HOT SPOT PREDICTION FOR ROUTING OF CAR SHARING AND OTHER VEHICLES USING AI

A Degree Thesis Submitted to the Faculty of the Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Christian Soler Lozano

In partial fulfilment of the requirements for the degree in

TELECOMMUNICATIONS TECHNOLOGIES AND SERVICES ENGINEERING

Advisor: Elisa Sayrol Clols

Barcelona, June 2020

Abstract

Deep learning techniques have shown to improve by far existing methods in many engineering problems whenever large amounts of data is available and powerful processors (GPUs) are available for training. In Urban Mobility issues, large amounts of data might be generated and analyzed, for example, to choose dynamic routing to reduce traffic congestion. In this project we pretend to apply deep learning techniques, specifically LSTM, to improve an aspect of Urban Mobility in the Metropolitan Area of Barcelona. The main goal of this project is to predict, from a sequence of pick-ups and drop-offs, the next pick-up of car sharing, taxis, and other passenger vehicles. The pick-up must be a hotspot, being a hotspot a place that stands out above the other collecting points.

Resum

Les tècniques d'aprenentatge profund han demostrat que milloren els mètodes existents en molts problemes d'enginyeria sempre que hi hagin grans quantitats de dades i es disposin de processadors potents (GPUs) per a l'entrenament. En problemes de mobilitat urbana es poden generar i analitzar gran quantitats de dades, per exemple, per elegir de manera dinàmica una ruta per tal reduir la congestió del trànsit. En aquest projecte es pretén aplicar tècniques d'aprenentatge profund, específicament LSTM, per tal de millorar un dels aspectes de la Mobilitat Urbana a l'Àrea Metropolitana de Barcelona. L'objectiu principal d'aquest projecte és predir, a partir d'una seqüència de punts d'inici i finalització, el proper punt de recollida de *car sharing,* taxis i altres vehicles de passatgers. Aquest punt de recollida és un *hotspot*, és a dir, un punt que destaca per sobre dels altres punts de recollida.

Resumen

Las técnicas de aprendizaje profundo han demostrado que mejoran los métodos existentes en muchos problemas de ingeniería siempre que haya grandes cantidades de datos y se dispongan de procesadores potentes (GPUs) para el entrenamiento. En problemas de movilidad urbana se pueden generar y analizar grandes cantidades de datos, por ejemplo, para elegir de manera dinámica una ruta para reducir la congestión del tráfico. En este proyecto se pretende aplicar técnicas de aprendizaje profundo, específicamente LSTM, para mejorar uno de los aspectos de la Movilidad Urbana en el Área Metropolitana de Barcelona. El objetivo principal de este proyecto es predecir, a partir de una secuencia de puntos de inicio y finalización, el próximo punto de recogida de *car sharing*, taxis y otros vehículos de pasajeros. Este punto de recogida es un *hotspot*, es decir, un punto que destaca por encima de los otros puntos de recogida.

Acknowledgements

I would like to thank Elisa Sayrol for her dedication along these months and for all the contribution and advice in order to successfully develop this project.

To Javer Hernando and Miquel Angel India for the advice and improvements of the implemented model. As well as Miquel Estrada for providing the taxi's dataset to train the model.

To Judith Soler and Àlex Martinavarro for the translation and linguistic correction of this project. I totally encourage my sister to keep learning NLP since she has a strong potential to do it.

Finally, to my parents, family and friends for their support along the project and career.

Revision history and approval record

Revision	Date	Purpose
0	1/06/2020	Document creation
1	29/06/2020	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Christian Soler Lozano	christian.soler.lozano@gmail.com
Elisa Sayrol Clols	elisa.sayrol@upc.edu

Written by:		Reviewed and approved by:	
Date	30/05/2020	Date	29/06/2020
Name	Christian Soler Lozano	Name	Elisa Sayrol Clols
Position	Project Author	Position	Project Supervisor

Table of contents

List of Figures

List of Tables:

1. Introduction

1.1. Statement of purpose

Nowadays, car sharing and taxi predominates over personal cars since it presents a huge advantage especially in large cities. This is because it may be difficult to find a parking slot, it is much cheaper if the ride is shared and finally it reduces the carbon footprint. In order to reduce the impact of shared vehicles or taxis turning around or having to go to distant points in order to attract customers, it would be good to make a prediction of the nearest place where there may be a request for service.

The main goal of this project is to predict from a sequence of pick-ups and drop-offs of car sharing, taxis, and other passenger vehicles (figure 1), the next pick-up. The pick-up must be a hotspot, being a hotspot a point that stands out above the others.

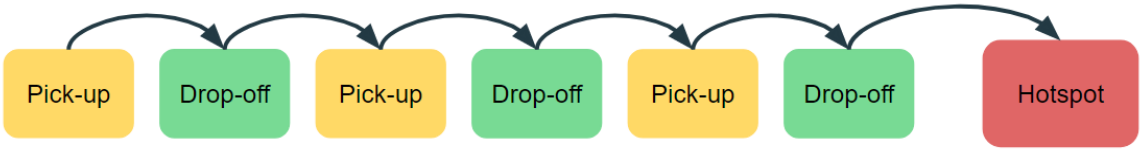


Figure 1. Sequence of pick-ups and drop-offs.

The model used to make predictions consists of embeddings, recurrent neural networks (RNNs) and attention models (AMs). These modules are normally used in Natural Language Processing (NLP), as for example, in machine translation. In this project, the RNNs, the AMs and the embeddings are implemented in order to perform a completely different task such as hotspot prediction.

Using this model, car sharing vehicles can strategically place their cars in the areas of greatest demand. When self-driving cars become an everyday reality, they will be able to go automatically to the area that the model deems appropriate. While this is not possible, users may be encouraged to leave the vehicle right in the area of the predicted hotspot.

In the case of taxi drivers, they can maximize the number of rides they do, always directing them to the areas where demand is assured. It also prevents them from circulating without customers, which causes them to consume resources without making a profit, while at the same time having a negative impact on the environment.

The model has been trained and tested using data from the taxi rides carried out in 2014 and 2015 in Barcelona. Taxi data has been collected since 2008, though GPS was not implemented until 2010, and from 2010 to 2013 the amount of data was lower.

Rides do not consist of all the journey, only of the pick-ups and drop-offs. In fact, it is not necessary to have the whole journey, as stated above, the model only predicts from the pick-up and drop-off geolocation points. It is an advantage that the model works this way as it can make a prediction without having to complete part of the journey as stated in [1]. As soon as the drop-off has been done, it will be able to predict the next pick-up. Nevertheless, prediction is more difficult when the trajectory of the vehicles is not known.

Although the data that has been used comes from taxi rides provided by [2], the model can also be adapted to car sharing. Taxi drivers have their own habits, but the model has been trained without the taxi identifiers so that the model learns from the mobility patterns

of Barcelona and not the taxi drivers habits, thus making this model able to work with car sharing vehicles.

Weather data has also been gathered to improve predictions from [3]. This data consists of the precipitation data by time slots and the average temperature of the three reference weather stations from Barcelona, which are the Fabra observatory, El Prat airport and the city of Barcelona.

The project has been divided into different sections. In section 2, or state of the art, published matter related to this work is presented. Academic concepts and used methods are also introduced. Section 3, or project development, which is divided into two parts: pre-processing, where data is modelled to be able to introduce it into the model, and creation of the model, where the different parts of the model and their features are detailed. Section 4 shows the results obtained from the prediction model. Section 5 is about the budget. Finally, section 6 covers the conclusions and future development.

1.2. Requirements and specifications

In order to carry out the development of the project, Python has been used as a programming language, and in tasks that required a lot of computational load, Cython [4] has been used to compile.

The libraries used in this project are Pytorch, Pandas and Geopandas. Pytorch is an open source machine learning library widely used in computer vision and NLP applications, and it has been used to create the model. The Pandas library is used to process the data before using it in the model, and the Geopandas library is used to process the geospatial data. In addition, QGIS software [5] has also been used to edit geospatial contours.

In the beginning, the working environment was Google colab notebooks, where basically tests were done. Afterwards, the code was executed on the Image Processing Group (GPI) servers using Pycharm, an integrated development environment where code can be edited and executed on the server remotely.

There is a previous work [6] that used the same data but with different purposes. The aim of the mentioned work was focused on drop-off prediction rather than pick-ups. Also, the procedures to develop the model are different as well as the data pre-processing. This project is not a follow-up and the codes have been generated from scratch.

