

A Recommender System For Logistics

Field of the Invention

This invention relates to a logistic recommender system and method. Particularly,
5 this invention relates to a logistic recommender system and method for recommending
logistic orders to logistic services. More particularly, this invention relates to a logistic
recommender system and method that automatically determines a best match for a logistic
order to one of logistic services.

Prior Art

10 While logistics is a well-established industry, the rate of technology adoption is slow.
According to a PwC report, the percentage of transport and logistics companies that rated
themselves as 'advanced' on digitization was just 28 percent. This suggests that many
tasks in logistics are still labour intensive. With the current state of high growth in logistics
demand, it is worth questioning how logistics companies, with the traditional labour-
15 intensive model, could keep up with the demand to fully enjoy the benefit the rapid growth
brought about. The situation gets even more complicated given the industry's highly
fragmented market.

Embracing technology breakthrough such as Artificial Intelligence (AI), according to
industry experts, provides vast opportunities for logistics companies to improve
20 performance and serve customers better.

Recommender systems (RS) are machine learning applications that are able to
learn the need and preferences of users, filter the huge collection of data accordingly and
present the best suited option by using some well-defined mechanism (Sohail et al., 2017).
Presently, one could easily find themselves making use of recommender systems in their
25 daily activities such as with movies suggestion on Netflix, video recommendation on
YouTube or product recommendation on e-commerce websites. RS have been reported to
have an important business impact on such kind of businesses. Businesses find it as an
efficient way to access its potential customers while customers/individuals are pretty much
saved from information overload problem through personalized recommendations, content,
30 and services to their taste, that were provided by recommender systems.

Recently, several tech start-up such as Uber Freight, Convoy and Transfix
announced that they have fully utilized their digital freight matching practice. However,
while RS have been mostly studied, developed and applied for the ecommerce business,
research and application on the logistics business is still under-developed. Moreover, the
35 unique nature of freight transportation poses different challenges to the recommendation
task as compared to other kinds of businesses. Each order has specific requirements of

specialized equipment types, shipments transported via multiple modes and handling service, just to mention few of the variables involved. Freight's high-value and time-and-location-sensitive nature add more complexity to the matter. Moreover, orders, unlike movies or ecommerce products, expire as soon as they are consumed (fulfilled) by a single user (a truck driver). As a result, a different kind of recommender system than that of other sectors like e-commerce and e-entertainment, is required to provide efficient freight matching.

Therefore, those skilled in the art are striving to provide a system and/or a method to overcome the above mentioned problem of freight matching in the logistics environment.

Summary of the Invention

The above and other problems are solved and an advance in the state of the art is 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 that the system and method accelerates work and reduces costs. This is because it allows automatic recommendations without human intervention. A second advantage of the system and method in accordance with this invention is that the system and method reduces delays. This is because it allows immediate retrieval of suitable drivers as soon as an order is posted. A third advantage of the system and method in accordance with this invention is that the system and method increases matching accuracy. This is because it is fully AI-based, which means that dozens of variables can be considered for tens of thousands of orders and drivers when deciding the best match between an order and a driver. As a result, the system and method creates more jobs for drivers, increases customer retention (both drivers and good owners), and reduces overall costs for all the stakeholders in the market.

A first aspect of the disclosure describes a logistic recommender system for automatically matching freight service to an order in the logistics business. The logistic recommender system comprising: a processor, memory and instructions stored on the memory and executable by the processor to: receive a plurality of driver and order data; pre-process the received plurality of driver and order data and translate the plurality of driver and order data to form a training dataset expressed as a matrix of shape, $(n * b) \times m$, where, n is the number of drivers, b is the average number of biddings per driver, and m is the length of a vector where each vector element is an element of an order_vector and a driver_profile_vector at time T ; and train a model of a Siamese neural network with a triplet loss function based on the training dataset with the objective of finding the degree to which the input order_vector and the input_driver_profile_vector are a good match

In an embodiment of the first aspect of the disclosure, the instruction to pre-process the received plurality of driver and order data and translate the plurality of driver and order

data to form the training dataset comprises instructions to: generate an order_vector for each of the plurality of driver and order data, the order_vector representing the characteristics of an order that the driver has bid for at time T and containing order number, route, cargo weight, truck model type, price, pickup and dropoff time; generate a driver_activity_history_vector for each driver, the driver_activity_history_vector representing activity of the driver up to time T-1 when the driver made the bidding and containing driver identification number, time stamp, location, route history, cargo weight history, truck model type history, price history, pickup history and dropoff time history; generate a driver_stable_profile_vector for each driver, the stable_driver_profile_vector includes driver identification, truck type and preferred routes; combine the driver_activity_history_vector and the driver_stable_profile_vector to form a driver profile vector; and combine the order_vector and the driver profile vector to form the (n * b) rows in the matrix.

In an embodiment of the first aspect of the disclosure, the instruction to combine the driver_activity_history_vector and the driver_stable_profile_vector to form the driver profile vector further comprises instructions to: encode categorical variables in the driver profile vector into integer lists; and pad with replacement the integer lists to align the length of each integer list to a maximum length for that categorical variable.

In an embodiment of the first aspect of the disclosure, the instruction to encode categorical features into integer lists is one of one-hot-encoding and integer encoding.

In an embodiment of the first aspect of the disclosure, the instruction to combine the driver_activity_history_vector and the driver_stable_profile_vector to form the driver profile vector further comprises instructions to: standardise numerical variables using z-score standardization which can be expressed in the following expression: $x' = (x - \text{mean}(X)) / \text{std}(X)$ where x is one instance of a numerical variable, mean(X) is the mean of X, std(X) is the standard deviation of X, and x' is the standardized version of x.

In an embodiment of the first aspect of the disclosure, the instruction to train the model of the Siamese neural network with the triplet loss function based on the training dataset comprises instructions to: obtain the driver_profile_vector at a particular time point as an anchor; obtain one of the order_vector representing the order that the driver bid for at that time point as a positive_order_vector; and obtain one of the order_vector representing the order that the driver did not bid for at that time point as a negative_order_vector.

In an embodiment of the first aspect of the disclosure, the instruction to train the model of the Siamese neural network with the triplet loss function based on the training dataset further comprises instructions to: feed the anchor, positive_order_vector and

negative_order_vector to the Siamese neural network where two vectors along a horizontal axis are concatenated and thereafter flattened to a required length before applying the triplet loss function which can be expressed in the following expression,

$$L(A, P, N) = \max(\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \alpha, 0)$$

5 where $f(A)$, $f(P)$, $f(N)$ are representation of the flattened anchor vector, positive vector and negative vector respectively, and α is a margin.

In an embodiment of the first aspect of the disclosure, the instruction to train the model of the Siamese neural network with the triplet loss function based on the training dataset further comprises instructions to: repeat the training of the model for 200 epochs
10 to obtain a trained model.

In an embodiment of the first aspect of the disclosure, the logistic recommender system further comprising instructions to: receive a driver_vector and an order_vector; feed the driver_vector and the order_vector to the trained model to output a driver encoded vector and an order encoded vector; calculate a cosine distance between the driver
15 encoded vector and the order encoded vector; and classify a match in response to the cosine distance being less than 0.8.

In an embodiment of the first aspect of the disclosure, the logistic recommender system further comprising instructions to: receive a driver_vector and a plurality of order_vectors; feed the driver_vector and each of the plurality of the order_vectors to the
20 trained model to output a driver encoded vector and a plurality of order encoded vectors; calculate a cosine distance between the driver encoded vector and each of the plurality of the order encoded vectors; recommend the order encoded vectors with cosine distance being less than 0.8.

A second aspect of the disclosure describes a logistic recommender method for
25 automatically matching freight service to an order in the logistics business. The logistic recommender method comprising: receiving a plurality of driver and order data; pre-processing the received plurality of driver and order data and translate the plurality of driver and order data to form a training dataset expressed as a matrix of shape, $(n * b) \times m$, where, n is the number of drivers, b is the average number of biddings per driver, and m is the
30 length of a vector comprising the order_vector and the driver_profile_vector at time T ; and training a model of a Siamese neural network with a triplet loss function based on the training dataset.

In an embodiment of the second aspect of the disclosure, the step of pre-processing the received plurality of driver and order data and translate the plurality of driver and order
35 data to form the training dataset comprises: generating an order_vector for each of the plurality of driver and order data, the order_vector representing a characteristic of an order

that the driver has bid for at time T and containing order number, route, cargo weight, truck model type, price, pickup and dropoff time; generating a driver_activity_history_vector for each driver, the driver_activity_history_vector representing activity of the driver up to time T-1 when the driver made the bidding and containing driver identification number, time stamp, location, route history, cargo weight history, truck model type history, price history, pickup history and dropoff time history; generating a driver_stable_profile_vector for each driver, the stable_driver_profile_vector includes driver identification, truck type and preferred routes; combining the driver_activity_history_vector and the driver_stable_profile_vector to form a driver profile vector; and combining the order_vector and the driver profile vector to form the (n * b) rows in the matrix.

In an embodiment of the second aspect of the disclosure, the step of combining the driver_activity_history_vector and the driver_stable_profile_vector to form the driver profile vector further comprises: encoding categorical variables in the driver profile vector into integer lists; and padding with replacement the integer lists to align the length of each integer list to a maximum length for that integer list.

In an embodiment of the second aspect of the disclosure, the step of encoding categorical features into integer lists is one of one-hot-encoding and integer encoding.

