

An Automated Logistic Price Prediction System

Field of the Invention

This invention relates to an automated price prediction system and method.

- 5 Particularly, this invention relates to a system and method of predicting logistic cost upon receiving an order.

Prior Art

10 In a business setting, determining the price of an order is a key factor in the operation process of the company. Most of the times, the incentives of a business are to complete as many orders as possible and hence, coming up with a methodology to quickly estimate the price of each order is very crucial. However, it is not always easy to do so because the price depends on many variables and there is no established formula to manually calculate the price in real time.

15 Estimating the shipping costs is a very difficult problem in the logistics industry as there are several factors that can affect the prices. Some skilled in the art chose to calculate the fares based on their domain knowledge of the market and some estimate the prices based on a linear formula. The former is inaccurate most of the time for two reasons. First, it is very likely that the domain knowledge will have some human bias. Second, knowledge
20 from a person or a group of people cannot be comprehensive and deep enough to infer the prices of orders that take place in less popular areas. The use of linear formula has its disadvantage as well. For example, a formula that some may use to calculate the prices of delivering an order is as follows:

$$\text{final price} = \text{cargo's weight} \times \text{price per km} \times \text{distance}$$

25 Formulas such as the one above are problematic because (i) it does not consider the fact that it is harder to find trucks in some cities than others; (ii) it does not consider the truck's

model and truck's type; and (iii) it does not consider cases where a large truck can carry more than one order, which reduces the prices significantly.

For example, when a client wants to ship a cargo, he will have to find a truck owner and pays an amount of money to have the cargo delivered to the desired destination.

5 However, different drivers will demand different compensations, which will lead to a lot of difficulties in setting an appropriate price. This leads to inefficiency in the work flow in logistics because more time needs to be spent on researching the fares to find the best deal. This problem is not easy to solve as there is no standard approach for determining the correct price.

10 Therefore, those skilled in the art are striving to provide a system and/or a method of real time price prediction specifically for logistic services.

Summary of the Invention

The above and other problems are solved and an advance in the state of the art is
15 made by a system and method in accordance with this invention. A first advantage of the system and method in accordance with this invention is the ability to estimating the prices instantly after knowing the orders' details. A second advantage of the system and method in accordance with this invention is the capability of improving itself based on new changes in the market, which allows more flexibility and accuracy in a fast-changing industry such
20 as logistics. A third advantage of the system and method in accordance with this invention is that.

A first aspect of the disclosure describes a logistic price prediction system comprising: a processor, a memory and instructions stored on the memory and executable by the processor to: receive an order; pre-process the order to form a vector of length 174
25 relating to the order; and apply a neural network based on the vector to predict a price.

In an embodiment of the first aspect of the disclosure, the instruction to pre-process the order to form the vector of length 174 relating to the order comprises instructions to:

obtain information about the order; generate features based on the information; and convert the order with the generated features to form the vector of length 174.

In an embodiment of the first aspect of the disclosure, the information about the order comprises truck model identification number, truck quantity, pick up date/time, drop
5 off date/time, boolean value indicating whether or not tax invoice is requested, boolean value indicating whether the shipper wants the whole truck or not, truck type identification number, weight of the cargo, length of the cargo, width of the cargo, height of the cargo, type of bidder, pick up area identification number, drop off area identification number, distance from the pick up point to the drop off point, duration from the pick up point to the
10 drop off point, weight of the truck, and type of cargoes.

In an embodiment of the first aspect of the disclosure, the instruction to generate features based on the information comprises instructions to: generate continuous features comprising: volume of truck based on the dimensions of input truck type identification
number, volume of cargo based on the length, width, and height of the cargo, expected
15 delivery speed based on pick up and drop off date/time; in response to features that are not binary, construct one-hot vectors where an array has a single high bit for an index indicating correct class and low bits for others in the array.

In an embodiment of the first aspect of the disclosure, the instructions further comprise instructions to: train a model of the neural network.

20 In an embodiment of the first aspect of the disclosure, the instruction to train the model of the neural network comprises instruction to apply a loss function with the following expression:

$$L(y, f(X)) = \frac{1}{N} \left| \frac{f(X) - y}{y} \right| \cdot g(y, f(X))$$
$$g(y, f(X)) = \begin{cases} (f(X) - y)^2, & \text{if } f(X) - y \geq 0 \\ p \cdot (f(X) - y)^2, & \text{otherwise,} \end{cases}$$

where X is the data, f is the model used by the neural network, N is the total number of records in a training batch, y is the true prices collected from past orders, and p is the pre-configured penalty weight.

In an embodiment of the first aspect of the disclosure, the instruction to apply the
5 loss function further comprises instruction to minimizing $L(y, f(X))$.

A second aspect of the disclosure describes a logistic price prediction method comprising: receiving an order pre-processing the order to form a vector of length 174 relating to the order; and applying a neural network based on the vector to predict a price.

In an embodiment of the second aspect of the disclosure, the step of pre-processing
10 the order to form the vector of length 174 relating to the order comprises: obtaining information about the order; generating features based on the information; and converting the order with the generated features to form the vector of length 174.

