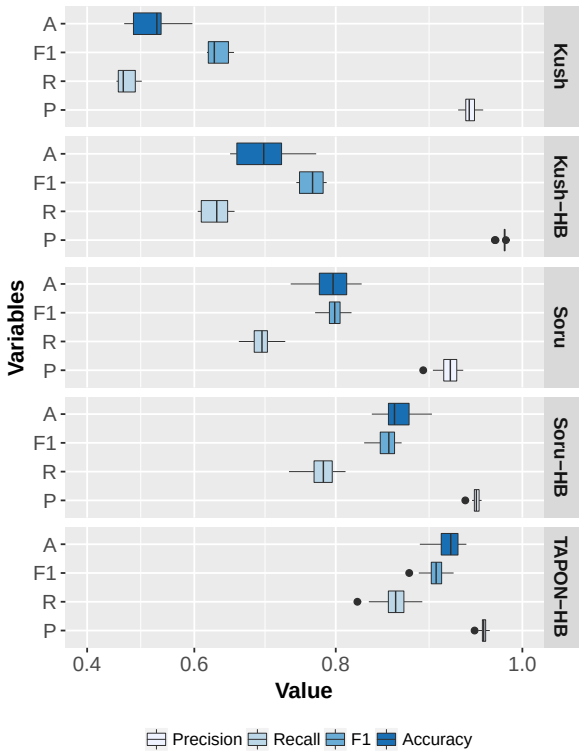


Figure 7: Boxplot with results obtained by different feature catalogues.

Technique	Precision	Recall	F1	Accuracy
TAPON-HB	[0.96, 0.96]	[0.87, 0.87]	[0.91, 0.92]	[0.92, 0.93]
Soru-HB	[0.95, 0.96]	[0.77, 0.80]	[0.85, 0.87]	[0.86, 0.89]
Soru	[0.92, 0.93]	[0.70, 0.70]	[0.80, 0.80]	[0.79, 0.80]
Kush-HB	[0.98, 0.98]	[0.62, 0.65]	[0.76, 0.78]	[0.68, 0.73]
Kush	[0.94, 0.95]	[0.47, 0.49]	[0.63, 0.65]	[0.50, 0.56]

Table 5: Results summary displaying the 95% confidence interval (features).

lieve this to be caused by the way it is trained using actual classes (while we use labels inferred by the first model). The quality of the predictions heavily depends on the quality of the labels. When applied to the testing dataset, the model is applied even when there are no labels that can be used to properly compute hint-based features and consequently, performance is poor. However, if iterations are able to slightly improve the labels, there is a slow approximation towards the actual classes which requires many iterations. If the set of labels is too distant from real classes and an iteration is not able to improve the labels, then there is convergence and the process stops. In some cases, there is no early convergence and iterations slowly reach a good set of predictions with accuracy around 0.85. In other cases, there is early convergence and the set of poor predictions remains unchanged, resulting in accuracy around 0.40.

4.4. Comparison of TAPON and other feature catalogues

We compare the results obtained by the following techniques: TAPON including hints-based features (TAPON-HB); a version of our classification technique that only uses Kushmerick [13]'s feature catalogue(Kush); the former including hint-based features(Kush-HB); a version of our classification technique that only uses Soru and Ngomo [30]'s feature catalogue (Soru); and the former including hint-based features(Soru-HB). This experiment aims to prove that TAPON's feature catalogue achieves better results than the other feature catalogue in the related work, in the application scenario we have described, and that adding hint-based features to a features catalogue improves results, not only with our specific catalogue.

Table 6 summarizes the features that each technique uses to model information. The techniques by Lerman et al. [15] and McCann et al. [18] use feature catalogues that are very similar to Kushmerick's, and so they have not been included.

Figure 7 shows the macro-average of the precision, recall, and F_1 score obtained by the aforementioned techniques. Table 5 shows a numerical summary. Our technique achieves better accuracy than the other techniques that we have applied in our experiments. Regarding the other techniques, the version with hint-based features outperforms the original catalogue in both cases.

