*Article*

International Journal of Market Research

International Journal of Market Research
2019, Vol. 61(6) 601–620
© The Author(s) 2019
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/1470785319841217
journals.sagepub.com/home/mre
SAGE

# Machine learning applied to pack classification

**Weiqiang Hang**
National University of Singapore, Singapore

**Timothy Banks**
Nielsen, USA

**Abstract**
Pack or product classification is a quite common task in market research, particularly for sales tracking audits and related services. Electronic data sources have led to increased volumes, both in the sales volume being tracked and also the number of packs (or stock keeping units). The increase in packs needing to be classified presents a problem, in that, it needs to be done accurately and quickly. Traditional solutions using people for the classifications can be costly, due to the large number of people required to process the classifications in a timely and accurate manner. Reducing the manual work is a priority for audit-based market research businesses, leading to interest in automation, such as through machine learning techniques. In this article, we apply such methods. These include support vector machine, decision tree, XGBoost, AdaBoost, random forest, and neural network–based methods that are trained on the textual descriptions of already classified packs. We also implement a hierarchical classification method to take advantage of the structure of classes of the products. Once the models are trained, they can be used on unclassified data. Where the methods are not confident in their classifications, humans can be asked to classify. The hope is that the methods can learn to classify accurately enough that the manual workloads are reduced to manageable levels. This article reviews various methods and then outlines tests using these methods on two datasets collected by Nielsen, showing good performance.

**Keywords**
machine learning, natural language processing, pack classification

**Corresponding author:**
Weiqiang Hang, Department of Statistics & Applied Probability, National University of Singapore, Block S16, Level 7, 6 Science Drive 2, Singapore 117546.
Email: E0010758@u.nus.edu

## Introduction

Tracking services or audits have been the staple of a number of marketing research companies, such as IQVIA with its pharmaceutical sales tracking audits and Nielsen with its fast-moving consumer goods' (FMCG) audits. Such audit services provide a regular read of a defined market, supporting marketing decisions by client companies. Clients rely on such services to provide insights informing business decisions, such as whether to enter a new market (through marketing sizing and competitive analysis) through to measuring their performance inside a market against their competitors. More granular services can provide information, such as inside sales territories allowing client companies to size the market accessible to their individual sales representatives, target their sales activities better, measure their performance against competitors and against the market available inside a territory, decide the boundaries of sales territories, and finally use the data as the basis of incentive programs for the sales staff. Quality is of utmost importance to such services, increasingly so as the granularity of the service increases in both geography and time, leading audit companies to vigorously explore data sources to expand coverage and accuracy.

Such new data sources are presenting opportunities and challenges to these companies. "Big Data" is frequently defined with references to the "three v-s": volume, variety, and velocity. Data volumes have been increasing at a rapid rate in recent years, for example, via the rapid popularity of smart phones and Internet-connected devices that all can provide data. This is a trend that looks certain to continue for at least the immediate future. Processing these data in a timely manner had led to the development of new cluster computing technologies, such as Spark (Zaharia et al., 2011), where a group of computer processors can divide up a work problem between them, process the individual chunks, and then consolidate the results obtained across the group. Such parallel computing can lead to dramatic improvements in not only the volume of the data able to be processed, but also how quickly it can be processed. Variety comprises of data changing, such as in its structure (as available fields or elements), consolidating across different data sources containing different levels of information, or literally in the number of different cases or types being presented. In this study, variety involves the information available in the written descriptions of pack items, which can vary across pack descriptions both across suppliers and inside an individual supplier, along with the changes in descriptions with time as data capture changes. This is separate to volume that can be thought of as the number of lines to be processed. Velocity concerns the speed of the data flowing in and needing to be processed, as well as the speed that the data need to be processed into actionable information. A fourth "v" is sometimes added: value. Not all data are equal, some provide useful insights into an issue and others do not. Data might be generated for a particular problem, but with ingenuity could be turned to another use case. A simple example would be use of night-time satellite imagery by Nielsen to measure urbanity in developing countries (cities are brighter than rural areas), leading to improvements in statistical sample design (Tung, Zhao, & Banks, 2014). Volume and velocity are usually the easiest of the "v-s" to address, through sampling or extra computing facilities. Value is the most important, and the problem of variety will be discussed further in this article.

This article will concentrate on FMCG data sourced electronically from the panels of thousands of panelists, as prototypes where successful classification methods (from these pilot studies) could later be applied to data sourced from the panels of millions of consumers.[1] These large data sets can lead to higher accuracy reporting, as well as more granular insights into a market place, but they come at the cost of substantially increased processing. The price of this increasing processing and handling is the need for increased automation and judicious assignment of human involvement. Automation can lower production costs, allow faster delivery to clients, and free up human involvement from less mundane tasks to higher impact ones.

Pack labeling or classification is a fundamental task in market research, accurately assigning details, such as manufacturer, brand, pack size, price, active ingredient, and the like. After accurate labeling of packs sold in the market, we can base market analyses on concrete sales data for different kinds of products, assisting clients to better monitor markets, and make better business decisions for those markets. However, pack labeling poses a great challenge to market research. As noted above, new data sources are increasingly being explored, such as sales audit panels consisting of literally millions of individuals contributing their online purchases. The volume of data is huge, leading often to improvements in quality over previous data sources and for the subsequent market trackers based on them. The number of classes being reported on can be counted in thousands and the number of transactions in the data sets is of the order of many millions. Moreover, the data are generated at rapid speed. Quick processing and classification of packs are, therefore, required. Finally, we need the pack be classified thoroughly, at different levels (such as manufacturer, brand, and packing) in order to provide more comprehensive information to users of these market tracking services. Extracting this information can be difficult, given the abbreviations and truncations present in many pack descriptions, particularly where the data source is based on the receipts and not on an inventory tracking system.

It is not really practical to simply scale-up manual solutions to pack classification. Manual labeling is suitable for small-scale and complex datasets that may require some domain knowledge. As we will explain in this article, we see manual processing as helpful in an automated process to handle exceptions and generate labeled cases that can be injected back into the (machine) learning process.

However, the advance in data science provides the possibility of using automated solutions to pack classification and reducing the level of manual work to manageable levels and costs. There are many methods available for classification. They use some labeled data as training data and learn the relationship between the information about the pack itself and that contained in the label description about the pack, then these methods can apply the learned relationships to classify new data. There are many tools providing very efficient and user-friendly implementation of these methods, such as sklearn (Pedregosa et al., 2011), Keras (Chollet, 2015), TensorFlow (Abadi et al., 2016), PyTorch (Paszke, Gross, Chintala, & Chanan, 2017), and XGBoost (Chen & Guestrin, 2016). In this article, we will introduce several methods using these libraries, including support vector machines (SVMs), neural networks, classification trees, and random forests. All of these techniques will use the textual description of product for classification.

