December 07, 2017

Configuring Alarm System Based on Time to Arrive at Appointment

Nick Felker

Recommended Citation

Felker, Nick, "Configuring Alarm System Based on Time to Arrive at Appointment", Technical Disclosure Commons, (December 07, 2017)
http://www.tdcommons.org/dpubs_series/890

Configuring Alarm System Based on Time to Arrive at Appointment

Author: Nick Felker

Overview

Generally, the present disclosure is directed to setting an alarm to alert a user based on an appointment. In particular, in some implementations, the systems and methods of the present disclosure can include or otherwise leverage one or more machine-learned models to predict a time to alert a user based on a user's schedule and location.

Example Figures

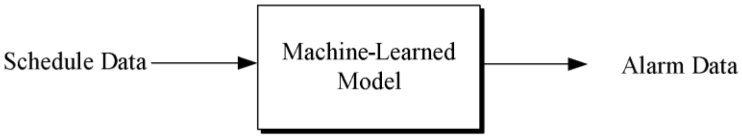

Figure 1

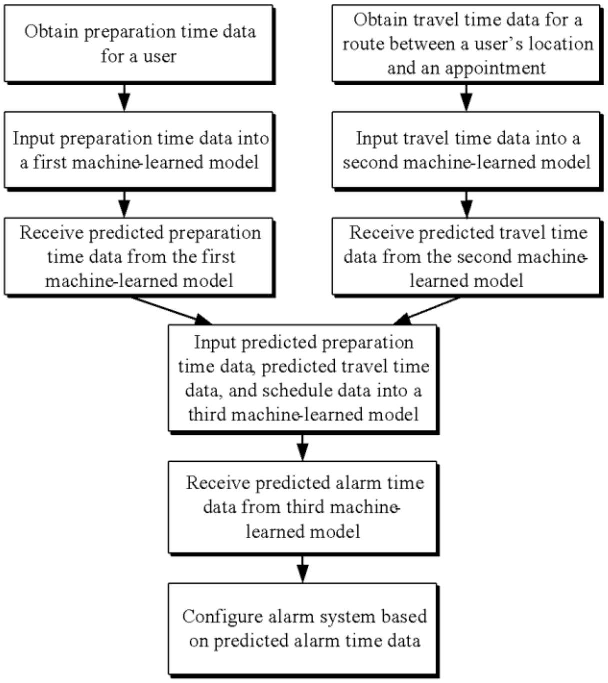

Figure 6

Introduction

When setting an alarm system to alert oneself for an appointment (e.g. meeting, school, work, or other scheduled appointment), many individuals set the alarm before the time of the appointment to allow time for preparation and/or commuting. Some individuals, however, incorrectly estimate how much time a commute will take and/or how long to allow themselves to prepare. This can result in individuals wasting time by arriving too early for the appointment or being late to the appointment.

Further, for an individual that has appointments at different times on different days (e.g., start work at 8am on Mondays but 9:30am on Tuesdays), the individual may be unable to set a single recurring alarm for the appropriate time and may forget and/or get frustrated with having to set different alarms for different days of the week.

What is needed is a system to automatically set an alarm system based on an individual's appointment to allow the individual sufficient time between the alarm and the appointment to prepare and/or commute.

Summary

Generally, the present disclosure is directed to setting an alarm to alert a user based on an appointment. In particular, in some implementations, the systems and methods of the present disclosure can include or otherwise leverage one or more machine-learned models to predict a time to alert a user based on a user's schedule and location.

A computing system can configure (e.g. set) an alarm system to alert a user at a time which allows sufficient time between the alarm and an appointment for the user to arrive at the appointment. The computing system and alarm system can be part of a larger system (e.g. a mobile device such as a mobile phone, tablet, laptop computer, etc.) or may be separate (e.g. physically separate) systems. In particular, the computing system can employ a first machine-learned model to predict the length of time that the user needs to prepare for the appointment. Additionally and/or alternatively, the computing system can employ a second machine-learned model to predict travel time. Additionally and/or alternatively, the computing system can employ a third machine-learned model to predict a time to alert the user that allows the user a sufficient amount of time to prepare for and/or commute to the appointment. Additionally and/or alternatively, the computing system can include a single machine-learned model that performs each of the tasks assigned to the three machine-learned models described above.

In some example implementations of the present disclosure, a first machine-learned model can predict a length of time that a user needs to prepare for an appointment based on location data. For instance, the location data may comprise historical location data illustrating a location of the user with respect to time. The location of the user can be determined from one or more of any known methods in the art, including but not limited to satellite positioning systems (e.g. GPS, GLONASS, etc.), land-based positioning systems, and/or connection-based positioning (e.g. network connectivity, Wi-Fi triangulation, cell tower triangulation, etc.). In some embodiments, the first machine-learned model may provide a prediction based at least in part on a time, such as day, week, month, season, year, day of week, time of day, etc. The time may be a digital representation of time, e.g. a timestamp, and may be sourced from an external location (e.g. the Internet), or from a clock within a system (e.g. within the computing system and/or the alarm system).

Based on the location data, the first machine-learned model can predict a length of time that a user needs to prepare for an appointment based on a length of time between the time that the user is alerted by an alarm and the time that the user departs in response to the alert. For example, if the alarm wakes the user up in the morning, the first machine-learned model may predict the length of time that the user needs to prepare based on previously collected data illustrating the time when the user is woken up by an alarm and the time when the user leaves his or her residence. The alert time and response time can be determined in various manners. Continuing with the above example, the time when the user is woken up by an alarm may be determined from an alarm application on the computing system or on another computing system. Additionally, the time when the user leaves his or her residence may be determined by a change in position, such as by a change in location (e.g. moving from within an area known to be the user's residence to a different area, a change in generalized location such as street, city, ZIP code, etc.) an occurrence of a velocity associated with driving a vehicle (e.g. if a device detects the user is travelling at a speed above average human walking speed), a change in network connectivity (e.g. a user disconnects from his or her home internet connection), a local positioning system (e.g. a sensor on the user's door), a user arming a security system, or other suitable methods.

The preparation time predicted by the first machine-learned model may be specified for a specific time, such as a specific day, week, month, season, year, day of week, week of year, etc. For instance, the model may predict that the user takes a different length of time to prepare on a certain day of the week (e.g. the user may have to start laundry before work on Monday resulting in a longer preparation time than on Tuesday through Friday) or during a certain season (e.g. the user may take longer to put on multiple layers of clothing during autumn or winter resulting in a longer preparation time than during spring or summer). Additionally and/or alternatively, the model may predict that the user takes more time to prepare for a holiday or event (e.g. for an event which consistently falls on the same day (e.g. a New Year's party held on December 31st) or day of week (e.g. a New Year's party held on the last Friday of December)).