In an embodiment of the second aspect of the disclosure, the step of combining the driver_activity_history_vector and the driver_stable_profile_vector to form the driver profile vector further comprises: standardising numerical variables using z-score standardization which can be expressed in the following expression: $x' = (x - \text{mean}(X)) / \text{std}(X)$ where x is one instance of a numerical variable, mean(X) is the mean of X, std(X) is the standard deviation of X, and x' is the standardized version of x.

In an embodiment of the second aspect of the disclosure, the step of training the model of the Siamese neural network with the triplet loss function based on the training dataset comprises: obtaining the driver_profile_vector at a particular time point as an anchor; obtaining one of the order_vector representing the order that the driver bid for at that time point as a positive_order_vector; and obtaining one of the order_vector representing the order that the driver did not bid for at that time point as a negative_order_vector.

In an embodiment of the second aspect of the disclosure, the step of training the model of the Siamese neural network with the triplet loss function based on the training dataset further comprises: feeding the anchor, positive_order_vector and negative_order_vector to the Siamese neural network where two vectors along a horizontal axis are concatenated and thereafter flattened to a required length before applying the triplet loss function which can be expressed in the following expression, $L(A, P, N) =$

$\max(\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \alpha, 0)$ where $f(A)$, $f(P)$, $f(N)$ are representation of the flattened anchor vector, positive vector and negative vector respectively, and α is a margin.

In an embodiment of the second aspect of the disclosure, the step of training the model of the Siamese neural network with the triplet loss function based on the training dataset further comprises: repeating the training of the model for 200 epochs to obtain a trained model.

In an embodiment of the second aspect of the disclosure, the logistic recommender method further comprises: receiving a driver_vector and an order_vector; feeding the driver_vector and the order_vector to the trained model to output a driver encoded vector and an order encoded vector; calculating a cosine distance between the driver encoded vector and the order encoded vector; and classifying a match in response to the cosine distance being less than 0.8.

In an embodiment of the second aspect of the disclosure, the logistic recommender method further comprises: receiving a driver_vector and a plurality of order_vectors; feeding the driver_vector and each of the plurality of the order_vectors to the trained model to output a driver encoded vector and a plurality of order encoded vectors; calculating a cosine distance between the driver encoded vector and each of the plurality of the order encoded vectors; recommending the order encoded vectors with cosine distance being less than 0.8.

Brief Description of the Drawings

The above and other features and advantages in accordance with this invention are 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;

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

Figure 4 illustrating a simplified plot of the model architecture;

Figure 5 illustrating the functions of the inference model;

Figure 6 illustrating a distribution of the distance between a driver_embedding_vector and an order_embedding_vector for positive vs negatives examples on the training set;

Figure 7 illustrating the 3 principal components obtained through t-SNE before training;

Figure 8 illustrating the 3 principal components obtained through t-SNE after training;

Figure 9 illustrating the distribution of the cosine distance between a driver_embedding_vector and an order_embedding_vector for positive examples vs negatives examples on the test set;

Figure 10 illustrating a full model architecture graph with the input to the model architecture as shown in figure 4;

Figure 11 illustrating a first magnified part of the full model architecture graph as shown in figure 10;

Figure 12 illustrating a second magnified part of the full model architecture graph as shown in figure 10;

Figure 13 illustrating a third magnified part of the full model architecture graph as shown in figure 10; and

Figure 14 illustrating an alternative embodiment to the model architecture as shown in figure 4.

Detailed Description

This invention relates to a logistic recommender system and method. Particularly, this invention relates to a logistic recommender system and method for recommending logistic orders to logistic services. More particularly, this invention relates to a logistic recommender system and method that automatically determines a best match for a logistic order to one of logistic services.

It is envisioned that a system and method in accordance with embodiments of this invention may be used for freight matching in the logistics business. The system and method in accordance with embodiments of this invention is also known as a Recommender systems (RS) – REEL (REcommender Engine for Logistics). The system and method in accordance with embodiments of this invention aims to solve freight matching which is the task of finding the best match between orders and drivers in the logistics business. Currently, this problem is tackled by most logistics companies in the following two methods:

1. First come first served method: calling up drivers until one accepts an order. Drivers may be chosen at random or through human knowledge of the person calling. This method is time consuming and costly (it requires a considerable amount of human work). Further, it does not ensure an optimal match between driver and order, and it may cause serious delays between the posting of an order and the finding of a suitable driver.

2. Filtering method: applying an automatic, filtering strategy to select the N best drivers for an order or the N best orders for a driver. Technically, this method is often

implemented as a simple SQL database query where database records are filtered under certain conditions. This method is highly reliant on human input, as the filters are decided beforehand by humans. Filters are only basic, and may include distance between current driver's location and order pickup point, or truck type. This method is unable to consider complex relationships among the many variables involved, and it is not able to consider the individual preferences of each single driver.

The system and method in accordance with embodiments of this invention improves on the above methods as follows:

1. Accelerating work and reducing costs: The RS in accordance with embodiments of this invention allows automatic recommendations, i.e. no human work is required and no middlemen are involved.

2. Reducing delays: The RS in accordance with embodiments of this invention allows immediate retrieval of suitable drivers as soon as an order is posted.

3. Increasing matching accuracy: The RS in accordance with embodiments of this invention is fully AI-based, which means that dozens of variables can be considered for tens of thousands of orders and drivers when deciding the best match between an order and a driver. This task could never be accomplished with the same accuracy by a human or by a non-AI-based filter

The benefits of faster and more accurate recommendations comprises:

1. More jobs for drivers
2. Increased customer retention (both drivers and good owners)
3. Reduced overall costs for all the stakeholders in the market

In machine learning terms, we are facing a recommendation task where we want to recommend items (orders) to users (drivers) in the absence of explicit ratings.

For this RS to work, however, we need at least some form of *implicit rating* to distinguish between positive examples (orders that we should recommend to the driver) and negative examples (orders that we should not recommend to the driver). These can be understood as two separate classes used to train a binary classifier. In our specific case, we have only one form of implicit rating, which is the type of driver interaction with an order, which could be:

1. Order bid (we will name it 'bidding')
2. Order opened on the app as a result of a notification received, but not bid (we will name it 'click'). We do not know why the driver did not bid for the order
3. Order for which a notification was received, but that was not opened on the app (we will name it 'no click')

4. All the other orders, for which a notification was never received and that were never opened nor bid (we will name them 'others').

Thus, we can select 'biddings' as our positive examples, and the rest ('click', 'no click', 'others') as our negative examples. In machine learning terms, this situation can be understood as a positive-unlabelled (or semi-supervised) learning problem, where only the positive class is known, and the negative class is unknown. In our case, the positive class (reliable positive examples) is represented by the orders that a driver has bid for in the past. The act of bidding for an order thus indicates interest in that type of order. Conversely, all the other available orders that the driver did not bid for do not easily provide us with obvious information about the degree of interest in that order, because the driver may have not bid for that order for a number of reasons. We do not have explicit ratings of an order to indicate that the driver was not interested in that order. However, based on implicit ratings, we identify different types of negative examples (in our case, 'clicks', 'no clicks' and 'others'). Moreover, we filter out examples that cannot provide useful information about the driver's interest in an order, that are orders that could not possibly be fulfilled by the driver (we call these 'impossible' orders). In our specific case, we adopted two filtering criteria:

1. reachability: we filter out orders whose pickup location is too far to be reached on time by the driver, given the driver's location at the time of his interaction with the order. For example, a 'click' will be included in the training set if at the time the driver clicked on the order notification, he/she was at a location that allowed him/her to reach the order pickup point on time. If this condition is not met, the 'click' is not included in the training set as a reliable negative.

2. cargo size: we filter out orders which are too large or heavy to be delivered by the driver, given the particular order's cargo size and weight and the particular driver's truck size and capacity.

Excluding 'impossible' orders from training is done because when the model is used in production, 'impossible' orders will be mechanically filtered out before the machine learning model is applied.

Any other constrain could be applied to exclude impossible orders according to the specific application.

The specific types of interactions listed above are not a necessary condition to the application of our RS. In the presence of other types of explicit or implicit ratings, the necessary condition is that they can be used to create two separate yes/no classes indicating the orders that we should and should not recommend to the driver. In fact, our negative sampling methodology may be applied in any logistics or non-logistics related business given the following conditions:

1. The aim is to identify a sample of reliable negative examples for any machine learning task that requires two or more labelled classes

2. A sample of reliable positive examples is known

3. The business application allows the recording of at least two types of user activity

(implicit ratings):

a. Purchase (or acceptance, or bidding): any sort of behaviour that indicates that the user has definitively chosen/consumed/purchased the item/service offered by the business

b. Inspect (or click, open, query, inquiry): any sort of behaviour that indicated that the user has interacted in some active way with the item/service offered by the business, but has not finalized the purchase. For instance, the user has clicked on the item on the application, or has sent an inquiry about the item. Passive behaviour (e.g. the user has seen the item by chance) does not count in this category

4. Users and items are time and location dependent. This means that the user behaviour will change according to the user location, and so should the class assignment of an item. In other words, an item may be classified as a positive example for a certain user if the user is in one location, or as a negative example if the user is in another location.

5. Because of condition 4, it is also necessary that the business application allows the recording of time and location at which the user performed the activities a and b in point 3 above.