Application of machine learning techniques is still relatively new to marketing research. For this reason, we will outline in this article the various methods of the National University of Singapore/Nielsen partnership trialed in this research project to provide both background to the work and also in the hope that this article will encourage other market research professionals to explore both big data and machine learning applications. This project is part of a wider partnership between the two organizations working on a range of projects from advanced statistical analysis to better verbal reporting of purchase data (Olsen, Lin, & Banks, 2017) through to machine learning techniques applied to improve the efficiency of large-scale sampling (in the published case, some 32,000 respondents), data collection, and processing (Seah, Tung, & Banks, 2015). Further details on this wider program can be found in Banks and Budding (2018).

## Techniques

In this section, we will explain the underlying ideas of the algorithms for SVMs, neural networks, XGBoost, and random forests.

While later in the article we go in detail into these methods, a short overview might help set the scene. The SVM is an algorithm that tries to find the optimal boundary separating classes of data points. In two dimensions, this would be some boundary line. The greater the number of groups needing to be classified, the greater the number of dimensions required for this boundary which, in these cases, is called a hyperplane. Programming details on SVMs may be found in the very readable introduction of James, Witten, Hastie, & Tibshirani (2013). This text is also a good practical introduction to random forests that are basically the collections of decision trees. A neural network is a collection of connected input/output units (commonly called neurons by analogy to the similarly named brain cells), which individual perform usually simple computations, such either "firing" a signal should a particular condition be reached. The collection of these computation units into a network, where the cells communicate with each other, can lead to astonishing results. If supplied with suitable training data, a neural network can be "taught" to classify similar data provided later. Finally, XGBoost is an example of a boosting algorithm, where consecutive learning algorithms are used to improve their predecessor's fit to the dataset. Chapter 8 of Han, Kamber, and Pei (2012) provides good background on these last two techniques.

First, we want to introduce several notations that will be used in the article. We first consider binary classification. Denote $x \in \mathbb{R}^p$ as the $p$-dimensional covariate variables and $y \in \{-1,1\}$ as the class given $x$. We have $n$ pairs of data $(x_1, y_1),\ldots,(x_n, y_n)$ following some unknown distribution as training data. We want to find a map $f(\cdot)$ between the domain of $x$ to $\{-1,1\}$, which minimizes the prediction error, $E[L(y, f(x))]$, where $L(\cdot,\cdot)$ is predefined loss function.

## Data preparation

Machine learning methods only receive numerical values as input. It is necessary to convert the textual data into the numerical data. The frequency-inverse document frequency (Tf-idf; Salton & Buckley, 1988) provides a convenient method for conversion. Tf-idf flags the prevalence of a word of interest in a given text as compared with its prevalence in an overall dictionary. Let $V = (w_1,\ldots,w_m)$ denote the dictionary that contains $m$ words. The, $m$, most frequent words in the records are usually included in the dictionary. Denote $d_i = (w_{i_1},\ldots,w_{i_L})$ as the textual description of length $L$ of the $i$ th record. Denote $tf(d_i, j)$ as the count of $w_j$ in $d_i$ and $idf(j) = log(n / c_j)$, where $c_j$ is the number of textual descriptions that contain $w_j$. Then the corresponding covariate variable $x_i = (x_{ij})_{j=1}^n$ of $d_i$ can be generated by $x_{ij} = tf(d_i, j) \cdot idf(j)$. The $idf$ term offsets the weights of frequent but non-informative words like "the."

Moreover, if the maximum length of item descriptions in the dataset is quite short, one-hot encoding[2] can be a useful technique. An array is initialized equivalent in length to the number of words[3] to be encoded, with each element set to zero. For a given word in a string, the corresponding element in the array is set to 1, indicating the presence of the word. Moreover, we can also consider the collocation, since some combinations of words may carry different meanings compared to that of individual word. If there is enough data, wider semantic units beyond a single word can be considered in the dictionary, such as collocation. Some collocations will carry more accurate meaning than taking words alone. For example, many drink products have "*chocolate drink*" in their descriptions. If we decompose it into "*chocolate*" and "*drink*," it is likely that it will be classified in food category. A problem with one-hot encoding is that large arrays can be built for strings, with the majority of elements being set to zero. However, there are methods for handling such sparse matrices, reducing their impact on computer memory, such as classes of sparse matrices in SciPy (Jones, Oliphant, & Peterson, 2014). We also discuss below word embedding techniques that address this problem.

## Support vector machines

This is a popular classification method (see, for example, Chapter 9 of James et al., (2013)) for a general introduction), sometimes known as maximal margin classifiers. The following explains the general principle of the method.

We first define a hyperplane by

$$\{x : f(x) = xw + w_0 = 0\}$$

where $w \in \mathbb{R}^{p\times 1}$ and $x \in \mathbb{R}^{1\times p}$. The output of the classifier is determined by the sign of $f(x)$ when $x$ is given

$$G(x) = sign(f(x))$$

Assume that we have a training dataset of size $n$, $(x_1, y_1), \ldots, (x_n, y_n)$. If the data are linearly separable, we can find the hyperplane satisfying

$$\begin{aligned} f(x_i) \geq 1, \quad &\text{if} \quad y_i = 1 \\ f(x_i) \leq -1, \quad &\text{if} \quad y_i = -1 \end{aligned}$$

Denote the hyperplanes $H_0 = \{x : f(x) = 1\}$ and $H_1 = \{x : f(x) = -1\}$. The margin is defined as the distance between the hyperplanes $H_0$ and $H_1$. The SVM method finds the hyperplane that separates the data by the largest margin. Figure 1 illustrates the concept of margin of the hyperplane.

By calculation, we can find that the margin can be expressed the reciprocal of $\| \beta \|$. Therefore, the hyperplane can be estimated through the following constraint optimization problem

$$\begin{aligned} &\min \| w \| \\ s.t. \quad & y_i f(x_i) \geq 1 \end{aligned} \tag{1}$$

The above method is known as the hard margin linear SVM method. We can introduce “slack” variables if the data are not linearly separable, which allow small deviations from the constraints

$$\begin{aligned} &\min \| w \| \\ s.t. \quad & y_i f(x_i) \geq 1 - \xi_i, \forall i \\ & x_i \geq 0, \sum\nolimits_i \xi_i \leq C \end{aligned} \tag{2}$$

where $C$ is some constant that controls the sum of deviation from the original constraints.

Moreover, if the data are not linearly separable (i.e., no straight line margins are presented between the different classes), we can use the feature function $\phi(x)$ to map the covariate variable to a high-dimensional feature space and then find the hyperplane in that feature space. In practice, such a feature map is realized by the so-called "kernel trick" with different kernels. There are three commonly used kernels:

- Linear kernels correspond to identity maps that do nothing to the original features.
- Polynomial kernels map the feature to its higher order polynomial factor, such as map two-dimensional vector $v = (v_1, v_2)$ to $(v_1, v_2, v_1^2, v_2^2, v_1 v_2,)$.
- Radial basis function kernels that map the feature to very high-dimensional space.

