Understanding the Semantic Structures of Tables with a Hybrid Deep Neural Network Architecture

Kyosuke Nishida, Kugatsu Sadamitsu, Ryuichiro Higashinaka, Yoshihiro Matsuo
NTT Media Intelligence Laboratories, NTT Corporation
1-1 Hikarinooka, Yokosuka, Kanagawa, 239-0847 Japan
nishida.kyosuke@lab.ntt.co.jp

Abstract

We propose a new deep neural network architecture, TabNet, for table type classification. Table type is essential information for exploring the power of Web tables, and it is important to understand the semantic structures of tables in order to classify them correctly. A table is a matrix of texts, analogous to an image, which is a matrix of pixels, and each text consists of a sequence of tokens. Our hybrid architecture mirrors the structure of tables: its recurrent neural network (RNN) encodes a sequence of tokens for each cell to create a 3d table volume like image data, and its convolutional neural network (CNN) captures semantic features, e.g., the existence of rows describing properties, to classify tables. Experiments using Web tables with various structures and topics demonstrated that TabNet achieved considerable improvements over state-of-the-art methods specialized for table classification and other deep neural network architectures.

Introduction

The World Wide Web consists of a huge amount of structured data in the form of HTML tables. Initial studies by (Cafarella et al. 2008a) investigated 14 billion tables and showed that 154 million of them contained relational knowledge. More recently, (Lehmberg et al. 2016) released the WDC Web Table Corpora consisting of 233 million tables extracted from the July 2015 Common Crawl containing 1.78 billion pages. Recent work exploring the power of tables has yielded some interesting applications: table search (Tam et al. 2015), table extension (Lehmberg et al. 2015), query answering (Yin, Tan, and Liu 2011), and knowledge base construction (Dong et al. 2014).

The most fundamental technology for such applications is the means to examine the structures of tables, i.e., *table type classification*. (Crestan and Pantel 2010; 2011) proposed a fine-grained classification taxonomy as to whether they contain semantic triples of the form (subject, property, object) or whether they are used for layout purposes.

A number of studies on table type classification have proposed various features of tables and used machine learning for classification. The most recent work by (Eberius et al. 2015) listed 127 handcrafted features considering global features for tables as a whole and local features per row and per column, e.g., the deviation of the number of characters per cell, the ratio of cells containing colons, and so on. However, their features do not contain ones capturing the *semantic structures* of tables despite that table types are defined on the basis of semantic relations among cells. The building blocks of the structures have different semantics, shapes, and sizes: e.g., columns listing subjects, rows describing properties, and blocks of object cells that belong to the same class. It is difficult to form such complex building blocks by using handcrafted features.

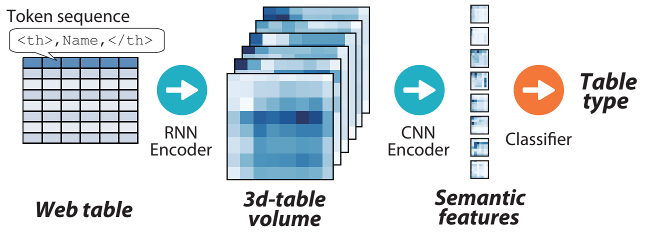
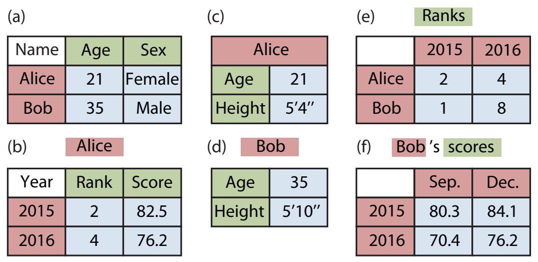


Figure 1: Concept of TabNet. An RNN first encodes a sequence of tokens in each cell, and a CNN then extracts semantic features for table type classification.

Here, we should notice that a table is a *matrix* of texts, analogous to an image, which is a matrix of pixels, and each text consists of a *sequence* of tokens (html tags and words). Recently, deep neural networks have been very successful in capturing semantic representations of sequential and matrix data. The recurrent neural network (RNN) is a neural sequence model that has state-of-the-art performance on various important tasks, including machine translation (Bahdanau, Cho, and Bengio 2015). On the other hand, the convolutional neural network (CNN) can compose a local patch of lower level features into a higher level representation and achieves state-of-the-art performance for computer vision tasks, especially image classification (He et al. 2015).

In this study, we try to determine whether deep neural networks can capture the semantic structures of tables in order to classify them by type. The contributions of this study are summarized as follows.

