AITA
ARTIFICIAL INTELLIGENCE THEORY AND APPLICATIONS

■ RESEARCH ARTICLE

Text2Price: Deep Learning for Price Prediction

Beyzanur Saraçlar[a†], **Birol Kuyumcu**[a], **Selman Delil**[a]

[a] Sefamerve R&D, İstanbul, Turkey
[†] beyzanur.saraclar@sefamerve.com, corresponding author

RECEIVED JULY 8, 2022
ACCEPTED AUGUST 18, 2022

CITATION Saraçlar, B., Kuyumcu, B., & Delil. S. (2022). Text2Price: Deep Learning for Price Prediction. *Artificial Intelligence Theory and Applications, 2(*2), 28-38.

Abstract

There are many methods and strategies that can be used when determining the selling price of a product in the online marketplace. Correct pricing of a product is an important factor affecting the overall success and profitability of the e-commerce business. Considering all these issues, the need to develop tools that will help the seller in the process of deciding the price of a product arises. In this paper, we designed a model that predicts the price of a product using its title, supplier, category and description information. Our technique is based on using only a single text data for price estimation. For this purpose, we concatenate product information in a string while preserving their attribute information. The task of preprocessing various feature types becomes simple and quick using this method. The main contribution of our approach is designing a model that is applicable for various prediction tasks without task-oriented implementation. To build the prediction model, we used deep learning methods which are based on RNN and CNN and we compared their performances. According to the results, LSTM-based models have achieved more accurate predictions with 6.1646 mean absolute percentage error (MAPE). Also, CNN-based models had 3x times faster running time advantage while having a minor increase in MAPE with 7.1387 compared to LSTM-based models.

Keywords: price prediction, deep learning, LSTM, CNN

1. Introduction

In the last few years, the tendency of people to buy products online through e-commerce sites has been increasing. Considering that there are many online marketplaces, it becomes a major problem for the seller to determine the price of alternatives to the same type of products. While determining the price of the product, the seller might wish to provide a good profit margin without reducing the value of the product. On the other hand, in order to compete in the online market, the buyer must be offered the best available offer. Also, the characteristics of a product such as category, brand, color or visual features have an impact on determining the price of a product. Considering all these, it requires expertise to know in detail which features determine the price of the product and which of them affect more or less on making a price decision. Accordingly, the need to automatically predict the price of a product to assist the seller in the pricing process has emerged.

Artificial Intelligence Theory and Applications, ISSN: 2757-9778. ISBN: 978-605-69730-2-4 © 2022 University of Bakırçay

Several studies have been conducted on price estimation of different items using diverse prediction methods. Fathalla et al. [1] focused on price prediction of second-hand items by applying a deep neural network based on images and textual descriptions of the items. In their study, one-hot encoding, Bidirectional LSTM (Bi-LSTM) and Convolutional Neural Network (CNN) models were combined to process product type, textual features and visual features respectively. Additionally, the authors used time series to forecast the price range of a second-hand item utilizing machine learning (Linear Regression), deep learning (LSTM) and statistical methods (SARIMA). Also, the quality scores of the items were calculated using predicted price and price range information. Focusing on the visual features in addition to textual and categorical features increased the model accuracy for price prediction. However, different kinds of features require separate data processing for each model. Carta et al. [2] proposed the software tool Price Probe for estimating the future price trend of e-commerce products. The model behind the tool is based on the time series forecasting model ARIMA. The authors used a dataset crawled from Amazon which consists of product features, product categories and price information over time. They also extracted information from Google Trends. These exogenous features have improved the prediction performance of the prediction model. However, this approach is not applicable to all products since Google Trends information is not available for most of the products. Tseng et al. [3] used sentiment analysis to forecast the price of e-commerce products based on ARIMA model. They focused on price prediction of house property and electronic products. They analyzed and extracted news content from Baidu related to the products. The results show that the price of products is affected by significant news events within the concept of sentiment. Kalaiselvi et al. [4] predicted the price of smartphones based on some features such as the specifications, name, product features, and reviews using neural networks. Also, they performed sentiment analysis of the products from the product reviews and combined sentiment analysis with the neural network model. These prediction models [3, 4] rely on the online product reviews feature which is not the main feature of the product such as category or brand of the product. Also, the products need to have enough reviews for these studies.

