# A neural network for semantic labelling of structured information

Daniel Ayala[a,*], Agustín Borrego[a], Inma Hernández[a], David Ruiz[a]

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

## Abstract

Semantic labelling consists in assigning known labels to the data from a source of structured information. This can be useful in a variety of tasks related to information extraction and integration into information systems and their local ontologies. Semantic labelling can be seen as a classification problem in which the input is structured information from which features can be computed in order to apply machine learning techniques. The existing proposals, based on machine learning so far, have focused on what features should be computed while relying on simple classification models like logistic regression or random forest, and may not be powerful enough to properly classify some classes, especially in scenarios in which a large number of features contain the necessary information but it is hard for the classifiers to properly combine them. In this paper, we propose and test the novel application of neural networks to semantic labelling, which benefits from non-linearity and can deal with the increasing number of features. Our proposal has been validated with datasets from three real world sources, and our conclusion is that state-of-the-art neural networks consistently improve the accuracy of the labelling when compared to traditional classification.

*Keywords:* Semantic labelling, Information Integration, Neural Networks

## 1. Introduction

The Web is a rich source of semi-structured data which usually has to be integrated into information systems before its exploitation (Knoblock et al., 1998). The first step towards the integration in one such system is the crawling of the Web to obtain a set of HTML documents (Hernández et al., 2018, Batzios et al., 2008). The second step is to extract structured information from them (Sleiman and Corchuelo, 2013, Wang et al., 2007). The extracted structured information lacks semantics, so the third step is to establish correspondences between the data and a known ontology. This is the goal of semantic labelling, which consists in labelling elements in data structures with known classes from a Web ontology (Pham et al., 2016). Semantic labelling proposals take the structured elements as input, and assign them one or several labels, which correspond to the classes that best describe each element according to its features. Figure 1(a) shows an example of a structured dataset from the Jisc repository (Jisc, 2018), displaying labelled information about a R&D project related to education. A semantic labelling proposal would learn from the examples in this and other datasets a classification model for each class, such as "jisc:name", "jisc:title", or "jisc:start-date". Then, when fed a new unlabelled dataset like the one in Figure 1(b), it would iterate every element in it and endow it with a known class. Consequently, semantic labelling can be seen as a classification problem in which the input is one of the elements in the structure and the features are whatever aspect are measured from them. In the former example, instance I2 could be classified as a "jisc:title" after an analysis of some of its features, including the number of words that start with an uppercase letter and the position of the instance in the structure, I3 could be classified as a "jisc:start-date" because of the number of digits, and I10 could be classified as a "jisc:status" because programme statuses only have a few possible values ("Complete", "Running", etc.), and the value of the instance matches

*Corresponding author
*Email addresses:* dayala1@us.es (Daniel Ayala), borrego@us.es (Agustín Borrego), inmahernandez@us.es (Inma Hernández), druiz@us.es (David Ruiz)

that of other known examples of the same class.

We can apply the same model to data from any source in order to label it with the same known classes, as long as the model was able to properly learn what features can be used to identify each class. Semantic labelling is therefore related to the integration of heterogeneous information from different sources by modelling classes in structured information. Beyond the direct integration of information, the modelling has other applications such as information extraction (Banko et al., 2007) (which, as we mentioned, is also a step of information integration), information verification (Kushmerick, 2000, Lerman et al., 2003, McCann et al., 2005), or ontology matching (Euzenat and Shvaiko, 2013). These areas are all tightly related to the Web and the integration of information from external sources.

The current trend in the state of the art proposals is to focus on feature engineering(Ayala et al., 2019, Ramnandan et al., 2015, Neumaier et al., 2016, Pham et al., 2016), that is, identifying new features that endow the classifier with enough power as to discern between different classes, even when those classes are highly similar like "jisc:name" and "jisc:title". Devising elaborate features is crucial to achieve good accuracy, and the most recent work related to semantic labelling (Ayala et al., 2019) has resulted in a large explosion of features, with potentially hundreds of them. However, our study of the literature reveals that existing proposals are based on baseline classification techniques, neglecting advanced classification techniques that use the features efficiently. The most recent proposals only use random forest or logistic regression classifiers, and do not study more elaborate alternatives, leaving room for improvement.

Our hypothesis is that neural networks can significantly improve the accuracy of a semantic labelling model, while using the same initial low-level features as a traditional classification model. While some areas like Natural Language Processing, Computer Vision, or even other tasks related to integrating information from external sources like information retrieval from the Web have been transformed by the successful application of modern neural network technology (Deng and Yu, 2014), semantic labelling has so far relied on the more traditional machine learning techniques we have mentioned. While the potential of neural networks has been tested in some related tasks like information extraction, to the best of our knowledge it remains completely unexplored in the field of semantic labelling, which motivated us to study it as a novel application, checking what strategies and architectures are applicable and what results they achieve. Our experiments, in which we use a neural network with dense layers for semantic labelling in several scenarios using real world data, reveal that the accuracy of the labels improves consistently when compared to four traditional classification techniques, even when there is little margin for improvement.