- We propose a new architecture, TabNet, combining an RNN and a CNN for classifying tables by type (Figure 1).
- TabNet has considerable advantages over state-of-the-art


(a)

Name	Age	Sex
Alice	21	Female
Bob	35	Male

(b) Alice

Year	Rank	Score
2015	2	82.5
2016	4	76.2

(c)

Alice	
Age	21
Height	5'4"

(d) Bob

Age	35
Height	5'10"

(e) Ranks

	2015	2016
Alice	2	4
Bob	1	8

(f) Bob's scores

	Sep.	Dec.
2015	80.3	84.1
2016	70.4	76.2

Figure 2: Examples of genuine tables. Left: (a,b) vertical relational, Middle: (c,d) horizontal entity, and Right: (e,f) matrix tables. Red, green, blue cells denote subjects, properties, and objects, respectively.

methods specialized for table classification and other deep neural network architectures. The best result of TabNet was 91.05% in terms of a weighted macro-averaged F_1 score, for 3,567 tables of 200 unseen websites.

The rest of the paper proceeds as follows. First, we present background and give definitions relevant to our study. Next, we describe our neural network architecture. After that, we show experimental results for Web tables covering various structures and topics. Before concluding the paper, we discuss the contributions and originality of our study.

Preliminaries

Web Tables

Let us give definitions for explaining Web tables. Table type is defined on the basis of the semantic knowledge that tables and text around the tables contain (Crestan and Pantel 2011).

Definition 1 *Web tables are tabular structures and consist of an ordered set of N rows and M columns.*

Definition 2 *Each intersection between a row and a column determines a cell c_{ij}, where $1 \leq i \leq N$ and $1 \leq j \leq M$.*

Definition 3 *Genuine tables and text around the tables contain semantic triples of the form (subject, property, object).*

Genuine tables The WDC Web Table Corpora classifies genuine tables into relational, entity, and matrix tables (Lehmberg et al. 2016), and we use this taxonomy in this study. Figure 2 shows examples of genuine tables.

Definition 4 *Relational tables list one or more properties for a set of subjects or key aspects of a subject.*

Relational tables contain complete semantic triples themselves (Figure 2(a)), or do not contain a subject (Figure 2(b)). The latter kind lists the key aspects of the subject appearing outside the table, and the combination of an aspect and a subject forms a reified subject. The tables of Figure 2(a,b) are vertical; horizontal ones present their subjects (or key aspects) in one row.

Definition 5 *Entity tables describe one or more properties for one subject.*

Entity tables also contain complete triples or do not contain a subject. Entity tables are different from relational ones in that they describe semantic knowledge about a single subject. Figure 2(c,d) shows horizontal tables; vertical ones present a pair of property and object in a column.

Definition 6 *Matrix tables have the same property for each cell object at the junction of a row and a column.*

Matrix tables do not contain complete semantic triples, as shown in Figure 2(e,f); that is, the property is not explicitly contained in the table. The object value corresponds to a combination of two or more subjects, including ones appearing outside the table.

Other genuine tables Another three types were described by (Crestan and Pantel 2011). *Enumeration* tables list a series of objects that have the same ontological relation (e.g., hyponymy). *Calendar* tables are different from matrix tables in that they have different properties (e.g., schedule names) for each cell. *Form* tables are similar to entity tables; they have empty object fields for the users to fill in or select.

Layout (Non-genuine) tables There are two categories. Navigational tables consist of cells organized for navigational purposes. Formatting tables account for a large portion of the tables on the Web; their only purpose is to organize elements visually. We treat both the other genuine tables and the layout tables as the 'Other' type.

Semantic Structures of Tables

Here, we present the main building blocks of the semantic structures of genuine tables and show examples of the features of previous studies (Crestan and Pantel 2011; Eberius et al. 2015) that indirectly capture the semantic structures.

Subject rows and columns Vertical (horizontal) relational tables have one or more columns (rows) listing subjects; matrix tables have both of them, and entity tables do not list the subjects. The previous studies used the ratio of distinct strings for capturing subject (i.e., key) rows and columns.

Property rows and columns Vertical (horizontal) relational and entity tables have a row (column) describing properties; matrix tables do not explicitly contain its property. In the studies, the ratio of cells containing colons was used for finding the property columns of horizontal entity tables.

Blocks of sibling object cells Let a *sibling* block be a group of cells that belong to the same class. The object cells of vertical (horizontal) relational tables form multiple $n\times1$ ($1\times m$) sibling blocks along each column (row), while those of matrix tables form one sibling block. Entity tables barely contain sibling blocks. The related studies used the variance in cell string length for capturing sibling blocks.