In some embodiments, the user may manually provide a length of time the user needs to prepare (e.g. in addition to or alternatively to the first machine-learned model). For instance, the user may be provided with a prompt asking how much time the user prefers to prepare. This may be used in replacement of the first machine-learned model or in addition to (e.g., if the user specifies more time than usual, the first machine-learned model may predict additional time is necessary).

The computing system may additionally or alternatively employ a second machine-learned model to predict travel time. The second machine-learned model can predict travel time based on traffic data. For instance, the second machine-learned model can be trained based on historical traffic data to learn magnitudes and trends of traffic for one or more locations. The magnitudes and trends may be associated with a time, such as day, week, month, season, year, day of week, time of day, etc. The magnitudes and trends may be associated with additional factors, e.g. weather, holidays, events, etc. The traffic data can be associated with vehicle routes, pedestrian routes, public transit, airlines, railways, or other suitable forms of transportation. In some instances, the traffic data can be accessed in real-time to reflect current traffic conditions.

For example, the second machine-learned model may predict that vehicles traveling through a road will experience ten minutes of delay (e.g. compared to an ideal vehicle travelling the speed limit through the road) at a certain time based on historical traffic data indicating that the vehicles that travel through the road at a related time (e.g. same day of week, week, month, same day in a different year, same week in a different year, etc.) experience ten minutes of delay. In another example, the second machine-learned model may predict that a vehicle will travel ten miles per hour down a portion of a road (e.g. at a certain time) based on historical traffic data indicating that vehicles typically travel ten miles per hour down the portion of the road (e.g. at a time related to the certain time). In some embodiments, the second machine-learned model may directly predict travel time for a portion of a road. For instance, the second machine-learned model may predict that a vehicle will take ten minutes to traverse a portion of a road (e.g. at a certain time) based on historical traffic data indicating that vehicles typically take ten minutes to traverse the portion of the road (e.g. at a time related to the certain time). In some embodiments, predicted delay time or travel speed may be translated into travel time (e.g. by comparing delay time to ideal travel time, travel speed to distance, etc.).

The second machine-learned model can predict travel time for an entire route. For instance, the second machine-learned model can chain together travel times for a plurality of portions of a route. For example, if a user travels from point A to point B through road A with a predicted travel time of five minutes, road B with a predicted travel time of five minutes, and road C with a predicted travel time of ten minutes, the predicted travel time from point A to point

B may be predicted to be twenty minutes. In some embodiments, the second machine-learned model can base the predicted travel time on additional travel factors such as intersections, left turns, right turns, construction, weather, accidents, etc.

The third machine-learned model can predict a time to alert a user that allows the user a sufficient amount of time to prepare for and/or commute to an appointment based on the predictions from the first and/or second machine-learned models and additional input data. For instance, the input data can comprise the location of the user. The location of the user can be determined from one or more of any known methods in the art, including but not limited to satellite positioning systems (e.g. GPS, GLONASS, etc.), land-based positioning systems, and/or connection-based positioning (e.g. network connectivity, cell tower triangulation, etc.). The input data can additionally comprise appointment characteristics such as the location and/or time of the appointment. The appointment characteristics may be input into the computing system and/or may be received from an application (e.g. a calendar application) on the computing system or another computing system. The location of the appointment can be determined in one or more of various methods including “dropping a pin” on a digital map (e.g. in a satellite positioning application), selecting from a pre-populated list, location history of the user, suggested by a computing system, or other suitable methods for determining location. The time of the appointment can be determined by numerical input (e.g. by entering a time via keyboard or other means) or graphical input (e.g. selecting a time from a dropdown menu or on a virtual clock). The input data can additionally comprise temporal data such as day, week, month, season, year, day of week, week of month, etc. The temporal data may be a digital representation of time (e.g.

a timestamp). Based on the input data, the third machine-learned model can output a predicted time to alert the user about an appointment to allow sufficient time for the user to prepare for and/or commute to the appointment. The prediction from the third machine-learned model may be used, for example, to configure an alarm system. The prediction from the third machine-learned model may include a buffer, for example a fifteen minute buffer. The buffer may be predicted for the user (e.g., based on user "snooze" patterns in which the user delays the alarm for a short period of time).

For example, the first machine-learned model may predict that the time between when the user is alerted about an appointment and the time when the user departs for the appointment will be thirty minutes. Additionally and/or alternatively, the user may have otherwise specified a thirty minute preparation time. Continuing with the above example, the second machine-learned model may predict that the travel time from the user's current location to the location of the appointment will be thirty minutes. The third machine-learned model may additionally include a fifteen minute buffer. If the appointment time is at 8:00 AM, the model may thus predict an alarm time of 6:45 AM.

The one or more machine-learned models described herein are discussed with respect to a plurality of separate machine-learned models, i.e. a "first machine-learned model", "second machine-learned model", etc. The terms "first", "second", "third", and so forth are used herein for distinction in function and do not necessarily correspond to an ordering, temporal or otherwise, of the machine-learned models. Additionally, the functions of the one or more machine-learned models can be incorporated into a single machine-learned model, or may be separate machine-learned models collectively referred to as a machine-learned model, i.e. separate machine-learned models within a larger machine-learned model.

Thus, in some implementations, a single machine-learned model can be trained based on data descriptive of user habits such as typical preparation time(s). The single model can receive input data descriptive of the user's current location and a location and time associated with an appointment. The single model can provide a predicted time for which an alarm should be set for the appointment.

Further to the descriptions above, a user may be provided with controls allowing the user to make an election as to both if and when systems, programs or features described herein may enable collection of user information (e.g., information about a user's calendar, alarm settings, current location, location history, or preparation time), and if the user is sent content or communications from a server. In addition, certain data may be treated in one or more ways before it is stored or used, so that personally identifiable information is removed. For example, a user's identity may be treated so that no personally identifiable information can be determined for the user, or a user's geographic location may be generalized where location information is obtained (such as to a city, ZIP code, or state level), so that a particular location of a user cannot be determined. Thus, the user may have control over what information is collected about the user, how that information is used, and what information is provided to the user.

Thus, a computing system can configure an alarm system to alert a user to allow the user sufficient time to arrive at an appointment. This can save the user the hassle of manually configuring an alarm, can prevent the user from being too early or late, and can respond to changes in the user’s schedule. For example, if the user has a different schedule from day to day, the system can adjust the alarm from day to day, thus saving the user the trouble of adjusting the alarm manually.

Detailed Description

As described above, the present disclosure is directed to setting an alarm to alert a user based on an appointment. In particular, in some implementations, the systems and methods of the present disclosure can include or otherwise leverage one or more machine-learned models to predict a time to alert a user based on a user’s schedule and location.

Figure 1 depicts a block diagram of an example machine-learned model according to example implementations of the present disclosure. As illustrated in Figure 1, in some implementations, the machine-learned model is trained to receive input data of one or more types and, in response, provide output data of one or more types. Thus, Figure 1 illustrates the machine-learned model performing inference.