Polynomial and radial basis function kernels can capture complex relationships between $x$ and $y$ at expense of larger size of training data. In this project, since the dimension of variables, equal to the number of words considered, is very large, and training size is limited, we use a linear kernel in practice. The simulation result also demonstrates that linear kernel is the best option for the current data.

## *Ensemble methods*

*Boosting method.* The boosting method is a very famous ensemble method (Freund & Schapire, 1997; Friedman, 2001; Friedman, Hastie, & Tibshirani, 2001), which in each iteration fits the data with a weak estimator based on the estimated model in the previous iteration, and then ensembles these models into a strong estimator.

Denote the loss function as $L$. It may take the form like $L(y, f(x)) = (y - f(x))^2$ for regression problems and $L(y, f(x)) = \exp(-yf(x))$ for classification problems. The empirical loss of model function $f(x)$ in the training data is defined as $\sum_{i=1}^{n} L(y_i, f(x_i))$. Generally, $f$ is estimated by minimizing the loss function with a constraint on the complexity of model $f(x)$. The boosting methods estimate $f(x)$ through iterative greedy algorithms. In the $k$ th iteration, the simple update $h_k(x)$ achieves the steepest descent of the empirical loss of training data

$$h_k = \arg\min_{h \in \mathcal{S}} \sum_{i=1}^{n} L(y_i, f_{k-1}(x_i) + h(x_i))$$

where $\mathcal{S}$ is the set of some simple functions. Since in each iteration, the update function is searched inside $\mathcal{S}$, this algorithm has good control on the complexity of the estimated model. $f_{k-1}(x)$ is the model estimated in the previous iterations and it is updated by $f_k(x) = f_{k-1}(x) + h_k(x)$.

Classification and regression tree (CART) is a famous and successful boosting method (Breiman, 2017). CART constructs a decision tree to predict the value of response. Figure 2 gives an example of tree model. Each node in the tree represents a split and the leaf gives the value of output. When constructing the tree, in each step, each possible split is evaluated based on the loss. The split with minimum loss is chosen to append to the tree (Chen & Guestrin, 2016). For the gradient boosting method (Friedman, 2001), the update is estimated through the gradient of empirical loss, which is like the steepest descent method. In practice, validation data are used by the algorithm as a watch to determine whether to stop the iterations.

*Bagging method.* The bagging method, which is short for bootstrap aggregating, is another ensemble method. It is useful as it reduces the variance of the estimated model. The method can be justified by a simple idea that when we have a disease or illness, we can visit several doctors for better and accurate diagnosis.

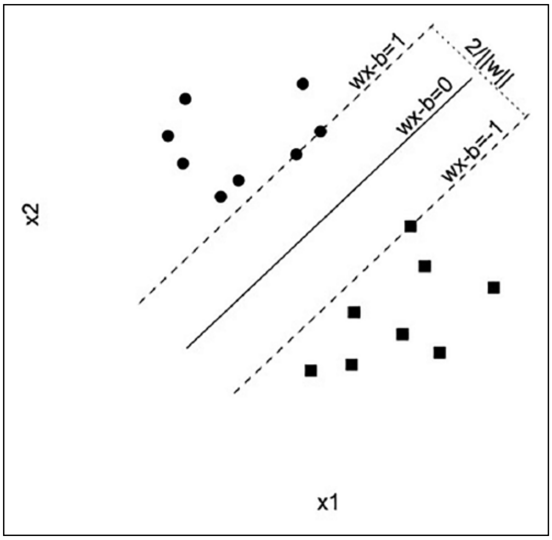

**Figure 1.** The margin of hyperplane. There are two classes of points: round and square points. The two classes are classified by the hyperplane: $wx - b = 0$. Moreover, the hyperplane is with largest margin that is defined as the distance between the hyperplane and the closest points.

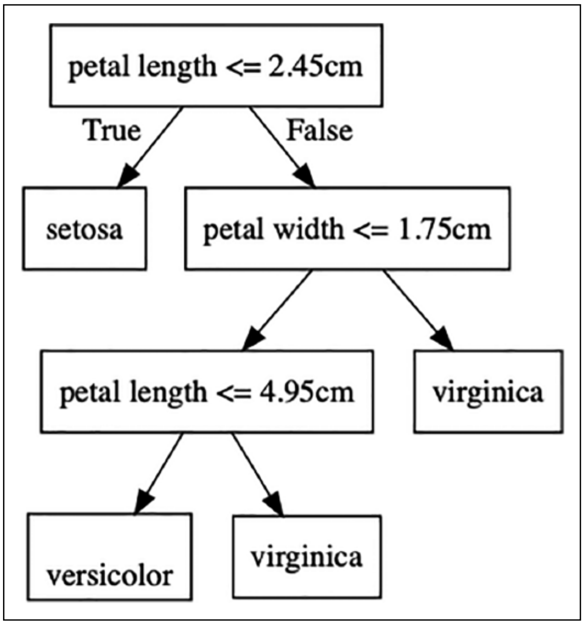

**Figure 2.** Example of decision tree. It predicts the three species of iris flowers (setosa, versicolor, and virginica), according to sepal length and width, and petal length and width.

Given a training data of size $n$, we can generate $m$ samples of size $n'$ by sampling uniformly from the original training data with or without replacement. Then $m$ different models are trained with the $m$ samples and the final result is determined by the votes of the $m$ estimated models.

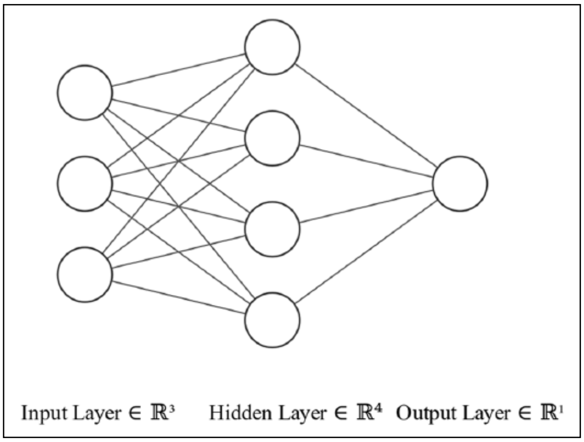
Input Layer ∈ $\mathbb{R}^3$
Hidden Layer ∈ $\mathbb{R}^4$
Output Layer ∈ $\mathbb{R}^1$

**Figure 3.** Single-layer neural network. The three-dimensional input layer receives three values and passes them to the hidden layer. The hidden layer contains four nodes and they receive inputs from the input layer. The output layer contains one node and its input are output of nodes in the hidden layer.

In practice, we will use the boosting and bagging methods simultaneously to achieve a better estimation of the model.