Network Architecture

Overview of Architecture

Figure 3 shows the overall architecture of our network. The input table is fixed-sized (cropped or padded), consisting of N rows and M columns and T tokens (vocabulary size: $|V|$) in the cells. Our network begins with a token embedding that creates a vectorial representation (size of E) of each token

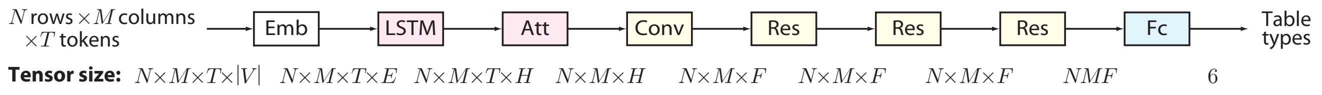
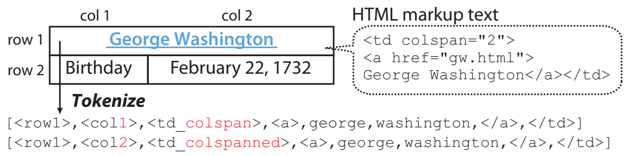

N rows $\times M$ columns $\times T$ tokens
Emb
LSTM
Att
Conv
Res
Res
Res
Fc
Table types
Tensor size: $N{\times}M{\times}T{\times}|V|$ $N{\times}M{\times}T{\times}E$ $N{\times}M{\times}T{\times}H$ $N{\times}M{\times}H$ $N{\times}M{\times}F$ $N{\times}M{\times}F$ $N{\times}M{\times}F$ NMF 6

Figure 3: Entire architecture of TabNet. An embedding layer creates a vector (size of E) from a one-hot representation (size of $|V|$) of each token. An RNN uses an LSTM with an attention mechanism to encode each cell into a vector (size of H). A CNN uses one convolutional layer that has F filters and several stacked convolutional blocks with residual units to extract semantic features for table classification. Fully connected classification layers compute the predictive probabilities for all six table types.



```
<td colspan="2">
<a href="gw.html">
George Washington</a></td>
```

```
[<row1>,<col1>,<td_colspan>,<a>,george,washington,</a>,</td>]
[<row1>,<col2>,<td_colspanned>,<a>,george,washington,</a>,</td>]
```

Figure 4: Example of tokenization of HTML markup text.

in the cells. Next, it encodes each cell (i.e., a sequence of token representations) into a fixed-size vector (size of H) using a recurrent neural network (RNN), which uses a long short-term memory (LSTM) with an attention mechanism, to obtain a semantic representation of each cell; the result is an $N{\times}M{\times}H$ third-order tensor. This tensor has the same structure as image data, i.e., height, width, and depth; hence, our network encodes the tensor using a convolutional neural network (CNN) to capture high-level semantic representations of the cell matrix. The CNN consists of a convolutional layer that has F filters and several stacked convolutional blocks with residual units. Finally, it flattens the output of the last convolutional layer (an $N{\times}M{\times}F$ tensor) into a vector and uses fully connected layers and a softmax function to compute the predictive probabilities for all the table types.

Embedding Layer

Tokenization Our network considers words, HTML tags, and row and column indexes of cells as tokens. The attributes of the HTML tags are ignored, except for `rowspan` and `colspan`, and spanned cells are discriminated from the original cell by cell indexes and the tag name, as shown in Figure 4. `<thead>`, `<tbody>`, `<tr>`, `<colgroup>`, `<col>`, `<caption>` tags are not used. All word and tag tokens are converted to lowercase.

Token embedding Each cell in a table is represented as a fixed-sized (cropped or padded) sequence of T one-hot vectors $(x_1, x_2, \dots, x_T)$. A one-hot vector of the v-th token in a vocabulary is a binary vector whose elements are all zeros, except for the v-th element, which is set to one. An embedding layer projects each of the one-hot vectors into a E-dimensional continuous vector space with a weight matrix $W_e \in \mathbb{R}^{E\times|V|}$,

$$e_t = W_e x_t, \tag{1}$$

where $|V|$ is the number of unique tokens in a vocabulary.

Recurrent Neural Network

After the embedding layer, each cell has a sequence of dense, real-valued vectors $(e_1, e_2, \dots, e_T)$. Our network uses a long short-term memory (LSTM), which reads the input sequence in reverse, with an attention mechanism to obtain a semantic representation of each cell.