1.3. Work plan

1.3.1. Tasks

Project: Hot spot prediction of car sharing using AI	WP ref: A
Major constituent: State of the art	Sheet 1 of 9
Short description: Search for information on the concepts being worked on, while also searching and analyzing academic articles in order to carry out the project.	Planned start date: 17/02/2020 (1W) Planned end date: 22/03/2020 (5W)
Internal task T1: Search and analyze papers	Deliverables: Project approach

Project: Hot spot prediction of car sharing using AI	WP ref: B
Major constituent: Data operations	Sheet 2 of 9
Short description: Because the raw data has unnecessary and incorrect values and at the same time must be adapted to the input of the model, operations with the data must be performed. Also, in order to display the data, it is necessary to implement a code capable of generating maps and graphs for visualization.	Planned start date: 24/02/2020 (2W) Planned end date: 19/04/2020 (9W)
Internal task T1: Data preparation Internal task T2: Data visualization	Deliverables: Processed data Data visualization

Project: Hot spot prediction of car sharing using AI	WP ref: C
Major constituent: Clusters generation	Sheet 3 of 9
Short description: A set of location clusters is defined from location points in the training data and applying a clustering algorithm.	Planned start date: 9/03/2020 (4W) Planned end date: 29/03/2020 (6W)
Internal task T1: Generate clusters	Deliverables: Results from clustering

Project: Hot spot prediction of car sharing using AI	WP ref: D
Major constituent: Neural Network - Word2Vec	Sheet 4 of 9
Short description: Model entries share common contexts, for this reason word2vec is used so that the model can observe these relationships.	Planned start date: 16/03/2020 (5W) Planned end date: 15/05/2020 (13W)
Internal task T1: Generate Word2Vec data representation. Internal task T2: Generate spatial cluster embedding.	Deliverables: Word2Vec data representation Spatial cluster embedding

Project: Hot spot prediction of car sharing using AI	WP ref: E
Major constituent: Neural Network - LSTM	Sheet 5 of 9
Short description: The use of recurrent neural networks is necessary because temporal data is used. LSTM will be used to perform the temporary process.	Planned start date: 30/03/2020 (7W) Planned end date: 26/04/2020 (10W)
Internal task T1: LSTM model training Internal task T2: LSTM model test	Deliverables: LSTM model

Project: Hot spot prediction of car sharing using AI	WP ref: F
Major constituent: Neural Network - Attention model	Sheet 6 of 9
Short description: To learn which part of the trajectory is more important to focus on.	Planned start date: 13/04/2020 (9W) Planned end date: 4/05/2020 (12W)
Internal task T1: Attention model training Internal task T2: Attention model test	Deliverables: Attention model

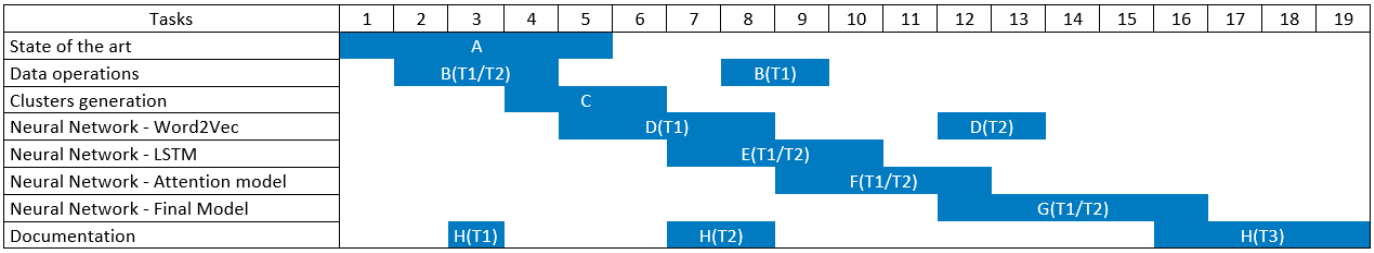
Project: Hot spot prediction of car sharing using AI	WP ref: G
Major constituent: Neural Network - Final Model	Sheet 8 of 9
Short description: When the different models have worked separately, they are put together into a single model. Hyperparameters are also varied and results are obtained in each case.	Planned start date: 4/05/2020 (12W) Planned end date: 7/06/2020 (16W)
Internal task T1: Union of the different parts in a single model Internal task T2: Hyperparameters variation and comparison of results	Deliverables: Final model Results and conclusions

Project: Hot spot prediction of car sharing using AI	WP ref: H
Major constituent: Documentation	Sheet 9 of 9
Short description: During the project, three documents have to be delivered, project proposal and work plan, critical review and final review.	Planned start date: 2/03/2020 (3W) Planned end date: 28/06/2020 (19W)
Internal task T1: Project Proposal and Work Plan Internal task T2: Critical Review Internal task T3: Final Review	Deliverables: Project Proposal and Work Plan Critical Review Final Review

1.3.2. Milestones

WP#	Task#	Short title	Milestone / deliverable	Date (week)
H	T1	Project Proposal and Work Plan	Project Proposal and Work Plan	8/03/2020 (3W)
H	T2	Critical Review	Critical Review	14/04/2020 (9W)
H	T3	Final Review	Final Review	28/06/2020 (19W)

1.3.3. Gantt diagram



1.3.4. Description of the deviations from the initial plan and incidences

Minor changes have been made to the critical review. These changes are detailed below. The semantic characterization has been integrated into the Word2Vec work package under the name “spatial cluster embedding”. This has been done since both tasks are related. Also, as the delivery date has changed, another week has been added. The final review is the same duration but it has been delayed to next week.

2. State of the art of the technology used or applied in this thesis:

First of all, this section starts by introducing the projects and studies on which this project has been based or which have addressed this problem from a different perspective. Hereafter, applied theory and used methods to carry out the project are introduced.

2.1. Related work

One of the papers on which part of this work has been based [7] is that of predicting drop-offs based on modeling the behavior of taxi drivers. In order to carry out the proposed objectives, the model they use has been adapted so that it does not depend on the identifier of a specific vehicle. Therefore, the model does not associate the data with the behavior of a specific driver. Also, instead of predicting drop-offs, it predicts hotspots.

There are other studies [8, 9] that use meteorological data in order to improve predictions about mobility in cities. In this case, data such as precipitation by time slots and the average temperature of the city of Barcelona have been included in the model.

Other papers as in [10], use the full trajectory and not just pick-up and drop-off points. The model in [10] uses the real time data collected along the day to define the upcoming trajectory. On the contrary, in the developed model in this project, the predictions are only made from pick-ups and drop-offs. This is done in order to be able to predict the hotspot instantaneously.

There is another study, [11], that predicts whether an event will occur in a specific area based on the analysis of web sites. These models do not use LSTM layers but convolutional layers as the events to be predicted do not have a cyclic behavior.

Albert Baldó, a civil engineering student, is currently working on his master thesis using the same data as in this project to analyse the profitability of cabs from a statistical point of view.

2.2. Haversine distance

The Haversine formula [12] is used to calculate the distance between two points over the earth given their longitudes and latitudes. This formula calculates the great circle distance between two points, that is, the shortest distance on a sphere.

Working with the Haversine distance is necessary as geospatial data is given in latitudes and longitudes.

The Haversine formula (1) is shown below with an explanation of its parameters.

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2-\phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2-\lambda_1}{2}\right)}\right) \quad (1)$$

φ and λ are respectively the latitude and longitude in radians of each point and r is the earth's radius.

2.3. Projective transformation

A projective transformation [13, 14] is a type of linear transformation. A transformation is a function that transforms a vector space in another. This transformation is linear if it keeps the scalar multiplication and the vector addition.

To apply a projective transformation to a vector, this has to be multiplied by a transformation matrix, and as a result it obtains a vector with transformed coordinates.

Below it is shown a transformation matrix (2) with the result of the application on a vector.

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \simeq \begin{pmatrix} kx' \\ ky' \\ k \end{pmatrix} = \begin{pmatrix} a1 & a2 & b1 \\ a3 & a4 & b2 \\ c1 & c2 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (2)$$

In the transformation matrix, *a* elements form the rotation matrix, *b* elements form the translation vector and the *c* elements form the projection vector.

As the transformation must be from a 2D vector space to another 2D vector space, but the result of multiplying the transformation matrix by the vector gives a 3D vector, the vector is divided by *k*.

An affine transformation is a particular case of projective transformation. In the case of affine projection, it has only 6 degrees of freedom, while in the case of projective transformation, it has 8. The difference in the transformation matrix is that in the case of affine transformation, the projection vector is zero.

2.4. Clustering

There are different algorithms for generating clusters, such as K-means, which is one of the most well-known and widely used, because it is fast and easy to understand. Though it is not really an algorithm to generate clusters but a partitioning algorithm as it does not find clusters but partitions the data. This algorithm has some drawbacks when generating clusters from geospatial data, unlike other algorithms such as DBSCAN and HDBSCAN [15, 16].

One of the problems with the K-means algorithm is that it is not good for geospatial data because it minimizes the variance and not the geodesic distance, so the algorithm gives worse results when it moves away from the equator [17].

Also, the K-means algorithm has a different operation compared to DBSCAN and HDBSCAN as you have to select a priori the total number of clusters to be generated, while in the case of DBSCAN and HDBSCAN they select automatically the total number of clusters that should be generated.

In the end, the algorithm for generating clusters will depend on the type of data available. In this case they are cardinal points and it is important to take into account the distance used to calculate the clusters. As mentioned above, the Haversine distance is used to calculate the distance between two cardinal points. The K-means algorithm uses Euclidean distance, which is why it is not a suitable algorithm for geospatial data, while DBSCAN and HDBSCAN can use the Haversine distance.

2.5. Word embeddings

In NLP, a word embedding [18, 19, 20] is a representation of a specific word. Word embedding is able to capture the context of a word in a document, for example, the semantic and syntactic similarities.

The reason for using the word embedding is because a lot of Machine Learning Algorithms and almost all the Deep learning architectures can not process text directly. In order to perform tasks, the data must be introduced as numbers rather than text.

Another way to introduce the data inside the model is by using one-hot vectors codification. In case of words, these vectors have the same dimension as the corpus length. All the values are zero except the element where the index represents the word inside the vocabulary.

One of the problems of coding with one-hot vectors is that the parameters are independent of each other. This makes words with similar meanings have no proximity relationship while totally opposite words will not have a distance relationship either.

Another problem is the size of the corpus, if it is very large, it implies that the dimensions with which the model has to work are also very large.

Word embeddings are the solution to these problems. One of its goals is for words with a similar context to occupy close spatial positions. This is impossible in the case of one-hot vectors because they are orthogonal to each other. Also because their encoding is no longer one-hot, the size of their vectors no longer has to be equal to the size of the corpus and then their size can be greatly reduced.

There are different ways to obtain embeddings, classified according to whether they are embeddings based on frequency or prediction. Frequency-based ones are deterministic methods and their results are limited. Predictive ones are based on neural networks and have far surpassed the results of those based on frequency.

The first embeddings to make the queen-king example work (figure 2) are prediction-based embeddings.