## *Neural networks*

The neural network is a very flexible machine learning method and has witnessed great advances in recent years in applications to some complex tasks like image classification and machine translation. It is a parametric model that is composed of many nodes connected with each other to form a network, usually in layers. Figure 3 illustrates an example of a single hidden-layer network. Each node in the hidden and output layers transforms the input by

$$f_i(x_i) = \sigma(x_i\beta_i + \alpha_i)$$

where $x_i$ is the input of the $i$ th node, $\beta_i$ and $\alpha_i$ are the parameters of that node, and $\sigma(\cdot)$ is called the activation function. The activation function provides a nonlinear transformation on the linear projection of features; otherwise, a neural network only transforms features linearly, which is not enough to capture complex relationships between labels and features. The following are several commonly used examples of activation functions:

- ReLu: $\sigma(z) = \max(z, 0)$,
- sigmoid: $\sigma(z) = \frac{1}{1+\exp(-z)}$,
- tanh: $\sigma(z) = tanh(z)$.

Then techniques like backpropagation (LeCun et al., 1990) and stochastic gradient descent (SGD; Bottou, 2010) are used to update the values of parameters to optimize the empirical loss.

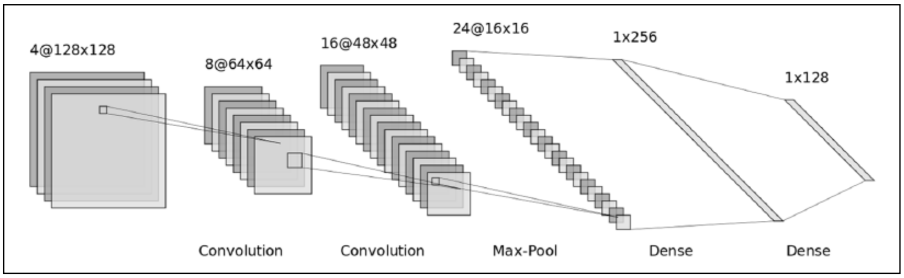

**Figure 4.** The example of CNN. The network is composed of several convolution layers, max-pooling layer, and dense layer. Each node in convolution layer is connected with a small square region of prior layer.

## *Word embedding*

Although we can use "Tf-idf" or one-hot encoding of text data as input to neural networks, these kinds of data are too sparse and the scale of data is too large, so it is very difficult for a neural network to handle.

The more efficient method is to embed each word $w_j$ in vocabulary $V$ into a $d$-dimensional word vector, $v_j \in \mathbb{R}^d$, whose values are the parameters in the neural network and will be estimated in the training process. The dimension $d$ is usually set to be around 100, like 64, 128, or 256, depending on the size of the data collected. Moreover, there are many word vectors available online, which are trained, for example, with the text from Wikipedia, such as Fasttext (Mikolov, Grave, Bojanowski, Puhrsch, & Joulin, 2017) or GloVe (Nenkova & McKeown, 2012). Then the text $(w_{i_1},\ldots,w_{i_L})$ of length $L$ is converted to a $L \times d$ matrix $D = \{v_{i_l}\}_{l=1}^{L}$. The matrix will be fed to the following network.

In this article, we built our own encoding model, since the words used in the textual descriptions are quite unusual. There are many abbreviations for brands and common words that are unlikely to be well represented by word vectors trained by data from Wikipedia and other public sources. Therefore, we just use embedding layer to convert the word into vector and the values of vectors are the parameters that will be trained by the data.

## *Convolutional neural networks*

Convolutional neural network (CNN) is widely used in image analysis. It greatly promotes the application and usefulness of neural networks. CNN is motivated by the local connectivity of visual neurons that are only stimulated by a restricted visual field. Figure 4 illustrates the structure of a typical CNN. CNNs usually are composed of several convolutional layers, max-pooling layers, and fully connected layers (which we explain below).

Unlike nodes in a typical neural network model, which use all the outputs from prior layer as inputs, CNN nodes are only connected to a subset of nodes in the immediately previous layer. A node in a convolutional layer generates new features based on values in a small sliding window on the feature matrix generated from the prior layer. This design will effectively reduce the number of parameters and makes it easy to capture local information. A pooling layer reduces the dimension of the prior layer. For example, the node in max-pooling layer will output the maximum value of its inputs, usually a small square region on the output of prior layer. A convolutional window can be understood as a detector sliding on the output, focusing on a certain pattern in the data, such as a human face, and outputs the strength of that pattern. A max-pooling layer retains the strongest

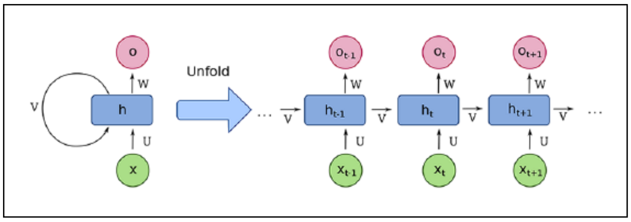

**Figure 5.** The example of RNN. Reprint from https://en.wikipedia.org/wiki/Recurrent_neural_network. For each step, the network receives input, updates the current state based on previous state, and then generates the output. The network can be unfolded as the long connected network.

value of signal in a local region, since a small value of strength implies that there is no significant presence of pattern of interest.

A fully connected layer is one where all the neurons in it have connections to all of the neurons in the previous layer, such as the hidden layer in Figure 3.

For simplicity, we will only introduce the one-dimensional convolution layer that is very useful for text classification. Denote the converted word vectors as $(v_1, v_2, \ldots, v_L)$, each neuron will generate new feature by

$$c_{ki} = \sigma((v_i, \ldots, v_{i+h-1})\beta_k + \alpha_k))$$

where $\beta_k \in \mathbb{R}^{hd\times 1}$ and $(v_i, \ldots, v_{i+h-1})$ form a new $hd$-dimensional row vector by joining the $h$ word vectors in the parentheses. $c_{ki}$ is generated by the neuron with the window that covers $h$ word vectors from $v_i$ to $v_{i+h-1}$. The new channel of feature is formed by concatenation of the results from the neuron when the window slides from the top of the matrix to the bottom, $c_k = (c_{k1}, c_{k(L-h+1)})$. Generally, we will pad the word vectors with some zero vectors to make the output of convolutional layer that has the same length as the input feature. Therefore, the output feature is of dimension $L \times K$.

## *Recurrent neural network*

Recurrent neural network (RNN) is a popular model designed for processing sequence data, such as time series or text data. It is very useful in natural language processing. Figure 5 illustrates the structure of a typical RNN. $h_t$ is hidden state at the $t$th step, $x_t$ is the input, and $o_t$ is the output based on the previous inputs. $V$, $U$, and $W$ are the parameters used to transform $h_t$, $x_t$, $o_t$, respectively. The network maintains hidden state, and when new input comes, a hidden state is updated according to the parameter and structure of the network, for example, $h_t = \sigma(Ux_t + Vh_{t-1})$. The output of the current step is based on the hidden state, $o_t = \sigma(Wh_t)$.[4] The final hidden state contains the information of the whole input, in our case, which is the item description. Then final hidden state will be fed to the layers like fully connected layer for classification. There are some famous variants of RNN, such as LSTM (Gers, Schmidhuber, & Cummins, 1999) and GRU (Chung, Gulcehre, Cho, & Bengio, 2014).