Long Short-Term Memory LSTM is a special kind of RNN capable of learning long-term dependencies (Hochreiter and Schmidhuber 1997; Gers, Schmidhuber, and Cummins 2000). It defines a sequence of hidden states h_t as

$$\begin{pmatrix} f_t \\ i_t \\ o_t \\ g_t \end{pmatrix} = W_h h_{t-1} + W_x e_t + b, \tag{2}$$

$$c_t = \sigma(f_t) \odot c_{t-1} + \sigma(i_t) \odot \tanh(g_t), \tag{3}$$

$$h_t = \sigma(o_t) \odot \tanh(c_t), \tag{4}$$

where $W_h \in \mathbb{R}^{4H\times H}$, $W_x \in \mathbb{R}^{4H\times E}$, $b \in \mathbb{R}^{4H}$ are model parameters, σ is the sigmoid function, and the $\odot$ operator denotes the Hadamard product.

Attention mechanism Our network has the same attention mechanism used by (Yang et al. 2016) for extracting important tokens that strongly contribute to forming a semantic representation of each cell:

$$u_t = \tanh(W_a h_t + b_a), \tag{5}$$

$$\alpha_t = \frac{\exp(u_t^\intercal u_a)}{\sum_t \exp(u_t^\intercal u_a)}, \tag{6}$$

$$s = \sum_t \alpha_t h_t, \tag{7}$$

where $W_a \in \mathbb{R}^{C\times H}$, $b_a \in \mathbb{R}^C$, and $u_a \in \mathbb{R}^C$ are model parameters.

Convolutional Neural Network

After the RNN, our network obtains an input volume, which is an $N{\times}M{\times}H$ tensor. A convolutional layer in a CNN consists of a set of filters, which have a small receptive field, but extend the full depth of the input volume. Each filter, which computes the dot product between the entries of the filter and the input, is convolved across the width and height of the input volume and produces a two-dimensional activation map of that filter.

Our network first applies one convolutional layer that has F filters of size 3×3 with a stride of 1, followed by several convolutional blocks that we explain below. It does not conduct any pooling operations because they have a negative effect on the predictive accuracy of TabNet.

Convolutional blocks with residual units Deep residual networks (ResNets) have high accuracy and nice convergence behavior for image classification, as a result of using many (over 100 layers) stacked residual units (He et al. 2015; 2016a). This unit, which has a shortcut connection, can be expressed in a general form:

$$x_{l+1} = f\left(h(x_l) + \mathcal{F}(x_l, \mathcal{W}_l)\right), \tag{8}$$

where x_l and x_{l+1} are the input and output of the l-th unit, and $\mathcal{F}$ is a residual function. $\mathcal{W}_l$ is a set of weights associated with the l-th unit.

Our network chooses two 3×3 convolutional layers with a stride of 1 as $\mathcal{F}$. The function f is the ReLU function (Nair and Hinton 2010). The function h is an identity mapping for all stacked units: $h(x_l) = x_l$. This shortcut connection can reduce the risks of overfitting.

Batch normalization Batch normalization (BN) is a technique to improve learning in neural networks by normalizing the distribution of each input feature in each layer across each minibatch to $\mathcal{N}(0, 1)$ (Ioffe and Szegedy 2015). Our network adapts BN right after each convolution and before ReLU activation, in the same way as the original ResNet (He et al. 2015). It does not use dropout (Srivastava et al. 2014) in the overall architecture, including the RNN.

Classification Layers

At the end of the last convolution, our network has an $N \times M \times F$ tensor. It flattens the tensor into a vector and uses several fully connected layers that reduce the size to six classes of table types. Our network uses ReLU as the activation function of the intermediate layers and adapts BN before ReLU activation. The output of the last layer is fed to a six-way softmax, which produces the predictive probabilities for all table types.

Learning of Networks

The entire network can be trained end-to-end by using stochastic gradient descent (SGD) with backpropagation and can be easily implemented using common libraries without modifying the solvers. It is worth noting that pre-training only the embedding matrix with Word2Vec (Mikolov et al. 2013a; 2013b) or GloVe (Pennington, Socher, and Manning 2014) from a very large text corpus is effective when the number of training tables is not so large.

Experiments

Dataset

The evaluation used a subset of April 2016 Common Crawl. We collected 272,888 tables from the crawl data and extracted 190,288 tables of the top 500 websites containing the most tables in the subset (the largest one is wikipedia.org). We eliminated small tables with fewer than two rows or two columns and nested tables from the dataset. In total, we obtained 64,245 tables covering various structures and topics. An expert annotated the type of table into six types: vertical and horizontal relational (VR and HR), vertical and horizontal entity (VE and HE), matrix (M), and other (O) tables.