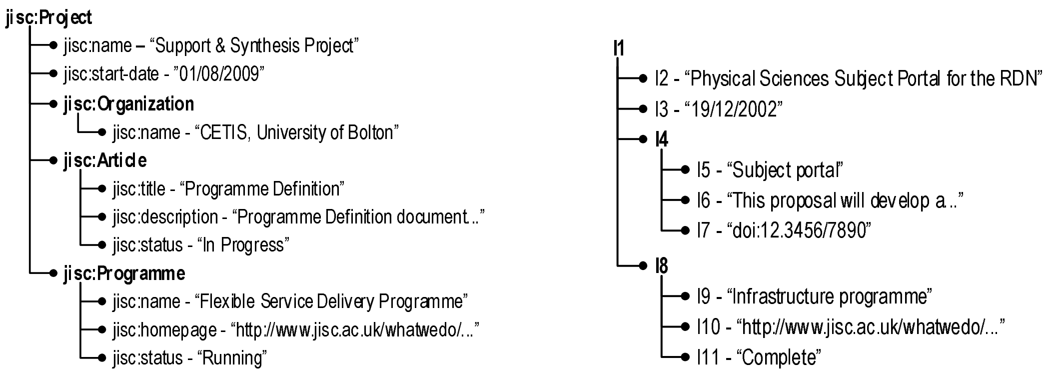
The rest of the paper is organised as follows: Section 2 reports on some preliminaries that are necessary to understand the domain of the problem; Section 3 describes the analysis of the relevant proposals we have identified in the literature; Section 4 describes the nature of features in semantic labelling; Section 5 contains a detailed description of the application of neural networks to semantic labelling; Section 6 describes the experiments we used to test our hypothesis and their result; finally, Section 7 recaps on our main conclusions.

## 2. Preliminaries

In this Section, we introduce definitions of concepts related to the problem of semantic labelling.

**Class:** a piece of text that denotes semantics in a Web ontology. The output of semantic labelling is a set of labels that should match the class of every data item. Example: classes "jisc:Project" and "jisc:start-date".

**Attribute:** A data item with a textual value that can be an instance of a class and have a label that denotes it. The textual value can represent a number, date, boolean, or any other data type. Note that in this context, an attribute does not refer to an element of the schema, but to a specific data item. It may be possible to have an attribute that does not belong to any class in a particular ontology, i.e., a piece of text that is automatically extracted from a website by a crawler but does not correspond to any known class. Example: in Figure 1(a), one of the two attributes of class "jisc:name" has a textual value of "Support & Synthesis Project", and the attribute of class "jisc:start-date" has a textual value of "01/08/2009". In Figure 1(b) there are several attributes: I2 (a name), I3 (a start date), I5 (a title), I6 (a description), I7 (a doi), I9 (a name), I10 (a home-




(a) Labelled dataset. (b) Unlabelled dataset.

Figure 1: Dataset examples.

page), and I11 (a status), but their class is unknown by the system. Attribute I7 is clearly a doi, but there is no doi class in the known ontology, so it would have no class in it.

**Record:** a text-less data item that has other attributes or records as children, may be an instance of a class and have a label that denotes it. Record classes admit a certain degree of variability in their schema, that is, different records of the same class may have variable attributes and records if some of them are optional or have different multiplicity. Example: in Figure 1(a) there are four records. The "jisc:Project" record contains instances of classes "jisc:name", "jisc:start-date", "jisc:Organization", "jisc:Article", and "jisc:Programme". Some of them are also records with their own instances, like the "jisc:Organization" record that has a "jisc:name. Figure 1(b) also shows several records: I1 (a project), I4 (an article), and I8 (a programme). Note that I1 belongs to class "jisc:Project", but it does not contain any "jisc:Organization" record, since it is optional.

**Dataset:** a set of attributes and records in a hierarchical structure. Usually, there is a single root record at the first level of the dataset, but nothing prevents the presence of several ones, having a forest-like structure. Example: Figure 1(a) displays a dataset with 4 records and 9 attributes, and the root is the "jisc:Project" record. Figure 1(b) displays a dataset with 3 records and 8 attributes, and the root is the I1 record.

**Model:** a classifier that takes attributes as the input, and outputs their label. A model could classify a single instance or a group of them. Example: a random forest classifier that takes the attributes in Figure 1(b), computes some features, and outputs a label for each of them.

**Feature:** a numeric or categorical measure that can be taken from an attribute or group of attributes. It can be seen as a function that takes an instance or group of attributes as input and outputs a feature value. Example: a feature that computes the number of digits in the textual value of an attribute, which in Figure 1(b) would output 0.0 for I2 and 8.0 for I3.

**Internal model:** a model that learns from a set of examples (labelled attributes) by using features obtained from the data item themselves, without relying on external sources of data. Example: a classifier that computes features related to the format of the attributes such as the number of uppercase letters or the average word length, and labels them using a random forest or logistic regression classifier.

**External model:** a model that learns from a set of examples by using at least one feature that requires an external knowledge base (e.g. YAGO, DBpedia) to be computed. These features are usually computed by mean of queries to the knowledge base. Example: a classifier that queries DBPedia using the textual value of attributes and labels them with the class of the result with the highest score.

## 3. Related work

In the literature, there are several types of proposals that are able to provide structured information with labels that describe it. These proposals have different goals, but they can all be applied to the problem of semantic labelling, which is why we include them in this analysis. Furthermore, these proposals work with different types of features; however, in our analysis, we focus on the type of classification technique on which they are based, regardless of the specific features. Note that none of them use neural networks, and instead use more traditional techniques like random forest, linear regression, and nearest neighbour classifiers.