In an embodiment of the second aspect of the disclosure, the information about the order comprises truck model identification number, truck quantity, pick up date/time, drop
15 off date/time, boolean value indicating whether or not tax invoice is requested, boolean value indicating whether the shipper wants the whole truck or not, truck type identification number, weight of the cargo, length of the cargo, width of the cargo, height of the cargo, type of bidder, pick up area identification number, drop off area identification number, distance from the pick up point to the drop off point, duration from the pick up point to the
20 drop off point, weight of the truck, and type of cargoes.

In an embodiment of the second aspect of the disclosure, the step of generating features based on the information comprises instructions to: generating continuous features comprising: volume of truck based on the dimensions of input truck type identification number, volume of cargo based on the length, width, and height of the cargo,
25 expected delivery speed based on pick up and drop off date/time; in response to features that are not binary, constructing one-hot vectors where an array has a single high bit for an index indicating correct class and low bits for others in the array.

In an embodiment of the second aspect of the disclosure, the logistic price prediction method further comprises: training a model of the neural network.

In an embodiment of the second aspect of the disclosure, the step of training the model of the neural network comprises applying a loss function with the following
5 expression:

$$L(y, f(X)) = \frac{1}{N} \left| \frac{f(X) - y}{y} \right| \cdot g(y, f(X))$$
$$g(y, f(X)) = \begin{cases} (f(X) - y)^2, & \text{if } f(X) - y \geq 0 \\ p \cdot (f(X) - y)^2, & \text{otherwise,} \end{cases}$$

where X is the data, f is the model used by the neural network, N is the total number of records in a training batch, y is the true prices collected from past orders, and
10 p is the pre-configured penalty weight.

In an embodiment of the second aspect of the disclosure, the step of applying the loss function comprises minimizing $L(y, f(X))$.

Brief Description of the Drawings

The above and other features and advantages in accordance with this invention are
15 described in the following detailed description and are shown in the following drawings:

Figure 1 illustrating an overview of the system in accordance with an embodiment of this disclosure;

Figure 2 illustrating a block diagram of a processing unit in accordance with an embodiment of this disclosure;

20 Figure 3 illustrating modules in a software application in accordance with an embodiment of this disclosure;

Figure 4 illustrating performance of RMSE loss function;

Figure 5 illustrating performance of MSLE loss function;

Figure 6 illustrating performance of the customised loss function in accordance with
25 an embodiment of this disclosure;

Figure 7.1 illustrating a process flow to train a model in accordance with an embodiment of this disclosure;

Figure 7.2 illustrating a process flow to initiate training in accordance with an embodiment of this disclosure;

5 Figure 8 illustrating the performance of the predicted price provided by the system in accordance with an embodiment of this disclosure measured against the actual price;

Figure 9 illustrating the performance of the predicted price provided by the system when using the customized loss function in accordance with an embodiment of this disclosure measured against the actual price;

10 Figure 10 illustrating the performance of the predicted price provided by the system when using the MSE loss function in accordance with an embodiment of this disclosure measured against the actual price; and

Figure 11 illustrating the performance of the predicted price provided by the system when using the MSLE loss function in accordance with an embodiment of this disclosure
15 measured against the actual price.

Detailed Description

This invention relates to an automated price prediction system and method. Particularly, this invention relates to a system and method of predicting logistic cost upon
20 receiving an order.

It is envisioned that a system and method in accordance with embodiments of this invention may be used for predicting logistic cost upon receiving an order. The system and method in accordance with embodiments of this invention uses a data-driven approach. First, data is collected from successful orders in the past in order to train a model that
25 produces a function mapping orders' details to numbers that are closest to the actual prices. This system and method is known as APPLE, which is an abbreviation for Auto Pricing Project for Logistics: code E.

The system and method in accordance with embodiments of this invention estimates the price based on statistics and machine learning. The model is constructed from a dataset that is closely related to logistics and transportation. More specifically, the model can predict the price of an order, which is a trip that delivers package(s) from one location to another using a truck, with a minimal error. Last but not least, the system and method in accordance with embodiments of this invention can be useful for every organization that has an association with trucking and logistics because it can be used as a general tool to predict the cost of a shipment.

Figure 1 illustrates system 100 for automatically predict the price of an order in accordance with this disclosure. The system 100 includes a server 105 and other servers or computing systems 115 for receiving data or orders.

Servers or computing systems 115 and server 105 are typical processing systems such as a desktop computer, laptop computer, or other computer terminal that are communicatively connected to each other via a network 110 via either a wired or wireless connection. Servers or computing systems 115 and server 105 executes applications that perform the required processes in accordance with this disclosure.

Network is a network such as the Internet that allows various servers and computing systems to communicate with one another. One skilled in the art will recognise that more than server 105 may be used without departing from the disclosure. Further details of the servers or computing systems 115 and server 105 will be described below with reference to figure 2.

