Özgür Kurt

Recommending next store visit for new customers in large shopping malls

Master’s thesis of mathematical information technology

November 8, 2021

Author: Özgür Kurt

Contact information: ozgurkurt80@yahoo.com

Supervisors: Oleksiy Khriyenko

Title: Recommending next store visit for new customers in large shopping malls

Project: Master's thesis

Study line: Web Intelligence and Service Engineering

Page count: 55+1

Abstract: Nowadays widespread availability of complimentary WI-FI inside large shopping malls and the increasing precision of WI-FI positioning systems make it possible to track a customer's trajectory inside shopping malls via their mobile devices. This trajectory data open the door for many useful applications that can help both customers and store owners. This study presents an application aimed for new customers of a large shopping mall, who are not familiar with the layout and available stores inside, to navigate the mall more effectively. To achieve this, we first find common customer intents (store visit patterns) inside the mall, and then fit a newly arrived customer's intent to one of these common intents. After finding possible intents for a customer, we use the movement patterns for available intents to produce a next-store recommendation for the customer. Fuzzy c-means clustering technique will be used to find intents from customer trajectories. All customer visits belonging to these intents will be processed as sequential trajectory steps. These sequential steps are enriched with some other peripheral information related to day, time, duration, and then are fed into a neural network architecture consisting of RNN and Dense layers to model the movement patterns related to intents. Results of this model will provide recommendations to new-coming customers for their next store visit. Finally, using a set of real life trajectory data, predictions from the model will be presented and interpreted.

Keywords: Shopping malls, Trajectory patterns, Fuzzy clustering, Machine learning, Neural networks, RNN

FIGURES

TABLES

Contents

1 Introduction

Whether it be a developed or an emerging economy, large shopping malls have a substantial part in economic transactions. However, these shopping malls are facing a danger of decrease in profits and efficiency for different reasons. In developed countries the attraction of shopping malls are decreasing because of the appeal of online shopping and the high number of available competitors. In emerging countries, every new mall that is opened claims some share of the consumers available, while potentially decreasing others'. On top of these, the troubles faced during COVID-19 pandemic might induce long lasting anxiety in customers' psyche, causing them to avoid spending more time than necessary within enclosed spaces, which may decrease the footfall to malls and/or the income for the stores due to reduction in time spent inside. All these factors put extra pressure for each shopping mall to maximise their profits using the resources they have. Targeted marketing efforts over available customers, and presenting innovative solutions to customers are just some ways to this end.

Bloch et al says enclosed shopping mall is a habitat for consumers (Bloch, Ridgway, and Dawson 1994, 23). They also state that it is expected that consumers who enter into the habitat seeking different benefits will behave differently inside the habitat. This is to say; people who are fashion-conscious will spend more time visiting clothing shops, people who want to be entertained will be visiting arcade shops, movie theatres and the likes rather than retailers. We can also speculate the intentions they have will affect the way they move inside the mall, and possibly the order by which they visit the stores. Vasquez et al. points to the fact that intentions and physical state complement each other; physical state is conditioned by intentions and intentions can be deduced by tracking the speed and position of an object (Vasquez, Fraichard, and Laugier 2009, 1486-1506).

Shopping malls can be large, and even have unintuitive layouts, and this might make the customers' visit to the mall a confusing and chaotic experience, even more so for first-time customers who are not familiar with the surroundings. New customers can arrive at a mall with a definite intention on their mind, or they can be after a more free-flowing visiting experience in which they can be attracted by different propositions. For both groups of new customers a recommender system that can direct them to relevant places could become a useful tool. For customers who have a certain intent on their mind, it can be argued that it would be enough to make them choose their intent from one of the predefined intents upon their arrival to the mall, and then recommend relevant places that could fit this intent. Second group of customers however, the ones who do not have a certain intent, will not be able to provide such preference accurately. This second group of customers are the main target of this study, even though other use-cases for all types of customers can be derived around this application.

With this context in mind, this study aims to produce a non-personalised recommender system that can be used to make new customers' navigation inside a large shopping mall more effective by promoting them stores they might be interested in visiting.

A software artefact (*NextStore*) will also be developed to realise these methods and the accuracy of the predictions made over the test data will be reported.

1.1 Objective and motivation

Mobile devices have become an integral part of human life and along with the problems this can cause, there also come some opportunities; for device users and for businesses alike. By providing their location data, mobile device users can enjoy location based content and applications that are more relevant to them at that point in time, whereas it is also good business to be able to analyse this kind of data and to provide relevant applications for interested users. Nowadays access to this location data is possible in many enclosed areas, since the businesses owning these areas provide complimentary WI-FI connections to visitors, and these systems can also be used to retrieve location data of users. Having this data inside an enclosed area can open the door for several location-based applications, such as next-location recommendation, location-based advertisement, as well as many different types of analysis from the static data.

A company I have worked (IPera Solutions[1]) for had already been working on products that facilitate complimentary Wi-Fi offerings in shopping malls and other large scale enclosed areas like airports. Discussions with the company yielded a possibility to focus on the case of new-coming customers who are not familiar with the mall, and build a recommender system around their short-term store visit history, a.k.a. session-based.

Trajectory data from one of the shopping malls in this company's portfolio will be used in the study. Main motivation is to show that it is possible to produce a recommender system that can handle the task of producing real-time next-location recommendations to mobile users who are visiting a brick and mortar shopping mall, using their session-based visitation data.

Related to this task, following research questions are raised;

- *Is it possible to find some common store visitation patterns of customers inside large shopping malls?*

  First question conjectures the possibility of customers visiting similar stores having similar intentions (a.k.a. belonging to same segment of customers). If a new-coming customer's movement fits one of the deduced intentions, their successive visit in the same session could also be in line with other people who had similar intentions.

  We first define the concept of *a customer's intent for visiting the shopping mall (*referred to as *intent* hereon*)*. This intent constitutes of visited stores by the customer throughout their trajectory in the whole session of visiting the mall.

- *Could a collaborative filtering method utilising fuzzy clustering and RNNs be a good model for next store recommendation?*

  Different collaborative filtering based recommendation methods have already been studied extensively, especially in the contexts of social media and online retail. The second research question of this paper puts forward the idea of applying this

[1] https://iperasolutions.com/

technique to the task of producing recommendations for stores inside a brick and mortar shopping mall. It will be attempted to show that an RNN structure taking sequential store visitations as input can successfully model the visit pattern of a customer, by using similar intents derived from historical visitation patterns of other users.

1.2 Research method

Conforming to Design Science Research Method, a real life problem (recommending a successive store visit to a customer) has been identified during discussions with a real life company. Trajectory data from one of the shopping malls in their portfolio, containing a month of customer movement information, is acquired for the study. This is just one possible format of data, but data in any different format will still be as useful so long as they contain the essential information used in the study. The difference in data formats can quite easily be made up for, during the data preprocessing phase.

To demonstrate the results produced by the proposed system, a prototype software artefact will be designed and implemented. Python programming language is used to implement this software. The most prominent libraries used in the development process are Pandas and Tensorflow/Keras for data manipulation and neural network implementation respectively. Also skfuzzy[2] library is used for fuzzy c-means clustering of customer trajectories.

Some portion of the sample data will be used to evaluate the prototype and the success rate of the recommendations according to different metrics will be reported in *Results* section.

1.3 Limitations

There are some limitations to achieving the best results from the proposed system.

[2] https://pythonhosted.org/scikit-fuzzy/

First of which is the unreliable location data. Even though GPS locationing systems are getting better at providing accurate coordinates, they are still far from providing exact coordinates, as it is observed from the location data we are provided with. Some measures are taken during the data preprocessing steps towards correcting this faulty information, but there will still be inaccuracies amongst the data points, related to the exact stores visited and their orderings in the trajectory, and particularly the durations indicating the time customers spent in there.

Current data available to the study raises another limitation regarding the distinction between first-time users and users revisiting the mall. It is reasonable to assume that if people are familiar with the mall and the stores inside it, the intent groups we can derive from their trajectories could have less noise in them. Contrarily, first-timers may go in and out of stores they do not actually find interesting until they find out by seeing what's inside. This distinction could have provided more insight into the difference between the movement patterns and the produced intent groups between these two types of customers. We can assume that customers who are already familiar with the mall would have a more purposeful, more optimised routes to their intents, and it could be a good idea to try and fit new-comers' intents to that of more "experienced" visitors. Theoretically this could be done with a sort of weighting mechanism during recommendations. It would be possible to find probabilities of intents from two separate intention groups, and experienced group's intention can take more prominence while recommending. This way of working can create a better system for enhancing collective intelligence of the customer crowd, since visitors' routes will get closer to the optimal route as they become more experienced. This will help our recommendations to be more on the spot for new-comers. Current version of this study, however, will not be able to make this distinction, and a single grouping of intents will be derived from both types of users' trajectories.

Another limitation relates to measuring the effectiveness of the recommendations produced by the system. To be able to rate the exact number of recommendations found useful by the customer, this study needs to be integrated into an already functioning mobile application and be able to collect feedback from the device's user about how useful was the recommendation produced. Since this integration will not be provided at the time of writing this thesis, we will need to resort to other types of metrics, perhaps less accurate than the real life scenario, to evaluate the effectiveness of the produced system.

1.4 Thesis structure

This thesis starts with a presentation of the building concepts of this study and reviews of literature related to these concepts. After that a section describes the available sample data and the preprocessing operations applied on it. It moves on to present the methods used in each step of the application. Justifications for the design choices are also made in this section.

After a thorough explanation of workings of the prototype, the metrics used to rate the success of this prototype is presented along with the actual results from tests made on sample data, in *Results* section. In the concluding sections, some ideas for improving the solution as a product, and a summary of the study are provided.

2 Main Concepts & Literature Review

Subject areas like collective intelligence, data clustering, semantic trajectories, neural networks, and recommender systems form the backbone of the proposed application in this paper. Literature related to these areas of research has been reviewed for this study.

2.1 Consumers and Collective Intelligence

With an analogy to wildlife, Bloch et al explain that, consumers flock to facilities where the climate, opportunities for social interaction, perceived feeling of safety, and the selection of consumable goods and experiences are of higher standards. Hence they describe the shopping mall as a "premier habitat for consumers", which also implies that consumers tend to form groups that are likely to have similar behavioural patterns inside the mall (Bloch, Ridgway, and Dawson 1994, 23). As an example, it is reasonable to assume that store visitation patterns of consumers who are seeking to buy women's clothing, will mostly include clothing retailers who sell women's garments.

These similarities in behaviour are what make this study viable. By being able to expose the different groups of behavioural patterns, we can produce recommendations for new-coming customers and create a more optimal shopping or browsing experience for them. As for the retailers, they would benefit from this by drawing more attention from the right sort of customers who would have a bigger interest in their products or services. Even if the customer in question is only visiting for browsing and not for spending, retailers still get to expose their products and promotions. This information might convert the browsing customer into a spending customer immediately or in the future, or simply, this customer might act as an opinion leader and spread the information, influencing others to buy (Bloch, Ridgway, and Sherrell 1989, 13-21).

When it comes to collective intelligence there are varying definitions, but the following one is fitting for our study; a group's ability to find more or better solutions than can be produced by its members (Heylighen 1999, 253-280). It can be said that the solutions we want to derive from the consumer crowd is how to get the best experience out of the facility in the most optimal and efficient way before leaving. This would mean that consumer is directed to stores those are actually in line with their shopping or browsing intentions.

Building blocks of collective intelligence systems comprise of 4 simple questions, which is illustrated in Figure 1 (Malone, Laubacher, and Dellarocas 2009);

- Who is performing the task? Why are they doing it?
- What is being accomplished? How is it being done?