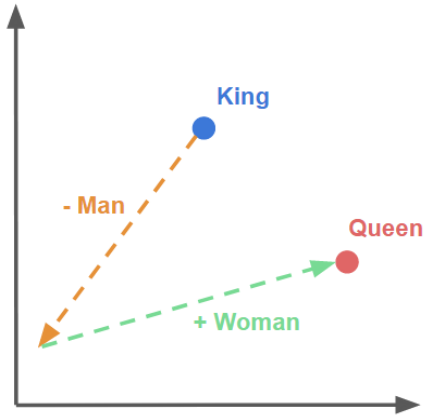


Figure 2. Queen-King embedding example[21].

Embeddings based on predictions appear with Word2Vec. The Word2Vec has two methods to obtain word embeddings which are the Skip Gram and the Common Bag of Words (CBOW) depicted in figure 3.

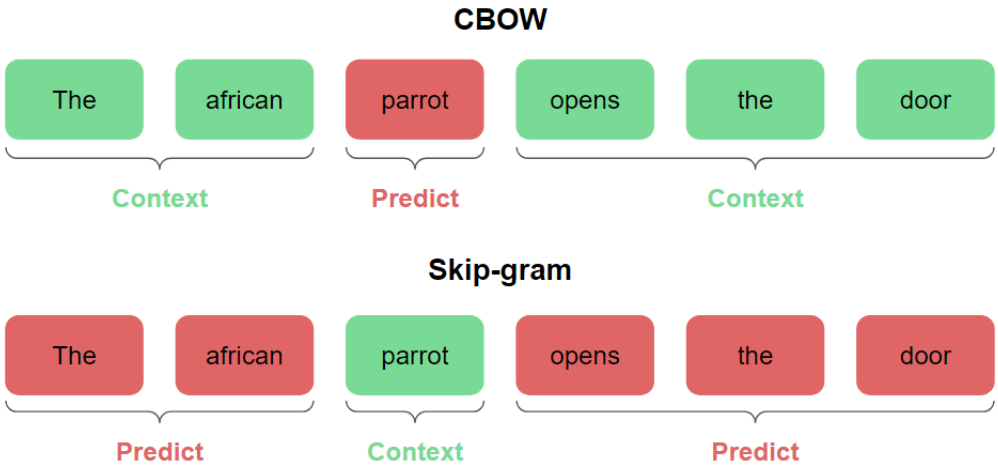


Figure 3. CBOW and Skip-gram comparison.

CBOW is based on predicting the probability of a word given a context. The context can be a word or a set of words. The size of the context is determined by the window, a window of size 1 implies that there is only one context word. Skip-gram does just the opposite, predicting the words around it from a context word.

Both methods have their advantages and disadvantages. The CBOW is faster and represents better the most common words. In the case of Skip-gram, it works better with little data and represents rare words better.

CBOW

CBOW is based on neuronal networks. In order to understand the concept, a context window of 1 is considered. In this case it implies that at the input of the model there is a context word and at the output the word to be predicted. These words are represented as one-hot vectors.

The model consists of two neuronal layers. The size of the first is equal to the size of the embeddings and the size of the second is equal to the size of the one-hot vectors. Once this model is trained, the first layer acts as a lookup table, as each of its rows represents an embedding.

2.6. RNN

The idea that lies behind recurrent neural networks (RNNs) [22, 23] is to use sequential information, that is, data sequences in which this data is not independent of each other.

In traditional neural networks, all inputs and outputs are independent of each other. They work taking a fixed amount of data at the input and produce a fixed amount at the output, and they do it with all the data at once. In contrast, RNNs do not capture all input data at once, instead they work capturing each one of the pieces of data in an input sequence at each instant of time.

The output at each instant of time of an RNN is known as the hidden state. This contains information about the current entry, as well as all entries from previous times. That is, at each instant of time, the output will be a function of the current input and a context, which is the hidden state calculated at the previous instant.

One of the reasons why a recurrent neural network can remember sequence information is because the hidden state acts as an internal memory and this makes it different from a conventional neural network that has no memory. Recurrent neural networks are called recurrent because they perform the same task for each of the elements of a sequence.

Typical application of RNN: voice recognition, machine translation, predictive analytics, text classification... among many other applications that deal with sequential information.

2.6.1. LSTM

One of the major problems that RNN networks have is the vanishing or explosion of gradients. This problem arises in backpropagation during training, especially in networks of very long sequences. Due to the chain rule, gradients must be multiplied continuously by matrices, causing the gradient to fade or explode depending on whether they are decreasing or increasing exponentially respectively. Then, having a small gradient causes the weights of the grid not to update, causing the model to not learn; or the gradients to become very large, causing the model to become unstable.

Due to this problem with gradients, RNNs cannot work with very long sequences. A very simple example would be a very long text in which, at the beginning, it is mentioned that someone has a parrot called Beethoven, and after a few sentences without mentioning it, it had to predict how the following sentence continues: “Beethoven, my pet…, can open doors”. In this case the model would not know which pet this sentence is referring to, because the relevant information from the beginning would have been lost, and would end up predicting that instead of a parrot, it is perhaps a cat or a dog.

A typical RNN network will only be able to use information from the nearby context to predict the word because it only has short-term memory, while a Long Short-Term Memory (LSTM) [24] network will be able to use contextual information from a text that has appeared earlier in some sentences, as it also has a long-term memory.

In each of the cells of a normal RNN, the way they work is very simple. The input in an instant of time and the hidden state of the previous instants go through an activation function to obtain the output in this instant of time, or what is the same, the hidden state in this instant.

The cells of an LSTM, unlike a normal RNN, have a more complex function. Each LSTM cell (figure 4) takes three pieces of data that are the current input and two from the previous cell that are short-term memory and long-term-memory. Both short-term memory and long-term memory are also called hidden states and cell states, respectively.

Each of the cells has gates in order to regulate the information that has to be discarded or maintained at each instant of time before passing the hidden state and the cell state to the next cell. The gateways for regulating information are called input gate, forget gate, and output gate.

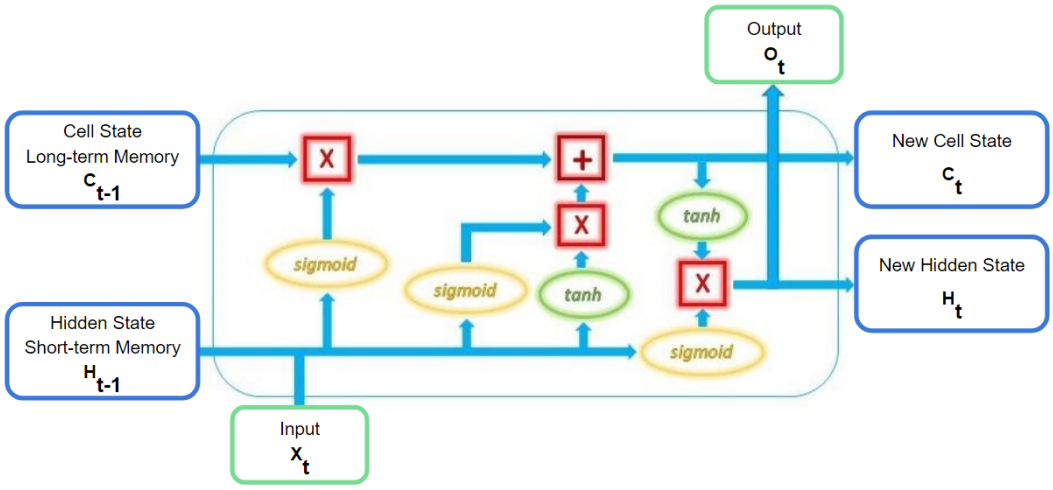


Figure 4. LSTM cell [24].

Input Gate

The input gate (formula 3) decides what new information should be stored in long-term memory. It does this based on information from the current entry and the hidden state of the previous instant. It has to filter the information by selecting which of this is relevant and discarding those pieces that are not useful.

$$i_{gate} = \sigma\left(W_{i1} \cdot (H_{t-1}, x_t) + bias_{i1}\right) * tanh\left(W_{i2} \cdot (H_{t-1}, x_t) + bias_{i2}\right) \quad (3)$$

The first multiplier term acts as a filter. The sigmoid (σ) makes its values go between 0 and 1 indicating how relevant the information is, being 0 not relevant and 1 very relevant. The second term is the current input and the hidden state to which the filter is applied. These values are passed by an activation function tanh to regulate the network.

Where the W are the weights, H_{t-1} is the hidden state and x_t is the input.

Forget gate

The forget gate (formula 4) decides which pieces of information in the long-term memory should be discarded or kept. The sum of the input gate and the forget gate create the long-term memory of the current instant (formula 5).

$$f_{gate} = \sigma\left(W_f \cdot (H_{t-1}, x_t) + bias_f\right) * C_{t-1} \quad (4) \qquad C_t = i_{gate} + f_{gate} \quad (5)$$

The formula (4) shows the filter, which is the first multiplier term, which is applied to long-term memory, which is the second multiplier term. Finally, C_{t-1} is defined as the cell state.

Output gate

The output gate (formula 6) captures the current input, the hidden state of the previous state, and the long-term memory of the current instant in order to generate the hidden state of the current instant, which matches the output of the current instant.

$$O_t, H_t = \sigma\left(W_{o1} \cdot (H_{t-1}, x_t) + bias_{o1}\right) * tanh\left(W_{o2} \cdot C_t + bias_{o2}\right) \quad (6)$$

The first multiplier term is the filter, and the second term is the long-term memory of the current instant to which the filter is applied.

2.7. Attention model

Attention models [25, 26] are used in neural networks in order to focus on the most relevant parts of the data.

There are different types of AMs. In order to understand the general operation of the different models we will explain it as an example applied in seq2seq models.