To the best of our knowledge, none of the existing studies utilize only one textual feature of the product for product price estimation. Different kind of features are extracted and processed for the previous models. Furthermore, most of the studies have focused on the price estimation of certain categories of products. The models are not suitable to predict the price of several product types simultaneously. Additionally, many studies used conventional methods such as time series models (ARIMA, SARIMA) for price prediction.

We considered this issue as a prediction task within the area of deep learning. Contrary to traditional machine learning methods, the use of deep learning architectures can extract information automatically without applying feature engineering such as creating and selecting features from raw text data. However, different types of product features which might consist of numerical, categorical or textual data are required to be handled before giving input to these models. In order to tackle this problem, we concatenated different types of features in a string keeping their attribute name information.

In this study, we applied RNN more specifically LSTM variations, and CNN architectures to estimate the price of items that are sold on Sefamerve.com which is the top e-commerce retail in Turkey for modest clothing. Products have different types of features such as text and numerical data. To cope with different types of data, we combined the features of the item in one string which is a single input of the models. The main contributions of this work are as follows:

- We proposed a price prediction model that concatenates product features in a text with the attribute information.
- The performance of RNN and CNN models are compared in a price prediction task using only a textual feature.
- For the CNN model, we designed Wawnet architecture which consists of a network block called waw-block and a regression layer.
- The models can predict the prices of different types of items from clothing to home and lifestyle categories.

The rest of this paper is organized as follows. Section 2 describes the research methodology and gives information about the price prediction models. Section 3 describes the experimental setup and Section 4 presents the results. Finally, the conclusions of the study are presented in Section 4.

2. Materials and Methods

2.1. Dataset

We collected data from Sefamerve.com which is an e-commerce website offering various categories to customers from clothing to home and lifestyle. The dataset consists of item title, item description, supplier id and category id of an item. The dataset consists of 26,070 observations, where the item prices range from 3.99 TRY to 699.9 TRY. To evaluate the price prediction models, the dataset was split into training and testing sets with a ratio of 80% and 20%, respectively.

2.2. Data Preprocessing

We proposed a model that predicts the prices using only a single textual feature for each product. We used title, supplier id, categorical id and description information of the product. We converted categorical information to the text data by aligning the attribute and its value together into a text using underscore between them. We concatenate all text values into one text with their attribute information. Figure 1 shows the method of text data generation from the features. With this technique, we aim for the model to extract the features from the text during training. In this way, we employ the prediction models without coping with preprocessing to make them suitable for the models.

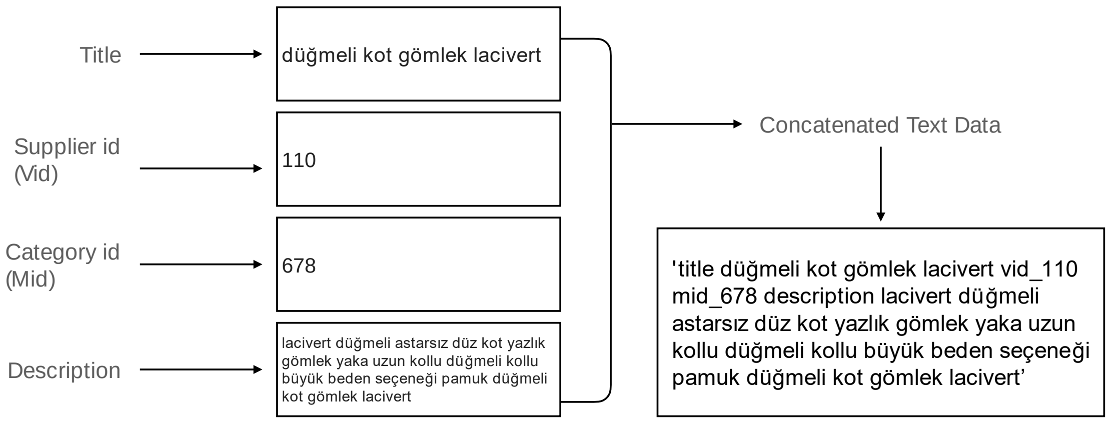


Figure 1. Text data generation example.