Processes are stored as instructions in a media that are executed by a processing system in servers or computing systems 115 and server 105 or a virtual machine running on the servers or computing systems 115 and server 105 to provide the method and/or system in accordance with this disclosure. The instructions may be stored as firmware, hardware, or software. Figure 2 illustrates processing system 200 such as the processing system in servers or computing systems 115 and server 105 that execute the instructions to perform the processes for providing a method and/or system in accordance with this invention. One skilled in the art will recognize that the exact configuration of each

processing system may be different and the exact configuration of the processing system in each device may vary. Thus, processing system 200 shown in figure 2 is given by way of example only.

Processing system 200 includes Central Processing Unit (CPU) 205. CPU 205 is a processor, microprocessor, or any combination of processors and microprocessors that execute instructions to perform the processes in accordance with the present invention. CPU 205 connects to memory bus 210 and Input/ Output (I/O) bus 215. Memory bus 210 connects CPU 205 to memories 220 and 225 to transmit data and instructions between the memories and CPU 205. I/O bus 215 connects CPU 205 to peripheral devices to transmit data between CPU 205 and the peripheral devices. One skilled in the art will recognize that I/O bus 215 and memory bus 210 may be combined into one bus or subdivided into many other busses and the exact configuration is left to those skilled in the art.

A non-volatile memory 220, such as a Read Only Memory (ROM), is connected to memory bus 210. Non-volatile memory 220 stores instructions and data needed to operate various sub-systems of processing system 200 and to boot the system at start-up. One skilled in the art will recognize that any number of types of memory may be used to perform this function.

A volatile memory 225, such as Random Access Memory (RAM), is also connected to memory bus 210. Volatile memory 225 stores the instructions and data needed by CPU 205 to perform software instructions for processes such as the processes required for providing a system in accordance with this invention. One skilled in the art will recognize that any number of types of memory may be used as volatile memory and the exact type used is left as a design choice to those skilled in the art.

I/O device 230, keyboard 235, display 240, memory 245, network device 250 and any number of other peripheral devices connect to I/O bus 215 to exchange data with CPU 205 for use in applications being executed by CPU 205. I/O device 230 is any device that transmits and/or receives data from CPU 205. Keyboard 235 is a specific type of I/O that receives user input and transmits the input to CPU 205. Display 240 receives display data from CPU 205 and display images on a screen for a user to see. Memory 245 is a device

that transmits and receives data to and from CPU 205 for storing data to a media. Network device 250 connects CPU 205 to a network for transmission of data to and from other processing systems.

Figure 3 illustrates a program 300 stored in memory or virtual memory of the server 105 for performing the processes in accordance with the disclosure. Program 300 includes a pre-processing module 310 and an APPLE module 320. Briefly, the processes executed by the 2 modules are as follows:

- 1) Pre-processing module 310 receives order and clean the data and create more features by combining and transforming existing features.
- 2) APPLE module 320 receives the data from the pre-processing module and apply a neural network to predict the price using the datasets from the training module 340.

More details of the 2 modules will now be described as follows.

Pre-processing module 310

To improve the quality of the data, we need to clean the data received for each order and create more features by combining and transforming existing features. Specifically,

- for pick up date and time feature, generate more information such as day of the week, whether or not that day is weekend, whether or not that day is a public holiday/
- remove the records where most of the information is missing or unknown.
- drop features with too many missing values.
- utilize fasttext, a natural language processing toolkit, to classify whether a cargo is light or heavy.
- remove records that distance and cargo's weight are 0.
- remove records with price less than a predetermined amount. For example, price less than 400,001 VND.
- remove high-distance order (longer than 300km) but price per km is less than 1,000 VND.

- remove records where price per km per ton is less than a predetermined amount.

For example, less than 500 VND.

- remove records where price per km per ton is higher than a predetermined amount. For example, higher than 10,000 VND.

- 5 - generate polynomials features such as truck's weight divided by duration, truck weight divided by distance, fuel price times distance, cargo's weight per truck times duration, and cargo's weight per truck divided by truck's weight.

The APPLE module 320 is an automated pricing system powered by statistics and machine learning algorithms. The way it works is fairly straightforward. First of all, we need

- 10 to prepare the information about the order that we want to know the price for. Specifically, the APPLE module 320 needs the following information to give the final prediction:

- truck model identification number
- truck quantity
- pick up date/time

15 - drop off date/time

- boolean value indicating whether or not tax invoice is requested
- boolean value indicating whether the shipper wants the whole truck or not
- truck type identification number
- weight of the cargo

20 - length of the cargo

- width of the cargo
- height of the cargo
- type of bidder
- pick up area identification number

25 - drop off area identification number

- distance from the pick up point to the drop off point
- duration from the pick up point to the drop off point

- weight of the truck
- type of cargoes

For example, the pre-processing module 310 can send a json request as below to the system to request for information:

```

5           data={
                'truck_model_id': 102,
                'truck_quantity': 1,
                'pickup_datetime': '2019-10-24 11:57:00',
                'dropoff_datetime': '2019-10-31 11:57:00',
10           'is_tax_invoice_requested': True,
                'is_whole_truck': True,
                'truck_type_id': 5,
                'cargo_weight': 1.0,
                'cargo_length': 0.0,
15           'cargo_width': 0.0,
                'cargo_height': 0.0,
                'bidder_type': 1,
                'pickup_area_id': 10,
                'dropoff_area_id': 35,
20           'distance': 500000,
                'duration': 22344,
                'truck_weight': 7.0,
                'cargo_types': 'máy móc'
            }

```

25 After receiving the above information, the pre-processing module 310 will generate more features based on the input such as boolean value indicating whether or not the cargo is heavy, or boolean value indicating whether or not the pick up date is a public holiday.