The seq2seq model can be used in NLP to translate sentences from one language to another. These consist of an encoder that encodes the input sequence and a decoder that, from the output of the encoder, generates the sentence in the desired language. Both, the encoder and the decoder, are made up of RNNs seen in section 2.6.

The encoder input is the hidden state generated by the encoder RNN. All the information needed to generate the translated sequence is represented in the hidden state vector.

The fact that all the information is represented in a single vector results in a bottleneck being generated between the encoder and the decoder. This may cause problems in the case of very long sequences, because the model will find it difficult to retain the information from the beginning of the sequence to be translated.

Figure 5 represents a seq2seq model that shows the bottleneck problem. The encoder is represented in red and the decoder in green.

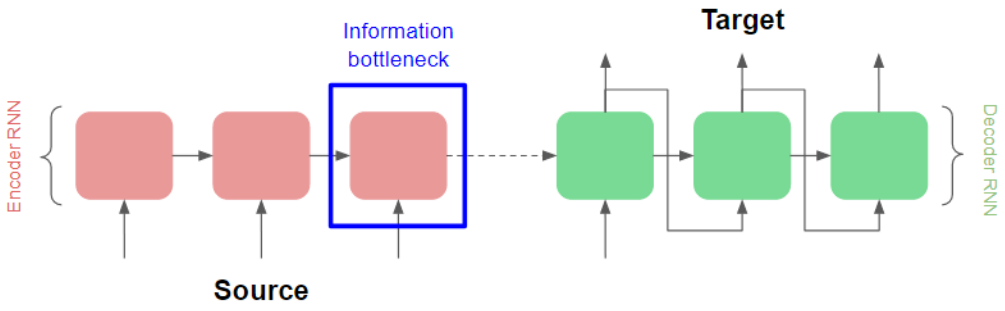


Figure 5. Problem with RNN [25].

In order to solve the bottleneck problem, an attention model (figure 6) has been used. The core idea of the attention module is to focus on the most relevant parts of the input sequence for each output.

Its operation is as follows. A context vector is added to each cell in the decoder. This context vector is obtained from a weighted sum of the encoder hidden states. The way to calculate weights is by passing the attention scores through a softmax. Attention scores are obtained by multiplying the hidden state of the decoder at a specific instant with all the hidden states of the encoder.

Once this is done, the decoder has direct access to the entire sequence of the decoder and the information it obtains is no longer only the hidden state.

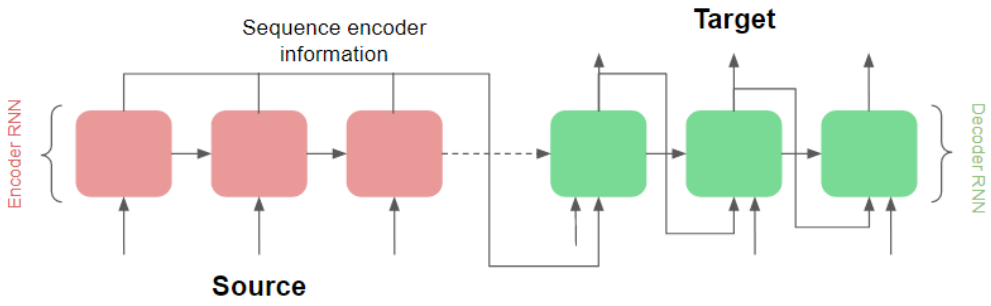


Figure 6. Attention model [25].

3. Project development:

The development of the project has been mainly divided into two parts: the preprocessing of the data and the creation of the model. In the preprocessing of the data, data is transformed into a proper format in order to be able to enter it in the model, and in the creation of the model the different elements that form it and its operation are observed.

3.1. Preprocessing

3.1.1. Flowgraph of the preprocessing

The following figure 7 shows the different steps that have been followed to obtain the data to enter the model. Each of these steps will be thoroughly explained below.

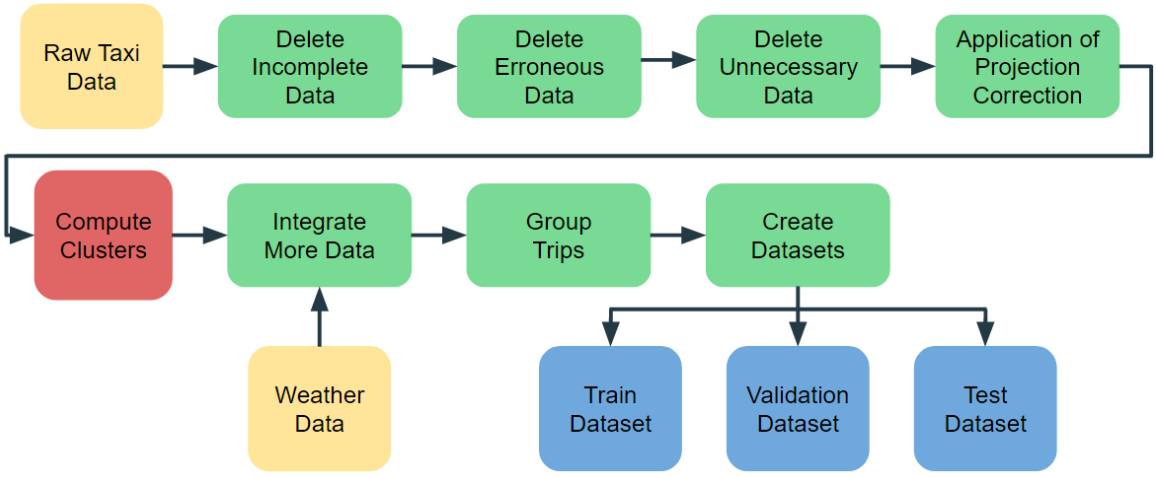


Figure 7. Preprocessing flowgraph.

3.1.2. Data format

The data used to carry out the research includes meteorological data from the city of Barcelona and data retrieved from the taxis working in the metropolitan area of Barcelona and other regions of Catalonia.

The data from the taxis is a representative sample from the taxi rides during the period 2014-2015

Each of the rows of the data represents a ride of a taxi, to be more specific, the pick-up and drop-off. Each of its rides has the fields shown in table 1. The data used is in green.

TAXIS	CONDUCTOR	JORNADA	CARRERA	DATA_INICI
DIA_SETMANA	HORA_INICI	MINUTS_INICI	DATA_FINAL	HORA FINAL
MINUTS_FINAL	TARIFA	TARIFA_INTEL	IMPORT	SUPLEMENTO
TEMPS_OCUP	KM_OCUPATS	VELCOM_CAR	VELMAX_CAR	TEMPS_LLIU
KM_LLIURES	VELCOM_LLI	VELMAX_LLI	ESTAT	TIPUS
LATIINI	LONGINI	LATIFIN	LONGFIN	

Table 1. Taxi data fields.

The meteorological data has been gathered from the observatories of Barcelona city, the airport and the Fabra observatory. The data used was the average temperature during the day and the precipitation by time slots. A total of four slots: from 0 to 5, from 6 to 11, from 12 to 17, and from 18 to 23 hours.

As there were days where some weather stations had no information, weather data comes from different sources. In first place from the observatory of Barcelona which better represents all the regions where the forecast has been made. If the average temperature was missing, the weather station used is El Prat airport due to its altitude above the sea level. In the case of precipitation data, an average has been made between the Fabra observatory and El Prat airport.

3.1.3. Delete data

There was missing data, for instance there were rides where relevant information to the research had no assigned value (*NaN*). In cases like this, the rides had been removed.

Rides with wrong values had also been removed. In this case, the wrong values correspond to latitude and longitude values equal to zero because perhaps the vehicle did not have a built-in GPS or was not working at that time. This step would not really be necessary since the values of latitude and longitude equal to zero are outside the established perimeter to work as will be seen below.

Eventually, data from outside the perimeter of the region formed by Barcelona, L'Hospitalet de Llobregat and El Prat de LLobregat has also been deleted. Also, within this region, the areas where data had a low density of pick-up or drop-off points has been removed. In order to delimit these areas, the census districts have been used, and some parts of the fringing districts have been deleted. The districts that have been affected are Nou Barris, Horta-Guinardó and Sarrià-Sant Gervasi. One of the reasons why these areas have a low density of points is because they are located within the natural park of the Serra de Collserola. The Prat de LLobregat area has also been modified, annexing some parts of the municipalities of Viladecans and Sant Boi de Llobregat close to El Prat airport.

The map below (figure 8) shows the different areas that have been used for the research.

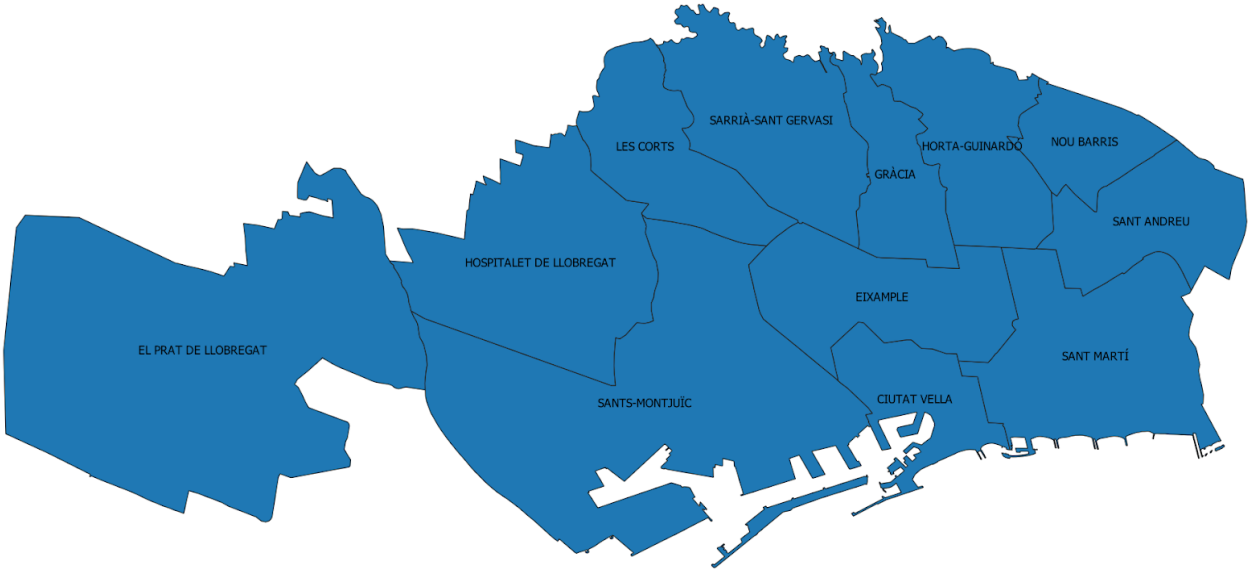


Figure 8. Map of the regions used.