2.3. Vectorization of the Text Data

To implement Machine Learning models, text data needs to be encoded by converting it to a numeric value or vector. Textual data needs to be prepared for encoding by pre-processing it. Text processing is the first and essential step for NLP tasks and it affects the network performance. We utilized lower casing and tokenization process techniques. We also removed product code numbers from the product title and description. The input text is tokenized using Keras library's tokenizer function [5].

To vectorize concatenated text data, we used word embedding layer to obtain an efficient representation of text. In word embedding methods, words with similar meanings have a similar encoding. Word vectors are initialized with random weights and weight parameters are learned during training by the neural network model while the model learns the network parameters. The size of the word embedding vectors depends on the size of the dataset. Higher-dimensional embedding obtains a better representation of words with large datasets while embedding with lower dimension is sufficient for small datasets. We chose the embedding size 256 for our dataset.

2.4. RNN Models

In recurrent networks, the connection between the terms is represented by a chain structure that current output depends on long-distance features. The input layer of the RNN represents features at time t. Contrary to the feedforward network which considers only current input and has no notion of order in time, RNNs combine the hidden state and current input to preserve the sequential information. Mathematically, hidden and output layers are implemented as below:

$$h_{(t)} = f(U_{x(t)} + \left(W_{h(t-1)}\right) \quad (1)$$

$$y_{(t)} = g(V_{h(t)}) \quad (2)$$

$h_{(t)}$ represents the hidden state at time t and can be considered as "memory" of the network. It is calculated by adding the input $x_{(t)}$ at the same time step with $h_{(t)}$ to the previous hidden state $h_{(t-1)}$. U, W and V are the weight matrices which are updated in the backward pass for input, hidden state and output respectively. $f_{(z)}$ squashes the sum of weighted input and hidden state, it can be either sigmoid or tanh function. $g_{(z)}$ is the softmax function that gives the output probabilities between 0-1.

2.4.1. LSTM Network

LSTM [6] which is a variant of RNN is one of the most effective models for time sequence modeling and context dependencies tasks because it encodes the long-term dependencies with its gate mechanism in the hidden layer. A memory cell of LSTM formula is computed as follows:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \quad 3$$

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right) \quad 4$$

$$c_t = f_t c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad 5$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \quad 6$$

$$h_t = o_t \tanh(c_t) \quad 7$$

i_t, c_t and o_t are the input gate, forget gate, cell state vector and output gate respectively. These gates have a similar formula to RNN. According to the (5) equation, the previous cell state c_{t-1} is modulated by the forget gate f_t and input is modulated by input modulation gate to calculate the current cell state c_t. With these gates, LSTM decides how much information should be blocked or transferred based on their own sets of weights.

Bidirectional LSTMs improve the performance of LSTMs by training the network in two ways with forward and backward LSTM [7]. The network combines the past and future information in a hidden state by feeding it with original and reverse-copy of the input sequence timesteps.

In this paper, we used LSTM and bidirectional LSTM to predict prices. To achieve the best results, we performed hyperparameter tuning to obtain the optimal set of parameter values by using Mean Absolute Percentage Error (MAPE) function as an objective function.

2.5. Convolutional Models

RNN has a sequential structure in which the output depends on previous hidden states. This structure makes the computation cost high because of many linear operations. As a solution to this problem, CNN models are performed for NLP tasks. CNNs are mainly used in image recognition problems. However, this method achieved successful results in text data in recent years. CNN has a hierarchical structure based on two main layers which are convolutional and pooling layers. Each convolutional layer extracts different features from the input by sliding the local receptive fields on it. Since the same feature is mapped onto the convolutional layer, as much as the size of local receptive field weights are shared. This considerably reduces the number of parameters that lead to less computation time compared to RNNs [8]. The pooling layer extracts the most important features by reducing the dimension with different pooling techniques such as max pooling or average pooling. The pooling layer also decreases memory usage and the number of parameters because it represents the input in a more compact way [9]. Also, the hierarchical structure of CNN enables parallelization during computation and simultaneous computation of the input. However, in recurrent networks output depends on the previous state which precludes parallelization of the model. Due to less parameters and memory usage advantages, the use of CNN increases in the area of NLP [10].