6. Items are not stable in time, that is, they expire as soon as they are consumed by a user. In this case, orders are no longer valid to be fulfilled by other drivers once they are fulfilled by one driver. Similarly, orders expire once the pickup deadline has passed whether or not they are fulfilled by any driver.

Another condition that must be satisfied for the RS to be applied is the possibility to characterize entities (drivers and orders) by at least some features. In our case, we have data on driver's preferred routes, past biddings, truck type owned etc.; and on order's route, price, truck type requested etc. In other words, the conditions for content-based filtering must be satisfied.

Finally, a particular characteristic of the input (drivers and orders) and the output (recommendations) is that they are both time and location-dependent. The representation of a driver's input vector to the model, for instance, is different according to the driver's location at the time they bid. Consequently, the recommendation is different according to the driver's location. Similarly, a certain order may be a good recommendation for a driver only up to a certain time point, after which it would be too late for the driver to fulfill the order. Moreover, the items (orders) are not stable as in the most common applications of a

RS (movies, songs, products). Instead, they expire as soon as they are fulfilled by a driver or the order pickup deadline has passed.

In summary, the conditions that make the model unique and possible to apply are:

1) A recommendation task

2) Users are drivers and items are orders that expire as soon as they are fulfilled by a driver or as soon as the pickup deadline has passed

3) Availability of implicit ratings to label orders as 'positive' or 'negative' in the absence of explicit ratings. If explicit ratings are available, we can still apply our RS without the need to creating a negative sample by using implicit ratings

4) Features to characterize orders and drivers allowing content-based filtering

5) Input representation and recommendations are time and location dependent

The scenario depicted above represents the majority of freight matching problems faced by logistics companies, and it is thus highly relevant and applicable across the logistics industry.

The model selected is a Siamese neural network with triplet loss function that takes as input, at training phase, three vectors:

a) A `driver_profile_vector`, that represents the driver history (i.e. past biddings, searches, truck type owned etc.) at a particular time point, and the driver location at that time point (anchor vector)

b) A `positive_order_vector` representing an order that we should recommend to the driver given the driver history and location at that time point (positive vector)

c) A `negative_order_vector` representing an order that we should not recommend to the driver given the driver history and location at that time point (negative vector).

At inference phase, the model receives as input a `driver_profile_vector` (anchor_vector) and an `order_vector`, and outputs the dissimilarity score (cosine distance) between the input `driver_profile_vector` and the input `order_vector`. Based on the value of the dissimilarity score, the input `order_vector` can be classified as positive (i.e. relevant for the particular input `driver_profile_vector`) or as negative (i.e. not relevant for the particular input `driver_profile_vector`). The optimal threshold to establish whether to classify the order as positive or negative is defined at training phase, and will thus depend on the particular dataset used. For example, in our specific case, the threshold used was 0.8, that is, order-driver pairs with a dissimilarity score below or equal to 0.8 are classified as matching pairs (the order is classified as positive for the particular input driver), while order-driver pairs with a dissimilarity score above 0.8 are classified as non-matching pairs. The threshold of 0.8 used in our case can be identified by looking at the distribution of dissimilarity scores for positive and negative orders at training phase as shown in Figure 6. Similarly, at

inference phase, given a list of driver_profile_vector – order_vector pairs, the model can output the dissimilarity score between each pair. The dissimilarity score can be used as above to classify matching and non-matching pairs, or to rank pairs from the most matching (lowest dissimilarity) to the least matching (higher dissimilarity).

5 Figure 1 illustrates logistic recommender system 100 for automatically matching freight service to an order in the logistics business in accordance with this disclosure. The logistic recommender system 100 includes a server 105 and other servers or computing systems 115 for receiving data or orders.

10 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.

15 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.

20 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.

30 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 REEL module 320. Briefly, the processes executed by the 2 modules are as follows:

- 1) Pre-processing module 310 receives driver and order data and translates the driver and order data in to the required vector form.
- 2) REEL module 320 receives the data from the pre-processing module and applies a Siamese neural network to recommend order to a driver.

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

Pre-processing module 310

Below are the most important features used to represent the driver and the order vectors.

Driver vector:

- Driver information
 - Truck type: the type(s) of truck owned by the driver
 - Truck capacity: the maximum weight that the driver's truck(s) can carry
 - Preferred routes: the preferred route(s) inputted by the driver when

5 registering to the application

- Activity information
 - Searches
 - Searched routes: the routes searched through the app in the past by the driver

10

- Timestamp: the time when the driver made the search

- GPS location: the GPS location of the driver when they made the search

- Biddings
 - Bid orders: the order ID of orders bid in the past by the driver
 - Timestamp: the time when the driver made the bidding
 - GPS location: the GPS location of the driver when they made the bidding

15

Order vector:

- Route: pickup and dropoff points of the order
- Price: the price offered for carrying out the delivery
- Pickup and dropoff time: the required time for picking up and dropping off the goods
- Truck type: the truck type required for this order
- Cargo weight: the weight of the goods to be carried

20

25 From these basic features, other features can be engineered, such as the length of the route, the duration of the trip, etc. See Appendix 1 for a table with the full feature list after feature engineering.

The pre-processing module 310 will receive a number of driver and order data. Based on the data received, the pre-processing module 310 will process the data and form a dataset for all drivers. The dataset includes driver_activity_history_vector, stable_driver_profile_vector and n order vectors for each driver, where $n=0,1,2,3,\dots,n-1$.

30

Datasets for recommender systems are conventionally thought of as $n \times m$ sparse matrices where n is the number of unique users (in our case, drivers) and m is the total number of items (in our case, orders). For this invention, we created a different type of dataset that will be described below.

35

The basic unit of investigation (a dataset row) in our case is represented not by each individual driver, but by each driver's biddings, so that multiple rows may include information related to the same driver, but at different points in time. The dataset for this invention can be expressed in the following matrix, $(n * b) \times m$, where n is the number of drivers, b is the average number of biddings per driver, $(n * b)$ forms a row in the matrix and m is the length of vector comprising `order_vector` and `driver_profile_vector` at time T . This means that instead of n rows, we obtain $n * b$ rows, where b is the average number of biddings per driver. Since each bidding happens at a certain time, the information included in each dataset row is also time-dependent. For example, row R is related to bidding N made at time T made of two vectors concatenated horizontally:

1) The **order_vector**, representing the characteristics of the order the driver has bid for bidding N at time T

2) The **driver_profile_vector**, which is a combination of two vectors:

a) The **driver_activity_history_vector** up to time $T-1$, representing the driver's activity history up to the time they made for bidding N .

b) The **driver_stable_profile_vector**, including information such as preferred routes and truck type that are always the same for a driver regardless of time and location.

The way the two vectors a) and b) are combined is described as follows.

Order Vector

The `order_vector` is made of the features listed above in "order vector" (i.e. features route, price, pickup and dropoff time, truck model type, cargo_weight). The `driver_activity_history_vector` at time T is made of all information included in the driver's `order_vectors` up to time $T-1$, plus timestamp and location. Let's consider the following example.

Given that the driver A has made N biddings in the past:

i) Dataset row 0 (time T_0 , bidding 0) for driver A will include:

- The `order_vector` relative to the first bidding made by the driver (bidding 0).

For simplicity and economy of space, we will characterize an order only by the route, cargo weight and truck type in this example, but several other characteristics may be present, which can be expressed with the following vector 1 (`Order_vector` for the first driver's bidding):

```
{
  'bidding_id':[0]
  'order_id':[1243],
  'order_route':['HCMC-Hanoi'],
  'cargo_weight':[100],
  'truck_type':['container'],
  'other_order_info':['...']
}
```

- The driver_activity_history_vector relative to the driver's activity history up to time $T(0-1) = T(-1)$. Since this is the first driver's bidding, the driver_activity_history_vector will be empty at row 0, and it will only include the time and location where the driver made bidding 0 which can be expressed with the following vector 2 (Driver_activity_history_vector for the first driver's bidding):

```
{
  'driver_id':[0],
  'timestamp':['21:30:00'],
  'location':[(10.7601,106.3929)],
  'route_history':[],
  'cargo_weight_history':[],
  'truck_type_history':[],
  'other_order_info_history':[]
}
```

The activity history is made of the information of the orders bid in the past up to time T (route, cargo weight, etc.). In this example, no activity history is available.