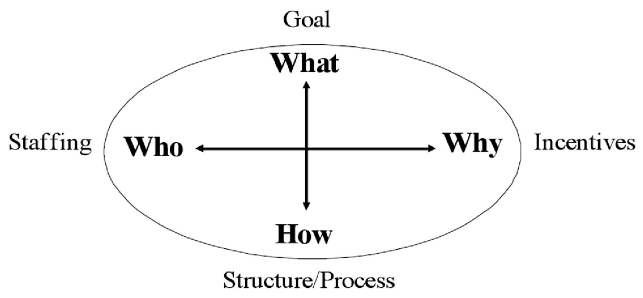


Figure 1 Elements of collective intelligence building blocks or “genes”

Who, corresponds to the shopping mall crowd that are undertaking the navigation activity in our study. *Why*, is the motivation for the crowd for being in the shopping mall, which is supposedly different for each individual amongst the crowd. *What*, is the decisions made about navigating inside the mall. *How*, is the product of this study, namely, a recommender system that derives the intelligence from the crowd and figure out possible stores of interest, hence a way of navigating, for a given customer.

To get the most out of the intelligence in a crowd it should have the following characteristics; varying opinions amongst individuals, freedom of forming and expressing of opinions, ability to focus on different sets of knowledge, and the possibility to aggregate all opinions and make a collective decision (Surowiecki 2005). Surowiecki claims that if a crowd have these characteristics, it will be smarter than the smartest person in the crowd. Shopping mall’s habitat will have a crowd satisfying these conditions; it has many individuals with different opinions about how to navigate inside, their opinions are collected through the location data we have, each customer has a different motivation for being in the mall, and all ideas produced by the shopping mall crowd will be interpreted and recommendations will be made by our system.

These recommendations will help first-time visitors make their decisions easier and arguably, better informed. Since human brains are wired to avoid complexity and try to make faster decisions (Bonabeau 2009, 45-52), there is some value in a product such as *NextStore*.

2.2 Clustering

In their clustering techniques survey, Grira et al. defines the aim of clustering as organising data points into groups such that they will bear some similarity to other data points in the same group as them and less so to the points in other groups (Grira, Crucianu, and Boujemaa 2004).

This study makes use of two different clustering techniques for grouping data in different phases of the application. First, we use a density-based clustering method to during data-cleaning phase of location point data and get a more accurate store-visit trajectory. In a later step, we use a fuzzy clustering method to find trajectories with similar store interests.

2.2.1 Density-Based Clustering

2.2.1.1 DBSCAN

The main idea behind density-based clustering is to create clusters where each point in the cluster will have a minimum number of neighbouring points in a given radius, thus creating dense areas of data points (Ester et al. 1996, 226–231). DBSCAN is the most well-known algorithm in this category. It requires two meta-parameters to function; *eps*, which defines the radius of the neighbourhood for a point, and a *minPts* parameter, which denotes the minimum number of points that should be contained in the neighbourhood. Compared to other popular clustering methods, DBSCAN has the advantage of not requiring the number of produced clusters beforehand, and the ability to discover clusters of arbitrary shapes (Ester et al. 1996, 226–231).

2.2.1.2 ST-DBSCAN (Spatio-Temporal DBSCAN)

ST-DBSCAN is a variation on DBSCAN algorithm, developed by Birant et al. specifically for data that consists of both spatial and temporal features, like trajectory data (Birant and Kut 2007, 208-221). While standard DBSCAN utilises only one non-spatial/non-temporal distance between data samples for producing clusters, ST-DBSCAN considers spatial and temporal distances amongst data points, too.

While a single *eps* parameter is needed for regular DBSCAN algorithm, ST-DBSCAN requires one additional *eps* parameter that will be used to calculate similarities according to temporal features. This allows the algorithm to define similarity of points by using two separate density values (Birant and Kut 2007, 208-221).

Deciding on meta-parameter values for clustering algorithms is usually a challenging task. Birant et al. describe a heuristic to calculate these parameters optimally and this method is also utilised in our work. They suggest that the ideal value of *minPts* parameter should be the natural logarithm of the number of data points in the set. For *eps1* and *eps2* values, distances to the *k-nearest* neighbours are found for each point in dataset, where *k* is equal to *minPts*. Then these distance values are sorted in descending order to decide the point where the graph of distance values starts forming a *valley.* A value smaller than this valley point should be picked for the *eps* parameter. These operations are done separately for spatial and temporal values so that *eps1* and *eps2* can be set respectively.

2.2.2 Fuzzy clustering

Clustering visitations patterns of customers in the mall is the first step in our study after pre-processing steps. The aim of this clustering step will be to get a reasonable number of different customer intents, while having similar patterns of store visits inside any given intent. These intents can be thought of as a way of customer segmentation. As cited by Öztayşi et al. from (Wind and Bell 2008, 222-244), it is necessary to separate the market into segments, understand what they need and desire, and produce suitable products and services to satisfy them, while directing the marketing efforts effectively to reach the segment in question (Öztayşi et al. 2017). Acquiring this segmentation in the first step of this proposed system forms the backbone of our attempt at making relevant marketing efforts (recommendations).

Clustering methods according to the nature of clusters they produce can be divided into two types; crisp and fuzzy. This division explains how the membership of the data points for the clusters are represented; for crisp clustering membership values are either 0 or 1, and for fuzzy clustering they are a real number between 0 and 1 (Grira, Crucianu, and Boujemaa 2004). In other words, when using fuzzy clustering, a data point will have a membership probability for each produced cluster, unlike in crisp clustering where they will only belong to one of the clusters.

This study will use fuzzy C-means clustering to discover customer intents. Main purpose of using a fuzzy clustering method is to provide the ability to model situations where clusters are overlapping (Grira, Crucianu, and Boujemaa 2004). However, since this is a centroid-based partitional clustering method, it has the downside of requiring the number of available clusters as a parameter to the algorithm, and it will be decided empirically during this study.

2.3 Semantic Trajectories

Spaccapietra et al. defines a trajectory as being user defined record of change in a moving, goal oriented object’s position during a given time interval. They put trajectories under three segments; metaphorical, naïve geographical, and spatio-temporal. Our study is concerned with spatio-temporal trajectories, which are defined as expressing the position of a traveling object by using spatial coordinates (Spaccapietra et al. 2008, 126-146). As Parent et al. state in their detailed survey, raw trajectories are good for applications that aim to locate a mobile object or analyzing spatio-temporal characteristics of trajectories, but most applications make use of some context for these trajectories (Parent et al. 2013). Providing this application-specific context gives us semantic trajectories.

Semantics of a trajectory however, can be miscellaneous and specific to the context at hand. Spaccapietra et al. defines the semantic facet of trajectories as the application-oriented meaning and its related characteristic. According to Alvares et al., enriching trajectory data with semantic geographical information is the most important part of trajectory data analysis applications. In the same paper, they also present a model for extracting *stops and moves* information from trajectory data. In summary *stops* can be defined as the parts of a trajectory where the object has stayed for at least a minimum duration and *moves* are the parts where the object uses to move between *stops* (Alvares et al. 2007).

There are also few other papers dealing with finding trajectory patterns using some variation of *stops and moves* approach (Giannotti et al. 2007, 330–339; Monreale et al. 2009, 637–646; Wang et al. 2013, 100–111; Zhang, W., Wang, and Huang 2019). Data used in this study already contain the *stops* information in the form of visited store ids, easing the burden of computation of finding *stops and moves* of trajectories.

This study will make use of semantic trajectories with contextual information including the *stops* (store visits) along the trajectory, visit durations and days/times of visits. Trajectories consisting only spatial information (GPS coordinates) and lacking any semantics will be noisy and contain minute details about the customer's movement that will not be of much use for the purposes of this study. This point is also backed up by the work of Ying et al., where their semantic behaviour based prediction model outperforms the model that only utilises geographic behaviour (Ying et al. 2011, 34–43).

2.4 Neural Networks

Carbonell asserts that when a learner has a greater capacity to infer, the teacher or the external world needs to do less to teach a task to learner. Rather than explaining every step of the task, it will be much easier to teach by just showing how other similar tasks are handled. This is the idea behind the usage of neural networks in machine learning, they are meant to be general purpose learning tools which require little or no task specific knowledge (Carbonell, Michalski, and Mitchell 1983, 3-23).

There are two main types of neural networks according to the architectures they use; Feed-forward networks and recurrent (feedback) networks. The most important difference between the two architectures is that recurrent networks have loops in their structures due to feedback connections, while feed-forward networks have none. The loops in their structure cause recurrent neural networks to have a memory. This means that when a new input is given the output is not only affected by that input but also by the previous states of the network (Jain, Mao, and Mohiuddin 1996, 31-44). This feature makes them especially useful for processing sequential data, like trajectory data which this study uses an example of.

2.4.1 RNNs

In their review Lipton et al. explain that while it is possible to use a standard (feed-forward) network to model sequences, by way of integrating sequence information into current input, it will not be feasible to reason about long range dependencies this way. Instead, the limitation of having restricted number of time-steps embedded in inputs can be overcome by including memory cells that can carry information to adjacent time-steps, thus introducing a notion of time to the model (Lipton 2015).

One of the earlier works utilising this idea is Elman's solution. He argued that time need not be explicitly added as a dimension to the input but rather, it could be interpreted by the effect it has on the process itself. His proposed method is to make the neural network dynamic with the addition of parts that are sensitive to temporal sequences (Figure 3). This is essentially the *memory* part in recurrent neural networks. One of the important conclusions he came to was that, due to the unpredictable nature of the sequential data, the network's output error will vary, and these variations could be used as a feedback to the system (Elman 1990, 179-211). In Figure 3, it can be seen that hidden units are connected to context units. Following time steps will then utilise the information from preceding context units (Lipton 2015).

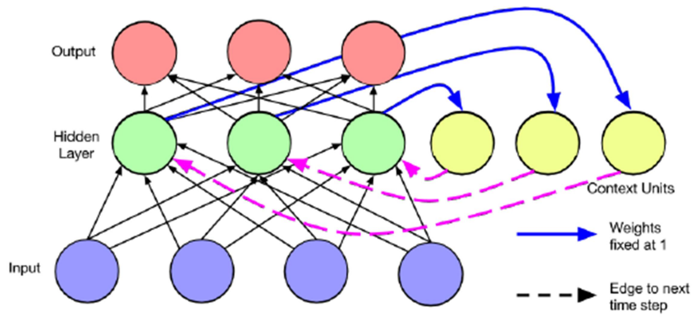


Figure 2 Simple RNN structure proposed by Elman

2.4.1.1 Backpropagation

Backpropagation is a learning procedure described by Rumelhart. It essentially gives the neural network the ability to learn from previous network states by adjusting the weights so that the difference between actual output and desired output are minimised (Rumelhart, Hinton, and Williams 1986, 533).

This is done according to a loss function, which penalises larger differences between actual and expected outputs more. Backpropagation calculates the derivatives of this loss function by using the chain rule with respect to each weight in the network. These weights are then updated using gradient descent, as shown in the following equation (Lipton 2015).

$$w \leftarrow w - \mu \nabla_w F_i$$

where μ is the learning rate and $\nabla_w F_i$ is the gradient of objective function w.r.t. weights.

For the recurrent networks, this weight update should be done for each time step rather than only considering the input and output of the whole model, and this method is called *backpropagation through time*.

2.4.1.2 Vanishing gradients problem

The capability to keep a memory of past data mentioned here is achieved by keeping a hidden state for each time-step. This hidden state is a function of the output of the neuron and previous hidden state. At any time-step t RNN takes the input x_t and updates the hidden state (h) for respective time-step as following;

$$Z_t = Wh_{t-1} + Ux_t$$

$$h_t = f(z_t)$$

where W is recurrent weights and U is the input weights of the network, and f is a non-linear activation function. If the input sequences are very long, Chandar et al. state that using saturating activation functions can cause gradients to vanish during calculations of loss for time-steps. They propose using non-saturating activation functions, e.g. ReLu, to prevent this issue (Chandar et al. 2019, 3280-3287).