## *Multiple classes classification*

From the introduction above, we can find that neural network method is qualified for multi-class classification. However, the classical SVM, boosting, and tree-based classifiers can only be used for binary classification. Therefore, we need to extend these methods to multi-class classification. Generally, there are multiple ways of extension.

First, the most common method is the "one-versus-rest" strategy. Generally, if there are $q$ classes in the data, $q$ models are trained for each class, generating the confidence level whether the instance belong to the corresponding class or not. The class with the largest confidence level will be reported as output. Another strategy is "one-versus-one" approach. For each pair of classes, we train a binary classifier to make classification between the two classes. Therefore, $q(q-1)/2$ classifiers will be trained. Then the output is determined by the votes of these classifier. If the number of classes is very large, then a one-versus-one strategy will consume lots of time and memory space during training, so a one-versus-rest strategy is usually used in practice.

Another method worth noticing is the hierarchical classifier. It imposes a hierarchical structure, such as a tree, on the classes. Each parent node will have several child nodes, and a classifier is trained for each parent node to make classification. This strategy is very useful in pack labeling, since the data have a hierarchical structure in nature. Generally, the classes of product have three types of granular levels: super groups, categories, and brands. Super group may contain the largest groups like *food & drink*, and *clothing*. Under the super group, there are several categories, such as *coffee, soft drink* under *food & drink*. There are many brands under each categories. This kind of structure is very suitable for hierarchical classifier. At the root node, we train a multi-class classifier with one-versus-rest strategy to make classifications among the super groups. At each super group node, the classifier is trained to classify the categories under the super group, and then in turn for the category nodes.

## Numerical studies

In this section, we introduce the data, along with the tools and settings used in the analyses discussed in this article.

In these numerical studies, two data sets are used. Both data sets are small, being extracts from larger sets. This was to allow faster computing and, therefore, comparison of methods, before scaling to larger data sets. The first dataset is collected by the Nielsen company in an Asian country. It contains around 24,000 records of packs and 24 super groups (a high-level grouping of like products), 248 modules (a lower level grouping of like products), and 649 brands. The second dataset is from a European country. It contains 74,000 records of packs, being in 24 super groups, around 300 categories, and 5,000 brands. The item description is used for covariate variables. In both datasets, the descriptions are quite short. More than 90% of texts are less than 12 words. The 5,000 most frequent words are selected in the vocabulary in both datasets and other words are removed when training. To evaluate the model, we divide the full datasets into training and test datasets with $20\%$ of the data (by lines) assigned to be the test data. Figures 6 and 7 show the distribution of super groups in training data in the two datasets. We can see that both datasets are highly unbalanced. To tackle this issue, we remove some duplicate cases in some dominant labels and oversample cases of rare labels to make the distribution of labels more balanced.

Different e-commerce companies will likely use different abbreviations in their item descriptions, along with sometimes different levels of details and order of elements inside the descriptions. In addition to these differences, the two data sets used in this pilot were in different languages. As Nielsen is a global company, the ideal is to develop as standard a process as possible so that it can be scaled (operationally deployed to different country markets) more easily. One of the goals of the pilot was to develop a robust methodology that would allow this explaining the interest in these diverse datasets.

We now turn to the test accuracy from the simulations. In the simulations, we have used the SVM method, AdaBoost (random forest in the "*sklearn*" package), *XGBoost* (an efficient gradient

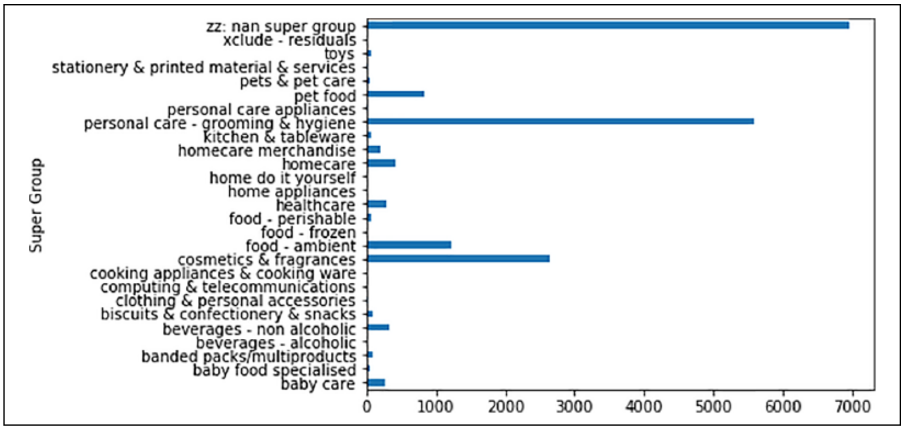

**Figure 6.** The distribution of super groups of training data of first dataset.

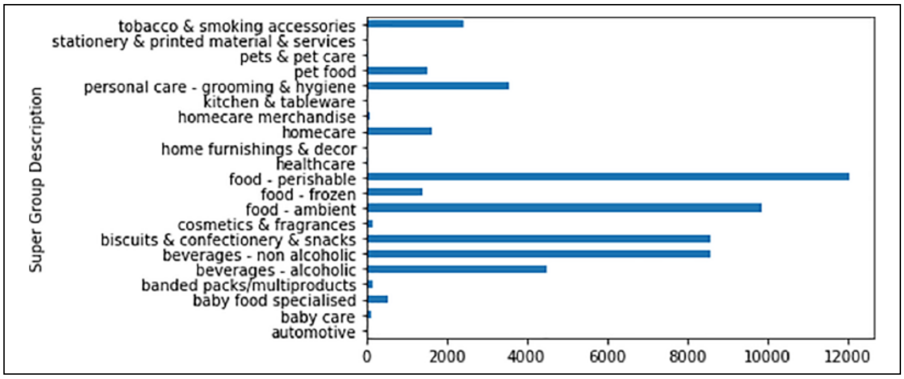

**Figure 7.** The distribution of super groups of training data of second dataset.

boosting library), CNN, and RNN (the last two implemented by *Keras* (Chollet, 2015)). The kernel used for SVM model is a linear kernel that showed the best performance in cross-validation that is a method for selecting hyper-parameters. The method splits the original data into training and validation sets. The model is trained on the training data and we evaluate the performance on the validation data. We can split the data for several times to achieve a stable evaluation by its performance. In k-fold cross-validation, the original data are split into k equal-sized subsamples. Each sample is used as validation and remaining k-1 subsamples are used for training. For the SVM and *AdaBoost* methods, we used three-fold cross-validation to tune the parameters. *AdaBoost* uses a decision tree method as the base estimator and the random forest uses a bagging method (implemented in the *sklearn* package) with *AdaBoost* as its base estimator. Hier-CNN stands for the hierarchical classification with CNN method.