4.5. TAPON applied to worst-case scenarios

Figure 8 contains a more fine-grained representation of some of the results. It depicts a comparison between TAPON and other techniques. For each technique, we compare the 3 most problematic classes for that technique (that is, those with the worst F_1 score). The number of classes (3) is arbitrary, since this representation of the results is not intended to be a strict report, but rather a visualization of how our technique seems to solve whatever problems cause really low accuracy when using other techniques. Each point represents the F_1 score obtained by a class during one of the 45

Technique	Features
TAPON	Features described in tables 1 and 2
Kushmerick	Density of letter characters
	Density of numeric characters
	Density of uppercase characters
	Density of lowercase characters
	Density of puntuation characters
	Number of HTML tags
	Number of words
	Mean word length
Soru and Ngomo	Mean edit distance to sets of examples of each attribute class
	Mean trigram similarity to sets of examples of each attribute class
	Mean cosine distance to sets of examples of each attribute class

Table 6: Features used by each technique.

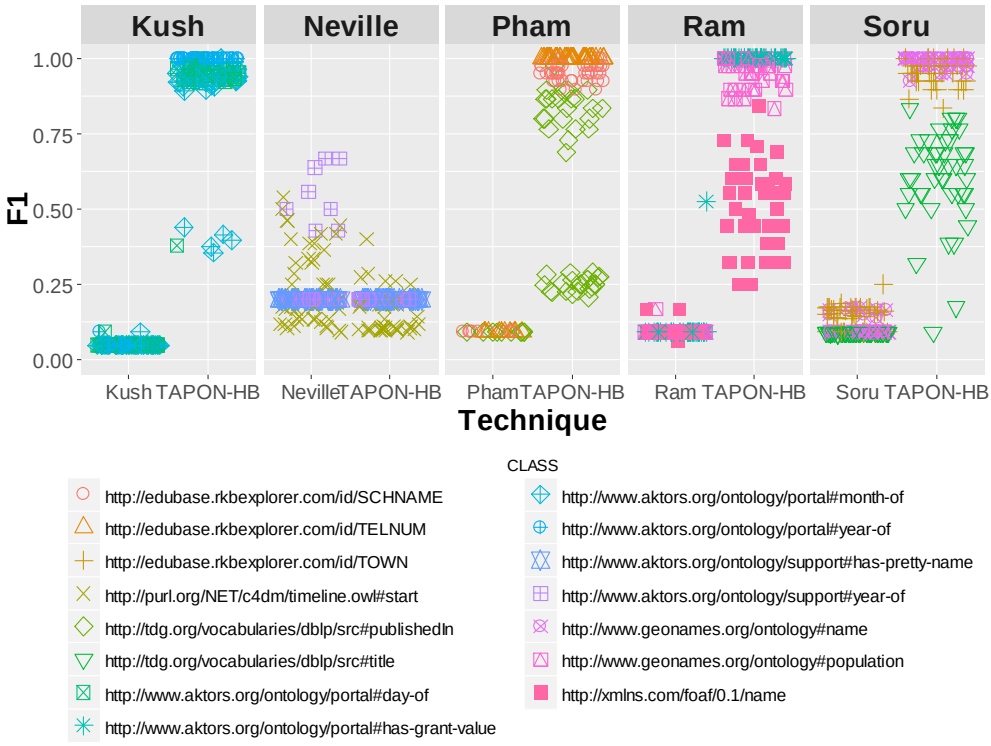


Figure 8: F_1 score obtained by the more problematic classes for each technique, compared to ours.

experiments (we use the F_1 score instead of accuracy, since these are results per class). It can be observed how, to a greater or lesser degree, the cases that are more problematic for other techniques are dealt with relatively well, obtaining high F_1 when using TAPON. The only exception is when comparing to Neville and Jensen [21]. Both techniques behave similarly, probably because the similarities between them make them fail in similar situations. Still, these are only the worst cases, and our technique obtains better results overall, as explained earlier.

4.6. Statistical analysis

In order to test the significance of our accuracy results, we have applied statistical tests with $\alpha = 0.05$. Since the variable (accuracy) is not normally distributed, we are performing a multiple comparison and there are less than five variables, we applied a Friedman Test with control estima-

Technique	Ranking
TAPON-HB	1.07
Soru-HB	2.16
TAPON-HF	3.20
Soru	4.27
Kush-HB	5.93
Neville	5.98
RAM	5.98
Pham	7.71
Kush	8.71

Table 7: Average rank of each technique.