2.4.1.3 Overfitting problem

Another common issue while training neural networks is overfitting. Overfitting can be described as the situation where network learns how to predict given training data with very high accuracy, but is not as effective when it is exposed to unseen data, thus lacking the ability to generalise well. Dropout is one of the commonly used techniques to prevent overfitting. It works by deactivating some of the neurons on the network randomly at each epoch. Both (Pascanu et al. 2014; Srivastava et al. 2014, 1929-1958) get successful results using this technique. However Srivastava et al. note that using dropout can slow down the training process and suggest that using regularisers and lower dropout ratios can alleviate the problem.

2.4.1.4 Gated Units: *GRU (Gated Recurrent Unit) and LSTM (Long Short-Term Memory)*

There are also other recurrent network types that rely on modified cell (neuron) types. The most notable additions to this area are LSTM and GRU networks.

LSTMs are proposed by (Hochreiter and Schmidhuber 1997, 1735-1780), and the main problems it is concerned are vanishing or exploding gradients. The method they use to solve these problems is to keep the error flowing backwards through time constant by replacing regular RNN cells with memory cells which add special state units and gates to regular neurons. They claim that this way LSTMs can be effective learners even at time-steps higher than 1000.

GRU is a more recent modification on LSTM networks. It does not have a memory cell like the LSTM but instead has two gates for resetting and updating the activation value. The differences between GRU and LSTM can be seen in Figure 4, where (a) *i, f* and *o* represent the input, forget and output gates, respectively. c and $\tilde{c}$ denote the memory cell and the content for new memory cell. (b) r is the reset gate and z is the update gate, and h and $\tilde{h}$ represent the activation and the candidate activation (Chung et al. 2014).

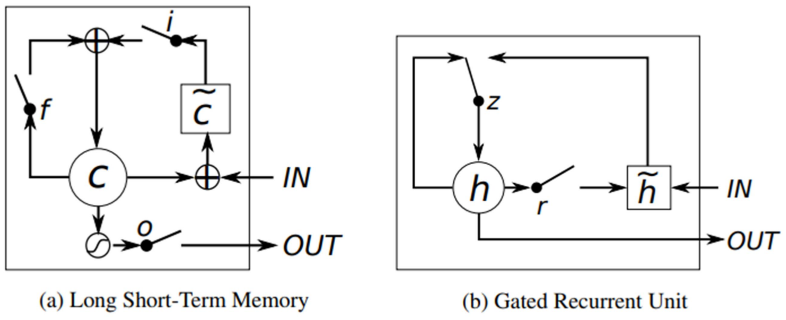


Figure 3 The difference between LSTM and GRU structures

As Chung et al. state, the main difference between simple RNN cells and LSTM/GRU units is simple RNN cells do not accumulate activations from previous time-steps but only replaces it with the new one, while gated units have the ability to add up previous activations through time. As a conclusion they claim that LSTM and GRU perform better than traditional recurrent units, and that GRU's performance is comparable to that of LSTM's (Chung et al. 2014).

2.4.1.5 Architectural types

There are a few different types of RNN architectures when number of steps in input and output sequences are considered. Following figure by Karpathy gives an overview for these types of networks. Red boxes represent inputs, blue boxes outputs, and green boxes

RNN’s state. Some usage areas from left to right: 1) Image classification (not RNN) 2) Image captioning 3) Text sentiment analysis 4) Machine translation. 5) Video classification (Andrej Karpathy 2015).

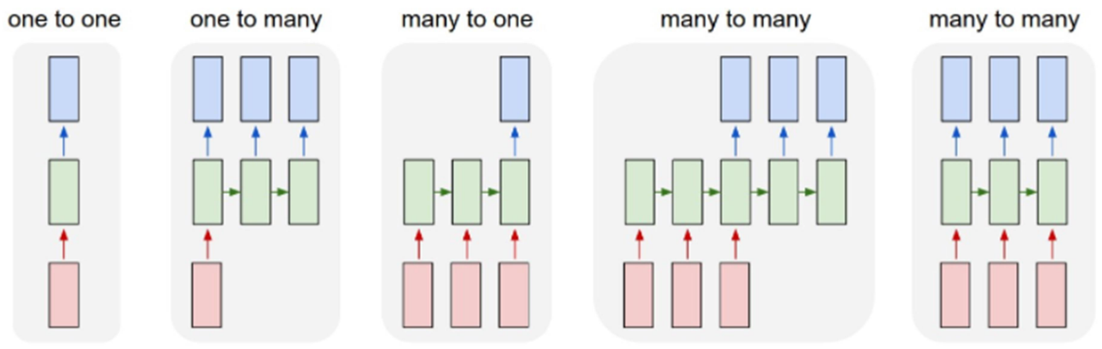


Figure 4 Neural network types

Most suitable network type for this study is a many-to-one network, since we are creating a model that takes a sequence of multiple store visits in a trajectory as input and produce the subsequent store visit as the outcome.

2.5 Recommender Systems

Recommender systems are software tools and techniques whose task is to provide useful suggestions to users for items (Ricci and others 2011). Most prominent techniques used in building a recommender system are;

- Content-Based Filtering, builds a representation of the item by analysing it, for example counting the terms and phrases in a document (Alag 2008). This way feature vectors are computed for items and similar items to the one in question can be found using these vectors
- Collaborative Filtering, whose purpose is to provide recommendations or predictions using the data from like-minded users (Sarwar et al. 2001)
- Session Based Recommender Systems are useful when history of the user’s past behaviour is not available and recommendations have to depend only on user’s actions on the current session

The data used by a recommender system on the other hand are classified by Ricci et al. as items, users, and interactions. For concerns of this study, *items* are the stores that will be recommended, *users* are customers, and *transactions* are the occasions of visiting a store. *Context*, displayed in Figure 2 (Alag 2008), is the stores in the mall from which we will make recommendations. Our system should be able to sift through the context to make relevant recommendations, for example a jewellery store should not be recommended to a customer whose visitation pattern resembles one that contains mostly toy stores.

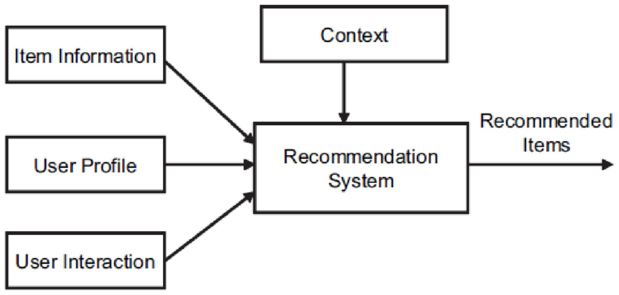


Figure 5 Basic recommender system structure

There are two main types of recommender systems according to data they use; item-based and user-based approaches. In item-based systems, items similar to each other are determined first, and when a given user interacts with one of the items in the list, another item they have not yet shown interest from the list can be recommended to this user. In user-based systems however, the similarity between the users rather than the items are taken into account. This similarity then makes it possible to make recommendation to a user for an item that is liked by one of the users from the list of similar users (Alag 2008).

In their paper, Sarwar et al. analyse different item-based algorithms, and conclude that item-based systems are more effective and better at providing successful recommendations (Sarwar et al. 2001). *NextStore* will also use an item-based collaborative filtering approach to producing recommendations. Using collaborative filtering has the advantage of treating the stores as black-boxes (Alag 2008), meaning that it is not necessary to have any explicit information about the stores and the similarities can be calculated using the interactions (visiting a store in this context) of the customers' with stores.

However, in the case of this study there is also a possibility to produce a system which can also take the store attributes into account in the future. This can be done through incorporating segment-belonging vectors for each store into the recommender model. These vectors will indicate in categorical terms which predefined segment a store's business belong to. To give an example if we take 10 as the maximum belonging value, a store might be 7 on women's clothing, 7 on men's and 4 on kids, 2 on jewellery… etc.

It is possible to construct a hybrid system that also incorporates a user-based approach, in the cases where the customer in the shopping mall provides some personal information while signing on to complimentary WI-FI network, which is not a case that will be considered in this study.

As stated by Altaf et al. many location recommendation models focus on geographical, temporal and semantic influence and built using matrix factorisation technique, which ignores the sequential nature of trajectory movements. This is where using recurrent neural networks can provide an advantage (Altaf, Yu, and Zhang 2018, 937-942).

2.5.1 Neural networks in recommender systems

There are some recent works adopting neural network architectures for implementing a recommendation system. Cheng et al. combines a linear model and a deep multi-layer feed-forward network and jointly train these to produce a recommendation (Cheng et al. 2016). He et al on the other hand build an ensemble architecture, where two different models are trained separately and then the outputs are combined (He and others 2017).

Hidasi et al. present a session-based recommender system utilising RNNs, comprised of GRU cells, to solve the problem of having only short session-based data rather than a long history of the user's behaviour. Relevant to convictions of this study, they also point out a problem with item-to-item similarity matrix solution for the session-based recommendation. Generally, a similarity matrix from the available historical session data is produced and when a user interacts with an item in current session, similar items are found and a recommendation is made. They deem this technique is effective but only takes into account the last interaction of the user and not the sequence of actions in that session (Hidasi et al. 2015). Finding a solution to this problem is the motivation behind using RNNs, as they are proven to be successful at handling sequential data.

Meanwhile Cheng et al. highlights an important challenge in finding the most relevant items to recommend. They first define the two concepts related to this; memorisation and generalisation. Learning the co-occurrence of items or features and finding the correlations in historical data is defined as memorisation, where generalisation is the focus on finding new correlations and combinations between items that have rarely or not at all occurred in the past. The outcome of this distinction is that, when a recommender system is based on memorisation the items recommended will be directly related to the item that has been interacted. On the other hand, if using generalisation, a more diverse set of items can be recommended by the system (Cheng et al. 2016).

What route does this study take in light of these definitions? *NextStore* will be implemented using an item-based Collaborative Filtering and Session-Based recommendation approaches together. While Collaborative Filtering provides item similarities derived from the like-minded customers' behaviours in the form of clustered intents, new-coming customer's movement data in a single session is progressively attempted to be fit into one of these intents without providing any other background information about that customer. Finally a recommendation is produced by evaluating the common movement patterns in related intent groups. Also, it will attempt to produce a system that can generalise well, rather than depending on pure memorisation, which is the reasoning behind using a fuzzy clustering algorithm to find customer intents. After all, marketing gain from incentivising a visit to a store that would not be naturally in a customer's radar should exceed the gain from giving them yet another one of their favourite store as a recommendation.

2.6 POI Recommendation / Making Predictions Using Trajectory Data

The research area of next location prediction has been getting attention in recent years. There are mainly two paths these studies follow; using probabilistic models and using neural networks.

Kim et al. argues that there is a causality between sequentially visited places and uses a Bayesian network model to predict users' next possible location from their visit history and collective visit patterns of other users (Kim et al. 2011).

(Yao and others 2017) and (Liu et al. 2016) on the other hand use recurrent neural networks to make next location predictions for users. Yao et al. use embeddings of semantic trajectory information, while Liu et al. use spatio-temporal transition matrices to train their neural network models.

Zhao et al. proposes a modified version of GRU network that contains spatio-temporal gates for finding spatio-temporal relations in users' check-ins (Zhao et al. 2019, 5877-5884).

Altaf et al. also use a recurrent neural network for the prediction task, but enrich the model with attention mechanism which they claim to allow the model to learn how spatial and temporal differences between successive visits affect the patterns (Altaf, Yu, and Zhang 2018, 937-942). Studies from Feng et al. and Zhang et al. similarly utilise an attention mechanism focusing on capturing the historical periodic patterns of human movements (Feng et al. 2018, 1459-1468; Zhang, X. et al. 2020, 1-8).