Table 1: Numbers of table classes in the dataset.

table type	train	test	total
Vertical Relational (VR)	11,397	328	11,725
Horizontal Relational (HR)	255	38	293
Vertical Entity (VE)	390	75	465
Horizontal Entity (HE)	15,618	879	16,497
Matrix (M)	662	18	680
Other (O)	32,356	2,229	34,585
total	60,678	3,567	64,245

Note that a website has a large number of very similar tables, and random partitioning of the dataset causes the test set to contain *seen* data. We therefore split up the dataset by website; we used 60,678 tables in the top 300 of the 500 websites for training and the remaining for testing (Table 1).

Evaluation Metric

The training and test datasets are imbalanced. They contain many more samples from VR, HE, and O types than from the rest of the types. We therefore used a weighted macro-averaged F_1 score as the evaluation metric as in (Eberius et al. 2015); it calculates F_1 scores for each table type and finds their average, weighted by the support (the number of true instances for each type).

Baselines

We compared our method with several baseline methods, including traditional approaches and neural network architectures that can be used for table type classification.

Random Forests with handcrafted features A number of studies proposed handcrafted features. We used Random Forests with the features of the following studies.

- **Cafarella08** (Cafarella et al. 2008b) described seven features of the whole table; e.g., the number of columns with non-string data and the variance in cell string length.
- **Crestan11** (Crestan and Pantel 2011) considered local features for the first two rows and columns as well as the last row and column instead of only considering global features for the table as a whole. They used 107 structural, HTML, and lexical features per tables.
- **Eberius15** (Eberius et al. 2015) extended Crestan11 by adding global and local features. They used 127 features per tables, and these features were used for building the Dresden Web Table Corpus.

Neural networks To the best of our knowledge, deep neural networks have not yet been used for classifying tables. We used a state-of-the-art document classification method that is suitable for classifying tables.

- **Hierarchical Attention Network (HAN)** (Yang et al. 2016) is a two-layered RNN with an attention mechanism for classifying documents that have a hierarchical structure (words form sentences, sentences form a document). Tables have a hierarchical structure as well (tokens from cells, cells from a row, rows from a table); therefore, we constructed a three-layered HAN for encoding tables. We used the same RNN blocks that our architecture used as the RNN encoder blocks of HAN.

- **Bidirectional HAN** uses another hierarchical structure in which cells form a column and columns form a table. The bidirectional HAN concatenates the outputs of rows-directional and columns-directional HANs, and this concatenated vector is fed into the classification layers.

Model Configuration

Pre-training We conducted pre-training for token embedding. For word tokens, we obtained a pre-training word embedding matrix using Word2Vec with the skip-gram model and negative sampling (Mikolov et al. 2013a; 2013b). We used full texts in Wikipedia article pages for pre-training, where the tables in the pages were not included in the test dataset. We only retained words appearing in the pre-training and replaced the other words with a special UNK token. For other tokens (HTML tags and row and column indexes), we randomly initialized the columns of the embedding matrix corresponding to the tokens. We set the size of the token embedding, E, to 100.

Training We preliminary confirmed that the top-left part of tables contains sufficient information to classify them; all tables consisted of 8 rows and 8 columns cropped from the top-left corner or padded with empty cells in the bottom-right part. Also, each cell had 50 tokens cropped from the beginning of sequences or padded with special PAD tokens at the end of the sequences, where the PAD tokens were ignored in the RNN encoding. This limitation can avoid the slow processing of the RNN for very long texts in a cell.

We used Chainer (Tokui et al. 2015), a framework of neural networks, for implementing our architecture. We initialized the weights for the CNN as in (He et al. 2015). For the LSTM, we initialized the forget bias with ones as in (Józefowicz, Zaremba, and Sutskever 2015), the hidden-to-hidden weights with orthogonal initialization (Saxe, McClelland, and Ganguli 2013), and input-to-hidden weights with Xavier initialization sampled from a uniform distribution (Glorot and Bengio 2010). We used SGD with a momentum of 0.9 and a minibatch size of 50. The number of epochs (one pass of all training tables) was five, and the learning rates were 0.1, 0.1, 0.1, 0.01, and 0.001 for each epoch. The hyperparameters of the models were tuned on a validation set generated by selecting 60 websites in the training dataset: $H = 100$, $C = 100$, and $F = 32$. The number of convolutional blocks with residual units was three, and the number of fully connected layers was two (100 and 6 neurons). The total number of weighted layers was 12.