In some implementations, the input data can include one or more features that are associated with an instance or an example. In some implementations, the one or more features associated with the instance or example can be organized into a feature vector. In some implementations, the output data can include one or more predictions. Predictions can also be referred to as inferences. Thus, given features associated with a particular instance, the machine-learned model can output a prediction for such instance based on the features.

The machine-learned model can be or include one or more of various different types of machine-learned models. In particular, in some implementations, the machine-learned model can perform classification, regression, clustering, anomaly detection, recommendation generation, and/or other tasks.

In some implementations, the machine-learned model can perform various types of classification based on the input data. For example, the machine-learned model can perform binary classification or multiclass classification. In binary classification, the output data can include a classification of the input data into one of two different classes. In multiclass classification, the output data can include a classification of the input data into one (or more) of more than two classes. The classifications can be single label or multi-label.

In some implementations, the machine-learned model can perform discrete categorical classification in which the input data is simply classified into one or more classes or categories.

In some implementations, the machine-learned model can perform classification in which the machine-learned model provides, for each of one or more classes, a numerical value descriptive of a degree to which it is believed that the input data should be classified into the corresponding class. In some instances, the numerical values provided by the machine-learned model can be referred to as “confidence scores” that are indicative of a respective confidence associated with classification of the input into the respective class. In some implementations, the confidence scores can be compared to one or more thresholds to render a discrete categorical prediction. In some implementations, only a certain number of classes (e.g., one) with the relatively largest confidence scores can be selected to render a discrete categorical prediction.

In some implementations, the machine-learned model can provide a probabilistic classification. For example, the machine-learned model can be able to predict, given a sample input, a probability distribution over a set of classes. Thus, rather than outputting only the most likely class to which the sample input should belong, the machine-learned model can output, for each class, a probability that the sample input belongs to such class. In some implementations, the probability distribution over all possible classes can sum to one. In some implementations, a softmax function or layer can be used to squash a set of real values respectively associated with the possible classes to a set of real values in the range (0, 1) that sum to one.

In some implementations, the probabilities provided by the probability distribution can be compared to one or more thresholds to render a discrete categorical prediction. In some implementations, only a certain number of classes (e.g., one) with the relatively largest predicted probability can be selected to render a discrete categorical prediction.

In some implementations in which the machine-learned model performs classification, the machine-learned model can be trained using supervised learning techniques. For example, the machine-learned model can be trained on a training dataset that includes training examples labeled as belonging (or not belonging) to one or more classes. Further details regarding supervised training techniques are provided below.

In some implementations, the machine-learned model can perform regression to provide output data in the form of a continuous numeric value. The continuous numeric value can correspond to any number of different metrics or numeric representations, including, for example, currency values, scores, or other numeric representations. As examples, the machine-learned model can perform linear regression, polynomial regression, or nonlinear regression. As examples, the machine-learned model can perform simple regression or multiple regression. As described above, in some implementations, a softmax function or layer can be used to squash a set of real values respectively associated with a two or more possible classes to a set of real values in the range (0, 1) that sum to one.

In some implementations, the machine-learned model can perform various types of clustering. For example, the machine-learned model can identify one or more previously-defined clusters to which the input data most likely corresponds. As another example, the machine-learned model can identify one or more clusters within the input data. That is, in instances in which the input data includes multiple objects, documents, or other entities, the machine-learned model can sort the multiple entities included in the input data into a number of clusters. In some implementations in which the machine-learned model performs clustering, the machine-learned model can be trained using unsupervised learning techniques.

In some implementations, the machine-learned model can perform anomaly detection or outlier detection. For example, the machine-learned model can identify input data that does not conform to an expected pattern or other characteristic (e.g., as previously observed from previous input data). As examples, the anomaly detection can be used for fraud detection or system failure detection.

In some implementations, the machine-learned model can provide output data in the form of one or more recommendations. For example, the machine-learned model can be included in a recommendation system or engine. As an example, given input data that describes previous outcomes for certain entities (e.g., a score, ranking, or rating indicative of an amount of success or enjoyment), the machine-learned model can output a suggestion or recommendation of one or more additional entities that, based on the previous outcomes, are expected to have a desired outcome (e.g., elicit a score, ranking, or rating indicative of success or enjoyment). As one example, given input data descriptive of a number of products purchased or rated highly by a user, a recommendation system can output a suggestion or recommendation of an additional product that the user might enjoy or wish to purchase.

In some implementations, the machine-learned model can act as an agent within an environment. For example, the machine-learned model can be trained using reinforcement learning, which will be discussed in further detail below.

In some implementations, the machine-learned model can be a parametric model while, in other implementations, the machine-learned model can be a non-parametric model. In some implementations, the machine-learned model can be a linear model while, in other implementations, the machine-learned model can be a non-linear model.

As described above, the machine-learned model can be or include one or more of various different types of machine-learned models. Examples of such different types of machine-learned models are provided below for illustration. One or more of the example models described below can be used (e.g., combined) to provide the output data in response to the input data. Additional models beyond the example models provided below can be used as well.

In some implementations, the machine-learned model can be or include one or more classifier models such as, for example, linear classification models; quadratic classification models; etc.

In some implementations, the machine-learned model can be or include one or more regression models such as, for example, simple linear regression models; multiple linear regression models; logistic regression models; stepwise regression models; multivariate adaptive regression splines; locally estimated scatterplot smoothing models; etc.

In some implementations, the machine-learned model can be or include one or more decision tree-based models such as, for example, classification and/or regression trees; iterative dichotomiser 3 decision trees; C4.5 decision trees; chi-squared automatic interaction detection decision trees; decision stumps; conditional decision trees; etc.

In some implementations, the machine-learned model can be or include one or more kernel machines. In some implementations, the machine-learned model can be or include one or more support vector machines.

In some implementations, the machine-learned model can be or include one or more instance-based learning models such as, for example, learning vector quantization models; self-organizing map models; locally weighted learning models; etc.

In some implementations, the machine-learned model can be or include one or more nearest neighbor models such as, for example, k-nearest neighbor classifications models; k-nearest neighbors regression models; etc.

In some implementations, the machine-learned model can be or include one or more Bayesian models such as, for example, naïve Bayes models; Gaussian naïve Bayes models; multinomial naïve Bayes models; averaged one-dependence estimators; Bayesian networks; Bayesian belief networks; hidden Markov models; etc.

In some implementations, the machine-learned model can be or include one or more artificial neural networks (also referred to simply as neural networks). A neural network can include a group of connected nodes, which also can be referred to as neurons or perceptrons. A neural network can be organized into one or more layers. Neural networks that include multiple layers can be referred to as "deep" networks. A deep network can include an input layer, an output layer, and one or more hidden layers positioned between the input layer and the output layer. The nodes of the neural network can be connected or non-fully connected.

In some implementations, the machine-learned model can be or include one or more feed forward neural networks. In feed forward networks, the connections between nodes do not form a cycle. For example, each connection can connect a node from an earlier layer to a node from a later layer.