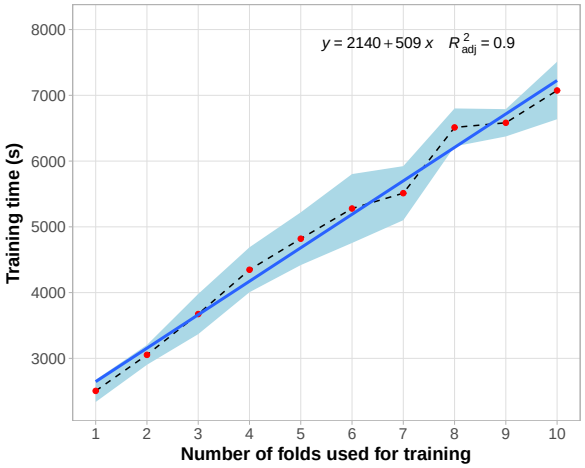


Figure 9: Training time of TAPON.

tion. The resulting p-value is $10e-10$, which implies there are significant differences between the ranks (whose average is shown in Table 7). We have applied Hommel's test to compare the results obtained by TAPON-HB to each other technique (including TAPON-HF). The resulting p-value is always below $2.84e-13$. Therefore, there is statistical evidence that the results obtained by our technique are significantly different from those obtained by other techniques. Note that hint-based versions of initially worse technique surpass TAPON-HB in accuracy when adding hint-based features, proving their usefulness.

4.7. Efficiency of TAPON

Regarding training time, Figure 9 shows how much time it takes to create our two models from different amounts of training data. We have trained our models using 1 to 10 folds (10% to 100% of our available data). For each different number of folds we performed 10 experiments, where the specific folds to be used among the 10 available ones were selected by using a moving window (wrapping around 1 in a cyclical fashion). We computed the average value and the 95% confidence interval. Our main observation here is that, since the model only has to be trained once, spending a small number of hours training is not at all unreasonable. The increase in training time seems to follow a linear trend. Table 8 shows the application time per instance of each technique. TAPON's application time is similar to most techniques, with the second phase adding additional application time. Ram and Kush have the lowest application time by a wide margin, thanks to their simplicity. The other techniques have less features, but they are more costly to compute. Adding hint-based features has a seemingly constant hit on the application time. Overall, we consider TAPON's application time to be reasonable.

Technique	App. time
TAPON-HB	[0.032, 0.035]
TAPON-HF	[0.020, 0.023]
Soru-HB	[0.056, 0.063]
Soru	[0.031, 0.034]
Ram	[0.002, 0.002]
Neville	[0.028, 0.032]
Pham	[0.030, 0.040]
Kush-HB	[0.015, 0.020]
Kush	[0.004, 0.004]

Table 8: 95% confidence interval of the application times in seconds.

5. Conclusions

In this article, we have presented a new two-phase, machine-learning semantic labelling technique that takes into account features that model the relationships between instances. It can model arbitrarily structured information from several sources, and is able to label datasets in an individual basis.

We use the information from nearby instances in a novel way. By creating two models, we can compute additional hint-based features. The first model is used to endow instances with a hint, allowing us to inject features that are used by the second model. We have performed semantic labelling of information from several linked data sources in order to evaluate the effect of the features and compare

our technique with other existing ones. Our experimental analysis shows that, while our rich feature catalogue helps our technique achieve better results in a stressful modelling scenario, it is the collection of hint-based features that creates a remarkable difference (from an average accuracy of 82.83% to 92.75%) by allowing us to measure features that require having a first idea of what classes the instances may belong to.

Consequently, we conclude that our technique contributes to the state of the art by improving information modelling accuracy while modelling both records and attributes in arbitrary structures. Existing techniques so far have focused on application scenarios that rely on certain conditions such as specific structures of data or the presence of instances groups. When that is the case (for example, we can group instances and label them as a group), making use of that information is an advantage, but in situations where it is not, it is convenient to use a more generic approach that is not adversely affected by the absence of the aforementioned conditions. Our technique offers this approach, while being able to integrate other techniques thanks to our two-phase solution.