The proposals by Limaye et al. (2010), Venetis et al. (2011), Mulwad et al. (2013), Ritze et al. (2015), and Zhang (2016) focus on labelling Web tables, which may include labels for individual cells, rows, columns, and relationships between columns. Tables can be transformed into generic structures, each row being a record, and its cells the attributes. These proposals use knowledge bases to perform the labelling. These contain a set of entities that belong to classes, and usually offer the possibility of querying them to obtain entities that seem to match the query. In most cases, tables are labelled in an iterative process by first obtaining a set of candidate entities for each cell, then labelling the columns according to the most frequent classes among the candidate entities, and then refining the candidates by limiting them to the column classes. These proposals are based on external models, since the classification is ultimately based on the score of queries to external sources, which in turn usually depends on the TF-IDF score and cosine distances computed from the documents in the knowledge base. The labels are limited to the existing classes in the external source.

The proposals by Ramnandan et al. (2015), Pham et al. (2016), Neumaier et al. (2016), and Ayala et al. (2019) label attributes by comparing them to sets of examples of known classes. The labels are obtained through a classification process, based on features such as the value of numeric attributes, string distance metrics, similarity metrics, or features related to the structure of the data. These proposals are based on internal modes. The proposal by Ramnandan et al. (2015) selects the class with the highest score when querying a Lucene index that contains examples of a class in each stored document. The proposal by Pham et al. (2016) uses a one-vs-all logistic regression classifier with several similarity measures. The proposal by Neumaier et al. (2016) uses a nearest neighbour classifier. The proposal by Ayala et al. (2019) uses a one-vs-all random forest classifier.

In addition to the former proposals, those by Kushmerick (1999), Lerman et al. (2003) and McCann et al. (2005) focus on information verification, and their goal is to confirm that a dataset is correct according to the reference model. They learn from a number of verified labelled examples, they compute the collections of values of each feature, and infer the statistical normal distributions that best fit them. When a dataset must be verified, the values of its features are compared to the inferred distributions. If some of the values associated to an element or the entire dataset deviate too much from the verified ones according to statistical tests, the dataset is considered to be anomalous. Information verification is very similar to semantic labelling, since verifying an already labelled dataset amounts to applying semantic labelling to re-compute the set of labels for the dataset and checking that the two sets of labels are identical.

We have observed that the classification of instances is not trivial when the number of classes is large. The similarity between classes may be such that even if the computed features hold enough information to differentiate classes, their efficient use by a model may require complex non-linear combinations that represent a challenge to most techniques. For example, instances of classes "jisc:title" and "jisc:name" are usually similar, and correctly separating their classes could require a combination of several features related to their length, presence of certain characters or tokens, and other measures. The existing proposals use techniques that do not deal well with cases that require non-linearity, which motivated us to implement the novel application of neural network techniques to semantic labelling.

## 4. Features

Features in the field of semantic labelling do not necessarily measure the occurrence of specific words in the textual value of attributes; instead, they are mostly related to its format, i.e., the kind of characters and tokens it contains, how long it is, or how similar it is to sets of examples according to different distance functions. The features catalogue does not necessarily depend on the particular classification algorithm that is being applied, i.e., we can create several classifiers for semantic labelling using the exact same features.

In the past, the features set used in related proposals was limited to around a dozen features (Kushmerick, 2000, Lerman et al., 2003, McCann et al., 2005). However, the most recent work has started to develop larger, more expressive sets of features to include as much information as possible in the input. One of the recent additions are the so-called parametric features (Ayala et al., 2019). They are a kind of feature that fits well this need to include as much low-level information as possible in the first layer. They take a parameter, which means that each parametric feature results in a family of features, each of them related to a different value of the parameter. The parameter can be one of the known classes, so that each variant of the feature gives information related to it. For example, feature $F_3$ expands into 6 different features of the same family.

Table 1 displays the final features that we have selected from the literature. Note that several features are parametrical, three of them on a per class basis. Features $F_1$, $F_2$, $F_3$, and $F_4$ give information about the textual format of the attribute. Features $F_5$ and $F_6$ help detect starting and ending patterns. Feature $F_7$ measures overall similarity to each class. Feature $F_8$ gives additional information when an attribute has a numeric value that can be considered a feature itself. Features $F_9$, $F_{10}$ and $F_{11}$ give information about the structure in which the attribute is present. For example, if we have trained a classifier with three known classes: "jisc:title", "jisc:name" and "jisc:start-date", feature $F_7$, "Average edit distance", would have three versions: "Average edit distance to examples of class jisc:title/jisc:name/jisc:start-date". With three classes there would be a total of 35 features. Since in the real world cases we have studied there are usually several dozens of classes, parametric features can result in a features explosion which is difficult to handle for traditional classifiers.

## 5. Our proposal

In this Section we present the neural network we have devised. First, we describe the application workflow in which the neural network is framed. Then, we describe in detail the architecture of the network. Finally, we justify the choices in the architecture and analyse why some popular strategies could not be applied.

### *5.1. Workflow*

Figure 2 summarizes the classification workflow. The original input is a dataset containing several records and attributes. Each individial attribute is fed to a features calculator that computes the low-level features. The features must be any measurement that we can take from the text of an attribute and the structure of the dataset that contains it. The neural network should benefit from a large number of low-level features that can later be combined.

The features are used to create a vector that is fed to the first layer of the neural network, whose size is always equal to the number of features. After going through the hidden layers, the output layer, whose size is always equal to the number of known classes, gives a score to each class, which is used to select the final label.