In some instances, the machine-learned model can be or include one or more recurrent neural networks. In some instances, at least some of the nodes of a recurrent neural network can form a cycle. Recurrent neural networks can be especially useful for processing input data that is sequential in nature. In particular, in some instances, a recurrent neural network can pass or retain information from a previous portion of the input data sequence to a subsequent portion of the input data sequence through the use of recurrent or directed cyclical node connections.

As one example, sequential input data can include time-series data (e.g., sensor data versus time or imagery captured at different times). For example, a recurrent neural network can analyze sensor data versus time to detect or predict a swipe direction, to perform handwriting recognition, etc. As another example, sequential input data can include words in a sentence (e.g., for natural language processing, speech detection or processing, etc.); notes in a musical composition; sequential actions taken by a user (e.g., to detect or predict sequential application usage); sequential object states; etc.

Example recurrent neural networks include long short-term (LSTM) recurrent neural networks; gated recurrent units; bi-direction recurrent neural networks; continuous time recurrent neural networks; neural history compressors; echo state networks; Elman networks; Jordan networks; recursive neural networks; Hopfield networks; fully recurrent networks; sequence-to-sequence configurations; etc.

In some implementations, the machine-learned model can be or include one or more convolutional neural networks. In some instances, a convolutional neural network can include one or more convolutional layers that perform convolutions over input data using learned filters. Filters can also be referred to as kernels. Convolutional neural networks can be especially useful for vision problems such as when the input data includes imagery such as still images or video. However, convolutional neural networks can also be applied for natural language processing.

In some implementations, the machine-learned model can be or include one or more generative networks such as, for example, generative adversarial networks. Generative networks can be used to generate new data such as new images or other content.

In some implementations, the machine-learned model can be or include an autoencoder. In some instances, the aim of an autoencoder is to learn a representation (e.g., a lower-dimensional encoding) for a set of data, typically for the purpose of dimensionality reduction. For example, in some instances, an autoencoder can seek to encode the input data and then provide output data that reconstructs the input data from the encoding. Recently, the autoencoder concept has become more widely used for learning generative models of data. In some instances, the autoencoder can include additional losses beyond reconstructing the input data.

In some implementations, the machine-learned model can be or include one or more other forms of artificial neural networks such as, for example, deep Boltzmann machines; deep belief networks; stacked autoencoders; etc. Any of the neural networks described herein can be combined (e.g., stacked) to form more complex networks.

In some implementations, one or more neural networks can be used to provide an embedding based on the input data. For example, the embedding can be a representation of knowledge abstracted from the input data into one or more learned dimensions. In some instances, embeddings can be a useful source for identifying related entities. In some instances embeddings can be extracted from the output of the network, while in other instances embeddings can be extracted from any hidden node or layer of the network (e.g., a close to final but not final layer of the network). Embeddings can be useful for performing auto suggest next video, product suggestion, entity or object recognition, etc. In some instances, embeddings be useful inputs for downstream models. For example, embeddings can be useful to generalize input data (e.g., search queries) for a downstream model or processing system.

In some implementations, the machine-learned model can include one or more clustering models such as, for example, k-means clustering models; k-medians clustering models; expectation maximization models; hierarchical clustering models; etc.

In some implementations, the machine-learned model can perform one or more dimensionality reduction techniques such as, for example, principal component analysis; kernel principal component analysis; graph-based kernel principal component analysis; principal component regression; partial least squares regression; Sammon mapping; multidimensional scaling; projection pursuit; linear discriminant analysis; mixture discriminant analysis; quadratic discriminant analysis; generalized discriminant analysis; flexible discriminant analysis; autoencoding; etc.

In some implementations, the machine-learned model can perform or be subjected to one or more reinforcement learning techniques such as Markov decision processes; dynamic programming; Q functions or Q-learning; value function approaches; deep Q-networks; differentiable neural computers; asynchronous advantage actor-critics; deterministic policy gradient; etc.

In some implementations, the machine-learned model can be an autoregressive model. In some instances, an autoregressive model can specify that the output data depends linearly on its own previous values and on a stochastic term. In some instances, an autoregressive model can take the form of a stochastic difference equation. One example autoregressive model is WaveNet, which is a generative model for raw audio.

In some implementations, the machine-learned model can include or form part of a multiple model ensemble. As one example, bootstrap aggregating can be performed, which can also be referred to as “bagging.” In bootstrap aggregating, a training dataset is split into a number of subsets (e.g., through random sampling with replacement) and a plurality of models are respectively trained on the number of subsets. At inference time, respective outputs of the plurality of models can be combined (e.g., through averaging, voting, or other techniques) and used as the output of the ensemble.

One example model ensemble is a random forest, which can also be referred to as a random decision forest. Random forests are an ensemble learning method for classification, regression, and other tasks. Random forests are generated by producing a plurality of decision trees at training time. In some instances, at inference time, the class that is the mode of the classes (classification) or the mean prediction (regression) of the individual trees can be used as the output of the forest. Random decision forests can correct for decision trees' tendency to overfit their training set.

Another example ensemble technique is stacking, which can, in some instances, be referred to as stacked generalization. Stacking includes training a combiner model to blend or otherwise combine the predictions of several other machine-learned models. Thus, a plurality of machine-learned models (e.g., of same or different type) can be trained based on training data. In addition, a combiner model can be trained to take the predictions from the other machine-learned models as inputs and, in response, produce a final inference or prediction. In some instances, a single-layer logistic regression model can be used as the combiner model.

Another example ensemble technique is boosting. Boosting can include incrementally building an ensemble by iteratively training weak models and then adding to a final strong model. For example, in some instances, each new model can be trained to emphasize the training examples that previous models misinterpreted (e.g., misclassified). For example, a weight associated with each of such misinterpreted examples can be increased. One common implementation of boosting is AdaBoost, which can also be referred to as Adaptive Boosting.

Other example boosting techniques include LPBoost; TotalBoost; BrownBoost; xgboost; MadaBoost, LogitBoost, gradient boosting; etc.

Furthermore, any of the models described above (e.g., regression models and artificial neural networks) can be combined to form an ensemble. As an example, an ensemble can include a top level machine-learned model or a heuristic function to combine and/or weight the outputs of the models that form the ensemble.

In some implementations, multiple machine-learned models (e.g., that form an ensemble) can be linked and trained jointly (e.g., through backpropagation of errors sequentially through the model ensemble). However, in some implementations, only a subset (e.g., one) of the jointly trained models is used for inference.

In some implementations, the machine-learned model can be used to preprocess the input data for subsequent input into another model. For example, the machine-learned model can perform dimensionality reduction techniques and embeddings (e.g., matrix factorization, principal components analysis, singular value decomposition, word2vec/GLOVE, and/or related approaches); clustering; and even classification and regression for downstream consumption. Many of these techniques have been discussed above and will be further discussed below.