- The driver_stable_profile_vector represents time-independent driver's information. For simplicity we will use only truck type and preferred routes in this example which can be expressed with the following vector 3 (driver_stable_profile_vector for the first driver's bidding):

```
{
  'driver_id':[0],
  'truck_type':['container'],
  'preferred_routes':['HCMC-Hanoi', 'Hanoi-HCMC']
}
```

ii) We then combine the driver_stable_profile_vector (vector 3) with the driver_activity_history_vector (vector 2) to obtain the driver_profile_vector which can be expressed with the following vector 4 (Driver_profile_vector for the first driver's bidding):

```
{
  'driver_id':[0],
  'timestamp':['21:30:00'],
  'location':[(10.7601,106.3929)],
  'route_history':['HCMC-Hanoi', 'Hanoi-HCMC'],
  'cargo_weight_history':[],
  'truck_type_history':['container'],
  'other_order_info_history':[]
}
```

The driver_profile_vector will include, for each feature, all available information from the driver's bidding history and stable characteristics. In this example, even though the driver has made no bidding yet, we can include two routes in his history because we find them in their preferred routes. Similarly, we can also include the truck type history.

iii) Dataset row 0 will thus be the horizontal concatenation of the order_vector (vector 1) and the driver_profile_vector (vector 4) which can be expressed with the following vector 5 (Dataset row for the first driver's bidding):

```
{
  # driver_profile_vector
  'driver_id':[0],
  'timestamp':['21:30:00'],
  'location':[(10.7601,106.3929)],
  'route_history':['HCMC-Hanoi', 'Hanoi-HCMC'],
  'cargo_weight_history':[],
  'truck_type_history':['container'],
  'other_order_info_history':[],
  # order_vector
  'bidding_id':[0]
  'order_id':[1243],
  'order_route':['HCMC-Hanoi'],
  'cargo_weight':[100],
  'truck_type':['container'],
  'other_order_info':['...']
}
```

iv) Dataset row 1 (time T1, bidding 1) for driver A will include:

- The order_vector relative to the second bidding made by the driver (bidding 1) which can be expressed with the following vector 6 (Order_vector for the second driver's bidding).

```
{
  'bidding_id':[1]
  'order_id':[1244],
  'order_route':['HCMC-Binh Duong'],
  'cargo_weight':[50],
  'truck_type':['cold container'],
  'other_order_info':['...']
}
```

- The driver_profile_vector relative to the driver's activity history up to time $T(1-1) = T_0$. Thus, driver_profile_vector of row 1 will be created by combining the driver_profile_vector of row 0 with the information of the order_vector of row 0, while timestamp and location will be updated with the current timestamp and location which can be expressed with the following vector 7 (driver_profile_vector for the second driver's bidding):

```
{
  'driver_id':[0],
  'timestamp':['19:15:00'],
  'location':[(10.7601,106.3929)],
  'route_history':['HCMC-Hanoi', 'Hanoi-HCMC', 'HCMC-Hanoi'],
  'cargo_weight_history':[100],
  'truck_type_history':['container','container'],
  'other_order_info_history':['...']
}
```

At row 1, this includes information about the driver_stable_profile_vector and the driver past biddings. Note that in comparison with the driver_profile_vector at row 0, the driver_profile_vector at row 1 has been updated with the route of order_vector at row 0 ('HCMC-Hanoi'), its cargo_weight (100) and truck type ('container').

v) Dataset row 1 will thus be the horizontal concatenation of the driver_profile_vector and the order_vector which can be expressed with the following vector 8 (Dataset row for second driver's bidding):

```
{
  # driver_profile_vector
  'driver_id':[0],
  'timestamp':['19:15:00'],
  'location':[(10.7607705,106.6954122)],
  'route_history':['HCMC-Hanoi', 'Hanoi-HCMC', 'HCMC-Hanoi'],
  'cargo_weight_history':[100],
  'truck_type_history':['container','container'],
  'other_order_info_history':['...'],
  # order_vector
  'bidding_id':[1]
  'order_id':[1244],
  'order_route':['HCMC-Binh Duong'],
  'cargo_weight':[50],
  'truck_type':['cold container'],
  'other_order_info':['...']
}
```

In general terms, if driver A has made N biddings, the dataset will include N rows, with the nth row at time T being the horizontal concatenation of two vectors: vector (6) the order_vector relative to the order bid at bidding n and vector (7) the driver_profile_vector including vector (3) driver_stable_profile_vector combined with vector (2) driver_activity_history_vector up to time T-1.

The latter includes all `order_vectors` for biddings (0, 1, ... n-1). Each row is characterized by a unique timestamp and location.

Vectors a) `driver_stable_profile_vector` and b) `driver_activity_history_vector` are combined in the following way:

1. Categorical variables can be encoded using either of the following encoding:
 - a. One-hot-encoding: two one-hot-encoded vectors can be combined by summation
 - b. Integer encoding: each vector will be a list of integers. Two vectors can be thus combined by appending one list to the other

Which option to adopt will depend on the model input structure and will be discussed below

2. Numerical variables from two vectors may be combined in several different ways. For instance:

- a. One or more measure of central tendency of all the previous occurrences of the numerical variable in the driver's activity history. For instance, the mean and/or the median and/or the maximum and minimum and/or the standard deviation. One new feature for each statistic will be created

- b. Cluster-centroids of the clustered distribution of all the previous occurrences of the numerical variable in the driver's activity history. One new feature for each cluster will be created

For example, the *n*th row of driver A dataset may be expressed in the following vector 9 (Encoded dataset row for the driver's *n*th bidding):

```
{
  # driver_profile_vector
  'driver_id':[0],
  'timestamp':['15:45:00'],
  'location':[(10.5137705,105.4254122)],
  'route_history':[1,1,5,1,7,14,1,5,23],
  'cargo_weight_history_min':[21],
  'cargo_weight_history_max':[200],
  'cargo_weight_history_median':[107.75],
  'truck_type_history':[2,1,5,5,5,1,3,5,5],
  'other_order_info_history':['...'],
  # order_vector
  'bidding_id':[9]
  'order_id':[1301],
  'order_route':[1],
  'cargo_weight':[97],
  'truck_type':[2],
  'other_order_info':['...']
}
```

In this example, categorical variables have been integer encoded and numerical variables have been combined through minimum, maximum and mean. As mentioned above, numerical variables can be encoded in different ways, one of which is to create one feature for every statistic chosen. In this example, we created one feature for min, max and median. If $n = 1$, where we have no element in the driver's history, $\min(E) = \max(E) = \text{median}(E) = \text{empty list}$ and this is shown in example for vector 5.

If $n = 2$, where we have only one single element, called E , in the driver's history, $\min(E) = \max(E) = \text{median}(E) = E$.

If $n = 3$, where we have 2 elements, called $E1, E2$, in the driver's history, we can calculate min and max and $\text{Median}(E1, E2) = \text{Mean}(E1, E2)$.

Route_history and truck_type_history are categorical variables. Encoding through one-hot-encoding or integer-encoding is indifferent. If you encode with one-hot-encoding, you need to pass the variables into Dense Layers in the model architecture. If you use integer encoding, you can pass them through embedding layers. For this invention, we have adopted integer encoding $\Rightarrow$ embedding layer in our particular application.

The resulting dataset used for training the model is presented in Appendix 2.

As stated above, this dataset structure is unique as compared to the standard way to define a dataset for a recommendation problem. It is important to highlight that we do not only use the latest driver's profile to train the model, but each single driver's profile corresponding to each single driver's bidding. This is done for a series of reasons:

1. It provides multiple representations of the same entity - a driver (we will discuss the implications of this in more detail in a later section)

2. It provides information about the relationship between location and driver's preferences. For instance, a driver may be interested for an order to be picked up in a city only if the driver himself is now in that city.

3. It allows the algorithm to perform better in a cold start scenario (when recommendations need to be made for a new driver with no bidding history) and a warm start scenario (when recommendations need to be made for a driver with a short bidding history). In fact, the training dataset includes rows with empty (row 0) or short (row 1) driver_activity_history_vectors.

Another key feature useful to solve the cold/warm start problem is the inclusion of driver_stable_profile_vector information in the driver_activity_history_vector.

We used elements of content-based filtering due to the unstable nature of our items, the orders, which expire as soon as they are fulfilled by a driver.

In summary, we have created our dataset as a $((n * b) \times m)$ matrix where n is the number of drivers, b is the average number of biddings per driver, and m is the length of

the vector described above (driver_profile_vector + order_vector at time T). Below we will present additional data preprocessing steps taken on the aforementioned dataset:

- we used a mixture of methods to fill-in missing data, including K-nearest-neighbours imputation for the variable 'offer_price', mean imputation for the other numerical variables and mode imputation for categorical variables

- after the construction of the driver_profile_vector, categorical variables in the driver_profile_vector are encoded into integer lists to be inputted to embedding layers. As mentioned above, categorical variables are encoded through one-hot-encoding or integer encoding. After integer encoding, we need to use padding with replacement, which consists in the following: each categorical variable has been integer-encoded and transformed into an integer-list (as in the example in vector 9 above for the feature route_history). However, each row of the dataset (i.e. each driver's history at a particular point in time) may include a different number of integers for each categorical variable. For instance, for feature route_history, a driver with N biddings will have, at time T0 an empty list, while at T1 is a list of length 1, and at T(N-1) a list of length N-1. Thus we use padding with replacement to ensure that all the integer lists for one features have the same length. The maximum length we allowed for a feature is the minimum between 20 and the length of the longest list for that feature. When cutting a list longer than 20, we select the list elements referred to the most recent time in order to give priority the most recent driver activity over the oldest one.

- we standardized numerical variables using z-score standardization:

$$x' = (x - \text{mean}(X)) / \text{std}(X)$$

where x is one instance of a numerical variable, i.e. a distribution X , $\text{mean}(X)$ is the mean of X , $\text{std}(X)$ is the standard deviation of X , and x' is the standardized version of x

We reframed the recommendation problem as a similarity problem, where the input is two vectors (the driver vector and the order vector) and the output is the similarity score between the two vectors. Recommendations may then be based on:

- 1) a threshold on whether the order is or is not relevant to the driver,
- 2) a ranking by ranking the orders based on the similarity score, or
- 3) a combination of 1 and 2 above by ranking the orders that are above a certain similarity threshold based on their similarity score.