A strengh of our proposal is that it labels individual instances as opposed to labelling a group of several attribute instances that are known to share the same class. For example, the proposal by Ramnandan et al. (2015) would take as input a set of several dozens or hundreds of instances and output a single label for them. We consider individual labelling to be a more challenging task due to the limited information available during classification. One possible real-world scenario in which the inputs are individual attributes is unsupervised information extraction (Roldán et al., 2017), which extracts general useful information from web pages in generic variable structures with no schema by means of universal rules that do not require training. However, the application to groups of attributes would be trivial, simply requiring a change of features, so that they are computed from several instances instead of a single one.

While structured datasets may include both records and attributes, our application of neural networks focuses on classifying attributes, so that our results are comparable with those in the related work, which does not include the labelling of records in many cases. However, the attributes used for training and testing are still positioned in a structured datasets, and consequently, features can make use of the records or their structure (for example, a feature could be "Number of adjacent records").

| ID | Feature | Description |
|---|---|---|
| $F_1(S)$ | 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. |
| $F_2(T)$ | 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. |
| $F_3(S)$ | Density of symbol type S | 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. |
| $F_4(T)$ | Density of token type T | The density in the attribute of token of type T. The density is computed as described in AF3 |
| $F_5(C)$ | 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. |
| $F_6(C)$ | 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. |
| $F_7(C)$ | Average edit distance to class C | Average Jaro edit distance between the attribute and a set of stored examples of class C. |
| $F_8$ | Numeric Value | The numeric value of the text of the attribute if it matches a number pattern. -1.0 otherwise |
| $F_9$ | Depth | The depth in the dataset of the attribute. |
| $F_{10}$ | Same level attributes | The number of attributes at the same structural level. |
| $F_{11}$ | Same level attributes | The number of records at the same structural level. |

Table 1: Features.

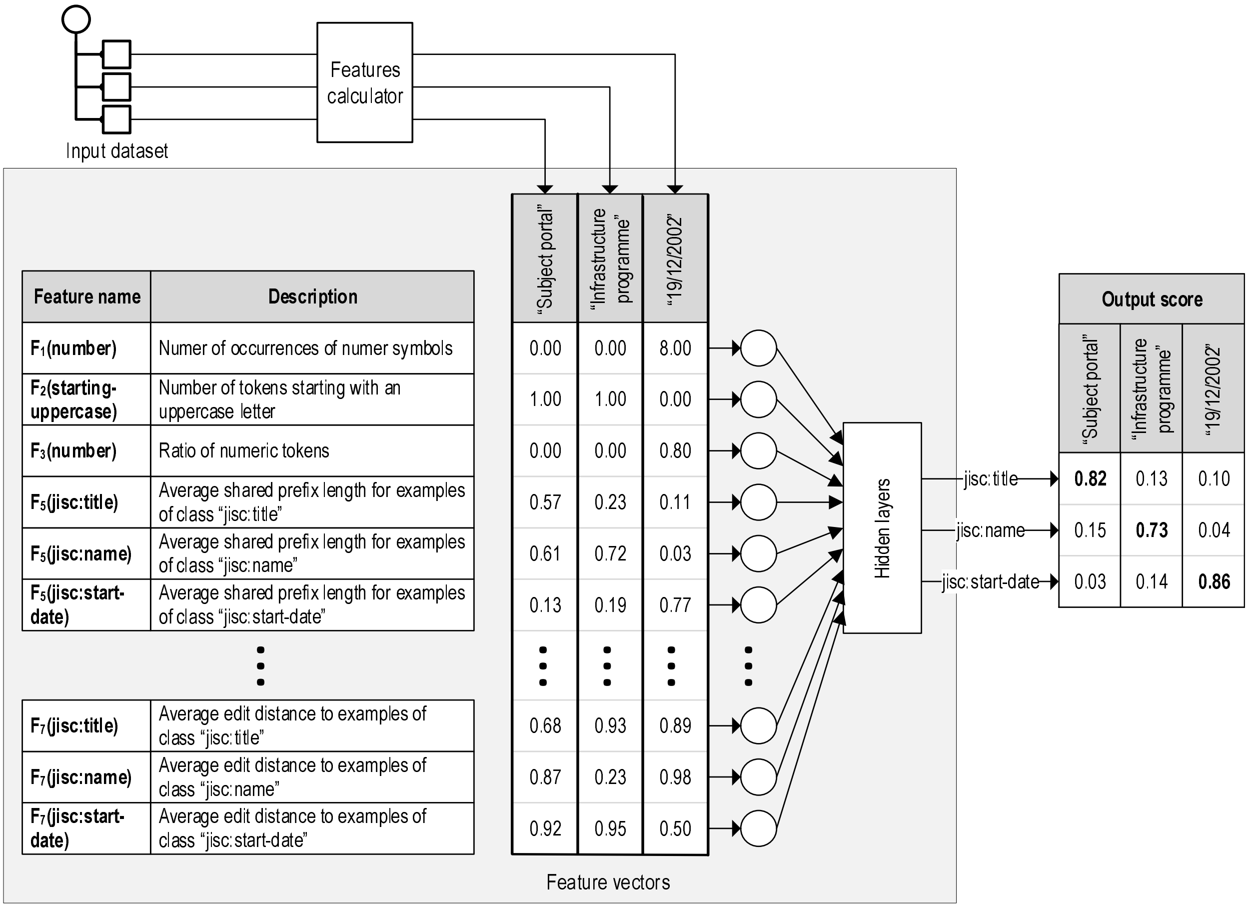



Figure 2: Workflow.