3.1.4. Projective transformation

Geospatial data is usually accompanied by a Coordinate Reference System (CRS) so that it can be accurately represented on a map. Apart from being able to represent the data on a map, it is also used to add geospatial data, georeferenced tags or to obtain information about which area they are in, for example, which neighborhood, district or municipality they belong to.

In the case of taxi data, the CRS used is unknown and could not be found out. If the data is represented directly without indicating which CRS it uses, the data is represented over the sea and it is impossible to work with geospatial information.

To solve this problem, a projective transformation has been made to pass the geospatial data from the unknown CRS to a known CRS like WGS 84 (EPSG: 4326). This CRS is used by GPS satellite navigation systems. Data on geospatial information such as neighborhoods are in the same CRC.

To apply the projective transformation it is necessary to know four points of the unknown CRS and its equivalent value in the desired CRS. When the different points of the data are represented as an image, the outline of Barcelona emerges, therefore, it is easier to select the four points.

Once the four points are set, the transformation matrix can be found. Once it is found, the projective transformation can be applied to all data as seen in the state of the art, thus obtaining data represented with the new CRS.

Once the projective transformation is applied, data from outside the desired perimetre can be removed, as stated in the previous section.

The chart below (table 2) shows the different points that have been used.

Used points for the projective transformation			
Unknown CRS		WGS84 CRS	
Latitude	Longitude	Latitude	Longitude
41.173598	2.043031	41.289430	2.072530
41.243839	2.079819	41.406391	2.133120
41.249379	2.108135	41.415700	2.180420
41.236262	2.119755	41.393757	2.199570

Table 2. Used points for the projective transformation.

3.1.5. Clusters generation

Within the dataset, there is a lot of geospatial data made up of latitudes and longitudes. This way of representing information is very redundant. For example, the first numbers to be represented are always the same as it is only being analysed in the region of Barcelona. Points that are very close or almost overlapping have different values of latitude and longitude, when it would be interesting for them to have the same value to indicate that they are in the same area or region.

A way to represent this spatial information in a more compact way is by generating clusters. This also makes it easier to enter the data into the model and at the same time indicates that a set of points belong to the same region.

The algorithm used to generate clusters is HDBSCAN, which, as seen in the state of the art, is a good option for generating geospatial data clusters as it uses the Haversine distance.

When the HDBSCAN algorithm is applied, this does not return the centroid of each generated cluster but it returns to which cluster points belong. Also, if the point does not belong to any cluster, it will be classified as noise.

To be able to calculate the distance among the different clusters is necessary to provide each cluster center. Also, it is needed to be able to visualize the clusters on a map as a single point. That center represents all the points that belong to the same cluster.

To calculate the centers of each cluster, the first thing to be done is calculating their centroids from all points that belong to the same cluster. Then, as the centroid may not match any point in the cluster, the point closest to the centroid is assigned as the center of each cluster.

Once the centers of all the clusters have been calculated, there are points to which no cluster has been assigned and have been classified as noise. With these points the Haversine distance among them and all the clusters is calculated, and the cluster with which they have a minimum distance is selected.

All points that would have been considered noisy and whose distance to any cluster were very large, exceeding a certain threshold, could have been dismissed. This has not been applied as all points have a close cluster that represents them accurately.

It is also possible to select which areas are more relevant depending on the available data. For example, in areas where there is a higher concentration of points, there will be a greater amount of clusters than in areas with a lower concentration.

To generate clusters, all available data is used, regardless of the relationships among them, such as a specific time or day. As clusters are a representation of the different latitudes and longitudes to introduce into the model, both pick-up and drop-off data are used to generate the clusters.

In the following illustration (figure 9), the generated clusters are represented in red, the pick-up points in green, and the drop-off points in blue. Drop-offs are more scattered because cabs usually leave customers at the place they ask for. This does not usually happen with pick-ups as it is usually the customer who goes to the area where there are cabs.

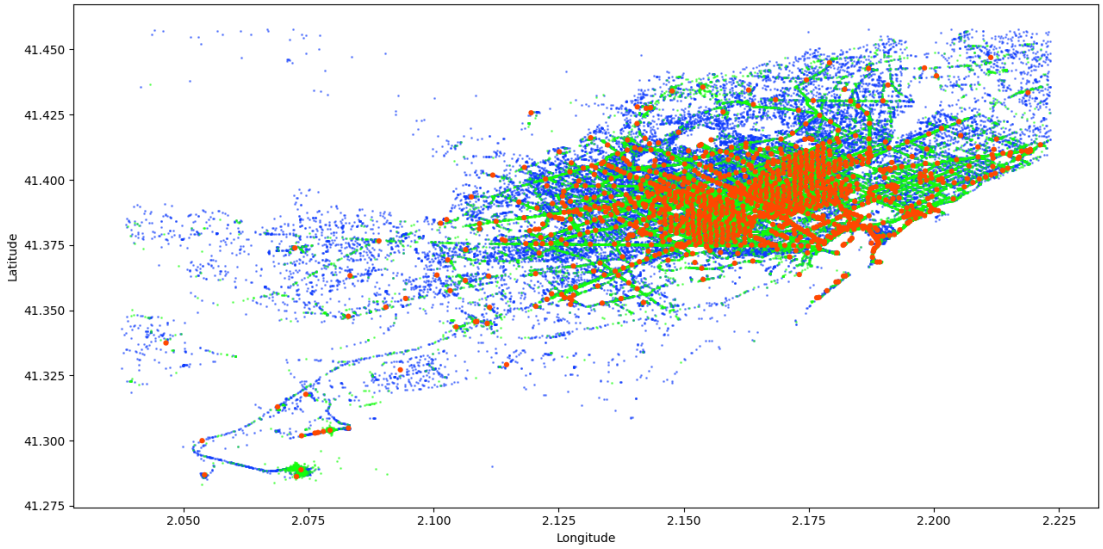


Figure 9. Pick-up, drop-off and cluster points on the map.

3.1.6. Hotspot generation

As mentioned above, clusters have not been generated with any time reference in mind, this is why hotspots are generated from previously calculated clusters, so hotspots do take these relationships into account.

The following three images (figure 10) show how the distribution of pick-ups is not uniform over time. On Saturdays night, from 2 to 3, the growing demand is located in nightlife areas such as the Olympic Port and the Paral·lel Street, while on Monday mornings, from 9 to 10, the demand is no longer around nightlife spots, and new areas are activated, like the ones around the cruises.

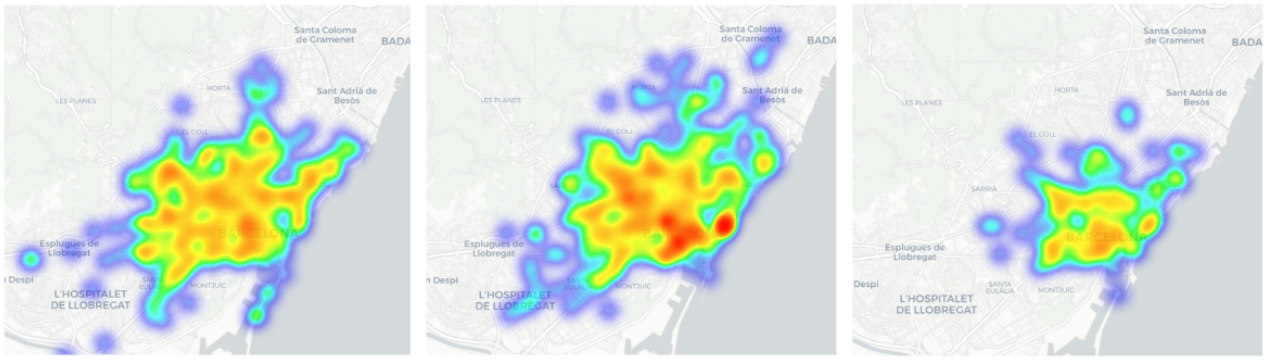


Figure 10. Heat maps generated at different times.

Hotspots are a subset of clusters that are generated taking into account the hours and days of the week. Each hotspot has the same identifier as a cluster. There are a total of 114 hotspots, but in a given hour and day there are only between 6 and 17 hotspots.

This is how the algorithm in charge of the generation of these hotspots works. Where each of the steps is displayed in the image below (figure 11):

1) First of all, the points of a given moment are selected, for example, a Monday morning from 8 to 9 hours. The selected points are in blue and the clusters previously calculated are in red. Then the closest points that each cluster has are assigned to each of them.
2) Clusters that have assigned points that exceed a certain threshold are considered hotspots. These are represented in yellow. Each cluster identifies which hotspot is the closest by calculating the Haversine distance between it and the different hotspots. The threshold used is 10.
3) Once each cluster knows which hotspot is the closest to it, the cluster assigns it to itself. In the picture, the hotspots are surrounded by a gray circle, and each of the clusters is represented by the color of the assigned hotspot.