2.5.1. Wawnet Architecture

In this study, we designed convolutional neural network architecture called Wawnet [11]. The network consists of a network block called waw-block and a regression layer. In this architecture, waw-block encodes the connection information to feed the regression layer. Figure 2 illustrates the architecture of the waw-block.

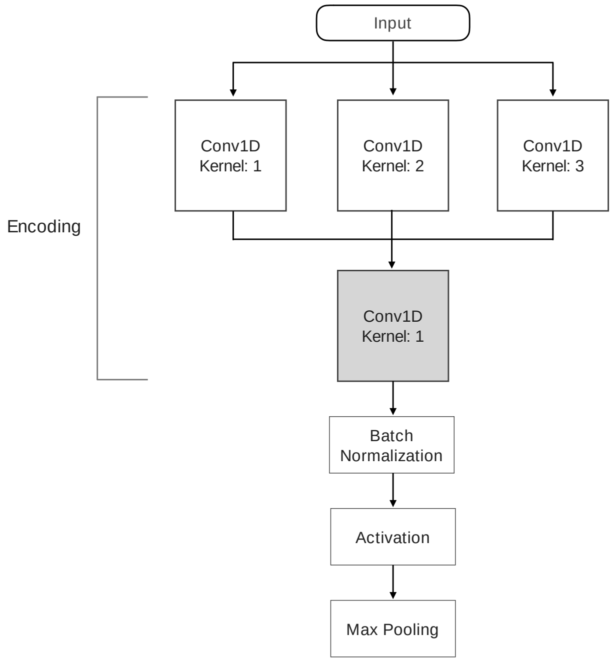


Figure 2. Waw-block architecture.

Each waw block input is passed through the 1, 2 and 3 one-dimensional convolution layers, the outputs of the convolution layers are combined and passed through the 1 one-dimensional convolution layer. In this way, Wawnet extracts the input features with these different sized filters. Then batch normalization and activation layers are applied to accelerate training and prevent the objective function from getting stuck in local minima. Lastly, we applied max pooling on outputs to reduce the dimension by half. The output of the network is the product price itself. Figure 3 shows the Wawnet architecture.

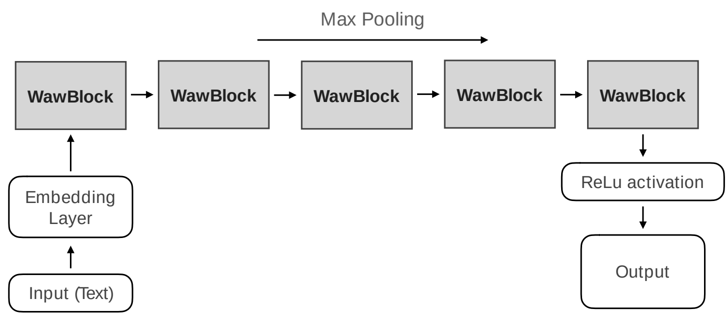


Figure 3. Wawnet architecture.

3. Experimental Setup

3.1. Accuracy Metric

In order to evaluate and compare the performance of the price prediction models, we used a metric namely MAPE. MAPE, which is frequently used to measure the accuracy of predictions in regression and time series models, measures the mean absolute percentage error between the actual prices and predicted prices. It is expressed in Eq. 8.

$$MAPE = \frac{100\%}{N}\sum_{i=1}^{N}\left|\frac{A_i - P_i}{A_i}\right| \quad (8)$$

where N is the number of observations. A_i and P_i denote the actual price and predicted price respectively. In order to express the result as a percentage, the mean absolute error is multiplied by 100%.

3.2. Text to Price Prediction

3.2.1. Baseline

We used Gradient Boosting Regression (GBR) as a baseline. We applied TF-IDF to concatenated text data. We selected the range of word n-gram as (1,3). We performed hyperparameter tuning for learning rate, maximum depth and number of estimators to obtain optimal parameters for GBR. The best performing parameter combination is when the learning rate is 0.5, maximum depth is 6 and number of estimators is 200. In order to perform TF-IDF and construct the models, we utilized Sklearn library [12].