Module 310 will also generate continuous features such as truck's volume based on the dimensions of input truck type's id, cargo's volume based on its length, width, and height, as well as expected delivery speed based on pick up and drop off time. For categorical features that are not binary, module 310 will construct one-hot vectors, which are arrays filled with low bits everywhere with a single high bit. The single high bit being a representation for the index indicating the correct class. For instance, if there are 5 possible areas, which can be represented in an array with 5 integers. Let's say the correct area is 3 and it is located in index 3, the one-hot vector that represents correct area is [0,0,1,0,0]. Some of such features that are not binary are day of the week, month interval, distance level, and truck type's id. After such process, for each order, we will have a vector of length 174 that describes the order in detail. That vector will then be fed to a trained neural network in APPLE to make the final prediction for the price.

APPLE module 320

The APPLE module 320 implements a neural network model using PyTorch. This is a 3-layer-neural network with 1024 neurons in layer 1, 512 neurons in layer 2, and 64 neurons in layer 3.

All of the weight matrices are initialized using Xavier Initialization with default setting and bias matrices are initialized as zero matrices. At the first hidden layer, the parameters go through a Rectified Linear Unit (ReLU) activation function, batch normalization layer, and a drop out layer with keep prob = 0.5. At the second hidden layer, the parameters go through a Tanh activation function, batch normalization layer, and a drop out layer with keep prob = 0.3. At the last hidden layer, the parameters go through a ReLU activation function, batch normalization layer, and a drop out layer with keep prob = 0.2. The loss function that we use is a customized loss function. During the optimization step, we use Adam algorithm (batch size=64) with learning rate decay (initial lr = 0.0016 and lr reduces by 70 % every 100 epoches). l2 regularization term (lambda = 0.0923) is also added. We trained the model for 200 epoches and saved the model with the lowest MAPE.

Customized loss function

In the pricing problem, the goal is to give estimates that are as close to the actual prices as possible. In most algorithms that are suitable for this task, predictions are made randomly at first and then adjusted gradually until the errors are acceptable. Those incorrect predictions will then be evaluated with a function to adjust the direction (go higher or lower) of the model's output. This function is known as a loss function. In short, loss function evaluates the error made by the current parameters of the model. The role of this function is to find the direction that leads to lowest possible errors.

Some popular loss functions

Mean squared error

One of the most popular choices for regression problem is MSE, which is defined as

$$\frac{1}{N} \cdot \sum_i (f(X_i) - y)^2$$

where f is the machine learning model, X is the data, N is the number of data points, and y is the vector consisting of corresponding prices for each data point X_i .

MSE measures the square differences between the predicted price and the actual price. It is an easy function to optimize and therefore widely used. However, regressive MSE does not care whether the predicted price is higher or lower compared to the actual price, which might not be ideal in a business setting. See figure 4 on the regressive MSE loss. The y-axis of the plot indicates the predictions and the x-axis indicates the MSE loss corresponding to those predictions when the actual value is 1. It is clear that the plot is symmetric at 0, which means higher predictions and lower predictions are penalized equally. In addition, regressive MSE tends to penalize outliers harshly, which can lead to some disadvantages in making prediction.

Mean square logarithmic error (MSLE)

MSLE is another popular choice to resolve regression problem. In cases where the target variable is high, evaluating the errors based on the natural log can increase the model's accuracy significantly. MSLE's formula is

$$\frac{1}{N} \sum_i (\log(f(X_i) + 1) - \log(y + 1))$$

where f is the model, X_i is the data and y is vector consisting of the corresponding prices of each data point in X .

One feature of this loss is that it penalizes lower predictions a lot more compared to higher ones. See figure 5 on MSLE loss. The y-axis of the plot indicates the prediction and the x-axis indicates the MSLE loss corresponding to that prediction when the actual value is 1. While MSLE is quite good for a business setting, this function does not optimize based on percentage error, which is one of the most important metrics in evaluating business performance.

The loss function that we propose is defined as follows:

$$L(y, f(X)) = \frac{1}{N} \left| \frac{f(X) - y}{y} \right| \cdot g(y, f(X))$$

where X is the data, f is the machine learning model, N is the total number of records in a training batch, y is the true prices collected from past orders, and squared error at the same time, but also assign higher penalties for predictions that are lower than the actual prices.

$$g(y, f(X)) = \begin{cases} (f(X) - y)^2, & \text{if } f(X) - y \geq 0 \\ p \cdot (f(X) - y)^2, & \text{otherwise,} \end{cases}$$

where p is the pre-configured penalty weight.

Here, $g: \mathbb{R}^2 \rightarrow \mathbb{R}$ is a function that takes two arguments, which are y , the correct vector prices and $f(X)$, the vector of predicted prices and output the error of the model.