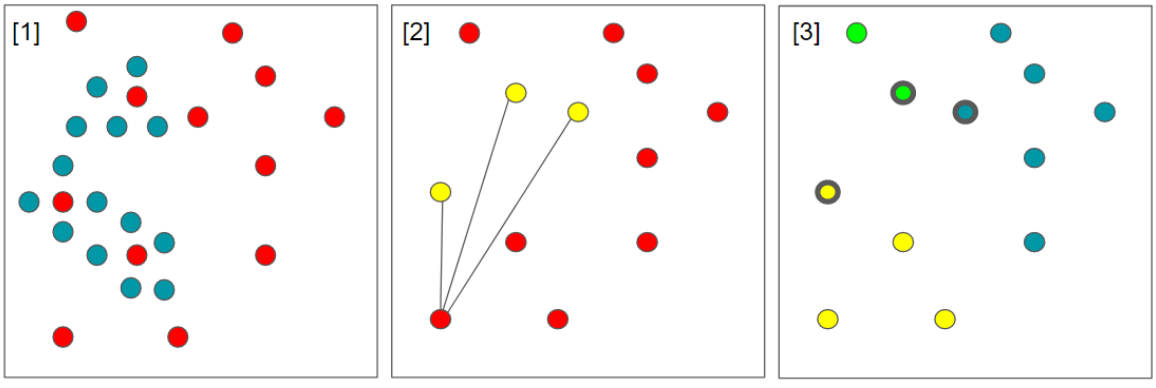


Figure 11. Algorithm used to generate hotspots.

3.1.7. Group trips

The way in which the data enters the model is in a format of pick-up and drop-off sequences. How these sequences are obtained is explained in this section.

First of all, all the data is sorted in the following order: starting with the taxi driver's identifier, starting date, starting hour and starting minutes. This is the only place where the taxi driver's identifier is used and it is only used to create the sequences. As mentioned previously, the taxi driver identifier is not introduced in the model.

Once all the data has been sorted by time, the different rides are gathered in groups of 4. In order that the different rides are consecutive, the time gap between a drop-off and a pick-up must not exceed more than two hours. The amount of rides and the time difference are hyperparameters to take into account.

The groups are made according to a sliding window taking into account the time gap and that within the window all the rides are from the same taxi driver. The following image (figure 12) shows how the sliding window works, where P refers to pick-up and D to drop-off.

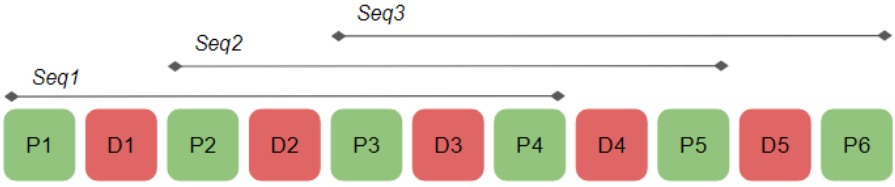


Figure 12. How the slider window works.

Once the sequences have been obtained, the pick-up to be predicted from the sequence is known at this point, and it is replaced by its equivalent hotspot.

Finally, the generated dataset is splitted between a training dataset and validation dataset. The training dataset has 80% of data and the validation dataset has 20% of data. A test dataset has not been created since there is not enough data available.

3.2. Model

3.2.1. Diagram of the model

The following diagram (figure 13) shows the model used to make the prediction. The model consists of an embedding layer, an attention model, an LSTM layer, a dropout layer, and specific classification or regression layers. The model is represented in the diagram as a classification, and the specific regression layers will be shown later.

Each of the parts that make up the model will be explained in more detail below.

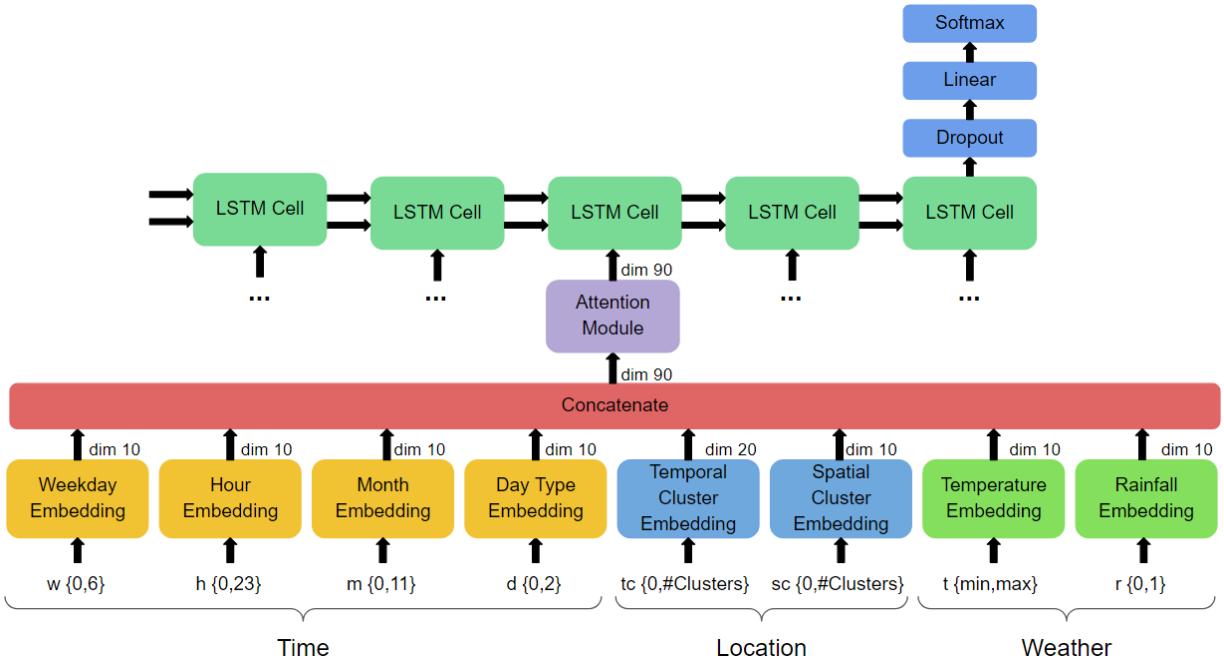


Figure 13. Diagram of the classification model.

3.2.2. Word embeddings

Embeddings for time, location, and weather data are created. Within the time embeddings there are the weekday (0,6), hour (0,23), month (0,11), and day type (0,2) embeddings. Within the location embeddings there are the temporal cluster (0, #clusters), and the spatial cluster (0, #regions) embeddings. Within the weather embeddings there are the temperature (minimum temperature, maximum temperature) and rainfall embeddings (0,1).

The day type embedding is used to classify days as workday, pre-holiday or holiday. To know what type day is, bank holidays that took place in Barcelona during the time period of the taxi data have been used.

The temperature has been taken in absolute value, obtaining an input range that goes from the minimum temperature to the maximum temperature. In the case of precipitation, it has been considered in binary mode, being 0 a non-rainy day and 1 a rainy day.

The temporal cluster embedding takes into account the temporal relationships between the different pick-ups and drop-offs of the sequences. Spatial cluster embedding takes into account the spatial relationships between clusters, assigning each cluster to a region and then embedding regions. The regions used include L'Hospitalet de Llobregat, El Prat de LLobregat, and the neighborhoods of Barcelona that were within the working perimeter, summing up to 65 neighbourhoods.

The output dimensions of all embeddings, except the temporal cluster embeddings is 10. On the other hand, the dimension of the temporal cluster embeddings is 20.

Embeddings have been generated in two different ways, pre-trained and trained during training. In the pre-trained ones, their weights are previously calculated before starting training the whole model; while the others are updated during the training initializing their

weights with random values at the beginning. The weights of the pre-trained can continue to be updated during training or freeze, so that their weights do not vary.

The temporal cluster embedding is pre-trained with unfreezed weights, thus they will keep being updated during training. All the others are trained during the training.

The way in which the weights of the temporal cluster embeddings have been generated is with the CBOW embeddings creation technique, seen in the word embeddings from the the state of the art section. Embeddings have been generated using Gensim [27], a module that implements Word2Vec algorithms. The clusters sequences of pick-ups and drop-offs have been used as sentences (figure 14) and a window of length 5 is applied on each side.

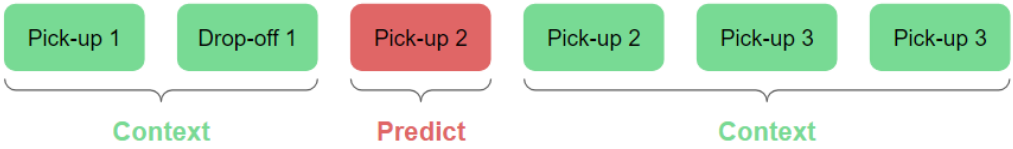


Figure 14. CBOW applied to clusters.

The model only took into account the prediction by hours and not by minutes, this is why there are no minute embeddings.

3.2.3. Attention model

The input of each LSTM cell in the model is the concatenation of the vectors of the time, location and weather embeddings. In order for the model to know which of this data to focus on, an attention model is added to decide which of this data is most relevant.

The attention model used in this model (figure 15) consists of three layers which are two permutation layers and a dense layer. Then there is a multiplication between the input vector and the vector resulting from going through the three layers.