3.2.2. LSTM and Bi-LSTM Model Hyperparameters

For the LSTM model, we used a two-layer network with a hidden size of 256. The activation function of LSTM and output layer was set to tanh and ReLU respectively. The model is optimized with the RMSprop [13] optimizer. For the Bi-LSTM, we constructed a one-layer network with a hidden size of 256. We set the activation function for the output layer to linear activation. RMSprop optimizer was used in the network. In order to prevent overfitting, we used dropout regularization [14] with a rate of 0.5 on both LSTM models. During the training, we employed a learning rate scheduler to adjust the learning rate while monitoring the validation loss metric which is MAPE. The learning rate is reduced by a factor of 0.5 when the validation loss does not improve for 10 epochs. The models are trained with 200 epochs. However, if no improvement is noted for 30 epochs, training is stopped early.

3.2.3. Wawnet Model Hyperparameters

Wawnet has the hyperparameter named depth which decides the number of waw-blocks in the network. When the depth is increased, the network becomes more complex and it improves the learning of the model. However, adding each waw-block increases the time required to train the model and causes overfitting. We experimented with different depth values from 1 to 7 to find the optimal value. As a result of the experiment, we set the depth parameter to 5. We employed 32 filters with 1, 2 and 3 filter sizes in waw-blocks. To optimize the model, Adam [15] optimizer is used. We used ReLU activation function for the output layer. As a learning rate scheduler, we used the same setup and early stop mechanism with the LSTM and Bi-LSTM models for Wawnet.

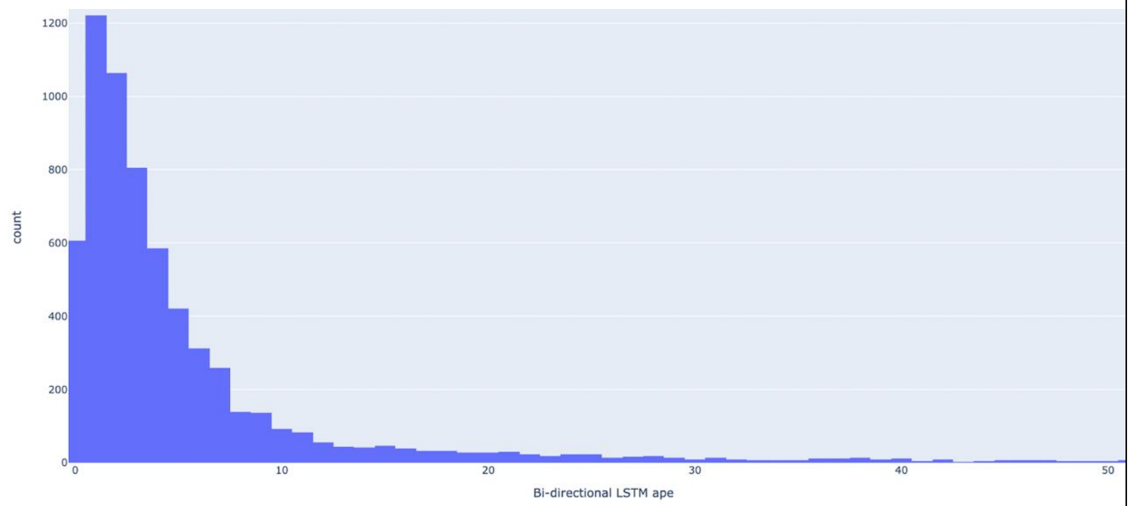
4. Results and Discussion

Table 1 presents the MAPE results on validation data for the Wawnet, LSTM, Bi-LSTM price prediction models and the baseline Gradient Boosting Regression (GBR) model. Wawnet, LSTM and Bi-LSTM outperformed GBR baseline model. The reason for this finding is that deep learning models obtain a better representation of data which results in higher accuracy for the prediction tasks. MAPE results of the deep learning models are close to each other and the two-layer LSTM network with a hidden size of 256 achieved the best result with a MAPE of 6.1646 on the validation data.