From Table 1, we find that the classification accuracy for brands is higher than that of super group or module. This phenomenon is caused by the distinct pattern of data. Since most of the item descriptions already contain the name of the brand, it makes brand prediction much easier. Generally, the SVM model achieves the best accuracy in all types of granular levels. Since the SVM model deals with the appearance of words directly, it may help classification in this dataset.

**Table 1.** Prediction accuracy in the first dataset.

| Method | Super group | Module | Brand |
|---|---|---|---|
| SVM | 85.1% | 79.7% | 96.1% |
| Decision tree | 78.4% | 71.4% | 83.1% |
| AdaBoost | 78.8% | 73.2% | 84.6% |
| Random forest | 81.1% | 73.5% | 84.8% |
| CNN | 85.5% | 78.1% | 95.5% |
| Hier-CNN | 86.9% | 76.9% | 65.6% |
| RNN | 84.7% | 77.5% | 95.6% |

SVM: support vector machine; CNN: convolutional neural network; RNN: recurrent neural network.

**Table 2.** Prediction accuracy in the second dataset.

| Method | Super group | Category | Brand |
|---|---|---|---|
| SVM | 97.9% | 96.6% | NA |
| Decision tree | 96.8% | 94%% | 88.5% |
| AdaBoost | 96.8% | 95.3% | NA |
| Random forest | 97.4% | 95.4% | NA |
| CNN | 98.2% | 96.1% | 88.1% |
| Hier-CNN | 98.1% | 95.4% | 83.7% |
| RNN | 98.3% | 96.5% | 83.8% |

SVM: support vector machine; CNN: convolutional neural network; RNN: recurrent neural network.

Moreover, the neural network method also works very well. CNN and RNN achieve similar results. However, Hier-CNN does not work well because its brand level output depends on the output of module and super group level, but the prediction results on these levels are not good as brand. Although the tree-based method does not have a good performance in this example, the boosting and bagging methods still improve the prediction accuracy.

As for the second dataset, from Table 2, we see that the SVM and neural network methods achieve very good results. The CNN method achieves the best result in brand-level classification. Hier-CNN does not perform as well as CNN. The possible reason is that there are many brands and categories belonging to multiple parents. It generates more classes than the data actually have. Although methods like SVM, *AdaBoost*, and random forest can achieve similar performance as the neural network method, they consume lots of time and resources, since they need to train the same number of classifiers as classes. For the second example, the decision tree needs 2.62 CPU seconds (on the computer used in this study) to train on the data of super group and 8.3 s for 300 categories, *AdaBoost* needs 55.1 s for super group and 557.4 s for categories, and the SVM method needs 38.2 and 92.2 s, respectively. The time needed to train model is in pace with the number of labels.

Since the two datasets we present in this article are quite small. The efficiency of training is not our concern. In practice, the volume of data might be several Gigabytes. It poses great challenge to hardware and software, and the efficiency of training is quite important. In this project, we also used dataset of several Gigabytes from a major e-tailer. The dataset contains around 20,000,000 textual descriptions of products. They are classified into more than ten departments, around 400 subgroups and more than 3,000 different labels. It took several days to train the

**Table 3.** The number of correct and wrong predictions of brands grouped by super group in the first example based on CNN method.

| Super group | Correct | Wrong |
|---|---|---|
| Automotive | 1 | 0 |
| Baby care | 54 | 2 |
| Baby food specialized | 6 | 1 |
| Banded packs/multi-products | 23 | 9 |
| Beverages—alcoholic | 1 | 0 |
| Beverages—non-alcoholic | 78 | 5 |
| Biscuits & confectionery & snacks | 27 | 1 |
| Clothing & personal accessories | 4 | 0 |
| Computing & telecommunications | 4 | 0 |
| Cosmetics & fragrances | 655 | 18 |
| Food—ambient | 275 | 17 |
| Food—frozen | 1 | 0 |
| Food—perishable | 10 | 0 |
| Health care | 69 | 5 |
| Home furnishings & decor | 0 | 1 |
| Homecare | 81 | 8 |
| Homecare merchandise | 37 | 6 |
| Kitchen & tableware | 22 | 3 |
| Personal care—grooming & hygiene | 1,357 | 65 |
| Personal care appliances | 4 | 1 |
| Pet food | 223 | 23 |
| Pets & pet care | 6 | 5 |
| Stationery & printed material & services | 7 | 1 |
| Toys | 11 | 2 |
| Nan super group | 1,645 | 48 |

CNN model on a Tesla 40K GPU. If we use ordinary SVM method or tree classifier, we need train more than 3,000 one-versus-rest classifiers or 9,000,000 one-versus-one classifiers. It is an unrealistic task. Moreover, electronic data are common to be streamlined. The method should be easily adapted to online learning scheme. In practice, we should carefully select the machine learning method according to the nature of data. We will have more discussion at "Extension topics" below.

Table 3 shows the summary of the number of correct predictions of brands grouped by the super groups in the first example by using CNN method. Since many textual descriptions already contain the name of brands, it is very easy for a machine learning method to capture that feature. From the table, we can find that even some minor brands have good prediction results, being assigned to their correct super group.

Table 4 shows the summary of the number of correct predictions of brands grouped by super groups in the second example by using CNN method. From the table, we can find that the dataset is highly unbalanced and for some rare classes such as "*automotive*," the prediction accuracy is not good, since the number of observations for training is too small. However, brands in super group "*baby food specialized*" achieve good results. Moreover, the super group of "*tobacoo & smoking accessories*" achieves very good results. Its salient features are different from other groups that

**Table 4.** The number of correct and wrong predictions of brands grouped by super group in the second example based on CNN method.

| Super group | Correct | Wrong |
|---|---|---|
| Automotive | 0 | 13 |
| Baby care | 38 | 10 |
| Baby food specialized | 183 | 27 |
| Banded packs/multi-products | 46 | 13 |
| Beverages—alcoholic | 1,602 | 315 |
| Beverages—non-alcoholic | 3,337 | 387 |
| Biscuits & confectionery & snacks | 3,259 | 387 |
| Cosmetics & fragrances | 40 | 22 |
| Food—ambient | 3,805 | 407 |
| Food—frozen | 452 | 150 |
| Food—perishable | 4,368 | 788 |
| Garden & flora | 0 | 1 |
| Health care | 6 | 10 |
| Home do it yourself | 0 | 1 |
| Home furnishings & decor | 0 | 1 |
| Homecare | 568 | 112 |
| Homecare merchandise | 35 | 8 |
| Kitchen & tableware | 5 | 0 |
| Personal care—grooming & hygiene | 1,279 | 257 |
| Personal care appliances | 0 | 1 |
| Pet food | 589 | 9 |
| Pets & pet care | 18 | 8 |
| Stationery & printed material & services | 2 | 8 |
| Tobacco & smoking accessories | 1,058 | 12 |

may account for the high prediction accuracy. We can notice that there are many wrong classifications in super groups of food and beverage.