Then, we can find the best machine learning model f by minimizing $L(y, f(X))$, which means we would not only decrease the mean absolute percentage error and mean squared error at the same time, but also assign higher penalties for predictions that are

lower than the actual prices. See figure 7 on the customized loss. The y-axis of the plot indicates the prediction and the x-axis indicates the proposed customized loss corresponding to that prediction when the actual value is 1.

During the training process in a machine learning problem, a loss function is used to represent the goal we want to achieve. By using the proposed loss function, we can achieve the following goals:

- Minimize the percentage error
- Minimize the mean squared error
- Direct the predicted price to be higher than the actual price instead of lower

Once the neural network is trained based on the above goals, the predicted price will increase compared to other regression objective functions such as MSE.

It is observed that when we used ordinary MSE loss function to optimize our model, we noticed that the price tended to be very low, especially in long-distance cases. During the training process, MSE and similar objective functions do not take into consideration the fact that a high predicted price is more preferable to a low one. Hence, by directly penalizing the low predictions more during the training process, we have improved the situation significantly.

The proposed function can be used not only in logistics but also various different fields such as e-commerce or real estate valuation. As long as the goal is to (i) predict a price for anything with low error and (ii) a high predicted price is preferable to a low one, the described loss function can come in handy and be the most optimal choice.

Example

An example of a process flow performed by program 300 is as follows.

Step 1: Pre-processing module 310 obtain information about the order's details

```
{'truck_quantity': 1,
  'is_tax_invoice_requested': False,
  'is_whole_truck': True,
  'truck_type_id': 5,
  'cargo_weight': 4.0,
  'cargo_length': 0,
  'cargo_width': 0,
  'cargo_height': 0,
  'cargo_types': 'Thức ăn gia súc',
  'price': 1,
  'bidder_type': 0,
  'distance': 65038,
  'duration': 6749,
  'truck_weight': 5.0,
  'pickup_zone': 65,
  'drop_zone': 65,
  'truck_model_id': 35,
  'pickup_datetime': '2019-10-14 04:00:00',
  'dropoff_datetime': '2019-10-21 04:00:00',
  'pickup_area_id': 22,
  'dropoff_area_id': 5}
```

See above sample input data to the model

Step 2. Pre-processing module 310 generates more features and convert the input to a vector of length 174 that describes the order in detail. Each value from the body request is then transformed to its numerical form accordingly based on its type. More specifically,

5 continuous features such as weight, duration, and distance will be transformed to real numbers in the scale [0,1] using min max scaling technique. Categorical features such as date time, pick up zone and truck type id will be transformed to one-hot vectors, which contain all zeroes except for a number one indicating the correct class of each instance.

10 Correct class refers to correct value in each feature and this is represented using one-hot vectors. Using pick up zone as an example, if there are 10 possible pick up zones, an array with 10 integers would be used to represent the 10 possible pick up zones. Assume the correct pick up zone is 4 and it is located in index 4, the one-hot vector that represents

correct pick up zone is $[0,0,0,1,0,0,0,0,0]$. Finally, we concatenate all numbers and vectors together into a giant array of length 174, which is illustrated in the figure below.

[illegible]

See above vector of length 174.

5 Step 3. APPLE module 320 gives the final prediction based on the vector above.
Specifically, the vector of length 174 is fed to the deep neural network.

```
tensor([[1.1701]])
```

Step 4. Final Prediction from APPLE

The output is a tensor containing exactly one float number, and this number is the
10 predicted price given by the system.

Training model for the APPLE module

When machine learning is applied to a real business setting, it is often the case that there will be more data coming in after training the model. For example, if the model is

trained on 10,000 orders and there are 200 new important orders that the model needs to learn from, training 10,200 orders from scratch is not the most optimal solution. In this case, our model utilizes the trained weights of the neural network to help model learn incrementally.

5 More specifically, after we train the first model, we have the good parameters/weights for our neural network. Once we have new data, we will use the same network structure as the original model. However, instead of initializing everything all the parameters randomly, we use the weights of the previous model and update the weights gradually based on the new data. Afterwards, the new weights is validated using an
10 updated test set, which includes both old orders and the new orders. If the performance of the new model increases compared to the old model, the new model will replace the old one.

Figure 7.1 illustrates a process flow 700 for training the model and updating the model incrementally. Process 700 begins with step 705 by collecting and storing data from
15 all successful delivery orders that are made via the services. Step 705 also involves the Pre-processing module 310 to prepare the vector of length 174 for each order. In the data, there are columns describing the orders' details and a column containing the price that has been accepted by both the shippers and the trucks' owners. In more details, the data includes: cargo's height, cargo's length, cargo's weight, cargo's width, cargo's type, the
20 distance between pick up and drop off area, drop off area's id, duration (which is how long it takes to travel from pick up and drop off area), truck's type, truck's weight, whether the shipper wants to use the whole truck or shared truck, how fast the shipper wants their cargoes to be delivered, date and time to pick up the cargo, date and time to drop off the cargo, is tax document required, the type of bidding.