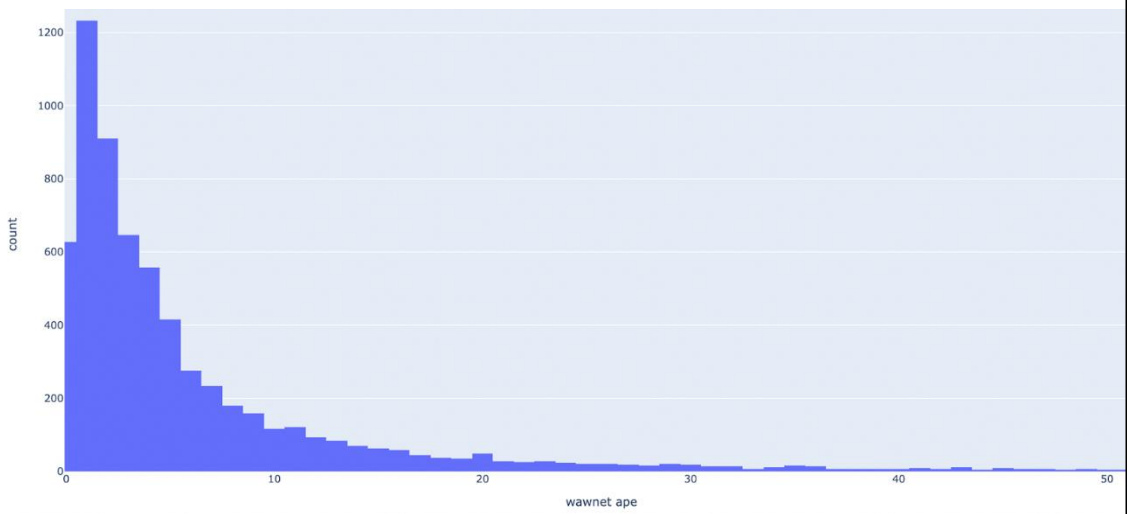
Table 1. Performance results of the models.

Model	Validation MAPE
GBR	12.5741
LSTM	**6.1646**
Bi-LSTM	6.4083
Wawnet (depth = 5)	7.1387

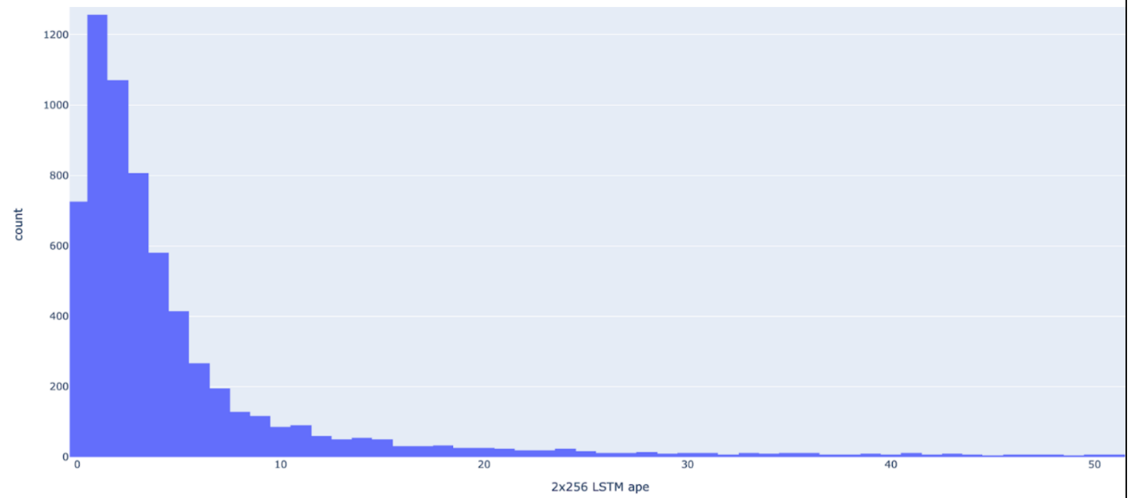
Also, we compared absolute percentage error (APE) of each sample to observe the distribution of price errors. Figure 4 shows the histogram of APE for each model. In order to analyze the distribution of APE, we focused on less than 50% error rate of the graph. Although the models have similar APE distribution, Wawnet APE values are seen in Figure 4 (b) to be spread over a wider area. This shows that LSTM based models have slightly more accurate price predictions compared to Wawnet.