Sun et al. also studies a user's own movement history to make predictions for next location. They argue that RNN-based method lacks the ability to model user's long term preferences and the geographical relations amongst recent visits. They use two separate LSTM networks for modelling these attributes (Sun et al. 2020, 214-221).

3 Methodology, Data, and Design

NextStore application will be suitable to be available as a standalone application or a service that can be integrated into a mobile application provided by the mall for their customers.

3.1 Application and Interacting with the Customer

3.1.1 Location data collection

Complimentary WI-FI systems usually require the mobile device users to go through some kind of login/confirmation process if they want to use the WI-FI service offered. Some of these systems can ask for a mobile phone number, for some it is enough to provide an email address, and sometimes clicking on a button on a webpage anonymously is enough to be able to use the service. This process can also be provided through a mobile application provided by the shopping mall for use inside the facility. In any of these cases, the information provided by the user to access the WI-FI facility does not have to be extensive, and when they agree to use this facility, at the very least it will be possible to track these customers' location within the mall.

However, in the case the customer decides the use the application provided by the mall, it is also possible to communicate to this customer during their visit. The kind of mobile services provided by the mall can be presented to the customer, and they can opt-in to any of them. *NextStore* recommender can be formulated to be one of these integrated services.

When a customer starts using the application, it will be possible to tell if that customer is a first-timer or a revisiting customer, assuming that the customer still has the application or in the case of reinstallation, still uses the same unique identifier, such as the same e-mail address. Even if none of these cases are true, their mobile device's unique id can be stored and queried to decide if they are new to the mall or not.

As stated in the *Limitations* section, this study goes with the assumption that all location data is retrieved from first-time mall customers. This prevents us from devising separate intent groups for these two types of customers.

3.1.2 Providing recommendations

Location data for the customer will in any case be logged, as long as they agree to the terms of WI-FI service, but they need to opt-in to *NextStore* service to get store recommendations throughout their visit inside the mall. At this point, it can be argued that, rather than providing a neural net based recommender system, we could interact with the customer on their arrival and ask them to select one of the provided *intents*.

The first problem with this approach is that these ready-made intents need to be supervised, meaning, the intents found by the algorithm should be analysed and grouped under *human-readable* labels such as "Suit hunter" for menswear, "Gold digger" for jewellery store enthusiasts… and so on. The number of these groups could quickly escalate, especially if we consider the possible sub-groups (interested in menswear, but cheap/expensive? casual/classic?... etc). Complications also arise when it is considered that most people tend to visit cafes and restaurants or entertainment facilities in between their store visits. This behaviour is hard to represent with ready-made classifications since it will multiply the available choices.

Another point is that, not all customers come into the mall with a clear intention of what they want to get out of that shopping mall experience, perhaps more so in the case of new-comers. Sometimes they go in and browse a couple of shops, related or unrelated, and they see an item that arouse their interest but not just perfect enough to buy and they may want to search for similar items in other similar shops.

And finally there is the aspect of sequential visitations. The proposed system in this study takes the sequence of historical store visits for an *intent* into account and produces the recommendations accordingly. If a solely classification based system was to be used, the next store recommendation would need to be made from stores belonging to the detected class (*intent*) either randomly, or based on location proximity.

Proximity of the stores to the current location of the customer may not always be the defining factor. That would disregard the intra-intent segment variations. Consider the scenario where there is an intent cluster which contains 4 menswear stores, 1 fast food restaurant, and 3 electronic appliances stores. The common visitation patterns found in the cluster do not necessarily have to be based on location proximity. If the visits are around lunch time, people may visit 4 menswear stores back to back and then feeling peckish, they may opt to head for the fast food restaurant 2 floors above rather than visiting an appliances store 10 meters away. Using a neural network structure that takes the sequential nature of the visits into account can also make the distinction between different behavioural patterns during different times of different days or different days of the week, unlike the fixed mind-set of a purely proximity based recommender.

Considering these scenarios, it is plausible to use a recurrent neural network based recommendation system for mobile customer movements.

3.1.3 Getting feedback from customer

Throughout the customer's trajectory, the application will actively observe their movements via location data. It can evaluate the intent membership probabilities after each store visit of the customer (It is however reasonable to set some threshold for the minimum store visits as will be discussed later on).

Feeding these membership probabilities along with other contextual information to a neural network model, it will produce some recommendations with probabilities for each store. Depending on the application of the system, the highest or first *n* highest recommendations are presented to the user through the user interface of the application. In a real life setting, it is easy to collect the feedback for these recommendations. User of the application can rate the presented recommendation as being useful/not useful or by any other system of rating the context requires. This feedback can even be used while training the network after this user's session has ended and their trajectory became an input data for training.

However, the prototype produced in this paper does not have the possibility to collect real-life feedback at the time of writing. For this reason, rather than evaluating the effectiveness of the system through collection of user feedback, we will treat the produced recommendations as predictions, and train the neural network using the next step in the trajectory as being the expected outcome. Even though the produced ratings will not represent the real-life effectiveness as a recommendation, it should still provide a good insight of the system's capabilities of finding relevant stores.

3.2 Data

Data from a real life shopping mall is used during the development of the artefact. Main parts of the data are provided in separate files, consisting of store metadata information (referred to as *zones*) and location data for the customers with mobile devices. For privacy reasons the store names will be substituted with generic names.

3.2.1 Store data

Provided in *zones.json* is the metadata information for the available stores inside the shopping mall. This metadata consist of;

- **id:** a unique id for the store
- **floor_id:** unique id of the floor where the store resides
- **name:** public name of the store
- **coordinates:** location of the store inside the mall, presented as coordinates of a polygon

```
{
  "id": "364",
  "floor_id": "93",
  "name": "SomeStore-1",
  "coordinates": "[[128.01117716726387, 181.6765073424279], [138.42831824495954, 181.6765073424279], [138.48066568756104,
  175.9559571671161], [128.01117716726387, 175.90347505541604], [128.01117716726387, 181.6765073424279]]"
}
```

Figure 6 A sample entry in *zones.json*

3.2.2 Location data

Mobile customers' location data is provided in *location_logs.json* file. Every time the location of a mobile device is polled, an entry created for that device at that timestamp. Figure 7 shows an example entry from this file.

```
{
  "_id": "5facec8c1bg3440340606e8f",
  "coordinate_x": "125.52",
  "coordinate_y": "39.55",
  "floor_id": "93",
  "building_id": "88",
  "tenant_id": "18",
  "session_id": "69597465",
  "visit_type": "1",
  "created_at": "12/08/2020 08:24:26.756",
  "current_zone_id": "366",
  "previous_zone_id": "",
  "updated_at": "12/11/2020 08:23:38.357"
}
```

Figure 7 A sample entry in *location_logs.json*

The fields that are relevant to our study here are;

- **_id:** a unique id for the entry
- **session_id:** unique id of the session, indicates a single visit of the given device to the mall from entry to exit
- **coordinate_x:** x coordinate of the location w.r.t. origin point
- **coordinate_y:** y coordinate of the location w.r.t. origin point
- **created_at:** timestamp this log entry is created at
- **updated_at:** timestamp this log entry was updated at
- **floor_id:** id of the floor given GPS location is at

- **current_zone_id:** the zone (store) this device is currently at, at the moment of polling
- **previous_zone_id:** the zone (store) this device was at prior to current one

Each record in location_logs represents a momentary location of the customer during their visit. Along with coordinates inside the shopping mall, the zone (store) they are currently inside is indicated in this record. In the context of our application these zones represent the *stop* points of a trajectory.

3.3 Preprocessing the Data

Location data is imported as a Pandas DataFrame object. Main preprocessing steps applied on the data are data-cleaning, creating semantic trajectories, creating a matrix for store-visits for sessions, dimensionality reduction using Singular Value Decomposition, and splitting the data for training and test sets.

3.3.1 Primary data cleaning

As stated in *Location data* section not all fields of these files are necessary for the application, so they will be removed, and only the relevant columns are kept in the dataframe.

3.3.2 Semantic trajectories from raw trajectory data

At this point we have GPS location, timestamp, and zone information for each session. A session in this context can be defined as a single visit to the mall by a single mobile device, a.k.a. customer. This data is still cluttered with unnecessary data points those should be removed or grouped together. A couple of points to consider at this step are as follows,

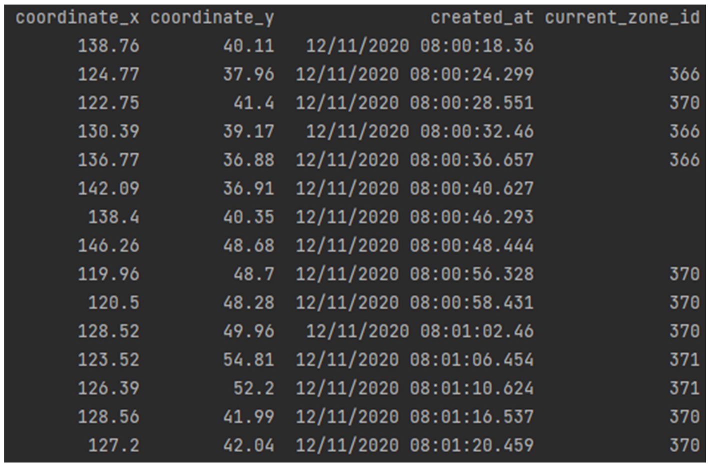
- One store visit is likely to be represented by many data points in the location data
- Location data is not always accurate, meaning there are likely deviations on data from the actual GPS location of the customer

- There can be sessions those do not fulfil the minimum threshold for store visits or minimum stay duration inside a store

After this processing step, we want to acquire a dataset with trajectories that have the contextual semantic information, namely, store visits with visit durations, visit hours, and week days of visits and ordered by visit timestamps.

Parent et al stresses that the main target of trajectory data cleaning process is to eliminate GPS data errors (Parent et al. 2013). This study is no exception and the unreliability of GPS data points are the main factor that makes this step of pre-processing challenging. It is possible to come across session records in data, which indicates that a customer has been in zone_1 at time *t*, in zone_2 at time *t+2* seconds, and zone1 again at *t+5* seconds, and it can go on in a similar unrealistic fashion. Furthermore, data does not only contain customer movements, but also the movement data of staff, which will not be a welcome contribution for the purposes of this study.

A session trajectory data where this unreliable data can be demonstrated is shown in Figure 8. In this excerpt from data, the mobile device is observed to be in zones with three different ids, 366, 370, 371. However, careful look reveals that transformations between zones are quite unrealistic; a move from zone 366 to 370 is indicated just after 4 seconds and then back to zone 366 after 5 seconds. GPS data at these rows show that measured coordinates at given times are close to each other, which suggest that these zones are probably next to each other or in very close proximity otherwise. Therefore, if the measured location at the polling time is not highly accurate, using *current_zone_id* information alone will be misleading.



```
coordinate_x coordinate_y              created_at current_zone_id
      138.76        40.11   12/11/2020 08:00:18.36
      124.77        37.96  12/11/2020 08:00:24.299             366
      122.75         41.4  12/11/2020 08:00:28.551             370
      130.39        39.17   12/11/2020 08:00:32.46             366
      136.77        36.88  12/11/2020 08:00:36.657             366
      142.09        36.91  12/11/2020 08:00:40.627
       138.4        40.35  12/11/2020 08:00:46.293
      146.26        48.68  12/11/2020 08:00:48.444
      119.96         48.7  12/11/2020 08:00:56.328             370
       120.5        48.28  12/11/2020 08:00:58.431             370
      128.52        49.96   12/11/2020 08:01:02.46             370
      123.52        54.81  12/11/2020 08:01:06.454             371
      126.39         52.2  12/11/2020 08:01:10.624             371
      128.56        41.99  12/11/2020 08:01:16.537             370
       127.2        42.04  12/11/2020 08:01:20.459             370
```

Figure 8 Unreliable location data

Since without sequential store-visit information, these trajectories are not of much use for this application, we need to provide a plausible approximation for the customer's movements. Considering the absence of a real-life agent which can be tracked to calibrate these measurement, it is not possible to perfectly align these data to real-life locations, but a correction algorithm will be applied to get a sequence of store visits with reasonable accuracy.