The limitations of TAPON are mainly related to the assumptions we presented in Section 2.2; in different application scenarios, it will likely perform worse than more optimised techniques. For example, if a task involves labelling millions of attributes that are known to share the same class, TAPON may not scale well, since it labels them individually, while the technique by Pham et al. [23] would not have this problem. There may be workarounds for some of these cases, like labelling a few attributes and taking the most frequent label in the example, but other techniques in the related work have more optimal solutions. Consequently, future research could focus on how to apply the same principles of TAPON (performing a second iteration to use additional information) in other techniques that focus on different application scenarios.

Furthermore, TAPON is not optimised for large numbers of classes (several hundreds or thousands) due to the use of a one-vs-all classifier. Although the scenarios we have observed have a few dozens, it would be of interest to develop a different version that sacrifices accuracy in favour of scalability and speed, in the same vein as Ramnandan et al. [24]'s.

Acknowledgements

Our work was supported in the Spanish R&D&I programme by grant TIN2016-75394-R.

References

[1] Carlos Buil Aranda, Aidan Hogan, Jürgen Umbrich, and Pierre-Yves Vandenbussche. Sparql web-querying infrastructure: Ready for action? In *International Semantic Web Conference (2)*, pages 277–293, 2013. URL `http://dx.doi.org/10.1007/978-3-642-41338-4_18`.

[2] Sylvain Arlot, Alain Celisse, et al. A survey of cross-validation procedures for model selection. *Statistics surveys*, 4:40–79, 2010.

[3] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. Open information extraction from the web. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 2670–2676, 2007. URL `http://ijcai.org/Proceedings/07/Papers/429.pdf`.

[4] Boris Chidlovskii. Automatic repairing of web wrappers by combining redundant views. In *ICTAI*, pages 399–406, 2002. URL `http://dx.doi.org/10.1109/TAI.2002.1180831`.

[5] Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *VLDB*, pages 109–118, 2001. URL `http://www.vldb.org/conf/2001/P109.pdf`.

[6] Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching, Second Edition*. Springer, 2013. ISBN 978-3-642-38720-3.

[7] The Apache Software Foundation. Apache spark. `https://spark.apache.org/`. Accessed: 2018-06-08.

[8] Hugh Glaser, Ian Millard, and Afraz Jaffri. Rkbexplorer.com: A knowledge driven infrastructure for linked data providers. In *ESWC*, pages 797–801, 2008. URL `http://dx.doi.org/10.1007/978-3-540-68234-9_61`.

[9] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The elements of statistical learning: data mining, inference, and prediction, 2nd Edition*. Springer series in statistics. Springer, 2009. ISBN 9780387848570. URL `http://www.worldcat.org/oclc/300478243`.

[10] Aidan Hogan, Antoine Zimmermann, Jürgen Umbrich, Axel Polleres, and Stefan Decker. Scalable and distributed methods for entity matching, consolidation and disambiguation over linked data corpora. *J. Web Sem.*, 10:76–110, 2012. URL `http://dx.doi.org/10.1016/j.websem.2011.11.002`.

[11] Patricia Jiménez and Rafael Corchuelo. On learning web information extraction rules with tango. *Inf. Syst.*, 62:74–103, 2016. URL `http://dx.doi.org/10.1016/j.is.2016.05.003`.

[12] Patricia Jiménez and Rafael Corchuelo. Roller: a novel approach to web information extraction. *Knowl. Inf. Syst.*, 49(1):197–241, 2016. URL `http://dx.doi.org/10.1007/s10115-016-0921-4`.

[13] Nicholas Kushmerick. Regression testing for wrapper maintenance. In *AAAI/IAAI*, pages 74–79, 1999. URL `http://www.aaai.org/Library/AAAI/1999/aaai99-011.php`.

[14] Nicholas Kushmerick. Wrapper verification. *WWW*, 3 (2):79–94, 2000. doi: 10.1023/A:1019229612909.

[15] Kristina Lerman, Steven Minton, and Craig A. Knoblock. Wrapper maintenance: A machine learning approach. *J. Artif. Intell. Res.*, 18:149–181, 2003. doi: 10.1613/jair.1145.