Referring again to Figure 1, and as discussed above, the machine-learned model can be trained or otherwise configured to receive the input data and, in response, provide the output data. The input data can include different types, forms, or variations of input data. As examples, in various implementations, the input data can include location data. For instance, the location data may comprise historical location data illustrating a location of the user with respect to time.

The location of the user can be determined from one or more of any known methods in the art, including but not limited to satellite positioning systems (e.g. GPS, GLONASS, etc.), land-based positioning systems, and/or connection-based positioning (e.g. network connectivity, Wi-Fi triangulation, cell tower triangulation, etc.). In some embodiments, the first machine-learned model may provide a prediction based at least in part on a time, such as day, week, month, season, year, day of week, time of day, etc. The time may be a digital representation of time, e.g. a timestamp, and may be sourced from an external location (e.g. the Internet), or from a clock within a system (e.g. within the computing system and/or the alarm system).

The input data can additionally comprise appointment characteristics such as the location and/or time of the appointment. The appointment characteristics may be input into the computing system and/or may be received from an application (e.g. a calendar application) on the computing system or another computing system. The location of the appointment can be determined in one or more of various methods including “dropping a pin” on a digital map (e.g. in a satellite positioning application), selecting from a pre-populated list, location history of the user, suggested by a computing system, or other suitable methods for determining location. The time of the appointment can be determined by numerical input (e.g. by entering a time via keyboard or other means) or graphical input (e.g. selecting a time from a dropdown menu or on a virtual clock). The input data can additionally comprise temporal data such as day, week, month, season, year, day of week, week of month, etc. The temporal data may be a digital representation of time (e.g. a timestamp).

In some implementations, the machine-learned model can receive and use the input data in its raw form. In some implementations, the raw input data can be preprocessed. Thus, in addition or alternatively to the raw input data, the machine-learned model can receive and use the preprocessed input data.

In some implementations, preprocessing the input data can include extracting one or more additional features from the raw input data. For example, feature extraction techniques can be applied to the input data to generate one or more new, additional features. Example feature extraction techniques include edge detection; corner detection; blob detection; ridge detection; scale-invariant feature transform; motion detection; optical flow; Hough transform; etc.

In some implementations, the extracted features can include or be derived from transformations of the input data into other domains and/or dimensions. As an example, the extracted features can include or be derived from transformations of the input data into the frequency domain. For example, wavelet transformations and/or fast Fourier transforms can be performed on the input data to generate additional features.

In some implementations, the extracted features can include statistics calculated from the input data or certain portions or dimensions of the input data. Example statistics include the mode, mean, maximum, minimum, or other metrics of the input data or portions thereof.

In some implementations, as described above, the input data can be sequential in nature. In some instances, the sequential input data can be generated by sampling or otherwise segmenting a stream of input data. As one example, frames can be extracted from a video. In some implementations, sequential data can be made non-sequential through summarization.

As another example preprocessing technique, portions of the input data can be imputed.

For example, additional synthetic input data can be generated through interpolation and/or extrapolation.

As another example preprocessing technique, some or all of the input data can be scaled, standardized, normalized, generalized, and/or regularized. Example regularization techniques include ridge regression; least absolute shrinkage and selection operator (LASSO); elastic net; least-angle regression; cross-validation; L1 regularization; L2 regularization; etc. As one example, some or all of the input data can be normalized by subtracting the mean across a given dimension's feature values from each individual feature value and then dividing by the standard deviation or other metric.

As another example preprocessing technique, some or all or the input data can be quantized or discretized. As yet another example, qualitative features or variables included in the input data can be converted to quantitative features or variables. For example, one hot encoding can be performed.

In some implementations, dimensionality reduction techniques can be applied to the input data prior to input into the machine-learned model. Several examples of dimensionality reduction techniques are provided above, including, for example, principal component analysis; kernel principal component analysis; graph-based kernel principal component analysis; principal component regression; partial least squares regression; Sammon mapping; multidimensional scaling; projection pursuit; linear discriminant analysis; mixture discriminant analysis; quadratic discriminant analysis; generalized discriminant analysis; flexible discriminant analysis; autoencoding; etc.

In some implementations, during training, the input data can be intentionally deformed in any number of ways to increase model robustness, generalization, or other qualities. Example techniques to deform the input data include adding noise; changing color, shade, or hue; magnification; segmentation; amplification; etc.

Referring again to Figure 1, in response to receipt of the input data, the machine-learned model can provide the output data. The output data can include different types, forms, or variations of output data. As examples, in various implementations, the output data can include a length of time that a user needs to prepare for an appointment based on a length of time between the time that the user is alerted by an alarm and the time that the user departs in response to the alert. For example, the length of time that a user needs to prepare for an appointment may be an output from a first machine-learned model. For example, if the alarm wakes the user up in the morning, the first machine-learned model may predict the length of time that the user needs to prepare based on previously collected data illustrating the time when the user is woken up by an alarm and the time when the user leaves his or her residence. The alert time and response time can be determined in various manners. Continuing with the above example, the time when the user is woken up by an alarm may be determined from an alarm application on the computing system or on another computing system. Additionally, the time when the user leaves his or her residence may be determined by a change in position, such as by a change in location (e.g. moving from within an area known to be the user’s residence to a different area, a change in generalized location such as street, city, ZIP code, etc.) an occurrence of a velocity associated with driving a vehicle (e.g. if a device detects the user is travelling at a speed above average human walking speed), a change in network connectivity (e.g. a user disconnects from his or her home internet connection), a local positioning system (e.g. a sensor on the user's door), a user arming a security system, or other suitable methods.

The preparation time predicted by the first machine-learned model may be specified for a specific time, such as a specific day, week, month, season, year, day of week, week of year, etc. For instance, the model may predict that the user takes a different length of time to prepare on a certain day of the week (e.g. the user may have to start laundry before work on Monday resulting in a longer preparation time than on Tuesday through Friday) or during a certain season (e.g. the user may take longer to put on multiple layers of clothing during autumn or winter resulting in a longer preparation time than during spring or summer). Additionally and/or alternatively, the model may predict that the user takes more time to prepare for a holiday or event (e.g. for an event which consistently falls on the same day (e.g. a New Year's party held on December 31st) or day of week (e.g. a New Year's party held on the last Friday of December)).

In some embodiments, the user may manually provide a length of time the user needs to prepare (e.g. in addition to or alternatively to the first machine-learned model). For instance, the user may be provided with a prompt asking how much time the user prefers to prepare. This may be used in replacement of the first machine-learned model or in addition to (e.g., if the user specifies more time than usual, the first machine-learned model may predict additional time is necessary).

Output data can additionally include a predicted time to alert the user about an appointment to allow sufficient time for the user to prepare for and/or commute to the appointment. For example, the predicted time can be output from a third machine-learned model. The prediction from the third machine-learned model may be used, for example, to configure an alarm system. The prediction from the third machine-learned model may include a buffer, for example a fifteen minute buffer. The buffer may be predicted for the user (e.g., based on user "snooze" patterns in which the user delays the alarm for a short period of time).