In practice, this formulation works in the two scenarios that a logistics company may face:

1. Given a driver and a list of orders, recommend the most suitable N orders for this driver.

2. Given an order and a list of drivers, recommend the most suitable N drivers for this order.

REEL module 320

After the resulting datasets are prepared by the pre-processing module 310 as described above, we feed the resulting datasets to a deep learning architecture to train a model which would be used for inference. In brief, before the REEL module 320 can make any prediction, a model needs to be trained using the datasets. This step is known as the training phase. After a model is trained during the training phase, the trained model would be used in the inference phase, where the trained model would then be used by the REEL module 320 to predict an outcome.

Training phase

We adopted a deep learning architecture that at training phase takes as input three vectors representing (a) the driver_profile_vector at a particular time point (anchor vector) (b) the order_vector representing the order that the driver bid for at that time point (positive vector) (c) the order_vector representing an order that the driver did not bid for at that time point (negative vector). Thus, we horizontally concatenate the vector shown in vectors 8 or 9 to a vector equivalent to those shown in vectors 1 and 6, a 'negative' order_vector which can be expressed in the following vector 10 (Input vector to the model).

```
{  
  # driver_profile_vector  
  'driver_id':[0],  
  'timestamp':['15:45:00'],  
  'location':[(10.5137705,105.4254122)],  
  'route_history':[1,1,5,1,7,14,1,5,23],  
  'cargo_weight_history_min':[21],  
  'cargo_weight_history_max':[200],  
  'cargo_weight_history_median':[107.75],  
  'truck_type_history':[2,1,5,5,5,1,3,5,5],  
  'other_order_info_history':['...'],  
  # positive order_vector  
  'bidding_id':[9],  
  'order_id':[1301],  
  'order_route':[1],  
  'cargo_weight':[97],  
  'truck_type':[2],  
  'other_order_info':['...']  
  # negative order_vector  
  'order_id':[1724],  
  'order_route':[73],  
  'cargo_weight':[450],  
  'truck_type':[21],  
  'other_order_info':['...']  
}
```

In short, the input vector to the model (vector 10) comprises a triplet of a) driver_profile_vector (anchor vector), b) bid order_vector (positive vector) and c) not bid order_vector (negative vector).

The dataset thus becomes a $((n*b) \times (h + o * 2))$ matrix, where h is the length of the driver_profile_vector, and o is the length of the order_vector, that is identical for the positive order and the negative order. Positive and negative orders, in fact, represent the same entity and are thus characterised by the same variables. It is possible to allow multiple negative orders per each positive order, so that the dataset will become a $((n*b*k) \times (h + o * 2))$ matrix, where k is the average number of negative examples per positive example. For instance, the above vector 10 represented the bidding history of driver 0 up to time point 9-1, horizontally concatenated with the drivers' 9th bidding (order_id = 1301) and a negative order example drawn with our negative sampling methodology (order_id = 1724). We may add another row represented in the below vector 11:

```
{
  # driver_profile_vector
  'driver_id':[0],
  'timestamp':['15:45:00'],
  'location':[(10.5137705,105.4254122)],
  'route_history':[1,2,5,1,7,14,1,5,23],
  'cargo_weight_history_min':[21],
  'cargo_weight_history_max':[200],
  'cargo_weight_history_median':[107.75],
  'truck_type_history':[2,1,5,5,5,1,3,5,5],
  'other_order_info_history':['...']
  # positive_order_vector
  'bidding_id':[9],
  'order_id':[1301],
  'order_route':[1],
  'cargo_weight':[97],
  'truck_type':[2],
  'other_order_info':['...']
  # negative_order_vector
  'order_id':[1836],
  'order_route':[29],
  'cargo_weight':[300],
  'truck_type':[7],
  'other_order_info':['...']
}
```

Here, we have the same information as in vector 10 for driver_profile_vector and positive_order_vector, while we have included a different negative_order_vector (order_id = 1836) to provide an additional negative example to the model. For each drivers' bidding, we may have N rows where N = number of chosen negative order examples for that driver. Each of these rows, as in the example above, will have identical driver_profile_vector and positive_order_vector, and different negative_order_vector.

For each unique driver_profile_vector - positive_order_vector pair, we included the following types of negative_order_vector:

- a. clicks
- b. no clicks
- 5 c. others

As stated above, a, b and c are only made of 'possible' orders, as 'impossible' orders are filtered out based on reachability and cargo size criteria.

The full list of model input features is presented in Appendix 3.

This input structure mirrors the anchor - positive - negative structure described by
10 Schroff and colleagues (2015) in their Siamese neural networks adopting a triplet loss function for face recognition (FaceNet). The authors of this paper stress the importance of having multiple examples of an anchor in the dataset. They achieve this by using different pictures of the same person. Because ultimately our anchor is a driver, we have achieved this by including multiple examples of each driver, each example representing a bidding
15 that is, the driver's profile up to a certain point in time. For each anchor (i.e. bidding) we included multiple types of negative examples based on our negative sampling methodology described above. The exact configuration of the Siamese neural networks implemented according to this invention is briefly described below. More importantly, the dataset that needs to be feed to the Siamese neural networks has to be configured accordingly by the
20 pre-processing module 310.

We adopted a Siamese neural network with triplet loss function. Figure 4 illustrates a simplified plot of the model architecture. Before describing the model architecture, we define three operations:

- concatenate: refers to concatenation of two vectors along the horizontal axis
- 25 - flatten: refers to the re-shaping of a 3-d tensor of shape (1, max_length, embedding_dimension) to obtain the shape (1, max_length * embedding_dimension). The tensor is the output of an embedding layer, where max_length is the maximum length of the input vector, and embedding_dimension is the chosen dimension of the embedding.
- "passing through" a layer: refers to the process of matrix multiplication between
30 the input of that layer and the weights of that layer, which is common in every neural network and will thus not be described in more details.

After describing these three operations, we describe the model architecture. Each set of positive input and negative output will be concatenated with corresponding history input. As shown in figure 4, the full input to the model is a dataset comprising history inputs
35 – i.e. the anchor - (history categorical input 421, history numerical input 411, positive order inputs – i.e. the positive example – (positive order categorical input 422, positive order

numerical input 412) and negative order inputs – i.e. the negative example - (negative order categorical input 423, negative order numerical input 413). Categorical inputs are passed through an embedding layer 424 and 425, while numerical inputs are passed through two dense 414 and 415 (i.e. fully connected) layers of 10 nodes (i.e. units) with relu activation function and with applied dropout to reduce overfitting. Weights are shared between positive and negative order inputs, while they are not shared with history inputs. Specifically, history categorical input 421 is passed through embedding layer 'history embedding' 424 to produce the flattened history_categorical_encoded vector 426; positive 422 and negative 423 order categorical inputs are passed through the same embedding layer 'order embedding' 425 to produce the flattened positive 427 and negative 428 order_categorical_encoded vectors respectively; history numerical input 411 is passed through dense layers history dense 414 to produce history_numerical_encoded vector 416; positive 412 and negative 413 order numerical inputs are passed through the same dense layer 'order dense' 415 to produce positive 417 and negative 418 order_numerical_encoded vectors respectively. Each pair of categorical_encoded – numerical_encoded for history, positive order and negative order is concatenated and passed through three dense layers 431 and 432 of 64, 32 and 16 units, with relu activation. The dense layers 431 and 432 are shared between positive and negative order vectors, but they are not shared with the history vector. In particular, history_categorical_encoded 426 and history_numerical_encoded 416 are concatenated and passed through 'history_dense' 431 to output history_encoded vector 433. Positive_order_categorical_encoded 427 and positive_order_numerical_encoded 417 are concatenated; negative_order_categorical_encoded 428 and negative_order_numerical_encoded 418 are also concatenated and passed through the same three dense layers 'order_dense' 432 to output positive_order_encoded 434 and negative_order_encoded 435 respectively. The three final resulting vectors from history_encoded vector 433, positive_order_encoded 434 and negative_order_encoded 435 are finally object of the loss function 440. After the loss function 440 is calculated, the model weights and biases are updated through backpropagation, and the process is repeated for a defined number of times (i.e. epochs – which is defined below). After the process is repeated for the defined number of times, the model is considered as trained and would be used for the inference phase. Please refer to figures 9-12 showing the full model architecture graph with the input to the model for history input 411, positive order input 412 and negative order input 413.

Loss Function

The network learns by optimizing the triplet loss function 440:

$$L(A, P, N) = \max(\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \alpha, 0)$$

Where $f(A)$, $f(P)$, $f(N)$ are the embedded representation of the anchor, positive and negative respectively, and α , is a margin. In this case, the anchor is the driver_embedding_vector (i.e. history_encoded 433 in Figure 4), the positive is the positive_order_embedding_vector (i.e. positive_order_encoded 434 in Figure 4) and the negative is the negative_order_embedding_vector (i.e. negative_order_encoded 435 in Figure 4). Instead of adopting the L2 norm to calculate distance as in the formula above, we used cosine distance. The best triplets for training are mined through a 'batch-all' online mining strategy, that works by excluding from the loss calculation those triplets where $L < 0$.

Model Parameters

- Embedding length is set through the formula:

$$\sqrt[4]{d + \beta}$$

Where d is the input size and β is a bias set to 2 in this case.

- Dense layers are 2 and have 10 units each, with relu activation function.
- Optimizer is stochastic gradient descent
- Loss bias parameter is set to 0.2
- The model was trained for 200 epochs

Response Time

Model performance in terms of response time is 0.00056 seconds for each driver or order.