25 In step 710, process 700 feeds the vector of length 174 from step 705 to the deep neural network. The deep neural network has 3 layers with 1024 neurons in layer 1, 512 neurons in layer 2, and 64 neurons in layer 3. All hyper-parameters are optimized using

Bayesian optimization. More specifically, we use Bayesian optimization to search for the optimal learning rate in the interval $[0.0001, 0.01]$, l2 regularization in the interval $[0, 1]$, and learning rate decay in the interval $[0, 1]$. Each layer of the neural network is a matrix, and the system contains three such matrices. The initial data is a tensor of shape $(N, 174)$, where N is the number of records and 174 is the vector received from the pre-processing module 310. When the tensor is fit to the neural network, it will go through each layer by performing a matrix multiplication operation and entering an activation function as defined in the neural network. After going through all the layers, the original tensor will output a tensor of shape $(N, 1)$, which indicates the predicted prices for each order in the original tensor. This output tensor is then inserted to the loss function along with the vector of real prices, which is also of shape $(N, 1)$. Each matrix in each layer of the neural network contains the parameters of the model and will be updated accordingly to reduce the loss value. This is similar to the process as described in step 715. After the loss value is reduced to zero or close to zero, the initial model would be implemented for predicting price for each order.

In step 715, process 700 obtains a feedback from a user if the prediction by the current model and the actual price needed to complete that order in operation is 20% higher or 20% lower than the current model's predicted price. When running the current model in production, there will be cases where the suggested prices from the current model are far off compared to the actual prices. This can happen for various reasons and sometimes can indicate the changes in the market. More specifically, cases where the mean absolute percentage error between the predicted price and the actual operating price is equal to or larger than 20% are considered failures. To tackle this, the system allows feedback from users. A verified member or group of members can directly send the details of the failed cases and the corresponding correct price or prices to our system, which is described in Figure 1, unit 105.

In step 720, process 700 receives the feedback from users. The feedback data contains all the fields the initial data has, along with a suggested correct price for that trip. The suggested correct price is the actual cost needed to pay for that delivery trip in operation process. A sample feedback data is similar to below:

```
5          {
            'truck_model_id': 102,
            'truck_quantity': 1,
            'pickup_datetime': '2019-10-24 11:57:00',
            'dropoff_datetime': '2019-10-31 11:57:00',
10         'is_tax_invoice_requested': True,
            'is_whole_truck': True,
            'truck_type_id': 5,
            'cargo_weight': 1.0,
            'cargo_length': 0.0,
15         'cargo_width': 0.0,
            'cargo_height': 0.0,
            'bidder_type': 1,
            'pickup_area_id': 10,
            'dropoff_area_id': 35,
20         'distance': 500000,
            'duration': 22344,
            'truck_weight': 7.0,
            'cargo_types': 'máy móc',
            'price': 3.5
25         }
```

The feedback data received in step 720 has the exact same format as the initial training data. Therefore, the pre-processing and transforming procedures are exactly identical to those of step 705.

In step 725, process 700 determines whether the number of feedbacks has reached a predetermined batch size of feedback data. As mentioned above, if the model is trained on 10,000 orders and there are 200 new important orders that the model needs to learn from, training 10,200 orders from scratch is not the most optimal solution. In this invention, we utilize the trained weights of the current model to help the model learn incrementally. For example, if the predetermined batch size of feedback data is 64, process 700 proceeds to step 730 to initiate re-training after accumulating 64 new feedbacks. If the current accumulated feedbacks are less than the predetermined batch size of feedback data, process 700 repeats from step 715 to receive the next feedback.

In step 730, process 700 initiates re-training of the current neural network model and updates a validation set. Further details on this step will be described with reference to figure 7.2.

Figure 7.2 illustrates a process flow 730 for initiating re-training of the current deep neural network model and updating a validation set. The predetermined batch size of feedback data will be fed to the neural network to begin re-training. Particularly, the predetermined batch size of feedback data and the original training data will be trained using the same neural network structure as described above in step 710 and the initialized weights are the trained weights from the previous training session.

Process 730 begins with step 731 where the first training phase is forward propagation, which occurs when the deep neural network is exposed to the training data and these cross the entire deep neural network for their predicted prices to be calculated. That is, passing the input data through the network in such a way that all the neurons apply their transformation to the information they receive from the neurons of the previous layer and sending it to the neurons of the next layer. When the data has crossed all the layers,

and all its neurons have made their calculations, the final layer will be reached with a result of label prediction for those input data.

Next, we will apply the proposed loss function to estimate the loss and to compare and measure how good or bad our predicted prices are in relation to the correct prices in respective feedback data. As the deep neural network model is being trained, the weights of the interconnections of the neurons will gradually be adjusted until good predictions are obtained. The weights are updated based on Adam algorithm which is a known optimization algorithm where the value of the loss function gets as close to 0 as possible. Further details will be described below.

In step 733, process 730 proceeds with the subsequent training phase which is forward propagation of the value of the loss function determined in step 732. Specifically, once the loss has been calculated in step 732, this information, i.e. the value of the loss function, is propagated backwards. Starting from the output layer, that loss information propagates to all the neurons in the hidden layer that contribute directly to the output. However, the neurons of the hidden layer only receive a fraction of the total signal of the loss, based on the relative contribution that each neuron has contributed to the original output. This process is repeated, layer by layer, until all the neurons in the network have received a loss signal that describes their relative contribution to the total loss.