3.3.2.1 Trajectory pre-processing

3.3.2.1.1 Removing anomalous data points

The first step in the process is to remove anomalous (unrealistic) location points from session data. The main idea here is that, when a person is moving inside the mall, their velocity of movement would not deviate from an average value, i.e. they would go similar distances in similar durations rather than walking around leisurely and then running about.

Over the session data sorted by timestamp of creation time, we estimate the *unit velocity* of the movement for each location point change. Unit velocity, v, to reach a location can be expressed as the following, where l is location at point n, and t is the time of measurement at the same point, expressed in milliseconds from epoch;

$$v_n = \frac{|l_n - l_{n-1}|}{|t_n - t_{n-1}|}$$

The average of these velocity values is then used to calculate standard deviation of movement velocity for the session. If a data location is accessed with a velocity above one standard deviation of session average, then that point is deemed anomalous and is removed. One improvement to this step could be to ignore timesteps where location change is zero, or near zero, to compensate for customer's immobility, such as while sitting at a restaurant.

3.3.2.1.2 Clustering zone visits

After removing the unrealistic points from trajectory, it can be assumed that most of the remaining location readings will at least be in the ball-park of the actual positions of the customer. This means a clustering algorithm, that groups location points while taking the temporal aspects of the locations into consideration, can provide a good approximation for actual zone visits.

To this end we utilise ST-DBSCAN algorithm, which is detailed in *section 2.2.1.2*. ST-DBSCAN will produce clusters of location points which are spatially and temporally close to each other, therefore these clusters can be seen as ideal representations for a store visit. X and Y coordinates of the locations are used as the spatial property of the points, while timestamps at location polling are taken as temporal values. However, to assign a zone to represent the visit for a given cluster, the most frequently occurring *current_zone_id* inside the cluster is chosen, instead of using x and y coordinates. This k-nn like approach simplifies the calculations and reduces computational load.

After a sequential store-visit pattern is retrieved from clustering, successive visits to the same stores are merged and visit times are updated accordingly. This prevents the occurrence of visit patterns where a customer visits a certain store back-to-back.

3.3.2.1.3 Trajectory pruning

At this step, the trajectories that resemble a non-customer's movements are removed. Namely,

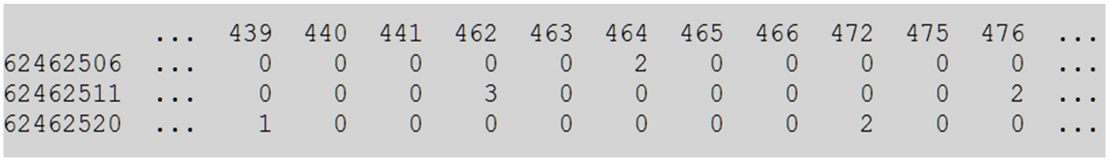
- Trajectories that have location records outside the working hours (up until 15 minutes later than closing time) of the mall. These are probably staff trajectories.

- Trajectories that span a time period of longer than 5 hours
- Trajectories that have visited 15 stores or more

After this data cleaning process, remaining location data is grouped by session and the sessions that have less than 3(configurable) zone visits are discarded as they are deemed insufficient to indicate an intent. Lastly, each zone visit is enriched with the information of binned values for duration, week day, and time of day.

3.3.3 Creating session-store-visits

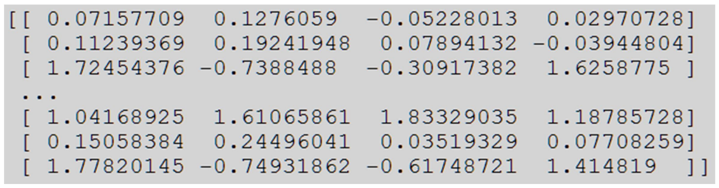
To cluster the intents of customers, each session's store visits are converted to a one-hot vector, without taking the sequence of visits into account. Instead of using a binary value (0/1) to indicate a store visit in this vector, binned value for the duration is used, i.e. 1=in&out visits, 2=average shopping/browsing visits, 3=long stay. The reason for this is to be able to distinguish between quick visits that may indicate a decision of disinterest, normal shopping visits with some browsing time, and movie theatre/restaurant customer scenarios (which would exhibit much longer stay durations). Figure 9 shows a possible matrix that could be formed by this operation.



```
          ...  439  440  441  462  463  464  465  466  472  475  476  ...
62462506  ...    0    0    0    0    0    2    0    0    0    0    0  ...
62462511  ...    0    0    0    3    0    0    0    0    0    0    2  ...
62462520  ...    1    0    0    0    0    0    0    0    2    0    0  ...
```

Figure 9 Sample session-store-visit matrix, leftmost column indicates unique session_id

Session-store-visits vectors produced after pre-processing steps, form a large matrix that will have the shape (#number_of_sessions, #number_of_stores). *NextStore* will use SVD as a dimensionality reduction technique before finding customer intents from the customer visit matrix. After applying SVD we will perform the clustering on a matrix reduced to a shape of (#number_of_sessions, 4), as shown in Figure 9.



```
[[ 0.07157709  0.1276059  -0.05228013  0.02970728]
 [ 0.11239369  0.19241948  0.07894132 -0.03944804]
 [ 1.72454376 -0.7388488  -0.30917382  1.6258775 ]
 ...
 [ 1.04168925  1.61065861  1.83329035  1.18785728]
 [ 0.15058384  0.24496041  0.03519329  0.07708259]
 [ 1.77820145 -0.74931862 -0.61748721  1.414819  ]]
```

Figure 10 Result of SVD operation on session-store-visits

3.3.4 Train/Test split of trajectories

Available trajectory data will be split as training and testing data with the ratios of 80% and 20% respectively. An important point while getting the test data split is that it should be retrieved from the whole array of sessions rather than from trajectory steps. This means that we will have two different sets of session data with their whole semantic trajectories for training and testing.

Since the entire trajectories are set aside for evaluation, we have the opportunity to reconstruct a customer's actual mall visit step-by-step, using a walk-forward evaluation technique. When a customer enters the mall, *NextStore* will not start making recommendations straight away. After the number of visited stores by the customer is over a threshold, e.g. 2 store visits, current set of store-visits can be clustered to find the appropriate intent for the customer at that moment, and a recommendation can be made according to this intent. Since we have the entire trajectory data, we then move through the visits step-by-step and iterate this process after every visit of the customer until their final store visit.

It should be noted that, for training data we consider all forward combinations of the sorted store-visits, whether they start from the first store-visit they make or the 5th, during sub-trajectory creation (section 3.5.1.2). For the evaluation of recommendations this is not meaningful. If we consider the real-life scenario, a customer's intent will be the accumulated store-visits during their presence in the mall. This implies that, their current intent will be calculated from a store-visit vector that includes all visits from the first store up until the last one they've visited.

It might be argued that clustering the training data (finding customer intents) can also be done using sub-trajectories rather than the whole, hence producing what could be called as *micro-intents*. While this kind of clustering might produce better results for a purely prediction-oriented task, it is unlikely to be a good technique for a recommendation task, since having intents for each possible combination (2 store visits, 3 store visits, 4 store visits… and so on) will impede the model's ability to generalise and will focus more on memorisation, ending up trying to find an exact match for a next store rather than a possible extension to customer's current intent.

3.4 Clustering Customer Intents

As stated, this study takes an item-based collaborative filtering approach to find similarities amongst items to produce a recommendation. The first step of finding these similarities is collecting users' ratings for items. These ratings can be explicitly given by the user, as in selecting from a set of ordinal ratings, such as rating a movie 1 star or 4 stars on a movie review web-site. Ratings for the items can also be retrieved implicitly without the need for the user to give a rating explicitly. An example of this is a user's watching a video on a streaming web-site without giving any rating. The action of watching the video can be considered as an implicit rating for that video from that particular user. In these cases a binary value is usually the rating, such as 0 for not watched videos and 1 for watched videos. In this light, the session-store-visits matrix in our study can be thought of as user ratings, since every visit will provide a positive implicit rating. The ratings however, will be an indicator of visit duration rather than a binary flag.

After collecting the item ratings from users, we apply a clustering model to find similarities. This method treats the task as a classification problem whereby clustering similar users into similar classes makes it possible to find the probability of a particular user being in a particular class, thus allowing the probabilities for ratings to be computed (Sarwar et al. 2001).

The result of clustering in our system is the *intents* (of customers), each of which contains the customers (sessions) those display similar store-visitation patterns inside the mall and the stores they tend to visit in single unique sessions. This implies a similarity amongst the stores belonging to the same intent, perhaps not by segment nor by product sold, but by tendency to be favoured by mall visitors with similar intentions.

3.4.1 Singular Value Decomposition (SVD)

Singular Value Decomposition is a method for representing the most relevant information from a large matrix in lower dimensions. Given a matrix A, SVD decomposes this matrix into three matrices;

$$\boldsymbol{A = UDV^T}$$

Where *U* and *V* are square matrices and *D* is a diagonal matrix with singular values, sorted in descending order to indicate the importance of its axis in representing matrix *A*. After acquiring these three matrices, one can use the first *k* singular values to get an approximation of *A*.

Su et al. explain in their survey that SVD is one of the data reduction techniques in collaborative filtering that can be used to alleviate the data sparsity problem, by removing unrepresentative and insignificant data points (Su and Khoshgoftaar 2009). However they also warn that this removal of “insignificant” data may also cause important information to get lost, degrading the quality of recommendations.

3.4.2 Fuzzy C-means clustering

Clustering is done using the fuzzy c-means algorithm, using cosine similarity as the distance metric. The reason for using a fuzzy clustering method is that a recommender system should have the flexibility to be “creative” in a sense. If a hard (single-cluster belonging) clustering method was used, the recommendations later on would only have limited set of possibilities to present to the customer (from within only stores belonging to that certain cluster). Usage of fuzzy clustering will give us the possibility to classify a session as belonging to **Intent-A** with 60% possibility and **Intent-B** with 17% possibility.

Since these membership ratios for each cluster will be added as inputs to the neural network model, it will be possible to produce a wider array of recommendations.

This clustering algorithm requires the number of clusters to be given beforehand. Using empirical knowledge gained from experimenting with hierarchical clustering, the number of clusters for this study will be taken as 11 (configurable).

3.4.3 Implementation of clustering

As stated the number of clusters is taken to be 11 for the fuzzy c-means algorithm. For input data points to clustering algorithm, the output matrix of SVD operation (with 4-dimensions) on session-store-visits is used (e.g. Figure 10). This matrix does not contain sequence information, but only all the visited stores and corresponding stay-duration values indicated by binned values.

Figure 11 shows a section from an output of a sample clustering result. The key in the map denotes the unique session (visitor) id, and the value is an array indicating the probabilities of that session belonging to the intent represented by that index of the array.

```
{
 '62462511': array([4.85346146e-12, 1.47472551e-11, 7.01022932e-08, 1.02415645e-11, 7.49501734e-10, ...
 '62462520': array([9.98861446e-01, 8.58167832e-08, 4.10163887e-09, 3.08040692e-06, 2.10370730e-08, ...
 '62462541': array([1.77034774e-12, 9.33184641e-12, 9.99999987e-01, 1.01049242e-12, 1.61684957e-11, ...
 '62462546': array([5.19593095e-06, 9.96630632e-01, 1.05298613e-05, 1.13700731e-06, 3.93311838e-07, ...
 ...
}
```

Figure 11 Cluster membership ratios for each session

3.5 Creating the Neural Network For Recommendations

The neural network architecture this study uses consists of two separate models that are jointly trained to produce a recommendation for the customer.