For example, the first machine-learned model may predict that the time between when the user is alerted about an appointment and the time when the user departs for the appointment will be thirty minutes. Additionally and/or alternatively, the user may have otherwise specified a thirty minute preparation time. Continuing with the above example, the second machine-learned model may predict that the travel time from the user's current location to the location of the appointment will be thirty minutes. The third machine-learned model may additionally include a fifteen minute buffer. If the appointment time is at 8:00 AM, the model may thus predict an alarm time of 6:45 AM.

As discussed above, in some implementations, the output data can include various types of classification data (e.g., binary classification, multiclass classification, single label, multi-label, discrete classification, regressive classification, probabilistic classification, etc.) or can include various types of regressive data (e.g., linear regression, polynomial regression, nonlinear regression, simple regression, multiple regression, etc.). In other instances, the output data can include clustering data, anomaly detection data, recommendation data, or any of the other forms of output data discussed above.

In some implementations, the output data can influence downstream processes or decision making. As one example, in some implementations, the output data can be interpreted and/or acted upon by a rules-based regulator.

Thus, the present disclosure provides systems and methods that include or otherwise leverage one or more machine-learned models to predict a time to alert a user based on a user's schedule and location. Any of the different types or forms of input data described above can be combined with any of the different types or forms of machine-learned models described above to provide any of the different types or forms of output data described above.

The systems and methods of the present disclosure can be implemented by or otherwise executed on one or more computing devices. Example computing devices include user computing devices (e.g., laptops, desktops, and mobile computing devices such as tablets, smartphones, wearable computing devices, etc.); embedded computing devices (e.g., devices embedded within a vehicle, camera, image sensor, industrial machine, satellite, gaming console or controller, or home appliance such as a refrigerator, thermostat, energy meter, home energy manager, smart home assistant, etc.); server computing devices (e.g., database servers, parameter servers, file servers, mail servers, print servers, web servers, game servers, application servers, etc.); dedicated, specialized model processing or training devices; virtual computing devices; other computing devices or computing infrastructure; or combinations thereof.

Thus, in some implementations, the machine-learned model can be stored at and/or implemented locally by an embedded device or a user computing device such as a mobile device. Output data obtained through local implementation of the machine-learned model at the embedded device or the user computing device can be used to improve performance of the embedded device or the user computing device (e.g., an application implemented by the embedded device or the user computing device). As one example, Figure 2 illustrates a block diagram of an example computing device that stores and implements a machine-learned model locally.

In other implementations, the machine-learned model can be stored at and/or implemented by a server computing device. In some instances, output data obtained through implementation of the machine-learned model at the server computing device can be used to improve other server tasks or can be used by other non-user devices to improve services performed by or for such other non-user devices. For example, the output data can improve other downstream processes performed by the server computing device for a user computing device or embedded computing device. In other instances, output data obtained through implementation of the machine-learned model at the server computing device can be sent to and used by a user computing device, an embedded computing device, or some other client device. For example, the server computing device can be said to perform machine learning as a service. As one example, Figure 3 illustrates a block diagram of an example client computing device that can communicate over a network with an example server computing system that includes a machine-learned model.

In yet other implementations, different respective portions of the machine-learned model can be stored at and/or implemented by some combination of a user computing device; an embedded computing device; a server computing device; etc.

Computing devices can perform graph processing techniques or other machine learning techniques using one or more machine learning platforms, frameworks, and/or libraries, such as, for example, TensorFlow, Caffe/Caffe2, Theano, Torch/PyTorch, MXnet, CNTK, etc.

Computing devices can be distributed at different physical locations and connected via one or more networks. Distributed computing devices can operate according to sequential computing architectures, parallel computing architectures, or combinations thereof. In one example, distributed computing devices can be controlled or guided through use of a parameter server.

In some implementations, multiple instances of the machine-learned model can be parallelized to provide increased processing throughput. For example, the multiple instances of the machine-learned model can be parallelized on a single processing device or computing device or parallelized across multiple processing devices or computing devices.

Each computing device that implements the machine-learned model or other aspects of the present disclosure can include a number of hardware components that enable performance of the techniques described herein. For example, each computing device can include one or more memory devices that store some or all of the machine-learned model. For example, the machine-learned model can be a structured numerical representation that is stored in memory. The one or more memory devices can also include instructions for implementing the machine-learned model or performing other operations. Example memory devices include RAM, ROM, EEPROM, EPROM, flash memory devices, magnetic disks, etc., and combinations thereof.

Each computing device can also include one or more processing devices that implement some or all of the machine-learned model and/or perform other related operations. Example processing devices include one or more of: a central processing unit (CPU); a visual processing unit (VPU); a graphics processing unit (GPU); a tensor processing unit (TPU); a neural processing unit (NPU); a neural processing engine; a core of a CPU, VPU, GPU, TPU, NPU or other processing device; an application specific integrated circuit (ASIC); a field programmable gate array (FPGA); a co-processor; a controller; or combinations of the processing devices described above. Processing devices can be embedded within other hardware components such as, for example, an image sensor, accelerometer, etc.

Hardware components (e.g., memory devices and/or processing devices) can be spread across multiple physically distributed computing devices and/or virtually distributed computing systems.

In some implementations, the machine-learned models described herein can be trained at a training computing system and then provided for storage and/or implementation at one or more computing devices, as described above. For example, a model trainer can be located at the training computing system. The training computing system can be included in or separate from the one or more computing devices that implement the machine-learned model. As one example, Figure 4 illustrates a block diagram of an example computing device in communication with an example training computing system that includes a model trainer.

In some implementations, the model can be trained in an offline fashion or an online fashion. In offline training (also known as batch learning), a model is trained on the entirety of a static set of training data. In online learning, the model is continuously trained (or re-trained) as new training data becomes available (e.g., while the model is used to perform inference).

In some implementations, the model trainer can perform centralized training of the machine-learned models (e.g., based on a centrally stored dataset). In other implementations, decentralized training techniques such as distributed training, federated learning, or the like can be used to train, update, or personalize the machine-learned models.

The machine-learned models described herein can be trained according to one or more of various different training types or techniques. For example, in some implementations, the machine-learned models can be trained using supervised learning, in which the machine-learned model is trained on a training dataset that includes instances or examples that have labels. The labels can be manually applied by experts, generated through crowd-sourcing, or provided by other techniques (e.g., by physics-based or complex mathematical models). In some implementations, if the user has provided consent, the training examples can be provided by the user computing device. In some implementations, this process can be referred to as personalizing the model.

As one example, Figure 5 illustrates a block diagram of an example training process in which a machine-learned model is trained on training data that includes example input data that has labels. Training processes other than the example process depicted in Figure 5 can be used as well.

In some implementations, training data can include examples of the input data that have been assigned labels that correspond to the output data. For instance, the first machine-learned model can be trained on sample location history labeled with a known preparation time. In addition, the third machine-learned model can be trained on sample batches of input data (e.g.

preparation time, travel time, location of user, location of appointment, etc.) labeled with a known alarm time.