[16] Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. Annotating and searching web tables using entities, types and relationships. *PVLDB*, 3 (1):1338–1347, 2010. URL http://www.comp.nus.edu.sg/~vldb2010/proceedings/files/papers/R118.pdf.

[17] Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. Generic schema matching with cupid. In *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy*, pages 49–58, 2001. URL http://www.vldb.org/conf/2001/P049.pdf.

[18] Robert McCann, Bedoor K. AlShebli, Quoc Le, Hoa Nguyen, Long Vu, and AnHai Doan. Mapping maintenance for data integration systems. In *VLDB*, pages 1018–1030, 2005. URL http://www.vldb2005.org/program/paper/fri/p1018-mccann.pdf.

[19] Varish Mulwad, Tim Finin, and Anupam Joshi. Semantic message passing for generating linked data from tables. In *ISWC*, pages 363–378, 2013. doi: 10.1007/978-3-642-41335-3_23.

[20] Sebastian Neumaier, Jürgen Umbrich, Josiane Xavier Parreira, and Axel Polleres. Multi-level semantic labelling of numerical values. In *International Semantic Web Conference (1)*, pages 428–445, 2016. URL http://dx.doi.org/10.1007/978-3-319-46523-4_26.

[21] Jennifer Neville and David Jensen. Iterative classification in relational data. In *Proc. AAAI-2000 Workshop on Learning Statistical Models from Relational Data*, pages 13–20, 2000.

[22] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: bringing order to the web. 1999.

[23] Minh Pham, Suresh Alse, Craig A. Knoblock, and Pedro A. Szekely. Semantic labeling: A domain-independent approach. In *International Semantic Web Conference (1)*, pages 446–462, 2016. URL http://dx.doi.org/10.1007/978-3-319-46523-4_27.

[24] S. K. Ramnandan, Amol Mittal, Craig A. Knoblock, and Pedro A. Szekely. Assigning semantic labels to data sources. In *ESWC*, pages 403–417, 2015. doi: 10.1007/978-3-319-18818-8_25.

[25] Dominique Ritze, Oliver Lehmberg, and Christian Bizer. Matching html tables to dbpedia. In *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics*, page 10. ACM, 2015.

[26] Carlos R. Rivero, Inma Hernández, David Ruiz, and Rafael Corchuelo. Discovering and analysing ontological models from big RDF data. *J. Database Manag.*, 26(2): 48–61, 2015. doi: 10.4018/JDM.2015040104.

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

[28] Hassan A. Sleiman and Rafael Corchuelo. Trinity: On using trinary trees for unsupervised web data extraction. *IEEE Trans. Knowl. Data Eng.*, 26(6):1544–1556, 2014. URL http://doi.ieeecomputersociety.org/10.1109/TKDE.2013.161.

[29] Dezhao Song and Jeff Heflin. Automatically generating data linkages using a domain-independent candidate selection approach. In *International Semantic Web Conference (1)*, pages 649–664, 2011. URL http://dx.doi.org/10.1007/978-3-642-25073-6_41.

[30] Tommaso Soru and Axel-Cyrille Ngonga Ngomo. A comparison of supervised learning classifiers for link discovery. In *SEMANTICS*, pages 41–44, 2014. URL http://doi.acm.org/10.1145/2660517.2660532.

[31] Mohsen Taheriyan, Craig A. Knoblock, Pedro A. Szekely, and José Luis Ambite. Learning the semantics of structured data sources. *J. Web Sem.*, 37-38: 152–169, 2016. URL http://dx.doi.org/10.1016/j.websem.2015.12.003.

[32] Petros Venetis, Alon Y. Halevy, Jayant Madhavan, Marius Pasca, Warren Shen, Fei Wu, Gengxin Miao, and Chung Wu. Recovering semantics of tables on the Web. *PVLDB*, 4(9):528–538, 2011. URL http://www.vldb.org/pvldb/vol4/p528-venetis.pdf.

[33] Ziqi Zhang. Effective and efficient semantic table interpretation using tableminer+. *Semantic Web*, (Preprint):1–37, 2016.