Recurrent neural networks are the go to choice for processing sequential data because of their ability to keep a memory of previously trained data. Due to the apparent sequential and time dependant nature of the trajectory data, our neural network architecture will contain a recurrent section that will be trained using these trajectory sequences.

Second part of the network is a feed-forward deep network that takes cluster memberships as an input. This section essentially acts as an embedding layer for cluster membership values.

3.5.1 RNN architecture

For the sequential trajectory data, a stacked deep RNN architecture is created, which is thought to provide much more efficient representations for some functions than single layer networks (Pascanu et al. 2014). A non-saturating non-linear function, e.g. leaky ReLu, will be used as activation functions to prevent vanishing gradients problem. Dropout is also added to the network to stop it from overfitting.

3.5.1.1 Input information

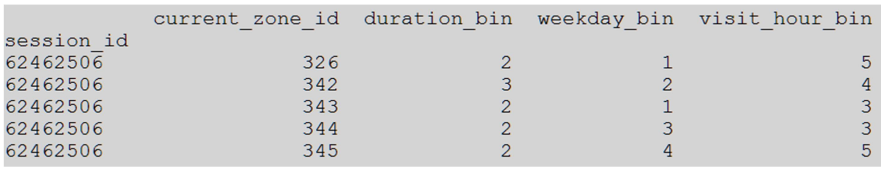
The input for this network is the sequence of customer's zone visits with accompanying semantic information. This visit sequence consists of the following attributes; *zone_id*, binned *duration*, *day of week* and *time of day* values. Resulting trajectory can be formulated as;

$$T_x = \{s_1, s_2, s_3, \dots s_t\}$$

where T_x is the whole semantic trajectory for session *x*, and s_t denotes the store visit at time-step *t*, sorted by timestamps. Since store visits include other information it could be presented as a tuple;

$$s_t = (z_t,\ d_t,\ w_t,\ h_t)$$

Where, for the given time-step t, *z* is the id of the zone (i.e. store), *d* is binned stay duration inside the zone, w is the binned value for day of week, and h is the binned hour of day information for the visit.



```
            current_zone_id  duration_bin  weekday_bin  visit_hour_bin
session_id
62462506                326             2            1               5
62462506                342             3            2               4
62462506                343             2            1               3
62462506                344             2            3               3
62462506                345             2            4               5
```

Figure 12 Sample semantic trajectory steps for a session

These attributes are converted to one-hot vectors, since they are categorical values in essence and bear no magnitude information. It can be argued that duration bin could also convey a magnitude, however, the bins in this study represent a semantic meaning for the visit (such as in-out visit, average shopping/browsing, or a shopkeeper stay) rather than the actual duration length. After these conversions to one-hot vectors, all four vectors are concatenated and fed into the network as one big vector.

Embedding information about time and day of visit sequences will give us the ability to distinguish between latent time-related connections between visits. For example, during lunch time weekdays, it may be more probable that higher number of customers visit cafes and restaurants for their lunch break first and then a couple of other stores of their interest. Over the weekends however, we may encounter more customers first visiting the stores and afterwards eating at the restaurants around the same hours. Binning of these values is also appropriate here, since the behaviour during weekdays should resemble one another and be different than the behaviour over the weekend, and weekend days might differ in themselves. Same thinking applies to hour of day values, even though it can be debatable as to what the best binning approach to these values is.

Following binning values are used in this study;

*Duration bins (in seconds)** = [0, 100), [100,180), [900,)

*Weekday bins*** = [sunday-thursday, friday, saturday]

Entry-time(visit) bins = every 2 hours from 00:00 to 23:59

** [inclusive,) exclusive*

*** Data is from a country where Friday and Saturday are the official weekends*

3.5.1.2 Sub-trajectories

Each customer semantic trajectory is augmented by using the sub-trajectories it contains, provided that these sub-trajectories contain more store-visits than the minimum configured (3 for this study). During training, the last store visit is always taken as the expected output of the network. Extending the training data for varying lengths of trajectories in this fashion should help the produced recommendation to be more accurate and more representative for different lengths of trajectories. Figure 13 shows some sample steps that can be created from a full trajectory;

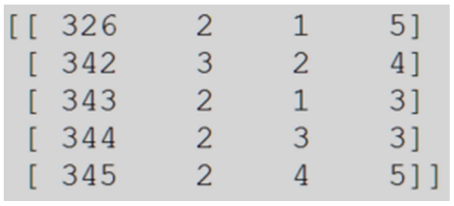
```
Full trajectory:
store 1 -> store 2 -> store 3 -> store 4 -> store 5 -> store 6

Sub-trajectories:
INPUT                                                        OUTPUT
store 1 -> store 2                                           store 3
store 1 -> store 2 -> store 3                                store 4
store 1 -> store 2 -> store 3 -> store 4                     store 5
store 1 -> store 2 -> store 3 -> store 4 -> store 5          store 6
store 2 -> store 3                                           store 4
store 2 -> store 3 -> store 4                                store 5
store 2 -> store 3 -> store 4 -> store 5                     store 6
store 3 -> store 4                                           store 5
store 3 -> store 4 -> store 5                                store 6
store 4 -> store 5                                           store 6
```

Figure 13 An example explaining the of sub-trajectory creation process

3.5.1.3 From trajectory to network input

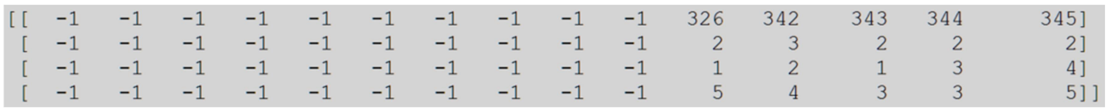
Following from the example trajectory from Figure 12, first step to produce the input for the recurrent model is to produce a tensor that will have semantic data from trajectory as time-steps. So this trajectory data becomes the tensor in Figure 14, where inner columns represent in order; *zone id, duration bin, weekday bin, entry-hour bin* for the time step;



```
[[ 326    2    1    5]
 [ 342    3    2    4]
 [ 343    2    1    3]
 [ 344    2    3    3]
 [ 345    2    4    5]]
```

Figure 14 A sample 5-step trajectory

However, even though it is possible to feed RNNs varying lengths of time-steps in separate batches, in any given batch all sequences should be of the same length. To adhere to this limitation, maximum number of store visits amongst all sessions in whole data is found and trajectory sequences are padded up to this length. All tensors produced in the step above are transformed to a new tensor with a fixed length, while padding the non-existing steps with a dummy value, such as -1. After applying this transformation and transposing the array to fit to training input, our new tensor will look like the one in Figure 15, where now inner rows represent in order, *zone id, duration bin, weekday bin, entry-hour bin,* and columns represent time steps;



```
[[  -1   -1   -1   -1   -1   -1   -1   -1   -1   -1   326   342    343   344        345]
 [  -1   -1   -1   -1   -1   -1   -1   -1   -1   -1     2     3      2     2          2]
 [  -1   -1   -1   -1   -1   -1   -1   -1   -1   -1     1     2      1     3          4]
 [  -1   -1   -1   -1   -1   -1   -1   -1   -1   -1     5     4      3     3          5]]
```

Figure 15 An example 5-step trajectory, padded as a sequence input with fixed length

All features in this matrix are converted to one-hot vectors, which only contain an indicator value (1) at the index which the categorical value corresponds to, and a placeholder (0) value at all other indices.

In the final step, instead of giving all features separately as inputs to recurrent network, they are concatenated at each step and one large one-hot vector is formed. This way, 4x15 matrix at Figure 15 becomes a tensor of 15 rows, indicating a 15 time-step input, where each element holds the concatenated one-hot features for the respective time-step.

It should be noted that, padding values (-1) will also be converted to one-hot vectors, which will in fact be a tensor of the size of the actual input filled with -1 values. Keras library has a feature called *masking*, which is defined as being a way to indicate to sequence-processing layers that some parts of the sequence are missing and they should be skipped during the process[3]. Using this feature, our application is able to feed the recurrent network varying lengths of store-visit sequences, where missing steps are indicated with tensors filled with padding values.

This array of features is now ready to be fed to recurrent network as customer's visit sequence input, an example of which is not presented here but in Appendix A, since it is essentially a large sparse array of zero values.

3.5.1.4 Recurrent cell types

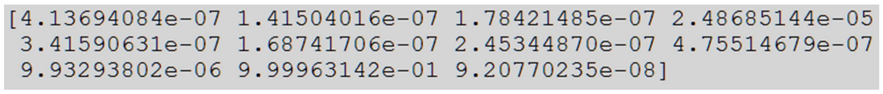
RNN model will be built using simple RNN cells, but GRU and LSTM cells have also been tested and results will be represented in the following section. It is observed that using simple RNN cells in the network is more time-efficient than LSTMs and GRUs, and accuracy of the recommendations produced is comparable to that of gated units. Another reason to prefer RNNs over gated units is that, in a real-life setting sequences for store visits are not long enough to require the long-term memorisation capabilities of LSTMs and GRUs.

3.5.2 Embedding cluster information

Cluster information for sessions is introduced to the network as a separate dense layer. This will be a simple feed-forward Keras Dense layer network which uses the same activation function as the RNN part of the model. It can be thought of as an embedding layer, where the dimensions of the input are reduced and the information is condensed before being concatenated with RNN structure's output.

For each visit sequence available, there will be an accompanying cluster membership array, and this enables the network to model the relationships between visit sequences and respective intent values for that sequence. Figure 16 shows an example input vector to this layer when there are 11 available customer intents;

[3] https://www.tensorflow.org/guide/keras/masking_and_padding



```
[4.13694084e-07 1.41504016e-07 1.78421485e-07 2.48685144e-05
 3.41590631e-07 1.68741706e-07 2.45344870e-07 4.75514679e-07
 9.93293802e-06 9.99963142e-01 9.20770235e-08]
```

Figure 16 A sample intent cluster membership array for 11 intent clusters

3.5.3 Combining outputs of network branches

The output from RNN and dense branches of the model will be combined through another dense layer, with one hidden layer and one output layer. Output layer has a number of neurons equal to the store count in the system. Since this is a classification task, where we want to get a probability distribution over available stores for recommendation, *softmax* is the most suitable activation function for the output layer (Goodfellow, Bengio, and Courville 2016). For this reason, the final layer in all models tried is always a *softmax* layer.

As the loss function for the whole model categorical cross-entropy fits our model, since it is used to calculate the differences between probability distributions over multiple classes.

Figures 17 describe our main neural network architecture with recurrent and feed-forward sections.

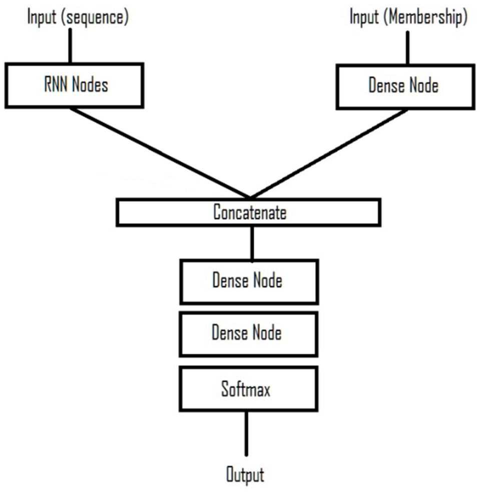


Figure 17 Combined (Sequence + Membership) neural network architecture

Figure 18 shows the second model we use for experimentation. This model only uses the RNN section for the sequential store visit data and disregards the intent membership information for the trajectories.