After the model has been trained for 200 epochs, the model would be used for inference. The model is re-trained periodically at varying intervals, depending on changes in the market or changes in our dataset. One skilled in the art will recognise that there may be no specific frequency of re-training. Alternatively, a specific frequency of re-training may be implemented without departing from the invention. Further, incremental learning (i.e. automatic addition of new data as new data is collected) may be implemented without departing from the invention. One skilled in the art will recognise that it is beneficial to re-train the model periodically as new data about driver behaviour is collected and the exact frequency of re-training is left to those skilled in the art.

Inference phase

For inferences, a request must be sent with only 2 inputs: the pre-processed data for the driver and the pre-processed data for the order. The model will output the dissimilarity score (cosine distance) between the two inputs where lower scores indicate better match.

At inference phase, the model as trained during the training phase is used to make inferences given the input from the pre-processing module 310. That is, at inference phase we want to use the model as trained during the training phase in REEL module 320 to:

a. given an input order and an input driver, classify this order-driver pair as matching or not matching – we can use this classification to decide whether or not to recommend the input order to the input driver, i.e. to decide whether the input driver is suitable to carry the input order.

b. similarly, given a list of input orders and an input driver, rank the input orders by the degree to which they match with the input driver – we can display the top N orders as recommendation to the driver.

c. similarly, given a list of input drivers and an input order, rank the input drivers by the degree to which they match with the input order – we can send a notification about the input order (or call, or other method) to the top N drivers.

Figure 5 illustrates the functions of the inference model. The inference model takes two inputs: a driver_history_vector and an order_vector. The input structure of driver_history_vector and order_vector is identical to the input to the training model as described in REEL module 320. Namely, history_driver_vector is composed of history_categorical_input 521 (which corresponds to 421 in Figure 4) and history_numerical_input 511 (which corresponds to 411 in Figure 4). Order_vector is composed of order_categorical_input 522 (which correspond to 422 and 423 in Figure 4 and both have identical structure) and order_numerical_input 512 (which correspond to 412 and 413 in Figure 4 and both have identical structure).

The input passes through the weights that have been calculated in the trained model to produce the history encoded (i.e. the resulting vector from the driver_history_input) and the order_encoded (i.e. the resulting vector from the order_input) vectors. This means that the history_categorical_input 521 and the history_numerical_input 511 are passed through the embedding layer 424 and history dense layer 414 respectively before both are concatenated and passed through 'history_dense' 431 to output history encoded vector 533 (which corresponds to history encoded vector 433 in figure 4). Further, the order categorical input 522 and order numerical input 512 are passed through the embedding layer 425 and order dense layer 415 respectively before both are concatenated and passed through 'order_dense' 432 to output order encoded vector 534 (which corresponds to order encoded vector 434 and 435 in figure 4). With “passed through the weights”, we mean a process of matrix multiplications between each input and each weight matrix, that is standard in the inference phase of any neural network and will thus not be described further. The dissimilarity score (cosine distance) is calculated between the history encoded vector

533 and order encoded vector 534 and, as stated above, the resulting number can be used to either classify or rank the order-driver pairs in terms of how much they match with each other. As illustrated in figure 5, if the dissimilarity score (or cosine distance) between the history encoded vector 533 and order encoded vector 534 is below a threshold, the order and driver match and the system recommend the order to the driver. If the dissimilarity score (or cosine distance) between the history encoded vector 533 and order encoded vector 534 is not below the threshold, the order and driver do not match and the system do not recommend the order to the driver. The threshold as illustrated in figure 5 is 0.8 which is based on the distribution of the distance between a driver_embedding_vector and an order_embedding_vector for positive 610 vs negatives 620 examples as shown in figure 6. For the avoidance of doubt, dissimilarity score (or cosine distance, $\cos(\theta)$) is a measure of similarity between two non-zero vectors (i.e. history encoded vector 533 and order encoded vector 534) and can be expressed in the following expression,

$$\cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|}$$

Where A is history encoded vector 533 and B is order encoded vector 534, or vice versa.

Possible deployment

The model can be deployed on Amazon server t2. large. It can be used to send automatic personalized notifications to drivers. It can also be used to show a list of relevant orders to drivers on a dedicated window of the application. The deployment can take an order ID as input and runs the model through all certified drivers in system's database to return the most suitable drivers. Another deployment can take a driver ID as input and runs the model through all available orders in system's database to return the most suitable orders. Some basic filters can be applied to filter out irrelevant drivers/orders before running the model, which are:

- Filter out drivers that are too far from the order pickup point to be reached on time according to the pickup deadline
- Filter out drivers that cannot carry the goods due to truck type being too small or not appropriate for the cargo type.

Alternative embodiment to the Model

Below, we present an alternative version of the model described above. The problem is reframed from a similarity problem to a binary classification problem. The input is restructured from a triplet of history, positive_order and negative_order vector, to a pair of history_vector and order_vector, labelled by 1 = the order_vector represents a positive order or 0 = the order_vector represents a negative order. As a consequence, the model

architecture changes to that displayed in Figure 14. The loss changes to Binary Crossentropy:

$$Loss = -y * \log(y^\wedge) - (1 - y) * \log(1 - y^\wedge)$$

Where y is the ground truth, (i.e. label, that can be either 0 or 1) and $y^\wedge$ is the predicted score (i.e. predicted probability).

Experimental Results for Experiment 1

The raw and engineered features used for experiment 1 are the same as those presented in the Appendix 1 section below. We have applied the pre-processing steps described above and the resulting dataset used for training has the same features as presented in the Appendix 3 section below. The training and test datasets had the following characteristics:

Table 1. Training and Test set characteristics

Info	Training set	Test set
Number of rows*	225397	852
Number of unique drivers	2542	405
Number of unique orders	8270	852
Number of unique positive orders	19303	426
Average positive : negative order ratio	1:12	1:1

*number of unique triplets (driver_profile_vector, positive_order_vector, negative_order_vector). For the test set, we chose a 1:1 ratio between positive and negative orders.

Table 2. Proportion of different types of negatives in the Dataset

Label	Percentage
0	10
-1	10
-2	40
-3	27
-4	13

Train - Test Split methodology

We choose 20% of drivers at random to create the test set. We excluded all orders that appeared in the test set from the training set. Thus, both drivers and orders in the test set are unseen by the training set.

Results

To report model performance, we consider 'biddings' as positive examples, and all other orders, including 'opened and not bid' orders, as negatives. To classify positives and negatives, we have chosen a threshold based on the distribution of distances between a. Driver_embedding_vector and positive_order_embedding_vector and b. Driver_embedding_vector and negative_order_embedding_vector as registered in the training set.

Figure 6 illustrates the distribution of the distance between a driver_embedding_vector and an order_embedding_vector for positive 610 vs negatives 620 examples on the training set. From figure 6, we can see that a threshold of 0.8 would be ideal to correctly classify most examples. Hence, we have used 0.8 in the example as shown in figure 5. We can then create a confusion matrix where:

True Positives: positive orders (biddings) correctly classified as positives

True Negatives: negative orders (clicks and not opened orders) correctly classified as negatives

False Positives: positive orders incorrectly classified as negatives

False Negatives: negative orders incorrectly classified as positives

Below, the evaluation metrics used and their results on the test set are reported:

Table 3. Experiment 1 evaluation metrics

Metric	Description	Value
Accuracy	$(TP+TN)/(TP+TN+FP+FN)^*$	0.84
Precision	$TP / (TP+FP)$	0.76
Recall	$TP/(TP+FN)$	0.98
F1-score	$(2 * Recall * Precision) / (Recall + Precision)$	0.86
Average ranking metric	$(1 - (rank/(n-1))) / N^{**}$	0.87
Hit rate	1 if positive order in top 5 recommendations, else 0***	0.85

*where: TP = True positive; FP = False positive; TN = True negative; FN = False

negative

**where rank is the position of the positive order in the list of recommended orders, n is the number of recommended orders and N is the number of samples. On average, 12 orders were given for each driver_profile_vector, with only 1 being positive

***On average, 12 orders were given for each driver_profile_vector, with only 1 being positive

Figure 7 illustrates the 3 principal components obtained through t-SNE before training and figure 8 illustrates the 3 principal components obtained through t-SNE after training. From figures 7 and 8, we can see a better separation between classes in Figure 8 as compared to Figure 7. Specifically in figure 7, we can see that before training, there is

only one big cluster and there is no clear separation among classes. In contrast, figure 8 shows that after training, there are two separated clusters, one on top left and one on bottom right. This shows that there is better separation among classes after training. Figure 9 illustrates the distribution of the cosine distance between a driver_embedding_vector and an order_embedding_vector for positive examples 910 vs negatives 920 examples on the test set. Comparing figure 6 (showing the training set) and figure 9 (showing the test set), we can see a clear separation between positive and negative orders using the threshold (0.8) calculated on the training set. Post-hoc analysis revealed that almost all the misclassified examples were clicked orders.

Experimental Results for Experiment 2

The model is the same as that used in Experiment 1. We built an API that takes as input a driver ID and a list of order IDs, queries and pre-processes the necessary data, and outputs a ranked list of order IDs where the top means better match between driver and order. We sampled 150 drivers and 1000 orders at random from the database. These may or may not be included in the training set. For each driver, we sampled 2 random locations where the driver had been in the past, in order to mimic a real life situation where the current driver GPS location has to be inputted to the model. We fed the driver IDs and order IDs to the API and compared the top 10 recommendations for each driver with the bottom 10 recommendations of each driver at each of the two sampled locations.