Results

Does our architecture work better than conventional methods and other deep architectures? Table 2 shows that weighted macro-averaged F_1 scores for the test dataset when each method was trained with the full training dataset. Our architecture, TabNet, achieved the best F_1 value by a single model, 88.42%, averaged over five trials with different initializations. TabNet statistically significantly performed better than Eberius15, a state-of-the-art method specialized for table classification, and Bidirectional HAN, another deep neural network architecture (t-test; $p < .05$).

Table 2 also shows that an ensemble (majority voting, where ties were broken randomly) of five TabNets, trained with different weight initializations, outperformed a single-model of TabNet, by up to 2.63%.

Table 2: Weighted macro-averaged F_1 for the test dataset. The results of NNs indicate mean ± SE over five trials.

method	weighted macro F_1
Cafarella08	0.6926
Crestan11	0.8114
Eberius15	0.8165
HAN	0.8409 ± 0.0056
Bidirectional HAN	0.8562 ± 0.0045
TabNet	**0.8842** ± 0.0070
Ensemble of 5 HANs	0.8471
Ensemble of 5 Bidirectional HANs	0.8652
Ensemble of 5 TabNets	**0.9105**

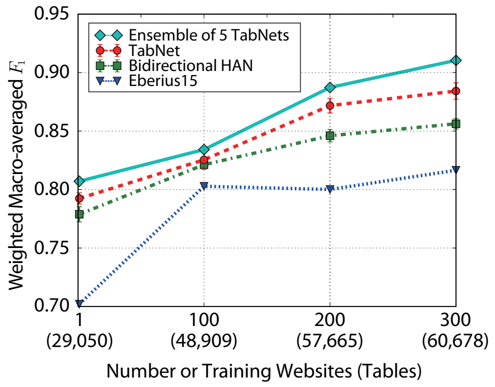


Figure 5: Effect of the number of training websites. The error bars of TabNet and Bidirectional HAN indicate standard errors over five trials.

Do deep neural networks work well on a large number of tables? We compared predictive accuracies while varying the number of training websites: 1, 100, 200, and 300 (29,050, 48,909, 57,665, and 60,678 tables, respectively). Figure 5 shows that TabNet and Bidirectional HAN performed well for larger numbers of training data. These results indicate that learning many tables covering various structures and topics helps deep neural networks to understand the semantic structures of tables.

Are the errors of our architecture reasonable? Figure 6 shows the confusion matrices of TabNet and Eberius15 across all six table types. Eberius15 overlooked most of the M and HR tables and had unacceptable levels of confusion in regard to the semantic definitions, e.g., VR vs. HE. On the other hand, the level of confusion of TabNet, e.g., VR vs. VE, was relatively reasonable. This analysis suggests that TabNet captures high-quality semantic features from tables.

Can our architecture capture the semantic features of tables? Figure 7 shows the averaged activations of parts of

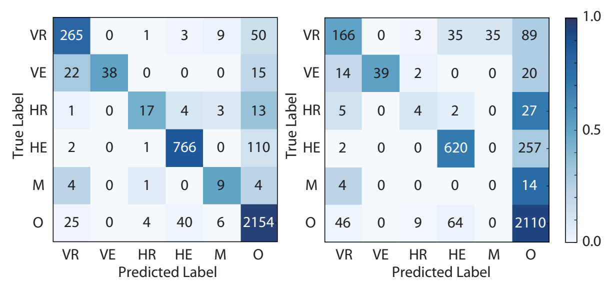


Figure 6: Confusion matrix normalized by row. Left: Ensemble of five TabNets. Right: Eberius15.

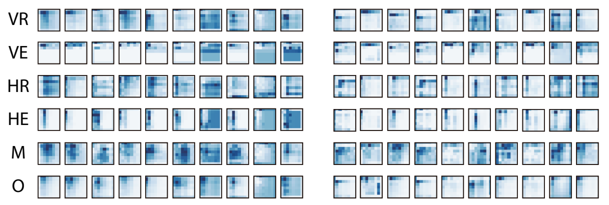


Figure 7: Visualization of activations of parts of filters in the first (Left) and last (Right) convolutional layer. Activations were grouped by table type and averaged over the test tables. The six vertical boxes correspond to the same filter.

filters in the first and last convolutional layers for each table type. Interestingly, relational and entity tables had line-type activations in the last convolutional layer, which capture rows and columns describing properties, while such activations hardly occurred in the maps of matrix tables that do not explicitly contain property cells.