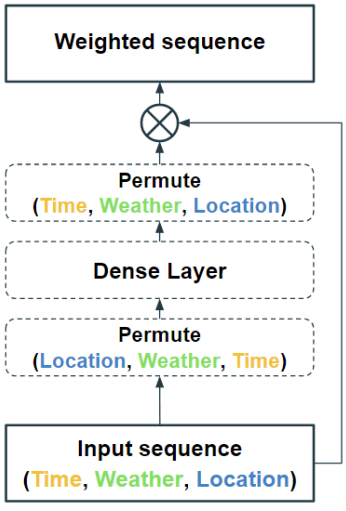


Figure 15. Attention model.

The dense layer generates the weights that afterwards will be multiplied by the input sequence, highlighting which part is more relevant, whether it is time, weather or location.

The authors do not explain in the paper the permutation functionality, but by performing those permutations the results have shown that the model is more robust [7].

3.2.4. LSTM

The model has a recurrent neural network layer (figure 16), which contains LSTM cells. The reason LSTM cells are used instead of basic RNN cells is because of the gradient vanishing and exploding problems that the latter has, as stated before in the state of the art section.

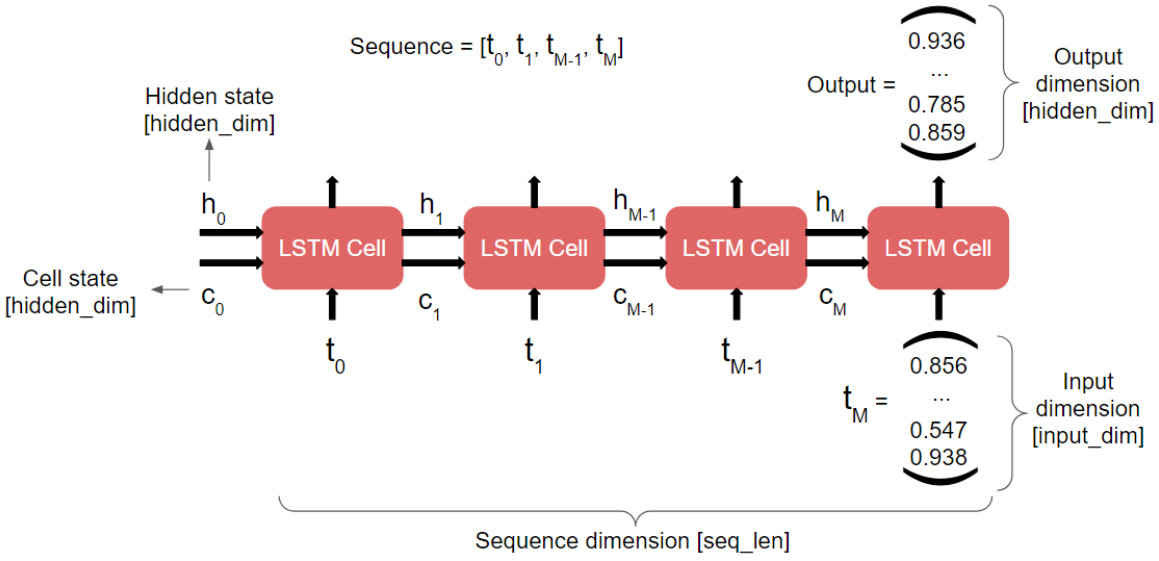


Figure 16. LSTM layer with dimensions.

The recurrent neural network consists of a single layer of LSTM cells. The total amount of LSTM cells that the model has is equal to the length of the sequence of pick-ups and drop-offs minus one, as the last pick-up in the sequence is the one that has to be predicted and this is compared to the output of the model. Then the input sequence dimension is equal to 6.

The result of the attention model is entered at the input of each LSTM cell, then the input dimension matches the output dimension of the attention model which is 90.

The hidden state and cell state dimension is 110. This dimension matches the output dimension of a cell because the output of an LSTM cell matches the hidden state.

Of all the outputs of the LSTM, the only output used is that of the LSTM cell with the last drop-off in the sequence as input. This output is passed through a dropout layer that during training, randomly zeroes some of the elements of the output of the LSTM with probability 0.5. A dropout layer is an effective technique for regularization, which makes the model generalize better.

3.2.5. Loss functions

3.2.5.1. As classification

When the model functions as a classifier, it is predicting the class among a group of classes, where each class has a unique identifier.

The model by classification (figure 17) has to predict a hotspot among a group of hotspots. That is, it considers each one of the hotspots like different classes.

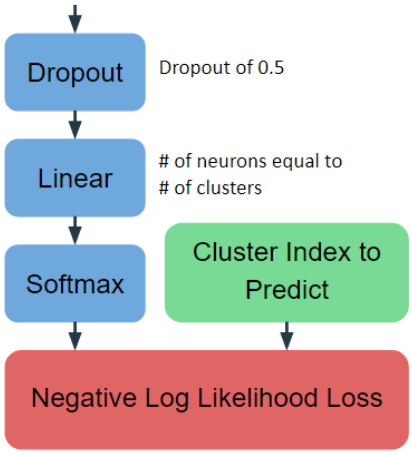


Figure 17. Loss function for classification.

The first layer that it has is a linear layer. The number of neurons in this linear layer equals the total number of clusters that have been created. The number of neurons could also be equal to the total number of hotspots but due to the indices they have match those of the clusters, and the hotspots are a subset of the clusters, which means that the total amount of hotspots is less than the total number of clusters. Hotspot indices must be mapped so that their values range from zero to the maximum number of hotspots minus one.

After the linear layer, there is a softmax layer which is used to sum the probabilities obtained in the linear layer up to 1. The output of the softmax describes the probability of each of the existing hotspots being predicted.

Finally, the softmax result and the target to be predicted are entered into the negative log-likelihood loss function (NLLLoss). The network assigns high confidence to the correct class, the “unhappiness” is low, but when the network assigns low confidence to the correct class, the “unhappiness” is high [28].

The merging of applying a softmax layer and a negative log-likelihood loss function is the same as applying the cross entropy loss directly.

The value of the loss function obtained is not enough to know if the model is really doing well, so it is necessary to apply other metrics. In this application we will use the confusion matrix and the histogram of distances. Despite the accuracy has also been used, it did not provide enough information.

3.2.5.2. As regression

The regression model (figure 18), instead of predicting a hotspot, predicts the latitude and longitude of a hotspot from the total number of existing hotspots.

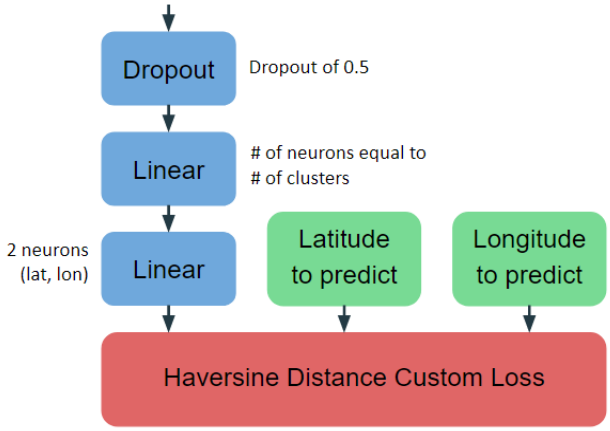


Figure 18. Loss function for regression.

The first layer is a linear layer with the number of neurons equal to the number of existing clusters. Also, as in the case of classification, the number of neurons can be equal to the number of hotspots. It should be noted then, that the values of the indexes must be reassigned so that their values range from zero to the maximum number of hotspots minus one.

Then, instead of having a softmax layer as in the case of classification, it has a linear layer. In this case, the total number of neurons in this layer is two. Each of these neurons represents the latitude and longitude to be predicted.

Finally, a custom loss function based on Haversine distance has been created. At the input of this function there are the predicted latitude and longitude, and the latitude and the longitude to be predicted when training.

4. Results

The following section will show the different results obtained from the execution of the model. The results have been divided into different subsections. The first shows the results obtained as a classifier. The second shows the results as a regression. The third discusses the results by balancing the data. The fourth compares the results of the model with the results obtained from decision trees. Finally, the last section describes other tests that have been done.

Initially, the optimizer used by the model was the SGD. This was later replaced by the Adam optimizer and there was a noticeable improvement in the results obtained. In the case of regression, the prediction goes from 2 Km to 1.4 Km. Adam has been used as an optimizer in all of the results below. Unless otherwise stated, all results refer to the validation dataset.

4.1. Model as classification

When the model operates as a classifier, it is predicting the class among a group of classes, where each class has a unique identifier. These classes can be hotspots and clusters or also the neighborhoods and districts of Barcelona for example. In this case the classes will be the hotspots, so the goal of this work is to predict hotspots.

The parameters used were: batch size of 128, a learning rate of 0,001 in the first 15 epochs and a learning rate of 0,0001 for the following 35 epochs.With an overall of 50 epochs.

The accuracy obtained is 36% of a total of 10925 predictions. However, by analysing the results, it can be clearly seen that the predictions bring the driver to the most concurrent hotspots. These concurrent hotspots have constant demand and do not represent a missed opportunity for the driver. Therefore, in 64% of the cases the vehicle will have assured demand (figure 19).

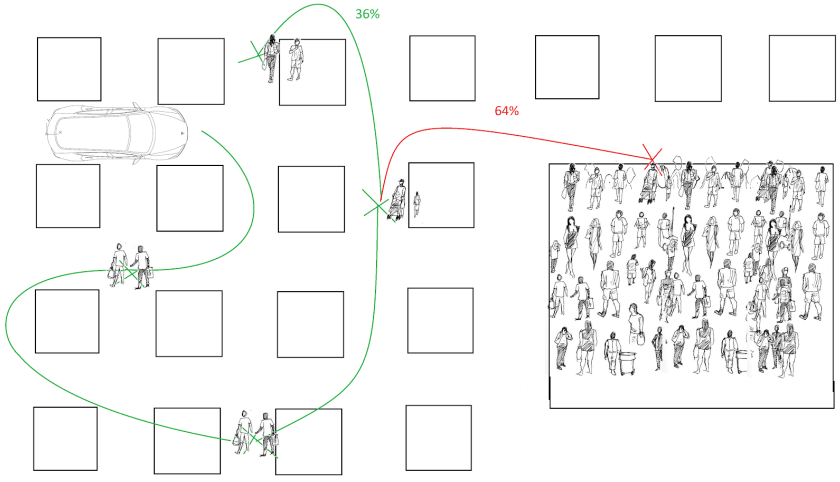


Figure 19. How the model works.