Below are the results of this experiment

Table 4. Experiment 2 evaluation metrics

Metric	Description	Top 10 (recommended)	Bottom 10 (not recommended)
Route Accuracy	% of recommendations where order route is within the driver's habitual area *	70%	55%
Novelty	% of 'good recommendations' that contained a route never explored before by the driver**	81%	n/a
Within-driver Personalization	Average cosine distance within a driver's recommendations (min = 0. Max = 2)	0.71	n/a
Between-driver Personalization	Average cosine distance within each driver's	0.85	n/a

	recommendations (min = 0, max = 2)		
--	---------------------------------------	--	--

*the driver's habitual area comprises all the locations where the driver had been in the past to pick up, carry or deliver an order. It is calculated as the rectangle having each side passing through the southernmost, northernmost, easternmost and westernmost point on the map ever reached by the driver in a completed journey.

5 **we define a 'good' recommendation here as a recommendation that passed the 'route accuracy' criterion (i.e. fell within the driver's habitual area)

Both experiments suggest that the model in the REEL module 320 has been able to learn the drivers' preferences and thus identify suitable orders for each driver. This model is able to recommend interesting orders (biddings and clicks) with very high accuracy.

10 Indeed, if we considered clicks as positives, our model would reach nearly 100% accuracy in all evaluation metrics showed in Table 4.

The model is able to recommend new, interesting routes to the driver. The high novelty score suggests that the model has learned the behavioural pattern of 'similar' drivers and not merely to repeat the routes that a driver has explored in the past.

15 Recommending new routes is also an important advantage from a business perspective, because it will allow drivers to discover new, interesting routes and thus take more jobs. Finally, the positive personalization score suggests that the RS did not merely learn to recommend the most popular orders/routes.

The high accuracy of the model recommendations on the test set suggest the
20 goodness of our negative sampling methodology. The negative sampling methodology not only was able to provide appropriate negative examples for a recommender system in a semi-supervised learning situation, but also to help us interpret its results. The validity of our negative sample is also confirmed by the principal components plot before and after training, that clearly groups our previously identified classes in separate clusters.

25 Below table 5 illustrates the business impact metrics. We have compared a set of metrics between before and after the model was used in production. The baseline is the recommendation system used prior to the model implementation, which comprises a basic filter based on location and driver preferred routes.

Table 4. Business Impact Metrics

Metric	Description	Baseline	REEL	Percentage Increase
--------	-------------	----------	------	---------------------

Click rate	Percentage of notifications received by a driver that are subsequently clicked-on	0.15%	0.32%	113%
Bid rate	Percentage of notifications received by a driver that are subsequently bid	0.3%	1.26%	320%
Fulfillment rate	Percentage of orders processed through the recommender system that were fulfilled	3.94%	20.77%	427%

In short, the RS in accordance with this invention adopts a Siamese neural network with triplet loss function represents a unique application of this kind of model to the logistics business. The uniqueness of this RS lies in the following factors:

- 5 1. The use of advanced, state of the art algorithms to solve the freight matching problem in the logistics business
2. The use of this particular neural network architecture in its application to the logistics business. We took elements of the Siamese neural network with triplet loss function and customized it to accept our particular input made of orders and drivers.
- 10 Differently, the present network accepts as input two entities of different domains - drivers and orders - that thus do not share embedding weights. Further, we have used a combination of embedding and dense layers to form the final embedding vector for the input data.
3. The particular input of the model, represented by orders and drivers. Specifically,
- 15 items (orders) are not stable: they expire as soon as they are fulfilled by a driver. Moreover, orders and driver vector representations are time and location-dependent. The data pre-processing methodology as described above is able to transform this kind of data to create a suitable model input.
4. The high accuracy of recommendations in the absence of an explicit rating
- 20 system.

To conclude, the RS in accordance with this invention is an effective method to solve the freight matching problem in the logistics business. The algorithm is an innovative application of a Siamese network architecture with triplet loss function to the logistics business, having drivers and orders as input data. Both experimental and business results

25 support the efficacy of this approach. The application of this RS to the development of an intelligent, automated notification system will reduce costs, accelerate work and increase the number of jobs available for drivers and the fulfilment rate of orders posted by goods owners.

The negative sampling methodology relying on knowledge-based and statistical techniques provide a better understanding of the data distribution and model results, and obtain a reliable negative sample for a positive-unlabelled learning problem. The success of our negative sampling methodology is clear from outstanding model results. The methodology not only allowed us to select a suitable input to the model, but also to better understand its output.

Appendix 1. Features

Feature	Engineered	Data Type (unit)	Description
route_id	Yes: combination of pickup_area_id and dropoff_area_id	int	Unique identifier of the route from order pickup point to order dropoff point
gps_area_id	Yes: closest driver's gps location at the time of bidding, up to 15 minutes before or after bidding. Mapped to the closest area_id	int	Unique identifier of the area where the driver was located at the time of bidding
gps_to_pickup_id	Yes: route between current driver gps and pickup point	int	Unique identifier of the route between the current driver's gps location at time of bidding and the pickup point
truck_model_id	No	int	Unique identifier of truck model
is_whole_truck	No	Boolean	If true, whole truck was requested in the order (Full-Truck-Load)
cargo_weight	No	Float (KG)	Weight of the cargo to be carried
pickup_on_weekend /dropoff_on_weekend	Yes: extracted from pickup_datetime/dropoff_datetime features	Boolean	If true, the pickup/dropoff date is on a weekend (Saturday or Sunday)
pickup_hour/dropoff_hour	Yes: extracted from pickup_datetime/dropoff_datetime features	int	The hour of the day at which the pickup/dropoff should take place

time_to_order	Yes: pickup_datetime - bidding_datetime	Float (seconds)	The number of seconds from the moment the driver bid for an order and the pickup deadline of that order
route_len	Yes: Vincenty distance between pickup and dropoff points	Float (km)	The Vincenty distance between pickup and dropoff point
dist_to_pickup	Yes: Vincenty distance between driver GPS location at time of bidding, and pickup point	Float (km)	The Vincenty distance between driver GPS location at time of bidding, and pickup point
speed_needed	Yes: dist_to_pickup/route_len	Float (km/h)	The speed required to arrive on time to the pickup point from the current driver location (i.e. where the driver made the bidding)
offer_price	No	Float (VND)	Price offered by the goods owner to carry out the order
offer_price_kg_km	Yes: offer_price/cargo_weight/route_len	Float (VND)	Price per kg per km offered by the goods owner to carry out the order
truck_weight	No	Float (Kg)	The weight that the truck can carry

Appendix 2. Dataset features after pre-processing

For the feature description, please see Appendix 1. Each feature in Appendix 1 is included here for both for the driver (prefixed as 'history_' to indicate that the feature belongs to the driver's activity history) and the order (prefixed as 'order_')

5

'history_route_id'	'history_truck_model_id'	'history_gps_to_pickup_id'
'history_pickup_on_weekend'	'history_pickup_hour'	'History_is_whole_truck'
'history_is_company'	'history_final_offer_price_kg_km'	'History_final_offer_price_kg_km_max'
'history_final_offer_price_kg_km_min'	'history_cargo_weight'	'history_cargo_weight_max'

'history_cargo_weight_min'	'history_route_len'	'history_route_len_max'
'history_route_len_min'	'history_dist_to_pickup'	'history_dist_to_pickup_max'
'history_dist_to_pickup_min'	'history_time_to_order'	'history_time_to_order_max'
'history_time_to_order_min'	'history_speed_needed'	'history_speed_needed_max'
'history_speed_needed_min'	'history_truck_weight', 'order_route_id'	'order_truck_model_id'
'order_gps_area_id'	'order_pickup_on_weekend'	'order_pickup_hour'
'order_is_whole_truck'	'order_final_offer_price_kg_km'	'order_cargo_weight'
'order_route_len'	'order_dist_to_pickup'	'order_time_to_order'
'order_speed_needed'	'order_id'	'created_at'
'history_id'	'driver_id'	

The last 4 features, 'order_id', 'created_at', 'history_id', 'driver_id' are not used for training and are presented here only to better illustrate the identity in a database row.

Appendix 3. Dataset features as model input

- For the feature description, please see Appendix 1. Each feature in Appendix 1 is included here for the driver (prefixed as 'history_' to indicate that the feature belongs to the driver's activity history), for the positive order example (prefixed as 'order_') and for the negative order example (prefixed as 'n_order').

'history_route_id'	'history_truck_model_id'	'history_gps_to_pickup_id'
'history_pickup_on_weekend'	'history_pickup_hour'	'history_is_whole_truck'
'history_is_company'	'history_final_offer_price_kg_km'	'History_final_offer_price_kg_km_max'
'history_final_offer_price_kg_km_min'	'history_cargo_weight'	'history_cargo_weight_max'
'history_cargo_weight_min'	'history_route_len'	'history_route_len_max'
'history_route_len_min'	'history_dist_to_pickup'	'history_dist_to_pickup_max'
'history_dist_to_pickup_min'	'history_time_to_order'	'history_time_to_order_max'
'history_time_to_order_min'	'history_speed_needed'	'history_speed_needed_max'
'history_speed_needed_min'	'history_truck_weight',	'order_route_id'
'order_truck_model_id'	'order_gps_area_id'	'order_pickup_on_weekend'
'order_pickup_hour'	'order_is_whole_truck'	'order_final_offer_price_kg_km',
'order_cargo_weight'	'order_route_len'	'order_dist_to_pickup'

'order_time_to_order'	'order_speed_needed'	'n_order_truck_model_id'
'n_order_gps_area_id'	'n_order_pickup_on_week end'	'n_order_pickup_hour'
'n_order_is_whole_truck'	'n_order_offer_price_kg_k m'	'n_order_cargo_weight'
'n_order_route_len'	'n_order_dist_to_pickup'	'n_order_time_to_order'
'n_order_speed_needed'		

Appendix 3. Model graph

See figures 10-13.