### 5.2. Architecture

Figure 3 summarises the architecture of our network. Keep in mind that we have devised a multi-purpose architecture for any scenario. However, it could be adapted for a specific situation. For example, the size of the hidden layers could be increased or decreased in concordance with the number of features (the size of the input layer). The following paragraphs describe the architecture, which is justified in the next subsection.

Our network has three wide, fully connected hidden layers (each neuron in a layer is connected to every neuron in the next layer). Their sizes are 2048, 1024 and 512. The size of the input layer is equal to the number of initial features, and that of the output layer, equal to the number of classes.

We have applied dropout, a probability of setting a value being transmitted between layers to 0 in order to decrease overfitting. The dropout rates of the layers are 0.01, 0.1 and 0.1. We have set ReLU as the activation function of all intermediary layers, and cross entropy as the loss function, since it is applicable to multiclass classification.

The final layer outputs the score of each label after a softmax function from which we select the one with the highest score. The user could also choose not to accept a label below a given threshold. The softmax function takes a vector of real values and turns it into a new vector of real values in the $(0, 1)$ range that add up to 1.

### 5.3. Discussion

Next, we justify our choices with regards to the architecture, and offer some insights on why we did not include some popular neural network strategies.

A popular machine learning practise is data augmentation (Witten et al., 2016), which consists in expanding the number of data points (in this case, attributes used for training) by creating new synthetic ones, derived from the original ones by means of transformations that create different but still valid data. For example, in computer vision this can be done by panning, zooming, or rotating the input images. Implementing data augmentation in semantic labelling would require manually creating transformation functions that slightly alter attributes while keeping them valid. For example, one such transformation could be to add the country code to phone numbers, so that apart from the training example "954123456", there is the example "+34 954123456". For dates, we could create several training examples for a particular date by changing the date format.

Transformations would have to be created for each of, potentially, several dozens of classes. Their creation is not trivial, and it would be needed to check that a transformation does not worsen training, i.e., always adding the same country code to phone numbers would lead to overfitting. Moreover, while some attributes allow simple changes of format like the aforementioned ones, others would require more complex alterations, such as classes "jisc:description" or "jisc:homepage". Altering a description would require somehow changing its contents while keeping it a valid description, and altering a homepage would require changing some parts of the url while keeping it a valid homepage. At this point, it is clear that the necessary analysis to determine when transformations of the original data can be applied to attributes of a class, and the manual work needed to create them is so large, that it would be easier to manually define rules to label attributes. Therefore, data augmentations does not seem to be applicable to semantic labelling.

Regarding the layer types, we decided not to include some layer types like convolution or pooling layers (LeCun et al., 2015). These and other similar layers aggregate the values of a region of "nearby", related features from a features vector, for example with a weighted mean (convolution) or by taking the maximum value (pooling). Evidently, these operations can only be performed when there is some kind of relation between features of the input that allows us to identify regions of nearby features, as is the case with pictures and sounds: the features from an image (the value of its pixels) have two spatial dimensions, and the features of a sound signal (the value of the samples) have a temporal one. Even in NLP tasks where the input is a sentence of a fixed size and there is a feature for each word of the sentence, we can apply convolution or pooling to groups of embeddings from nearby words. In semantic labelling, however, features are mostly related to the format of attributes, and there is no relation between them that makes it reasonable to talk about a region of features from which the mean or maximum is computed.

Regarding the amount and size of layers, since the initial features already have some level of abstraction, the network should not require a large depth to be effective, and three layers should be enough. The number of layers is in line with other architectures related to structured data in differ-

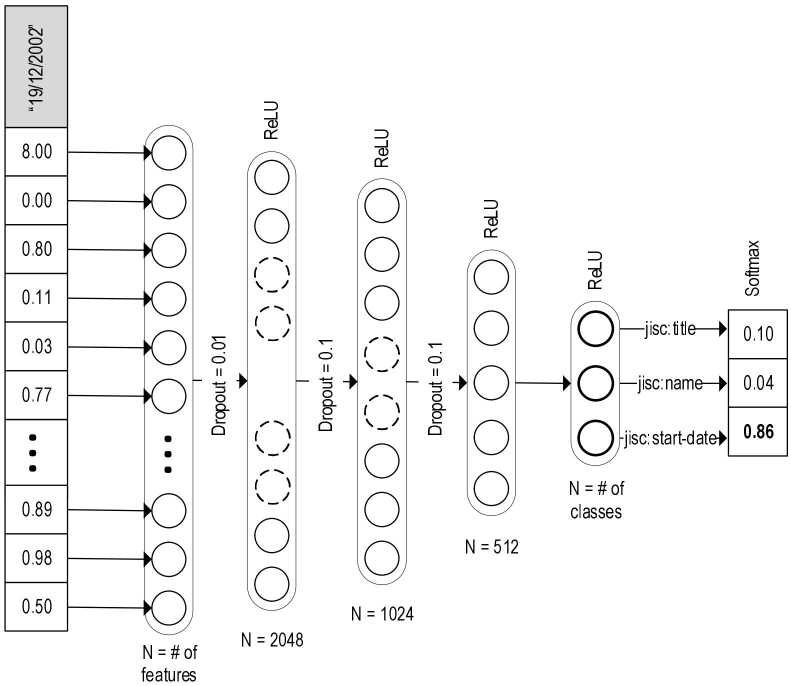



Figure 3: Architecture of our network.

ent tasks (Kazemi and Poole, 2018, Huang et al., 2015, Leng and Jiang, 2016), and is enough to allow nonlinear combinations of the input features which should correspond to more complex textual formats and data structures. The decreasing size helps force the abstraction of features and avoid overfitting.