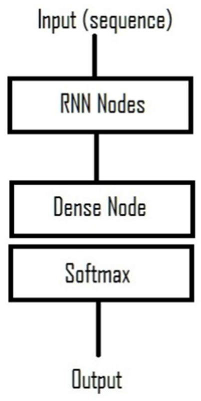


Figure 18 Single (sequence) neural network architecture

3.6 Making Predictions For New-coming Customers

Customers’ location data are continuously polled within the mall as long as they use the complimentary WI-FI service provided by the mall. This application should have access to each new polling event and query the new state of customer’s trajectory.

3.6.1 Cold-start problem

Bobadilla et al. point out the cold-start problem recommender systems can have and this is relevant to *NextStore* during intention querying phase (deciding which intent should be assigned to a current set of store visit steps). They define this problem as not having enough ratings (which corresponds to store visits in this study) to make a reliable recommendation caused by initial lack of ratings and define three kinds of cold-start problems (Bobadilla et al. 2013, 109-132);

- *New community problem*: This is defined as the hardship of obtaining initial data for the new recommender system, which is not an issue for *NextStore* since it can be initialised on already accumulated large data of customer mobility

- *New item problem*: This problem is caused when new items are introduced to the system and therefore no ratings being available for them. The items in this application, stores, are more or less static. However, that is not to say there will never be removals or additions to the set of stores. In these cases, it might take a while to for the application to adapt to new intent clusters that might be formed.
- *New user problem*: This problem can be encountered when a user is new to the system. Until they have provided enough ratings for items, they may not get decent recommendations.

The most relevant problem for *NextStore* is the new user problem. Upon their immediate arrival to the shopping mall the system can not have any idea about their intentions since it will not have any personal information. It will take some time for a customer to visit some stores, hence implicitly disclose their intentions for being there. So to minimise this type of cold-start problem we impose a minimum count of store visits before making a recommendation for the next visit.

3.6.2 Intentions and recommendations

Once the customer sets foot to the mall, after every store visit, customer's trajectory at given moment will be subjected to the same pre-processing steps as the training data. Then, cluster membership vector is found for store-visits up to that moment, using the same parameters from already trained clustering model. This gives us customer's current intent according to stores they have visited.

Then current semantic trajectory of the customer is used as input to the neural network model as a sequence of store visits. The output of the model will contain a probability vector for next store recommendation. This probability distribution presents the likelihood of customer's interest in any given store. *NextStore* can make its recommendations by taking the highest *n* probabilities from this vector.

4 Results and Analysis

During evaluation of the test data, we define the concept of *prediction ranks*. For a given input, we check the produced probabilities for the next store visit and find where the actual next visit lays on sorted probability array. If, for example, the probability of the actual next visit is the highest one, then we say that prediction rank for this input is 1. So in a way, the lower the prediction rank, the better the prediction is.

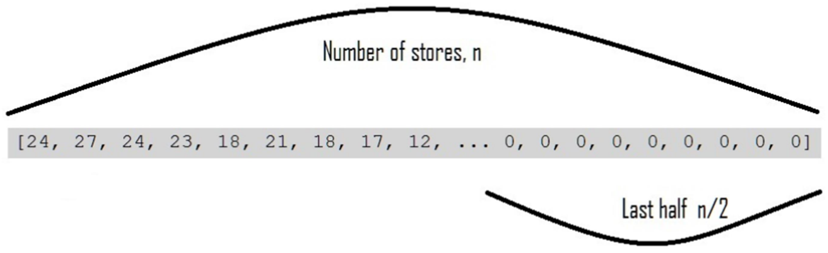


Figure 19 Predictions ranks of evaluation

In Figure 19, we can see a sample *prediction ranks* array produced for all available trajectories in a test set. This sample tells us that, over all predictions made on the test set, the highest probability next-store found by the model was the actual next-store 24 times, second highest probability was the actual next-store 27 times, and so on.

As a metric of successful recommendation, we measure the percentage of predictions that are covered by the first 1, 3, 5, and 10 prediction ranks, amongst all predictions. We also measure the prediction ranks that fall into *Last half* part of the array (worst predictions), to emphasize the effect of using intent-cluster-memberships during training.

It is obvious that, these metrics will give more of a prediction success result rather than a recommendation success since we are testing on already existing data. To be able to calculate the real success rate of recommendations, the system should be run in a real-life setting and push recommendations to the customer and observe to see if the recommendations were followed or get explicit feedback from the customer. Since real-life evaluation setup is not available for this study, we will go with prediction success rate, and we can assume that a high prediction success rate for next visited store will imply a high relevancy of recommendations, since visited zones provide implicit feedback.

4.1 Experiments and evaluation of recommendations

The test dataset which was set aside prior to model training was manipulated to produce each possible sub-step of the trajectory in a walk-forward manner. Consider the following trajectory for a given customer in the test dataset;

```
Customer x:
store 1 -> store 2 -> store 3 -> store 4 -> store 5 -> store 6
```

We can produce sub-trajectories as follows (when minimum-store-visit-for-training is configured as 2) and use each of them separately for evaluating the model;

```
INPUT                                                 OUTPUT
store 1 -> store 2                                    store 3
store 1 -> store 2 -> store 3                         store 4
store 1 -> store 2 -> store 3 -> store 4              store 5
store 1 -> store 2 -> store 3 -> store 4 -> store 5   store 6
```

Different neural network models and different hyper-parameters for these models have been tested for the given dataset. Following tables present the training and evaluation results for two different models and a default hyper-parameter set which was found to yield near optimal results. SimpleRNN, LSTM and GRU cell types were experimented for the recurrent part of the model with varying numbers of nodes and layers.

- ***Model 1*** uses 3 layers of sequential (RNN) nodes with following number of nodes (starting from the input layer); ***#number_of_feature_count -> 32 -> 16***
- ***Model 2*** uses 2 layers of sequential (RNN) nodes with following number of nodes (starting from the input layer); ***#number_of_feature_count/2 -> 16***

For both models, the outcomes of removing the dense layer for the intent-cluster-memberships are also presented (RNN only model).

Training parameters:

- Epoch: 300
- Batch size: 16
- Learning rate: 1e-06
- Dropout rate: 0.35

Evaluation results are presented in the following tables, where *n* denotes the percentage of the sum of first *n* prediction ranks. Hence, n=3 denotes the percentage of the total of first 3 prediction ranks in all predictions.

	MODEL 1	n=1 (%)	n=3 (%)	n=5 (%)	n=10 (%)	Last Half (%)	Approximate Training Time (seconds)
Combined Model	SimpleRNN	4.907	15.619	23.160	36.984	3.232	630 seconds
	GRU	4.608	15.021	23.399	36.326	3.232	1400 seconds
	LSTM	5.745	15.260	22.262	36.625	3.112	1500 seconds
Recurrent Only Model	SimpleRNN	6.104	15.859	25.135	37.283	27.050	670 seconds
	GRU	6.104	15.799	24.955	35.548	25.733	1400 seconds
	LSTM	6.104	15.799	24.955	35.189	23.100	1500 seconds

Table 1 Evaluation results from Model 1 training

	MODEL 2	n=1 (%)	n=3 (%)	n=5 (%)	n=10 (%)	Last Half (%)	Approximate Training Time (seconds)
Combined Model	SimpleRNN	6.403	15.859	23.399	37.163	3.172	400 seconds
	GRU	5.625	15.021	22.980	37.163	3.471	1000 seconds
	LSTM	4.548	14.303	23.399	38.600	3.232	1100 seconds
Recurrent Only Model	SimpleRNN	6.403	15.859	23.220	32.555	25.972	393 seconds
	GRU	6.104	15.799	24.955	36.445	25.613	1000 seconds
	LSTM	6.104	15.799	24.955	36.146	27.648	1100 seconds

Table 2 Evaluation results from Model 2 training

4.2 Analysing the results

It can be observed from the results that the type of the RNN node used does not make a significant change for the accuracy of predictions, even though the training time is significantly reduced by using SimpleRNN nodes. The reason for GRU and LSTM nodes not performing better than a SimpleRNN might be attributed to the fact that sequences in the data are not long enough to warrant the use of the memorisation capabilities of gated units.

Many different hyper-parameters (epoch count, dropout rate, learning rate…) for training have been tried out but have not been documented here, since they have not made a significant change for the better. Also it was observed that training accuracy and loss did not improve during the training of this dataset.

One probable cause of this could be the lack of substantial data. The data available for this study is not enough for presenting a general pattern of movement (only about 50 trajectories per day after data cleaning). If we also consider the fact that a great portion of these trajectories are very short after preprocessing steps, i.e. 3 or 4 steps, our network model may not find any meaningful relationships between store visits.

It was seen that using an RNN-only model (i.e. without adding the intent-cluster-memberships) does not really affect the rate of correct predictions for lower prediction ranks, sometimes even surpassing the combined model's accuracy. This could be due to RNN part of the model doing most of the work of predicting the consequent store visit after given input; hence producing similar results to the combined model for lower ranks (better predictions).

However, if we look at the number of predictions scattered across the last half of the prediction ranks array for RNN-only model, it becomes obvious that adding intent-membership information to training process increases the relevancy of predictions. This shows that without intent-membership information, failing predictions display a much more random characteristic.

4.2.1 Insufficient data

After preprocessing steps we are left with quite a small number of trajectories considering the context of finding patterns in large customer bases. For the available data set, number of trajectories (over 25 days) used for training is 1168, before producing sub-trajectories, which amounts to only about 47 trajectories a day, and that will not be enough to provide a pattern of movement for different sorts of intents on different times of the day / different days of the week… etc.

Another problem point is the length of the trajectories. Following table shows the count of trajectories grouped by the store visits they contain.

#store_visits	count
3	594
4	296
5	151
6	67
7	32
8	12
9	5
10	8
11	2
13	1

Table 3 Counts of store_visits contained in training data

Most of the trajectories are concentrated on 3 and 4 store visits and this is likely to prevent the application from forming more meaningful intent clusters. This can also limit the performance of RNN model on longer sequences, since it will need to rely on the shorter visit histories in the training data to make sequential predictions.

5 Future work and improvements

First and foremost the study needs to be replicated with substantially more amount of data, since I think the study is let down by limited available data. During the training process, many different hyper-parameters and different architectures were used, but none of them yielded satisfactory training performance. This could also be the result of a bug in the application code, and should be investigated.

It is also my conviction that the current preprocessing of data is computationally inefficient and the trajectories produced by the process are probably not perfectly accurate. Having so many short lived (3-4 steps) trajectories does not seem like a realistic scenario, which implies either inaccurate/not-representative data, or a problem with the preprocessing phase of the study.

During the clustering of intents a partition based fuzzy method was used. However, this forces us to give the number of possible customer intents that can be produced beforehand. Instead, a density based fuzzy algorithm will not require this parameter and provide a more flexible intent grouping. *Hdbscan*[4] would be an ideal candidate for such an algorithm.

Other than these practical concerns, the proposed application can be improved by using the aforementioned segment vectors. The segment information can help to embed more semantic information into trajectory data. By using this information for stores during intent clustering, we can achieve clusters based more on customers' actual interests in the products for certain segments, rather than a limited vision gained through store identities. Through the use of intent memberships during neural network training, segment information could also help recommendations to be more relevant.

[4] https://hdbscan.readthedocs.io/en/latest/index.html

6 Conclusions

This study aimed to produce a recommender system for customers of shopping malls, using their store visitation patterns inside the mall. To this end, it employed a fuzzy clustering technique to find groups of intents for customers, and neural network models based on RNN nodes to produce recommendations for a new-coming customer. Real life mobility data of customers encompassing a period of 25 days inside a shopping mall obtained from a company. General concepts related to the application were presented along with reviews of relevant literature. Limitations of the current study were highlighted.