In some implementations, the machine-learned model can be trained by optimizing an objective function. For example, in some implementations, the objective function can be or include a loss function that compares (e.g., determines a difference between) output data generated by the model from the training data and labels (e.g., ground-truth labels) associated with the training data. For example, the loss function can evaluate a sum or mean of squared differences between the output data and the labels. As another example, the objective function can be or include a cost function that describes a cost of a certain outcome or output data. Other objective functions can include margin-based techniques such as, for example, triplet loss or maximum-margin training.

One or more of various optimization techniques can be performed to optimize the objective function. For example, the optimization technique(s) can minimize or maximize the objective function. Example optimization techniques include Hessian-based techniques and gradient-based techniques, such as, for example, coordinate descent; gradient descent (e.g., stochastic gradient descent); subgradient methods; etc. Other optimization techniques include black box optimization techniques and heuristics.

In some implementations, backward propagation of errors can be used in conjunction with an optimization technique (e.g., gradient based techniques) to train a model (e.g., a multi-layer model such as an artificial neural network). For example, an iterative cycle of propagation and model parameter (e.g., weights) update can be performed to train the model. Example backpropagation techniques include truncated backpropagation through time, Levenberg-Marquardt backpropagation, etc.

In some implementations, the machine-learned models described herein can be trained using unsupervised learning techniques. Unsupervised learning can include inferring a function to describe hidden structure from unlabeled data. For example, a classification or categorization may not be included in the data. Unsupervised learning techniques can be used to produce machine-learned models capable of performing clustering, anomaly detection, learning latent variable models, or other tasks.

In some implementations, the machine-learned models described herein can be trained using semi-supervised techniques which combine aspects of supervised learning and unsupervised learning.

In some implementations, the machine-learned models described herein can be trained or otherwise generated through evolutionary techniques or genetic algorithms.

In some implementations, the machine-learned models described herein can be trained using reinforcement learning. In reinforcement learning, an agent (e.g., model) can take actions in an environment and learn to maximize rewards and/or minimize penalties that result from such actions. Reinforcement learning can differ from the supervised learning problem in that correct input/output pairs are not presented, nor sub-optimal actions explicitly corrected.

In some implementations, one or more generalization techniques can be performed during training to improve the generalization of the machine-learned model. Generalization techniques can help reduce overfitting of the machine-learned model to the training data. Example generalization techniques include dropout techniques; weight decay techniques; batch normalization; early stopping; subset selection; stepwise selection; etc.

In some implementations, the machine-learned models described herein can include or otherwise be impacted by a number of hyperparameters, such as, for example, learning rate, number of layers, number of nodes in each layer, number of leaves in a tree, number of clusters; etc. Hyperparameters can affect model performance. Hyperparameters can be hand selected or can be automatically selected through application of techniques such as, for example, grid search; black box optimization techniques (e.g., Bayesian optimization, random search, etc.); gradient-based optimization; etc. Example techniques and/or tools for performing automatic hyperparameter optimization include Hyperopt; Auto-WEKA; Spearmint; Metric Optimization Engine (MOE); etc.

In some implementations, various techniques can be used to optimize and/or adapt the learning rate when the model is trained. Example techniques and/or tools for performing learning rate optimization or adaptation include Adagrad; Adaptive Moment Estimation (ADAM); Adadelta; RMSprop; etc.

In some implementations, transfer learning techniques can be used to provide an initial model from which to begin training of the machine-learned models described herein.

In some implementations, the machine-learned models described herein can be included in different portions of computer-readable code on a computing device. In one example, the machine-learned model can be included in a particular application or program and used (e.g., exclusively) by such particular application or program. Thus, in one example, a computing device can include a number of applications and one or more of such applications can contain its own respective machine learning library and machine-learned model(s).

In another example, the machine-learned models described herein can be included in an operating system of a computing device (e.g., in a central intelligence layer of an operating system) and can be called or otherwise used by one or more applications that interact with the operating system. In some implementations, each application can communicate with the central intelligence layer (and model(s) stored therein) using an application programming interface (API) (e.g., a common, public API across all applications).

In some implementations, the central intelligence layer can communicate with a central device data layer. The central device data layer can be a centralized repository of data for the computing device. The central device data layer can communicate with a number of other components of the computing device, such as, for example, one or more sensors, a context manager, a device state component, and/or additional components. In some implementations, the central device data layer can communicate with each device component using an API (e.g., a private API).

The technology discussed herein makes reference to servers, databases, software applications, and other computer-based systems, as well as actions taken and information sent to and from such systems. The inherent flexibility of computer-based systems allows for a great variety of possible configurations, combinations, and divisions of tasks and functionality between and among components. For instance, processes discussed herein can be implemented using a single device or component or multiple devices or components working in combination.

Databases and applications can be implemented on a single system or distributed across multiple systems. Distributed components can operate sequentially or in parallel.

In addition, the machine learning techniques described herein are readily interchangeable and combinable. Although certain example techniques have been described, many others exist and can be used in conjunction with aspects of the present disclosure.

Thus, while the present subject matter has been described in detail with respect to various specific example implementations, each example is provided by way of explanation, not limitation of the disclosure. One of ordinary skill in the art can readily make alterations to, variations of, and equivalents to such implementations. Accordingly, the subject disclosure does not preclude inclusion of such modifications, variations and/or additions to the present subject matter as would be readily apparent to one of ordinary skill in the art. For instance, features illustrated or described as part of one implementation can be used with another implementation to yield a still further implementation.

A brief overview of example machine-learned models and associated techniques has been provided by the present disclosure. For additional details, readers should review the following references: *Machine Learning A Probabilistic Perspective* (Murphy); *Rules of Machine Learning: Best Practices for ML Engineering* (Zinkevich); *Deep Learning* (Goodfellow); *Reinforcement Learning: An Introduction* (Sutton); and *Artificial Intelligence: A Modern Approach* (Norvig).