5 The above is a description of exemplary embodiments of a system and method in accordance with this disclosure. It is foreseeable that those skilled in the art can and will design alternative systems and methods based on this disclosure.

Claims

1. A logistic recommender system for automatically matching freight service to an order in the logistics business, the logistic recommender system comprising:

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

receive a plurality of driver and order data;

pre-process the received plurality of driver and order data and translate the plurality of driver and order data to form a training dataset expressed as a matrix of shape, $(n * b) \times m$, where

10 n is the number of drivers,

b is the average number of biddings per driver, and

m is the length of a vector comprising the order_vector and the driver_profile_vector at time T ; and

15 train a model of a Siamese neural network with a triplet loss function based on the training dataset.

2. The logistic recommender system according to claim 1 wherein the instruction to pre-process the received plurality of driver and order data and translate the plurality of driver and order data to form the training dataset comprises instructions to:

20 generate an order_vector for each of the plurality of driver and order data, the order_vector representing a characteristic of an order that the driver has bid for bidding N at time T and containing order number, route, cargo weight, truck model type, price, pickup and dropoff time;

25 generate a driver_activity_history_vector for each driver, the driver_activity_history_vector representing activity of the driver up to time $T-1$ when the driver made the bidding N and containing driver identification number, time stamp, location, route history, cargo weight history, truck model type history, price history, pickup history and dropoff time history;

30 generate a driver_stable_profile_vector for each driver, the stable_driver_profile_vector includes driver identification, truck type and preferred routes;

combine the driver_activity_history_vector and the driver_stable_profile_vector to form a driver profile vector; and

35 combine the order_vector and the driver profile vector to form the $(n * b)$ rows in the matrix.

3. The logistic recommender system according to claim 2 wherein the instruction to combine the driver_activity_history_vector and the driver_stable_profile_vector to form the driver profile vector further comprises instructions to:

encode categorical variables in the driver profile vector into integer lists ; and

5 pad with replacement the integer lists to align the length of each integer list to a maximum length for that integer list.

4. The logistic recommender system according to claim 3 wherein the instruction to encode categorical features into integer lists is one of one-hot-encoding and integer
10 encoding.

5. The logistic recommender system according to claim 3 wherein the instruction to combine the driver_activity_history_vector and the driver_stable_profile_vector to form the driver profile vector further comprises instructions to:

15 standardise numerical variables using z-score standardization which can be expressed in the following expression:

$$x' = (x - \text{mean}(X)) / \text{std}(X)$$

where x is one instance of a numerical variable, mean(X) is the mean of X, std(X) is the standard deviation of X, and x' is the standardized version of x.

20 6. The logistic recommender system according to claim 1 wherein the instruction to train the model of the Siamese neural network with the triplet loss function based on the training dataset comprises instructions to:

obtain the driver_profile_vector at a particular time point as an anchor;

25 obtain one of the order_vector representing the order that the driver bid for at that time point as a positive_order_vector; and

obtain one of the order_vector representing the order that the driver did not bid for at that time point as a negative_order_vector.

30 7. The logistic recommender system according to claim 6 wherein the instruction to train the model of the Siamese neural network with the triplet loss function based on the training dataset further comprises instructions to:

feed the anchor, positive_order_vector and negative_order_vector to the Siamese neural network where two vectors along a horizontal axis are concatenated and thereafter
35 flattened to a required length before applying the triplet loss function which can be expressed in the following expression,

$$L(A, P, N) = \max(\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \alpha, 0)$$

where $f(A)$, $f(P)$, $f(N)$ are representation of the flattened anchor vector, positive vector and negative vector respectively, and α is a margin.

- 5 8. The logistic recommender system according to claim 7 wherein the instruction to train the model of the Siamese neural network with the triplet loss function based on the training dataset further comprises instructions to:
 repeat the training of the model for 200 epochs to obtained a trained model.
- 10 9. The logistic recommender system according to claim 8 further comprising instructions to:
 receive a driver_vector and an order_vector;
 feed the driver_vector and the order_vector to the trained model to output a driver encoded vector and an order encoded vector;
 15 calculate a cosine distance between the driver encoded vector and the order encoded vector; and
 classify a match in response to the cosine distance being less than 0.8.
- 20 10. The logistic recommender system according to claim 8 further comprising instructions to:
 receive a driver_vector and a plurality of order_vectors;
 feed the driver_vector and each of the plurality of the order_vectors to the trained model to output a driver encoded vector and a plurality of order encoded vectors;
 calculate a cosine distance between the driver encoded vector and each of the
 25 plurality of the order encoded vectors;
 recommend the order encoded vectors with cosine distance being less than 0.8.
- 30 11. A logistic recommender method for automatically matching freight service to an order in the logistics business, the logistic recommender method comprising:
 receiving a plurality of driver and order data;
 pre-processing the received plurality of driver and order data and translate the plurality of driver and order data to form a training dataset expressed as a matrix of shape, $(n * b) \times m$, where, n is the number of drivers, b is the average number of biddings per driver, and m is the length of a vector comprising the order_vector and the
 35 driver_profile_vector at time T ; and

training a model of a Siamese neural network with a triplet loss function based on the training dataset.

12. The logistic recommender method according to claim 11 wherein the step of pre-processing the received plurality of driver and order data and translate the plurality of driver and order data to form the training dataset comprises:

generating an order_vector for each of the plurality of driver and order data, the order_vector representing a characteristic of an order that the driver has bid for bidding N at time T and containing order number, route, cargo weight, truck model type, price, pickup and dropoff time;

generating a driver_activity_history_vector for each driver, the driver_activity_history_vector representing activity of the driver up to time T-1 when the driver made the bidding N and containing driver identification number, time stamp, location, route history, cargo weight history, truck model type history, price history, pickup history and dropoff time history;

generating a driver_stable_profile_vector for each driver, the stable_driver_profile_vector includes driver identification, truck type and preferred routes;

combining the driver_activity_history_vector and the driver_stable_profile_vector to form a driver profile vector; and

combining the order_vector and the driver profile vector to form the (n * b) rows in the matrix.

13. The logistic recommender method according to claim 12 wherein the step of combining the driver_activity_history_vector and the driver_stable_profile_vector to form the driver profile vector further comprises:

encoding categorical variables in the driver profile vector into integer lists ; and

padding with replacement the integer lists to align the length of each integer list to a maximum length for that integer list.

14. The logistic recommender method according to claim 13 wherein the step of encoding categorical features into integer lists is one of one-hot-encoding and integer encoding.

15. The logistic recommender method according to claim 13 wherein the step of combining the driver_activity_history_vector and the driver_stable_profile_vector to form the driver profile vector further comprises:

standardising numerical variables using z-score standardization which can be expressed in the following expression:

$$x' = (x - \text{mean}(X)) / \text{std}(X)$$

where x is one instance of a numerical variable, $\text{mean}(X)$ is the mean of X , $\text{std}(X)$ is the standard deviation of X , and x' is the standardized version of x .

16. The logistic recommender method according to claim 11 wherein the step of training the model of the Siamese neural network with the triplet loss function based on the training dataset comprises:

- obtaining the driver_profile_vector at a particular time point as an anchor;
- obtaining one of the order_vector representing the order that the driver bid for at that time point as a positive_order_vector; and
- obtaining one of the order_vector representing the order that the driver did not bid for at that time point as a negative_order_vector.

17. The logistic recommender method according to claim 16 wherein the step of training the model of the Siamese neural network with the triplet loss function based on the training dataset further comprises:

feeding the anchor, positive_order_vector and negative_order_vector to the Siamese neural network where two vectors along a horizontal axis are concatenated and thereafter flattened to a required length before applying the triplet loss function which can be expressed in the following expression,

$$L(A, P, N) = \max(\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \alpha, 0)$$

where $f(A)$, $f(P)$, $f(N)$ are representation of the flattened anchor vector, positive vector and negative vector respectively, and α is a margin.

18. The logistic recommender method according to claim 17 wherein the step of training the model of the Siamese neural network with the triplet loss function based on the training dataset further comprises:

- repeating the training of the model for 200 epochs to obtained a trained model.

19. The logistic recommender method according to claim 18 further comprising:

- receiving a driver_vector and an order_vector;
- feeding the driver_vector and the order_vector to the trained model to output a driver encoded vector and an order encoded vector;

calculating a cosine distance between the driver encoded vector and the order encoded vector; and

classifying a match in response to the cosine distance being less than 0.8.

- 5 20. The logistic recommender method according to claim 18 further comprising:
receiving a driver_vector and a plurality of order_vectors;
feeding the driver_vector and each of the plurality of the order_vectors to the
trained model to output a driver encoded vector and a plurality of order encoded vectors;
calculating a cosine distance between the driver encoded vector and each of the
10 plurality of the order encoded vectors;
recommending the order encoded vectors with cosine distance being less than
0.8.

15