To the best of our knowledge, there is no way to determine the optimal value for hyperparameters in a completely unsupervised way. The dropout probability in the first layer is very low to preserve most of the information in low-level features, while it is higher in the later layers that correspond to more abstract features. The exact value of hyperparameters were selected by fine-tuning the network in tests, using values that seem to be popular and make sense, i.e. a dropout value no bigger than 0.2. Changing them (for example, adding some additional layers or increasing dropout) did not seem to have a significant impact.

The softmax functions is an appropriate choice for the output layer, since each input is only given a single label. Note that, if several labels per instance are wanted, it is enough to replace it with a different function without altering the architecture of the network.

## 6. Experimental analysis

The experimental validation of our proposal consists in performing semantic labelling on individual attributes in three different scenarios with real-world datasets, which have been selected for their high number of classes:

**NSF** Datasets from the National Science Foundation Awards database (Foundation, 2018a), corresponding to the first 500 awards with the latest end date in 2017.

**Newcastle** Datasets from the Newcastle University repository (University, 2018), corresponding to article references. We set up a SPARQL server using the rdf dump, queried it to obtain resources with class "akt:Article-Reference", and used the first 250 results, each as the root of a dataset where linked resources are records and data properties are attributes.

**Jisc** Datasets from the Jisc repository (Jisc, 2018), corresponding to projects. We obtained 250 datasets in the same way as the Newcastle University datasets, using class "jisc:Project" as the root of each dataset.

| Scenario | Root class | # of datasets | # of classes | # of attributes | # of features |
|---|---|---|---|---|---|
| NSF | nsf:award | 500 | 34 | 17,723 | 135 |
| Newcastle | akt:Article-Reference | 250 | 23 | 7,657 | 102 |
| Jisc | jisc:Project | 250 | 18 | 9,985 | 87 |
| All | Variable | 1,000 | 75 | 35,365 | 258 |

Table 2: Scenarios.

**All** The datasets from the former 3 scenarios, added up.

Table 2 summarises some statistics about them. The number of features is obtained after fully computing all the parametric features in Table 1

The data we used in our experiments, including the computed features, have been made available online[1] for the sake of reproducibility.

We compare the results obtained by the dense network architecture we described to the following one-vs-all classifiers, which are common in the literature (Ayala et al., 2019, Pham et al., 2016), since they ease the separation of one class from the rest when there is a large number of classes:

- A random forest classifier with 20 trees, and maximum depth of 5.
- A logistic regression classifier.
- A linear SVC classifier with a maximum of 20 iterations, and tolerance of $10^{-4}$.
- A gradient boosted trees classifier with a maximum of 20 iterations.

We used the Spark (Foundation, 2018b) implementation of all classifiers, leaving all the unspecified hyperparameters at their default value.

For the implementation of our neural network, we used PyTorch (PyTorch, 2018). We used a single neural network as a multiclass classifier. The training of the neural network consisted of 5 training cycles of length 3 (15 epochs total) with learning rate $10^{-3}$, 2 training cycles of lengths 4 and 8 (12 epochs total) with learning rate $0.5 * 10^{-3}$, and 2 training cycles of lengths 4 and 8 (12 epochs total) with learning rate $0.1 * 10^{-3}$. In each fold, we took the best accuracy among all 39 epochs. The starting learning rate was determined by using the technique described by Smith (2017), in which the learning rate is set to a small value and progressively increased, showing the point at which the loss starts to increase. We diminish the learning rate in the later cycles to allow subtler changes in the weights. Further cycles did not improve the results.

We set the batch size to 16, which achieved the best results in optimal time, though this value could vary depending on the size of the training sets.

We have used 10-fold cross validation, measuring accuracy (fraction of correct labels), since it is the most appropiate metric for multiclass problems such as semantic labelling. Figure 4 shows the accuracy achieved by the traditional classifiers and the dense network implementation in a box plot, with separated results for each scenario, applying 10-fold cross validation. Table 3 shows a numerical summary. Dense networks achieve better accuracy consistently, even in the cases in which traditional classifiers have a high accuracy ("Newcastle" and "Jisc"), where there is a difference of approximately 2.7 percent points (in the median) when compared to the best traditional classifier (random forest). In the "NSF" scenario, where results are worse overall showing a greater labelling difficulty, the improvement is of 4.6 points. In the "All" scenario, the most complex one because of the high number of classes, the improvement is of 8.9 points. It could seem strange that classifiers achieve very similar, and in some cases even better results in the "All" scenario than in the "NSF" scenario, which has a lower number of existing classes. This is caused by the fact that we add relatively easy to classify cases from the "Jisc" and "Newcastle" scenarios to the harder "NSF" scenario, increasing the average accuracy. However, the easier cases become harder to classify due to the higher number of classes. The classification power of the dense network classifier is most visible in "difficult" scenarios, such as those in which there is a large number of classes or highly similar classes, in which the difference in accuracy is more noticeable.

Note that the dense network approach only needed a single multiclass classifier to outperform the one-vs-all classifiers despite the high number of classes, which was a cause for concern.

To prove the significance of the differences, we have applied the Wilcoxon signed ranked test. In all scenarios, the p-value is below 0.002. Since it is lower than the standard significance level of $\alpha = 0.05$, we reject the null hypothesis that differences in distributions are caused by chance.