In step 734, process 730 adjusts the weights. Now that we have spread this information, i.e. the value of the loss function, backwards in step 733, we can adjust the weights of connections between neurons, which are also our parameters. The goal is to make the loss as close as possible to zero. For this, we will use a technique called Adam optimization. Adam optimization is well known in the art and is omitted for brevity. The weights here refer to the deep neural network weights, which will be updated using Adam optimization until the value of the loss function gets as close to 0 as possible. These weights are updated based on original training data. After the weights have been adjusted, we save the trained parameters as a binary file to our processing system.

We then evaluate the deep neural network with the adjusted weights using a current validation set, which consists of orders that the model has never seen before in step 735. The validation set is updated with new successful orders once after a preconfigured period. After each period, the first 10% old validation data will be removed and new validation data will be concatenated at the end of the validation set. The new batch of validation data is sampled uniformly from most recent completed orders. It is important that the model has never seen any record from this batch. The current validation set is used to calculate the error on the old model, which includes the trained weights of the previous session and the error on the new model, which includes the new adjusted weights. Both new model and old model will be evaluated based on the F1- β score, with the following expression: F1 – β score = $(1 + \beta^2) \frac{x \cdot y}{(\beta^2 x) + y}$, where x and y are the accuracy on validation set and the set of feedback data, respectively. The accuracy here is defined as $1 - \epsilon$, where ϵ is the mean absolute percentage error between the predicted prices and the actual prices of the validation set. β is a hyper-parameter that will determine how much we want the model to be adjusted based on the feedback data. Based on experimental result, a good β score for the model is around 0.4. This means that we can insert $\beta = 0.4$ to calculate the F1- β score of the old model and new model to determine which model performs better. A higher F1- β score relates to better performance. If the new model perform better on the validation set, the new batch of validation data in the current validation set will be emptied and all parameters of the model are updated according to the new model. If the new model performs worse on the validation set when compared to the current model, the new batch of validation data in the current validation set will be kept and added to form the new validation set and trained with the next batch of feedback from users in the later sessions. This is because if the new weights perform worse, the old model will still be kept and all the new data did not have any impact on that retrain session. Hence, we need to save until we have more feedback data and retrain both old feedback data.

If the new weights perform better on the validation set, the new data will be emptied and all parameters of the model are updated. If the new adjusted weights perform worse on the validation set when compared to the old weights, new data will be kept, added and trained with the next batch of feedback from users in the later sessions. This is because if the new weights perform worse, the old model will still be kept and all the new data did not have any impact on that retrain session. Hence, we need to save until we have more feedback data and retrain both old feedback data.

Process 730 ends after step 735.

Overall result of the model

The error rate of our model varies depending on the distribution of the test set. Normally, assume a similarly-distributed test set, our model error rate should fall between 11 to 13% in mean absolute percentage error (MAPE). The mean absolute error (MAE) is about 500,000 VND and the root mean square error is about 920,000 VND.

See figure 8 on the performance of the predicted price vs actual price. As we can observed from figure 8, the two lines, one representing the predicted price 810 from the model and another line representing the actual price 820 correspond to each other. This figure illustrates how APPLE works in real life and how close the predicted prices and the actual prices are.

Experimental Result of implementing loss function

A test has been conducted using the customized loss function in a shipments' prices prediction project. We use mean absolute percentage error (MAPE) as the main evaluation metric and the result has been very promising.

The data we used for this project comprises columns describing the orders' details and a column containing the price that has been accepted by both the shippers and the trucks' owners. In more details, the data includes:

- cargo's height
- cargo's length

- cargo's weight
- cargo's width
- cargo's type
- the distance between pick up and drop off area
- 5 - drop off area's id
- duration, which is how long it takes to travel from pick up and drop off area
- truck's type
- truck's weight
- whether the shipper wants to use the whole truck or shared truck
- 10 - how fast the shipper wants their cargoes to be delivered
- date and time to pick up the cargo
- date and time to drop off the cargo
- is tax document required
- the type of bidding

15 Specifically, the error rate of our model varies depending on the distribution of the test set. Normally, assume a similarly-distributed test set, our model error rate should fall between 11 to 13% in mean absolute percentage error (MAPE). The mean absolute error (MAE) is about 500,000 VND and the root mean square error is about 920,000 VND.

20 Figure 9 illustrates the predicted price vs actual price using the customised loss function. As we can see from figure 9, the line 910 represents the predicted price from the model and the line 920 represents the actual price of the same order. This result indicates that in practice, the customised loss function can be optimized with a neural network to build a robust model.

25 To illustrate the capacity of our function, we have also tried training our network with MSE loss and MSLE loss. With reference to figure 10 illustrating the predicted price vs actual price using MSE loss where the line 1010 represents the predicted price from the model and the line 1020 represents the actual price of the same order and figure 11

illustrating the predicted price vs actual price using MSLE loss where the line 1110 represents the predicted price from the model and the line 1120 represents the actual price of the same order, we can see that with the exact same training procedure and data, the customised loss function outperforms the other two functions in every aspect.