Figure 8 shows the heat map of the distribution of super groups of the wrongly classified products of the first dataset. As can be seen, there are many products, especially *baby food specialized*, wrongly classified into brands in super group "*personal care—grooming & hygiene.*" The possible reason is that in the training dataset, there are lots of products in "*personal care—grooming & hygiene*," so the classifier tends to classify products into this super group. Moreover, we can find that there are many products wrongly classified as "*nan super group.*" Since "*nan super group*" is not well-defined and it may contain some products with similar features as "*food—perishable*" or "*personal care* appliances," the trained network may assign some products to this super group wrongly.

Compared with the result of Figure 9, we can find that many wrongly classified product are even wrong in their super group prediction. Although we ignore *nan super group*, there are many cases in *baby food specialized* and *kitchen & tableware* are wrongly classified to other super group. This might be caused by the similarity between the descriptions of the two super groups. For example, the descriptions of products in *baby food specialized* may contain many words like "*care*," "*healthy*," or "*organic.*" These words are common in the descriptions of *personal care—grooming & hygiene*, so it will be difficult for the classifier to make the discrimination. To tackle this problem, we can use hierarchical classification method. It will be easier to discriminate among super

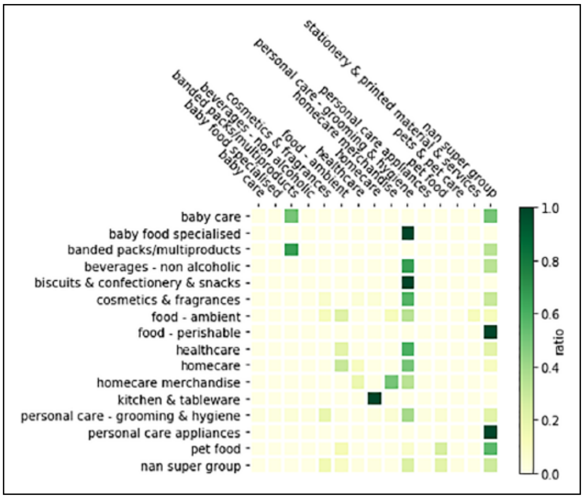

**Figure 8.** The heat map of distribution of wrongly classified products of the first dataset.

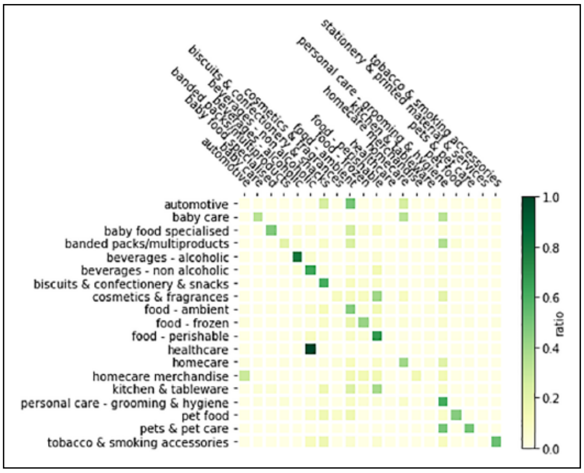

**Figure 9.** The heat map of distribution of wrongly classified products of the second dataset. Boxes in the diagonal running from upper left to lower right represent correct classification.

groups rather than specific labels. Moreover, if we have other information like price or information about the buyer. It will help in classification.

Figure 9 shows the heat map of the distribution of super groups of the wrongly classified products of the second dataset. From this figure, for the most wrongly classified products, they are assigned into correct super group (i.e., the classifier gets the product classification wrong, but correctly assigns the right super category). It is worth noting that there are many products in "*health care*," which are classified to "*beverages—non-alcoholic*." That may be caused by some non-alcoholic beverages being described as a healthy drink or a sports drink.

## Extension topics

### What kind of method is suitable for streaming data?

It is quite common that the available training data are generated continuously such as by consumer purchases. These new data may contain previously unseen item descriptions. If new data come, it is time and resource consuming if we need to train the model from the scratch. In that case, using a neural network is a good option as it can easily be adapted to handle the new data. The structure of the network will need to be adjusted at the output layer and the new training data injected—it is likely that human intervention will be needed to label the new cases. This is the first way human experience can be integrated into the production process.

In practice, a neural network is trained with via SGD, which is different compared to ordinary gradient descent that uses the whole data to update the parameters. SGD only uses subsamples of data, of size usually 128 or 256 observations, to update the parameters. This approach facilitates online training for a neural network.

### How to integrate human knowledge in product classification?

This pilot demonstrated that machine learning methods achieve very good results in classification of packs and their cost is very low compared with manual labeling. However, human knowledge is very accurate in classifying some rare observations. In general, a machine learning method is good at processing simple and common data at high speed and low cost, while humans are good at processing complex and unusual data but at slow speed and high cost. If we can combine the two, such as using humans to generate additional training data, so that the machine learning algorithm can learn the "new" pattern, then it can improve the performance of machine learning method and reduce the cost.

A second way of integrating human ability into a production process is to assign those observations with low confidence from machine learning method to humans. For example, the neural network assigns an estimated probability to each label, indicating its confidence in the prediction it is making. If the probability mass concentrates on one label, then it is very likely that the observations belong to that label. If the probability mass is highly distributed, then the given label may be wrong. We can assign the observation for a human to decide. If the prediction is wrong, we can add it to training data for further training.

## Conclusion and discussion

The decision as to which method to use for a particular data classification task is not a straightforward one. Even for the two datasets used in this article, we can find that it is hard to find a method achieving the best classification results across all granular levels and both data sets. In practice, the method we use depends on the data available and the task. For example, the authors have subsequently found that SVMs did not scale well with increased data sets, larger than the ones used in this pilot study, due to the increase in the number of categories to be classified to as well as the number of item descriptions to be classified. As we have shown in this article, different models will have to be trialed and evaluated, but good accuracy rates can be obtained. This evaluation will need to be ongoing, as the data sets change in content and quality with time and perhaps integration of additional new sources. Strict quality controls and monitoring are the keys to these automated systems, with regular validations and statistical process controls being important to ensure that high quality and reliability of the automated processes are maintained. Such rigor is the price of these “big data” sources, particularly in the cases where there are additional processing steps (such as optical character recognition) that will require ongoing validation to ensure significant biases are not being introduced. However, these are small prices to pay for the additional insights, these data sources can provide into the market place and consumer behavior, allowing marketing research companies to help their clients make faster and better decisions.