In addition, we see that the activations for matrix tables were broadly distributed in the maps. This captures the sibling relations between object cells that belong to the same property. It is difficult to directly capture a semantic block (not a sequence) of cells with the hierarchical RNNs of HANs; the CNN of TabNet is effective for capturing various building blocks that have different shapes and sizes.

Discussion

Contribution of this study to research on tables Table type classification is a fundamental technology for exploiting the potential of table knowledge. For example, Google's Knowledge Vault uses vertical relational tables as an information source for constructing a knowledge base (Dong et al. 2014). A Microsoft team constructed a fact lookup engine based on horizontal entity tables (Yin, Tan, and Liu 2011). Concept-instance pairs were extracted by (Dalvi, Cohen, and Callan 2012) from vertical relational tables. The concept expansion method proposed by (Wang et al. 2015) found many entities from few entity seeds and vertical relational tables. (He et al. 2016b) extracted attribute synonyms using query logs and vertical relational tables. Our study contributes to all of these studies through the development of a high-quality table corpus that they need.

Moreover, our study is independent of existing knowledge bases, unlike the conventional studies for annotating tables (Limaye, Sarawagi, and Chakrabarti 2010; Venetis et al. 2011) or entity linking (Mulwad, Finin, and Joshi 2013; Bhagavatula, Noraset, and Downey 2015). This is a good characteristic for collecting detailed information that is not included in the knowledge bases from Web tables.

Originality of our hybrid deep architecture Architectures combined with RNNs and CNNs have been extensively studied. Most representative studies are on image caption generation (Xu et al. 2015), and they have been used in other important tasks: image segmentation (Chen et al. 2015; Visin et al. 2015), speech recognition (Sainath et al. 2015), and document classification (Xiao and Cho 2016; Tang, Qin, and Liu 2015). These architectures first use a CNN and then an RNN, unlike our architecture. Our architecture is a novel one that first uses an RNN to encode the semantics of cells and then uses a CNN to learn relations among cell semantics. Experimental results showed that it captures the structure of tables, i.e., a matrix of token sequences.

Potential applications Although TabNet was designed to mirror the structure of tables, it can be applied to any matrix, including list ($N \times 1$ matrix), of sequences. Moreover, although we have not yet confirmed that it has high levels of accuracy on other datasets, we think that classification of structured documents will be a promising application. For example, in calculating the coherence of dialogue (Higashinaka et al. 2016), we can form a matrix of utterances: each cell c_{ij} of the matrix contains an utterance (a sequence of words) of speaker j at conversation turn i. Hierarchical RNNs (Yang et al. 2016) or very deep CNNs (Conneau et al. 2016) would be competitive baselines.

Conclusions

We proposed a new deep neural network architecture, TabNet, for table type classification, where six table types are defined on the basis of semantic triples of the form (subject, property, object) that the tables contain. Our architecture reflects the structure of tables: each cell has a sequence of tokens, and a table is a matrix of cells. It consists of an RNN that encodes token sequences and a CNN that extracts semantic features, e.g., the existence of rows describing properties, to classify table types.

We constructed a table corpus that contained 64,245 Web tables covering various structures and topics by using a subset of public Common Crawl data. Experimental results showed that an ensemble of five TabNets with different weight initializations achieved the best result, 91.05% in terms of a weighted macro-averaged F_1 score, for 3,567 unseen tables. This result is an improvement over those of the state-of-the-art method specialized for table classification (Eberius et al. 2015) and another deep architecture (Yang et al. 2016) up to 9.40% and 5.43%, respectively. We confirmed that the error confusions of TabNet were reasonable and that the semantic features for table type classification were in the output of the CNN.

In the future, we plan to improve the knowledge-base construction and knowledge search using the high-quality table corpus that TabNet classified. We will also demonstrate our architecture's potential by applying it to other datasets.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Bhagavatula, C. S.; Noraset, T.; and Downey, D. 2015. TabEL: Entity linking in web tables. In *ISWC*, 425–441.

Cafarella, M. J.; Halevy, A. Y.; Wang, D. Z.; Wu, E.; and Zhang, Y. 2008a. Webtables: exploring the power of tables on the web. *PVLDB* 1(1):538–549.

Cafarella, M. J.; Halevy, A. Y.; Zhang, Y.; Wang, D. Z.; and Wu, E. 2008b. Uncovering the relational web. In *WebDB*.