5 In short, the customised loss function has shown its potential in dealing with profit-focused regression problems such as price prediction in the logistics industry. With the current result, the customised loss function will be able to satisfy most of the requirements in building a pricing system that positively affects the business of a company.

10 The above is a description of exemplary embodiments of a device in accordance with this disclosure. It is foreseeable that those skilled in the art can and will design alternative devices based on this disclosure.

Claims

1. A logistic price prediction system comprising:

5 a processor, a memory and instructions stored on the memory and executable by the processor to:

receive an order;

pre-process the order to form a vector of length 174 relating to the order; and

apply a neural network based on the vector to predict a price.

10 2. The logistic price prediction system according to claim 1 wherein the instruction to pre-process the order to form the vector of length 174 relating to the order comprises instructions to:

obtain information about the order;

15 generate features based on the information; and

convert the order with the generated features to form the vector of length 174.

3. The logistic price prediction system according to claim 2 wherein the information about the order comprises truck model identification number, truck quantity, pick up
20 date/time, drop off date/time, boolean value indicating whether or not tax invoice is requested, boolean value indicating whether the shipper wants the whole truck or not, truck type identification number, weight of the cargo, length of the cargo, width of the cargo, height of the cargo, type of bidder, pick up area identification number, drop off area identification number, distance from the pick up point to the drop off point, duration from
25 the pick up point to the drop off point, weight of the truck, and type of cargoes.

4. The logistic price prediction system according to claim 3 wherein the instruction to generate features based on the information comprises instructions to:

generate continuous features comprising: volume of truck based on the dimensions of input truck type identification number, volume of cargo based on the length, width, and height of the cargo, expected delivery speed based on pick up and drop off date/time;

in response to features that are not binary, construct one-hot vectors where an array
5 has a single high bit for an index indicating correct class and low bits for others in the array.

5. The logistic price prediction system according to claim 1 wherein the instructions further comprise instructions to:

train a model of the neural network.

10

6. The logistic price prediction system according to claim 5 wherein the instruction to train the model of the neural network comprises instruction to apply a loss function with the following expression:

$$L(y, f(X)) = \frac{1}{N} \left| \frac{f(X) - y}{y} \right| \cdot g(y, f(X))$$

15

$$g(y, f(X)) = \begin{cases} (f(X) - y)^2, & \text{if } f(X) - y \geq 0 \\ p \cdot (f(X) - y)^2, & \text{otherwise,} \end{cases}$$

where X is the data, f is the model used by the neural network, N is the total number of records in a training batch, y is the true prices collected from past orders, and p is the pre-configured penalty weight.

20 6. The logistic price prediction system according to claim 5 wherein the instruction to apply the loss function further comprises instruction to minimizing $L(y, f(X))$.

7. A logistic price prediction method comprising:

receiving an order;

25

pre-processing the order to form a vector of length 174 relating to the order; and

applying a neural network based on the vector to predict a price.

8. The logistic price prediction method according to claim 7 wherein the step of pre-processing the order to form the vector of length 174 relating to the order comprises:

obtaining information about the order;

5 generating features based on the information; and

converting the order with the generated features to form the vector of length 174.

9. The logistic price prediction method according to claim 8 wherein the information about the order comprises truck model identification number, truck quantity, pick up
10 date/time, drop off date/time, boolean value indicating whether or not tax invoice is requested, boolean value indicating whether the shipper wants the whole truck or not, truck type identification number, weight of the cargo, length of the cargo, width of the cargo, height of the cargo, type of bidder, pick up area identification number, drop off area identification number, distance from the pick up point to the drop off point, duration from
15 the pick up point to the drop off point, weight of the truck, and type of cargoes.

10. The logistic price prediction method according to claim 9 wherein the step of generating features based on the information comprises instructions to:

generating continuous features comprising: volume of truck based on the
20 dimensions of input truck type identification number, volume of cargo based on the length, width, and height of the cargo, expected delivery speed based on pick up and drop off date/time;

in response to features that are not binary, constructing one-hot vectors where an array has a single high bit for an index indicating correct class and low bits for others in the
25 array.

11. The logistic price prediction method according to claim 7 further comprises:

training a model of the neural network.

12. The logistic price prediction method according to claim 11 wherein the step of training the model of the neural network comprises applying a loss function with the

5 following expression:

$$L(y, f(X)) = \frac{1}{N} \left| \frac{f(X) - y}{y} \right| \cdot g(y, f(X))$$
$$g(y, f(X)) = \begin{cases} (f(X) - y)^2, & \text{if } f(X) - y \geq 0 \\ p \cdot (f(X) - y)^2, & \text{otherwise,} \end{cases}$$

where X is the data, f is the model used by the neural network, N is the total number of records in a training batch, y is the true prices collected from past orders, and
10 p is the pre-configured penalty weight.

13. The logistic price prediction method according to claim 12 wherein the step of applying the loss function comprises minimizing $L(y, f(X))$.