Generally speaking, when the dataset is small, some simple methods, such as decision tree, linear model, or SVM with linear kernel are suitable. These can capture simple relationships between covariate and response. When we have a large volume of data, methods like neural network, SVM with a radial basis function kernel, or a boosting method can be used. However, the SVM and *XGBoost* methods are designed for binary classification, so for multiple classes classification, the two methods train the same number of classifiers as classes. When the number of classes is large, which is quite typical in practice, we recommend that it is better to use a neural network method, such as CNN or RNN. Moreover, the powerful deep learning libraries like *PyTorch* (Paszke et al., 2017) or TensorFlow (Abadi et al., 2016) make the neural network method easy and efficient to implement.

Furthermore, an ensemble method can be used to improve the accuracy (Chen & Guestrin, 2016), where several machine learning methods are “combined” together to improve prediction. A simple example could be a “majority vote” system, where several algorithms make predictions (“vote”) with the majority decision (say, above a minimum threshold of votes) being selected. This method proved to be quite useful in many data mining competitions. However, this method will consume lots of time and resource during training. Therefore, in practice, we need to consider whether the improvement from ensembling is worth the effort. Building off the pilot described in this article, the authors have implemented ensembles for data sources, such as scanned invoices, increased volumes of item descriptions from major e-commerce stores, and even associate retention probabilities, finding the additional effort worth the lift in prediction accuracy. In the use case described in this article, we built an ensemble from an SVM, a CNN, and a proprietary technique, trained on a larger dataset and which was later deployed via *Docker* containers. Where computer-intensive training is required, we have placed this training outside the production process and recommended instead regular “injections” of the updated models into the production process. By frequently updating the models based on the current data, we intend to avoid trend breaks (“steps”) and maintain high classification rates.

To conclude, the machine learning techniques reviewed in this article are revolutionizing market research audits, allowing not only larger data sources to be processed (and hence having the potential to increase the granularity and accuracy of the reports) but also allow such data to be processed more quickly into information actionable by client companies. Naturally, care still needs to be taken that these data sources are representative, or can be rebalanced so that the resulting insights are representative of the desired populations, but the combination of new data sources and processing techniques is leading market research into a new era. Care will need to be taken to avoid audit companies becoming commodified, being seen as simply providers of data to be ingested into decision support platforms, and instead market research increases its role as a partner in understanding the marketplace and people’s behavior, delivering actionable, grounded in fact, timely, and insightful views. The industry stands upon an exciting threshold! We hope that this review of available techniques and sample results will encourage others to explore the new data sources technology is opening up for the industry (even the so-called “exhaust” data sources that are side products generated by another processes) and techniques, with their promise to “level-up” our capabilities.

## Acknowledgements

We would like to express our special thanks of gratitude to Nielsen company and Statistics Department of National University of Singapore for providing this valuable opportunity to conduct research in this interesting topic. We also want to appreciate Mr Malykhin at Nielsen in Russia, Mr Mian at Nielsen at Turkey for their sharing of idea and data, and Professor Xia at NUS for his surveillance. Moreover, we thank the peer reviewers for their helpful comments that improved this article.

## Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

## Notes

1. In passing, we note that the methods are being actively applied by Nielsen to other data sources, such as receipt images (following image processing and text extraction steps).
2. For more details see, for example, https://en.wikipedia.org/wiki/One-hot
3. Note: words do not necessarily have to be actual "English" words, but could be groups of consecutive characters.
4. For more details, see http://colah.github.io/posts/2015-08-Understanding-LSTMs/

## References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., & Kudlur, M. (2016). November. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX conference on operating systems design and implementation* (OSDI'16, Vol. 16, pp. 265–283). Berkeley, CA: USENIX Association.

Banks, T., & Budding, E. (2018). Exoplanets and university-industry collaboration. *Southern Stars*, *57*, 10–14.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In Y. Lechevallier & G. Saporta (Eds.), *Proceedings of COMPSTAT'2010* (pp. 177–186). Heidelberg: Physica-Verlag HD.

Breiman, L. (2017). *Classification and regression trees*. Abingdon, England: Routledge.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785–794). New York, NY: ACM.

Chollet, F. (2015). Keras, GitHub. Retrieved from https://github.com/fchollet/keras

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv*. Retrieved from https://arxiv.org/abs/1412.3555

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55, pp. 119–139.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, *29*, 1189–1232.

Friedman, J. H., Hastie, T., & Tibshirani, R. (2001). *The elements of statistical learning* (Vol. 1, No. 10). New York, NY: Springer Series in Statistics.

Gers, F. A., Schmidhuber, J., & Cummins, F. (1999). *Learning to forget: Continual prediction with LSTM*. Paper presented at Ninth International Conference on Artificial Neural Networks (ICANN 99). Stevenage, England: IET.

Han, J., Kamber, M., & Pei, J. (2012). *Data mining: Concepts and techniques* (3rd ed.). Amsterdam, The Netherlands: Elsevier.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning: With applications in R*. New York, NY: Springer.

Jones, E., Oliphant, T., & Peterson, P. (2014). SciPy: Open source scientific tools for Python. Retrieved from https://www.scipy.org/

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., & Jackel, L. D. (1990). Handwritten digit recognition with a back-propagation network. In D. S. Touretzky (Ed.), *Advances in neural information processing systems* (pp. 396–404). San Francisco, CA: Morgan Kaufmann Publishers.

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., & Joulin, A. (2017). Advances in pre-training distributed word representations. *arXiv*. Retrieved from https://arxiv.org/abs/*1712*.09405

Nenkova, A., & McKeown, K. (2012). A survey of text summarization techniques. In C. Aggarwal & C Zhai (Eds.), *Mining text data* (pp. 43–76). Boston, MA: Springer.

Olsen, K., Lin, X. Y., & Banks, T. (2017). Evaluating data quality in reports of sales in a retail establishment survey. *International Journal of Market Research*, *59*, 301–320.

Paszke, A., Gross, S., Chintala, S., & Chanan, G. (2017). *Pytorch: Tensors and dynamic neural networks in python with strong GPU acceleration*. Retrieved from https://github.com/pytorch/pytorch

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management 24*, 513–523.

Seah, M. S., Tung, W. L., & Banks, T. (2015). A Novel Discrete Particle Swarm Optimization approach to large-scale survey planning. In Z. Xiao (Ed.), *11th international conference on natural computation (ICNC)* (p. 261). New York, NY: IEEE.

Tung, Q. L., Zhao, J. Y., & Banks, T. (2014). Eye in the sky: Rural-urban classification using remote sensing imagery. In *Asia—Pacific 2014—Celebrating Asian creativity* (ESOMAR Publication Series, *Vol. S364*, p. 1). Jakarta, Indonesia: ESOMAR.

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., . . . Stoica, I. (2011, July 19). *Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing* (Technical Report No. UCB/EECS-2011–82). Berkeley: EECS Department, University of California. Retrieved from http://www2.eecs.berkeley.edu/Pubs/TechRpts/2011/EECS-2011-82.pdf