Figures

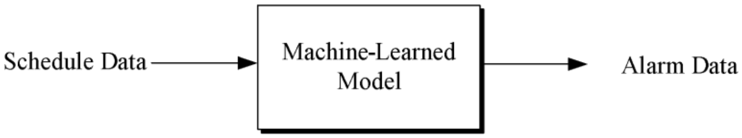

Figure 1

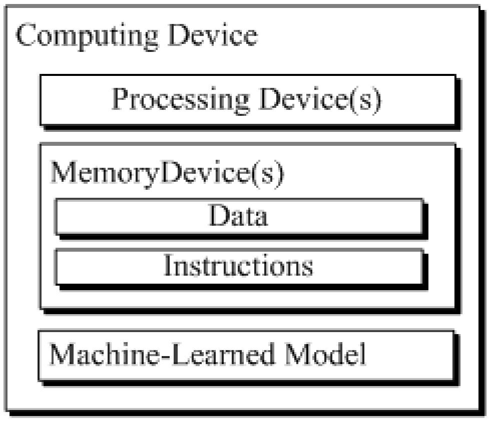

Figure 2

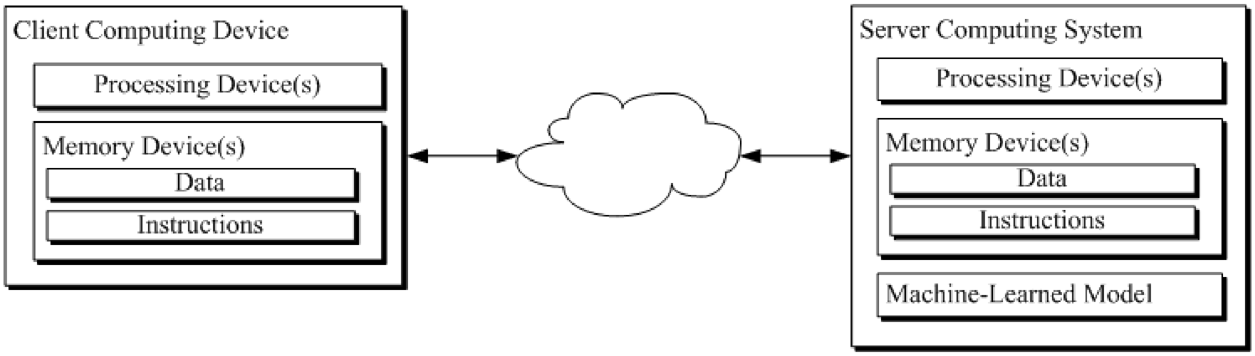

Figure 3

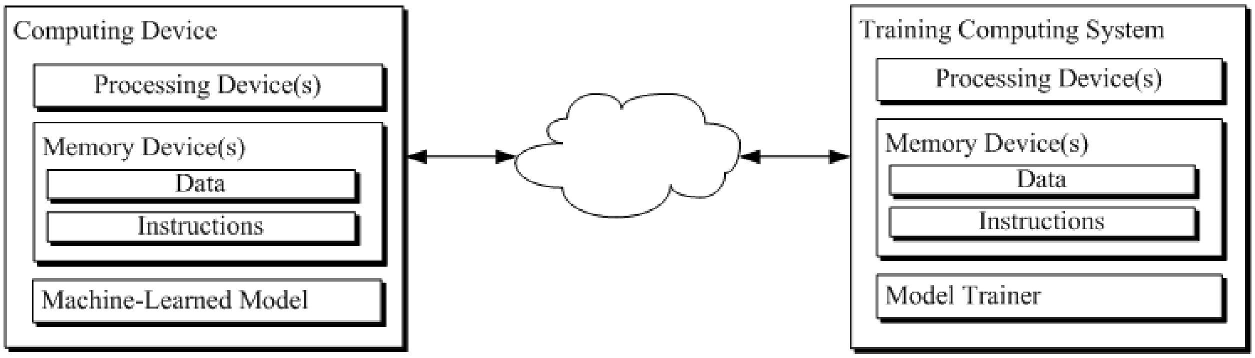

Figure 4

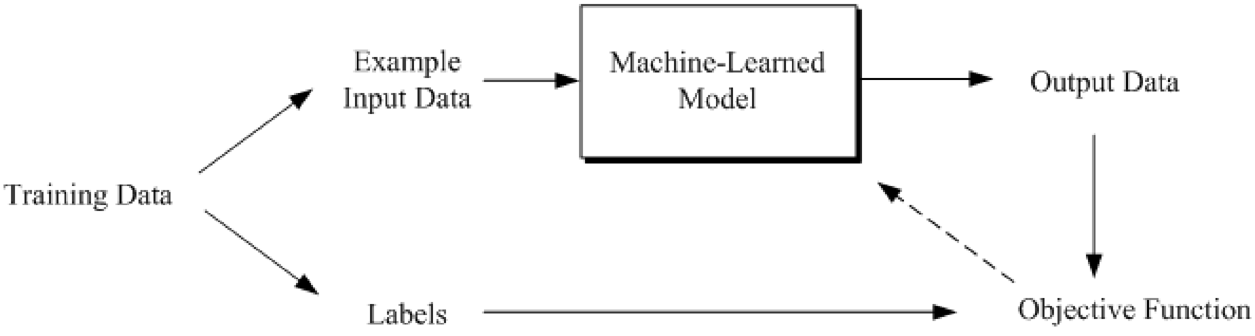

Figure 5

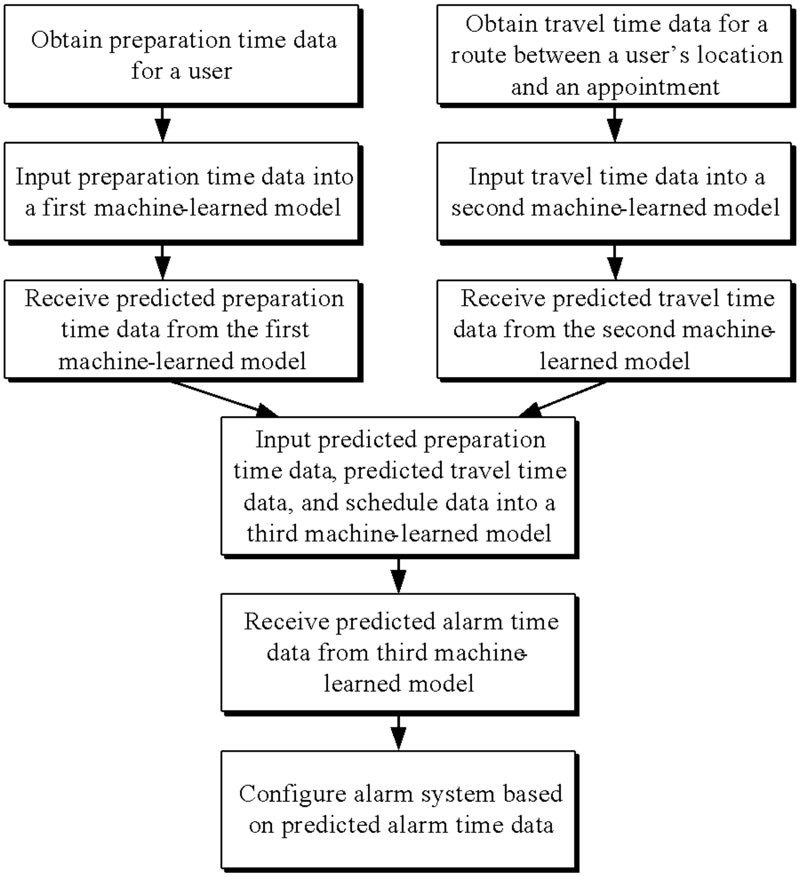

Figure 6