(a)



(b)



(c)

Figure 4. APE distribution of LSTM based and Wawnet models (a-c).

In order to compare the training time of the models, we measured training time during 1 epoch. Table 2 shows the 1 epoch run time of the models.

Table 2. The training time of the models during one epoch.

Model	1 Epoch Run Time	Validation MAPE
Bi-LSTM	21s	6.4083
LSTM	20s	6.1646
Wawnet (depth = 5)	6s	7.1387
Wawnet (depth = 4)	4s	7.7636
Wawnet (depth = 3)	3s	8.3561
Wawnet (depth = 2)	2s	9.4857

There is a considerable run time difference between LSTM models and Wawnet models. Also, the running time of the Wawnet decreases when the depth of the models decreases with minor validation MAPE changes. Although LSTM and Wawnet models have similar accuracy, Wawnet provides a major advantage over LSTMs in terms of running time.

5. Conclusion

In this paper, RNN and Convolutional models are used to predict the price of products sold on Sefamerve.com utilizing only one combined text feature. The main contribution of this study is a method which converts the different types of features which might consist of categorical, numerical and textual data to a single text to make a prediction. In this study, the models are able to predict the price of products in different categories from clothing to home and lifestyle. We applied LSTMs and Wawnet architecture for the price prediction task. We measure the performance of the models with MAPE metrics. Proposed models achieved better performance in this metric compared to the baseline model. Although the performance of the proposed models has similar scores, 2 layered LSTM is the most successful model with 6.1646 MAPE score. However, Wawnet architecture provides an advantage for the training time which is almost 3x times faster than the LSTM while making small concessions to accuracy. For future work, we plan to apply our “text to price” method to different price prediction tasks to forecast item prices.

References

[1] Fathalla, Ahmed, et al. "Deep end-to-end learning for price prediction of second-hand items." Knowledge and Information Systems 62.12 (2020): 4541-4568.

[2] Carta, Salvatore, et al. "Forecasting e-commerce products prices by combining an autoregressive integrated moving average (ARIMA) model and google trends data." Future Internet 11.1 (2018): 5.

[3] Tseng, Kuo-Kun, et al. "Price prediction of e-commerce products through Internet sentiment analysis." Electronic commerce research 18.1 (2018): 65-88.

[4] Kalaiselvi N, Aravind K, Balaguru S, Vijayaragul V (2017) Retail price analytics using backpropogation neural network and sentimental analysis. In: 2017 fourth international conference on signal processing, communication and networking (ICSCN). IEEE, pp 1–6

[5] Chollet, F., & others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras

[6] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

[7] Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural networks, 18(5-6), 602-610.

[8] E. Grave, A. Joulin, and N. Usunier, “Improving neural language models with a continuous cache,” arXiv preprint arXiv:1612.04426, 2016

[9] A. Ge ́ron, Hands-on machine learning with Scikit-Learn and Tensor- Flow: concepts, tools, and techniques to build intelligent systems.” O’Reilly Media, Inc.”, 2017.

[10] Yin, Wenpeng, et al. "Comparative study of CNN and RNN for natural language processing." arXiv preprint arXiv:1702.01923 (2017). [12] Kılınç, D. (2019). A spark-based big data analysis framework for real-time sentiment prediction on streaming data. Software: Practice and Experience, 49(9), 1352-1364.

[11] S. Delil, B. Kuyumcu, C. Aksakallı and İ. S. Akçıra, "Parsing Address Texts with Deep Learning Method," 2020 28th Signal Processing and Communications Applications Conference (SIU), 2020, pp. 1-4, doi: 10.1109/SIU49456.2020.9302154.

[12] Pedregosa et al., Scikit-learn: Machine Learning in Python, JMLR 12, pp. 2825-2830, 2011.

[13] Tieleman, Tijmen, and Geoffrey Hinton. "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude." COURSERA: Neural networks for machine learning 4.2 (2012): 26-31.

[14] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958

[15] Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980