[1] http://www.tdg-seville.info//Download.ashx?id=490

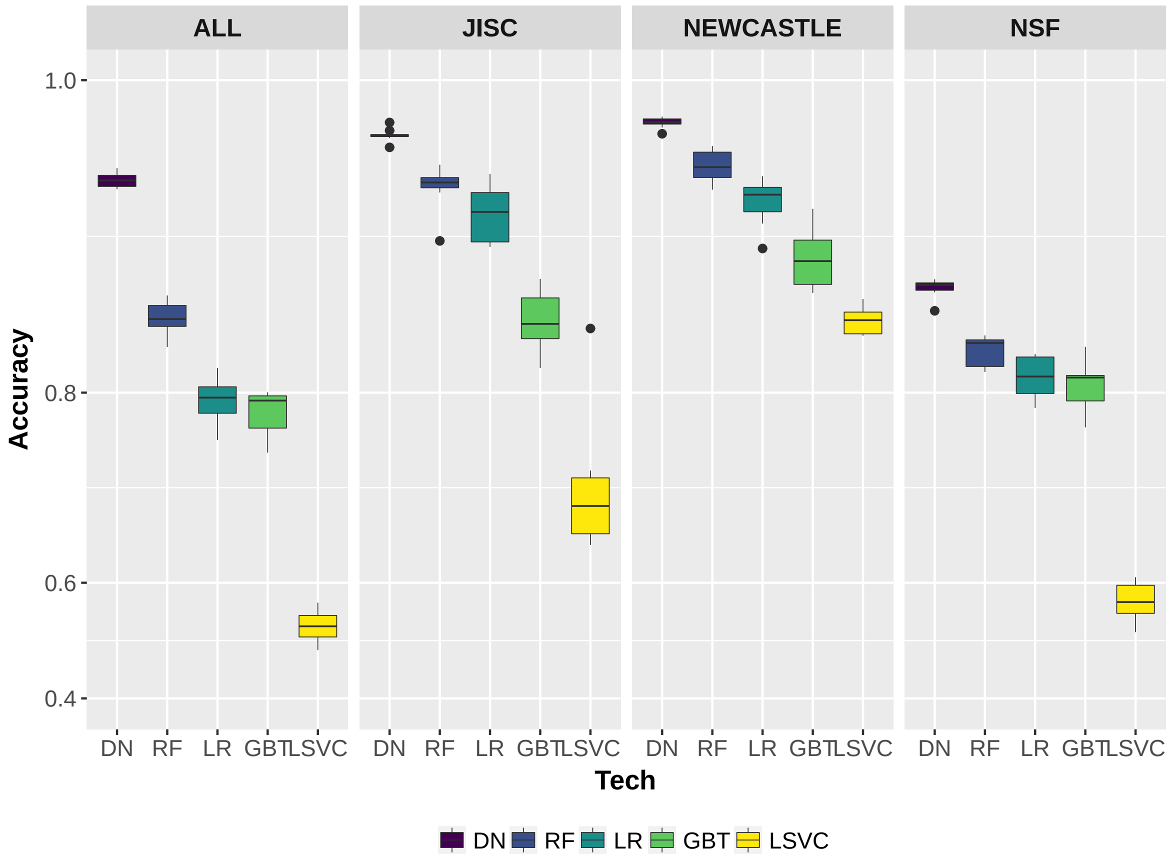



Figure 4: Experimental results. DN = Dense Network, RF = Random Forest, LR = Logistic Regression, GBT = Gradient Boosted Trees, LSVC = Linear SVC.

| Scenario | Median | | | | | Minimum | | | | | Maximum | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DN | RF | LR | GBT | LSVC | DN | RF | LR | GBT | LSVC | DN | RF | LR | GBT | LSVC |
| NSF | **0.88** | 0.82 | 0.81 | 0.81 | 0.57 | **0.86** | 0.82 | 0.79 | 0.77 | 0.53 | **0.88** | 0.84 | 0.83 | 0.84 | 0.61 |
| Newcastle | **0.98** | 0.95 | 0.94 | 0.90 | 0.86 | **0.97** | 0.94 | 0.90 | 0.88 | 0.84 | **0.98** | 0.97 | 0.95 | 0.93 | 0.87 |
| Jisc | **0.97** | 0.94 | 0.93 | 0.85 | 0.69 | **0.96** | 0.91 | 0.91 | 0.82 | 0.65 | **0.98** | 0.95 | 0.95 | 0.88 | 0.85 |
| All | **0.95** | 0.86 | 0.80 | 0.79 | 0.54 | **0.94** | 0.84 | 0.76 | 0.75 | 0.50 | **0.95** | 0.87 | 0.82 | 0.80 | 0.57 |

Table 3: Summary of the results (accuracy).

## 7. Conclusions

Semantic labelling and its many applications have become more relevant than ever thanks to the increasing availability of structured information in the Web and the need to homogenize heterogeneous data sources. Existing proposals have focused on the development of new features that contain the necessary information to classify instances properly, but have not explored the application of neural networks, whose recent development has proven effective in other fields. In this paper, we have explored semantic labelling as a novel application for neural network techniques by devising an architecture that suits well an input with a large number of features computed from attributes. We have tested our dense network implementation of semantic labelling in 4 scenarios created from real world structured data. The results show that neural networks of average depth outperform traditional classifiers in every scenario.

This confirms that the former work was not making full use of the information available in the features. Future semantic labelling proposals should take this into account and use classification techniques that allow the inference of abstract features through non-linear combinations.