It was shown with limited success that it is possible to produce relevant recommendations in the given setting. When prediction results from the neural network were considered to be indicative of recommendation performance, it was seen that only 20-25% of the actual next store visits will lie in the first 5 recommendations brought to the customer. However this does not necessarily tell us that other 4 recommendations made by the system are irrelevant to the said customer. To get an exact estimation of recommendation success, one approach is to deploy a pilot system in a real-life setting and to collect feedback from customers. Another approach is to produce a simulation system where we can model different type of customers with varying intents and produce artificial trajectories for these customers. This way we can collect feedback results representative of real customers.

Bibliography

Alag, Satnam. 2008. *Collective Intelligence in Action*. Greenwich, CT, USA: Manning Publications Co.

Altaf, B., L. Yu, and X. Zhang. 2018. "Spatio-Temporal Attention Based Recurrent Neural Network for Next Location Prediction.". doi:10.1109/BigData.2018.8622218.

Alvares, Luis Otavio, Vania Bogorny, Bart Kuijpers, de Macedo, Jose Antonio Fernandes, Bart Moelans, and Alejandro Vaisman. 2007. "A Model for Enriching Trajectories with Semantic Geographical Information." Seattle, Washington, Association for Computing Machinery, .

Andrej Karpathy. "The Unreasonable Effectiveness of Recurrent Neural Networks.", last modified May 21, http://karpathy.github.io/2015/05/21/rnn-effectiveness/.

Birant, Derya and Alp Kut. 2007. "ST-DBSCAN: An Algorithm for Clustering Spatial–temporal Data." *Data & Knowledge Engineering* 60 (1): 208-221.

Bloch, Peter H., Nancy M. Ridgway, and Scott A. Dawson. 1994. "The Shopping Mall as Consumer Habitat." *Journal of Retailing* 70 (1): 23.

Bloch, Peter H., Nancy M. Ridgway, and Daniel L. Sherrell. 1989. "Extending the Concept of Shopping: An Investigation of Browsing Activity." *Journal of the Academy of Marketing Science* 17 (1): 13-21.

Bobadilla, J., F. Ortega, A. Hernando, and A. Gutiérrez. 2013. "Recommender Systems Survey." *Knowledge-Based Systems* 46: 109-132. doi:https://doi.org/10.1016/j.knosys.2013.03.012. https://www.sciencedirect.com/science/article/pii/S0950705113001044.

Bonabeau, Eric. 2009. "Decisions 2.0: The Power of Collective Intelligence." *MIT Sloan Management Review* 50: 45-52.

Carbonell, Jaime G., Ryszard S. Michalski, and Tom M. Mitchell. 1983. "An Overview of Machine Learning." In *Machine Learning: An Artificial Intelligence Approach*, edited by Ryszard S. Michalski, Jaime G. Carbonell and Tom M. Mitchell, 3-23. Berlin, Heidelberg: Springer Berlin Heidelberg.

Chandar, Sarath, Chinnadhurai Sankar, Eugene Vorontsov, Samira Ebrahimi Kahou, and Yoshua Bengio. 2019. "Towards Non-Saturating Recurrent Units for Modelling Long-Term Dependencies." *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (01): 3280-3287. doi:10.1609/aaai.v33i01.33013280. https://ojs.aaai.org/index.php/AAAI/article/view/4200.

Cheng, Heng-Tze, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, et al. 2016. *Wide & Deep Learning for Recommender Systems*.

Chung, Junyoung, Caglar Gulcehre, KyungHyun Cho, and Y. Bengio. 2014. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling." .

Elman, Jeffrey L. 1990. "Finding Structure in Time." *Cognitive Science* 14 (2): 179-211.

Ester, Martin, Hans-Peter Kriegel, J. org Sander, and Xiaowei Xu. 1996. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise." Portland, Oregon, AAAI Press, .

Feng, Jie, Yong Li, Chao Zhang, Funing Sun, Fanchao Meng, Ang Guo, and Depeng Jin. 2018. *DeepMove: Predicting Human Mobility with Attentional Recurrent Networks*. doi:10.1145/3178876.3186058. https://doi.org/10.1145/3178876.3186058.

Giannotti, Fosca, Mirco Nanni, Fabio Pinelli, and Dino Pedreschi. 2007. "Trajectory Pattern Mining." San Jose, California, USA, Association for Computing Machinery, .

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning* MIT Press.

Grira, Nizar, Michel Crucianu, and Nozha Boujemaa. 2004. "Unsupervised and Semi-Supervised Clustering: A Brief Survey.".

He, Xiangnan, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. *Neural Collaborative Filtering*. doi:10.1145/3038912.3052569.

Heylighen, Francis. 1999. "Collective Intelligence and its Implementation on the Web: Algorithms to Develop a Collective Mental Map." *Computational & Mathematical Organization Theory* 5 (3): 253-280.

Hidasi, Balázs, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. "Session-Based Recommendations with Recurrent Neural Networks." .

Hochreiter, Sepp and Jürgen Schmidhuber. 1997. "Long Short-Term Memory." *Neural Computation* 9 (8): 1735-1780. doi:10.1162/neco.1997.9.8.1735. https://doi.org/10.1162/neco.1997.9.8.1735.

Jain, A. K., Jianchang Mao, and K. M. Mohiuddin. 1996. "Artificial Neural Networks: A Tutorial." *Computer* 29 (3): 31-44.

Kim, Byoungjip, Jin-Young Ha, Sangjeong Lee, Seungwoo Kang, Youngki Lee, Yunseok Rhee, Lama Nachman, and Junehwa Song. 2011. "AdNext: A Visit-Pattern-Aware Mobile Advertising System for Urban Commercial Complexes." .

Lipton, Zachary Chase. 2015. "A Critical Review of Recurrent Neural Networks for Sequence Learning." *CoRR* abs/1506.00019.

Liu, Qiang, Shu Wu, Liang Wang, and Tieniu Tan. 2016. "Predicting the Next Location: A Recurrent Model with Spatial and Temporal Contexts." *Proceedings of the AAAI Conference on Artificial Intelligence* 30 (1). https://ojs.aaai.org/index.php/AAAI/article/view/9971.

Malone, Thomas W., Robert Laubacher, and Chrysanthos Dellarocas. 2009. *Harnessing Crowds: Mapping the Genome of Collective Intelligence*. Cambridge, MA: MIT Center for Collective Intelligence.

Monreale, Anna, Fabio Pinelli, Roberto Trasarti, and Fosca Giannotti. 2009. "WhereNext: A Location Predictor on Trajectory Pattern Mining." Paris, France, Association for Computing Machinery, .

Öztayşi, Başar, Ugur Gokdere, Esra Simsek, and Sultan Ceren Öner. 2017. "A Novel Approach to Segmentation using Customer Locations Data and Intelligent Techniques." In .

Parent, Christine, Stefano Spaccapietra, Chiara Renso, Gennady Andrienko, Natalia Andrienko, Vania Bogorny, Maria Luisa Damiani, et al. 2013. "Semantic Trajectories Modeling and Analysis." *ACM Comput.Surv.* 45 (4).

Pascanu, Razvan, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. "How to Construct Deep Recurrent Neural Networks." *CoRR* abs/1312.6026.

Ricci, F., L. Rokach, B. Shapira, and P. (eds) Kantor. 2011. *Recommender Systems Handbook*. Boston, MA: Springer US.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. 1986. "Learning Representations by Back-Propagating Errors." *Nature* 323: 533.

Sarwar, Badrul, George Karypis, Joseph Konstan, and John Riedl. 2001. "Item-Based Collaborative Filtering Recommendation Algorithms." *Proceedings of ACM World Wide Web Conference* 1. doi:10.1145/371920.372071.

Spaccapietra, Stefano, Christine Parent, Maria Luisa Damiani, Jose Antonio de Macedo, Fabio Porto, and Christelle Vangenot. 2008. "A Conceptual View on Trajectories." *Data & Knowledge Engineering; Including Special Section: Privacy Aspects of Data Mining Workshop (2006) - Five Invited and Extended Papers* 65 (1): 126-146. doi:https://doi.org/10.1016/j.datak.2007.10.008. https://www.sciencedirect.com/science/article/pii/S0169023X07002078.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." *Journal of Machine Learning Research* 15: 1929-1958.

Su, Xiaoyuan and Taghi Khoshgoftaar. 2009. "A Survey of Collaborative Filtering Techniques." *Adv.Artificial Intellegence* 2009. doi:10.1155/2009/421425.

Sun, Ke, Tieyun Qian, Tong Chen, Yile Liang, Quoc Viet Hung Nguyen, and Hongzhi Yin. 2020. "Where to Go Next: Modeling Long- and Short-Term User Preferences for Point-of-Interest Recommendation." *Proceedings of the AAAI Conference on Artificial Intelligence* 34 (01): 214-221. doi:10.1609/aaai.v34i01.5353. https://ojs.aaai.org/index.php/AAAI/article/view/5353.

Surowiecki, James. 2005. *The Wisdom of Crowds* Anchor.

Vasquez, Dizan, Thierry Fraichard, and Christian Laugier. 2009. "Growing Hidden Markov Models: An Incremental Tool for Learning and Predicting Human and Vehicle Motion." *International Journal of Robotic Research - IJRR* 28: 1486-1506. doi:10.1177/0278364909342118.

Wang, Liang, Kunyuan Hu, Tao Ku, and Xiaohui Yan. 2013. "Mining Frequent Trajectory Pattern Based on Vague Space Partition." *Know.-Based Syst.* 50 (C): 100–111.

Wind, Yoram (Jerry) and David R. Bell. 2008. "Chapter 11 - Market Segmentation." In *The Marketing Book (Sixth Edition)*, edited by Michael J. Baker and Susan Hart, 222-244. Oxford: Butterworth-Heinemann. doi:https://doi.org/10.1016/B978-0-7506-8566-5.50015-7. https://www.sciencedirect.com/science/article/pii/B9780750685665500157.

Yao, Di, Chao Zhang, Jianhui Huang, and Jingping bi. 2017. *SERM: A Recurrent Model for Next Location Prediction in Semantic Trajectories*. doi:10.1145/3132847.3133056.

Ying, Josh Jia-Ching, Wang-Chien Lee, Tz-Chiao Weng, and Vincent S. Tseng. 2011. "Semantic Trajectory Mining for Location Prediction." Chicago, Illinois, Association for Computing Machinery, .

Zhang, Wanlong, Xiang Wang, and Zhitao Huang. 2019. "A System of Mining Semantic Trajectory Patterns from GPS Data of Real Users." *Symmetry* 11 (7).

Zhang, X., B. Li, C. Song, Z. Huang, and Y. Li. 2020. "SASRM: A Semantic and Attention Spatio-Temporal Recurrent Model for Next Location Prediction.". doi:10.1109/IJCNN48605.2020.9206935.

Zhao, Pengpeng, Haifeng Zhu, Yanchi Liu, Jiajie Xu, Zhixu Li, Fuzhen Zhuang, Victor S. Sheng, and Xiaofang Zhou. 2019. "Where to Go Next: A Spatio-Temporal Gated

Network for Next POI Recommendation." *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (01): 5877-5884. doi:10.1609/aaai.v33i01.33015877. https://ojs.aaai.org/index.php/AAAI/article/view/4537.

Appendices

A An example of a one-hot vector for recurrent network input

Where ... indicates similar tensors of same size (198), and the number of tensors in this array adds up to maximum sequence length for the data (15, following the example from Figure 15)

```
[<tf.Tensor: shape=(198,), dtype=float32, numpy=
array([1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 1., 0., 0., 0., 1., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.], dtype=float32)>,
       ...
       ...
       ...
<tf.Tensor: shape=(198,), dtype=float32, numpy=
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 1., 0., 0., 0., 0., 0.,
       0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0.], dtype=float32)>]
```