Chen, L.; Barron, J. T.; Papandreou, G.; Murphy, K.; and Yuille, A. L. 2015. Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform. eprint arXiv:1511.03328.

Conneau, A.; Schwenk, H.; Barrault, L.; and LeCun, Y. 2016. Very deep convolutional networks for natural language processing. eprint arXiv:1606.01781.

Crestan, E., and Pantel, P. 2010. A fine-grained taxonomy of tables on the web. In *CIKM*, 1405–1408.

Crestan, E., and Pantel, P. 2011. Web-scale table census and classification. In *WSDM*, 545–554.

Dalvi, B. B.; Cohen, W. W.; and Callan, J. 2012. Websets: extracting sets of entities from the web using unsupervised information extraction. In *WSDM*, 243–252.

Dong, X.; Gabrilovich, E.; Heitz, G.; Horn, W.; Lao, N.; Murphy, K.; Strohmann, T.; Sun, S.; and Zhang, W. 2014. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In *KDD*, 601–610.

Eberius, J.; Braunschweig, K.; Hentsch, M.; Thiele, M.; Ahmadov, A.; and Lehner, W. 2015. Building the dresden web table corpus: A classification approach. In *BDC*, 41–50.

Gers, F. A.; Schmidhuber, J.; and Cummins, F. A. 2000. Learning to forget: Continual prediction with LSTM. *Neural Computation* 12(10):2451–2471.

Glorot, X., and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 249–256.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep residual learning for image recognition. eprint arXiv:0706.1234.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016a. Identity mappings in deep residual networks. eprint arXiv:1603.05027.

He, Y.; Chakrabarti, K.; Cheng, T.; and Tylenda, T. 2016b. Automatic discovery of attribute synonyms using query logs and table corpora. In *WWW*, 1429–1439.

Higashinaka, R.; Funakoshi, K.; Kobayashi, Y.; and Inaba, M. 2016. The dialogue breakdown detection challenge: Task description, datasets, and evaluation metrics. In *LREC*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780.

Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 448–456.

Józefowicz, R.; Zaremba, W.; and Sutskever, I. 2015. An empirical exploration of recurrent network architectures. In *ICML*, 2342–2350.

Lehmberg, O.; Ritze, D.; Ristoski, P.; Meusel, R.; Paulheim, H.; and Bizer, C. 2015. The mannheim search join engine. *J. Web Sem.* 35:159–166.

Lehmberg, O.; Ritze, D.; Meusel, R.; and Bizer, C. 2016. A large public corpus of web tables containing time and context metadata. In *WWW*, 75–76.

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

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013a. Efficient estimation of word representations in vector space. eprint arXiv:1301.3781.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013b. Distributed representations of words and phrases and their compositionality. In *NIPS*, 3111–3119.

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

Nair, V., and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*, 807–814.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *EMNLP*, 1532–1543.

Sainath, T. N.; Vinyals, O.; Senior, A. W.; and Sak, H. 2015. Convolutional, long short-term memory, fully connected deep neural networks. In *ICASSP*, 4580–4584.

Saxe, A. M.; McClelland, J. L.; and Ganguli, S. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. eprint arXiv:1312.6120.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

Tam, N. T.; Hung, N. Q. V.; Weidlich, M.; and Aberer, K. 2015. Result selection and summarization for web table search. In *ICDE*, 231–242.

Tang, D.; Qin, B.; and Liu, T. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *EMNLP*, 1422–1432.

Tokui, S.; Oono, K.; Hido, S.; and Clayton, J. 2015. Chainer: a next-generation open source framework for deep learning. In *LearningSys Workshop at NIPS*.

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.

Visin, F.; Kastner, K.; Courville, A. C.; Bengio, Y.; Matteucci, M.; and Cho, K. 2015. Reseg: A recurrent neural network for object segmentation. eprint arXiv:1511.07053.

Wang, C.; Chakrabarti, K.; He, Y.; Ganjam, K.; Chen, Z.; and Bernstein, P. A. 2015. Concept expansion using web tables. In *WWW*, 1198–1208.

Xiao, Y., and Cho, K. 2016. Efficient character-level document classification by combining convolution and recurrent layers. eprint arXiv:1602.00367.

Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A. C.; Salakhutdinov, R.; Zemel, R. S.; and Bengio, Y. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2048–2057.

Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A.; and Hovy, E. 2016. Hierarchical attention networks for document classification. In *NAACL HLT*, 1480–1489.

Yin, X.; Tan, W.; and Liu, C. 2011. FACTO: a fact lookup engine based on web tables. In *WWW*, 507–516.