## Acknowledgements

Our work was supported the Spanish R&D&I programme by grant TIN2016-75394-R. We would also like to thank Prof. Dr. José Luis Ruiz-Reina, head of the Computer Science and Artificial Intelligence Department at the University of Seville, who kindly provided us with the unvaluable resources that helped us in our research.


## References


Ayala, D., Hernández, I., Ruiz, D., and Toro, M. (2019). Tapon: A two-phase machine learning approach for semantic labelling. *Knowledge-Based Systems*, 163:931–943.

Banko, M., Cafarella, M. J., Soderland, S., Broadhead, M., and Etzioni, O. (2007). 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.

Batzios, A., Dimou, C., Symeonidis, A. L., and Mitkas, P. A. (2008). Biocrawler: An intelligent crawler for the semantic web. *Expert Systems with Applications*, 35(1-2):524–530.

Deng, L. and Yu, D. (2014). Deep learning: Methods and applications. *Foundations and Trends in Signal Processing*, 7(3-4):197–387.

Euzenat, J. and Shvaiko, P. (2013). *Ontology Matching, Second Edition*. Springer.

Foundation, N. S. (2018a). NSF Awards API specification. `https://www.research.gov/common/webapi/awardapisearch-v1.htm`. Accessed: 2018-09-17.

Foundation, T. A. S. (2018b). Apache spark. `https://spark.apache.org/`. Accessed: 2018-09-17.

Hernández, I., Rivero, C. R., and Ruiz, D. (2018). Deep web crawling: a survey. *World Wide Web*, pages 1–34.

Huang, H., Heck, L., and Ji, H. (2015). Leveraging deep neural networks and knowledge graphs for entity disambiguation. *arXiv preprint arXiv:1504.07678*.

Jisc (2018). Jisc repository. `http://jisc.rkbexplorer.com/`. Accessed: 2018-09-17.

Kazemi, S. M. and Poole, D. (2018). Simple embedding for link prediction in knowledge graphs. *arXiv preprint arXiv:1802.04868*.

Knoblock, C. A., Minton, S., Ambite, J. L., Ashish, N., Modi, P. J., Muslea, I., Philpot, A. G., Tejada, S., et al. (1998). Modeling web sources for information integration. In *AAAI/IAAI*, pages 211–218.

Kushmerick, N. (1999). Regression testing for wrapper maintenance. In *AAAI/IAAI*, pages 74–79.

Kushmerick, N. (2000). Wrapper verification. *WWW*, 3(2):79–94.

LeCun, Y., Bengio, Y., and Hinton, G. E. (2015). Deep learning. *Nature*, 521(7553):436–444.

Leng, J. and Jiang, P. (2016). A deep learning approach for relationship extraction from interaction context in social manufacturing paradigm. *Knowledge-Based Systems*, 100:188–199.

Lerman, K., Minton, S., and Knoblock, C. A. (2003). Wrapper maintenance: A machine learning approach. *J. Artif. Intell. Res.*, 18:149–181.

Limaye, G., Sarawagi, S., and Chakrabarti, S. (2010). Annotating and searching web tables using entities, types and relationships. *PVLDB*, 3(1):1338–1347.

McCann, R., AlShebli, B. K., Le, Q., Nguyen, H., Vu, L., and Doan, A. (2005). Mapping maintenance for data integration systems. In *VLDB*, pages 1018–1030.

Mulwad, V., Finin, T., and Joshi, A. (2013). Semantic message passing for generating linked data from tables. In *ISWC*, pages 363–378.

Neumaier, S., Umbrich, J., Parreira, J. X., and Polleres, A. (2016). Multi-level semantic labelling of numerical values. In *International Semantic Web Conference (1)*, pages 428–445.

Pham, M., Alse, S., Knoblock, C. A., and Szekely, P. A. (2016). Semantic labeling: A domain-independent approach. In *International Semantic Web Conference (1)*, pages 446–462.

PyTorch (2018). Pytorch. `https://pytorch.org/`. Accessed: 2018-09-17.

Ramnandan, S. K., Mittal, A., Knoblock, C. A., and Szekely, P. A. (2015). Assigning semantic labels to data sources. In *ESWC*, pages 403–417.

Ritze, D., Lehmberg, O., and Bizer, C. (2015). Matching html tables to dbpedia. In *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics*, page 10. ACM.

Roldán, J. C., Jiménez, P., and Corchuelo, R. (2017). Extracting web information using representation patterns. In *Proceedings of the fifth ACM/IEEE Workshop on Hot Topics in Web Systems and Technologies, HotWeb 2017, San Jose / Silicon Valley, CA, USA, October 12 - 14, 2017*, pages 4:1–4:5.

Sleiman, H. A. and Corchuelo, R. (2013). A survey on region extractors from web documents. *IEEE Trans. Knowl. Data Eng.*, 25(9):1960–1981.

Smith, L. N. (2017). Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017, Santa Rosa, CA, USA, March 24-31, 2017*, pages 464–472.

University, N. (2018). Newcastle university repository. `http://newcastle.rkbexplorer.com/`. Accessed: 2018-09-17.

Venetis, P., Halevy, A. Y., Madhavan, J., Pasca, M., Shen, W., Wu, F., Miao, G., and Wu, C. (2011). Recovering semantics of tables on the Web. *PVLDB*, 4(9):528–538.

Wang, C., Lu, J., and Zhang, G. (2007). Mining key information of web pages: A method and its application. *Expert Systems with Applications*, 33(2):425–433.

Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. (2016). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

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