The following graphic (figure 20) shows the training and validation loss. This can be interpreted as, the lower its value, the better the model performs. On the contrary, when its value is high, the model is performing poorly. This graphic alone does not provide

enough information to know how well the model performs. Therefore, a confusion matrix and a distance histogram are required.

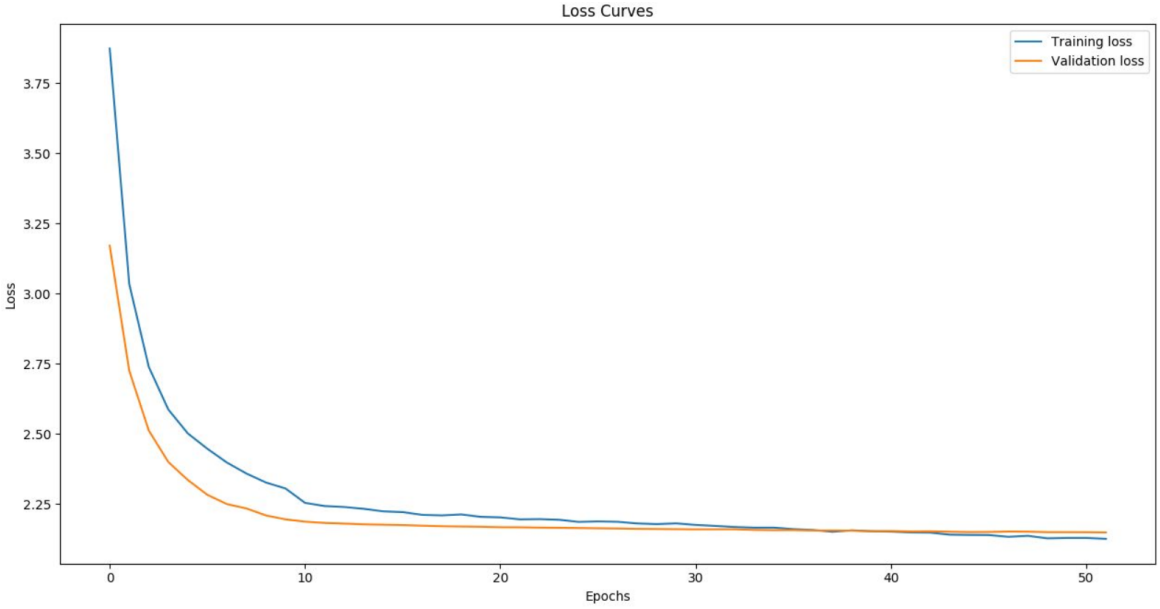


Figure 20. Classification loss curves.

In the confusion matrix (figure 21), the ordinate represents the classes to predict (the true classes) and the abscissa represents the predicted ones. If all the values fall into the diagonal, it means that the clases have been predicted successfully. In the generated confusion matrix can be seen the mentioned diagonal. However, some of the values fall outside diagonal meaning that the prediction has failed in some cases.

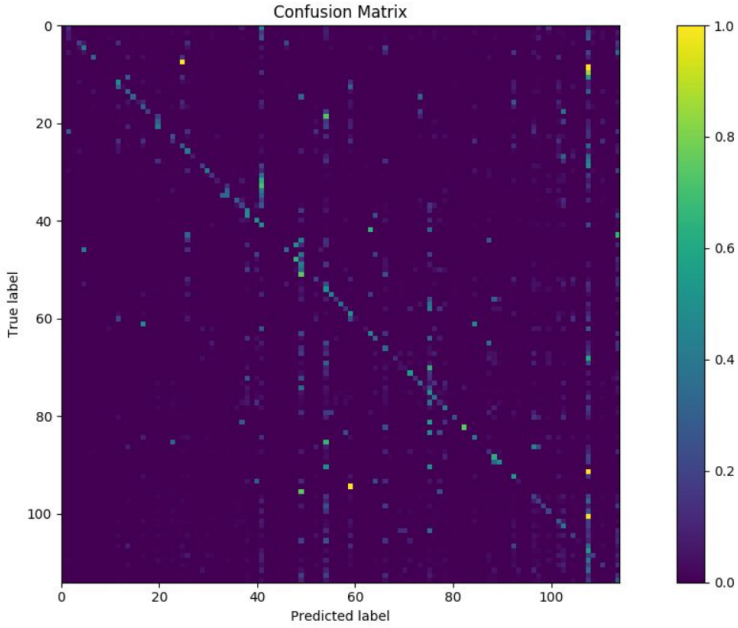


Figure 21. Classification confusion matrix.

The distances histogram (figure 22) represents the distances between the predicted hotspots and the expected ones.

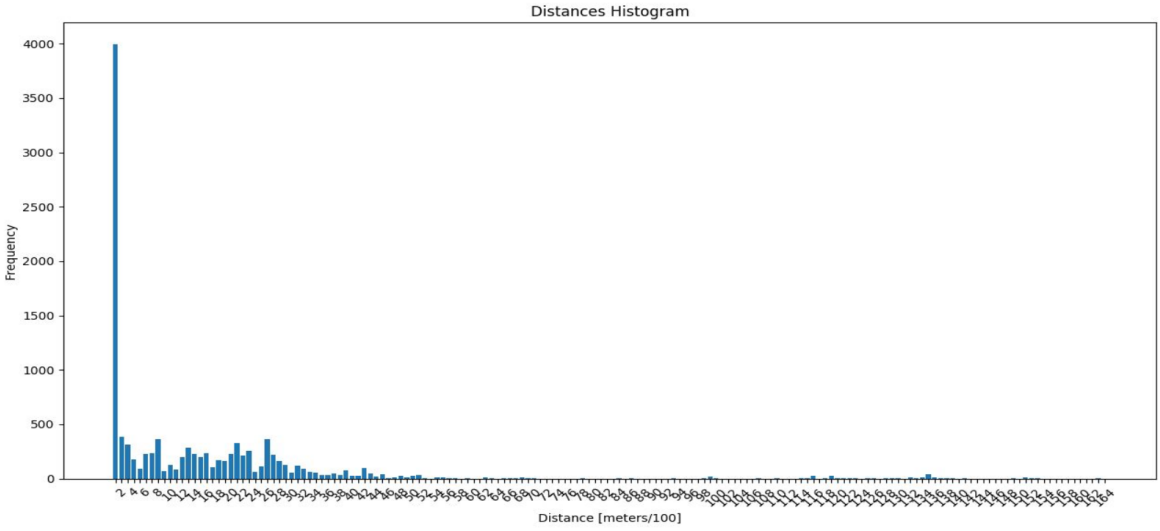


Figure 22. Classification distances histogram.

Every single bar represents an error of 100 meters. It can be seen that most of the cases fall between 0 and 100 meters which represents a successful prediction. Almost 36% of the prediction falls within the first bin. The rest falls abruptly and spread between 100 and 5000 meters beeing 5000 the most unlikely to happen. The rest of cases are considered exceptions.

4.2. Model as regression

When the model works as a regression, the prediction represents the latitude and longitude values of a particular hotspot instead of the hotspot index as in classification.

The parameters used were: batch size of 128, a learning rate of 0,0001 in the first 20 epochs and a learning rate of 0,00001 for the following 30 epochs.With an overall of 50 epochs.

The figure 23 represents a zoom in of the regression loss curves. This graphic can be interpreted better than the graphics obtained in classification because their values are the average of the distance error. The graphic shows that the curve stagnates which implies that the model is not learning.

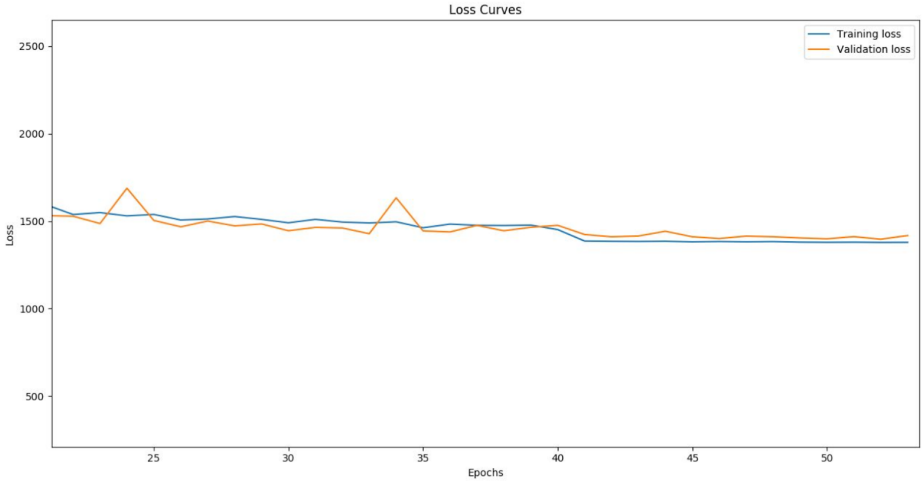


Figure 23. Regression loss curves.

Due to the nature of the analysis, the confusion matrix neither the accuracy value can not be obtained since these analyses are only valid for classification. However, the distance histogram (figure 24) could be obtained by using Haversine distance between the predicted coordinates and the expected ones.

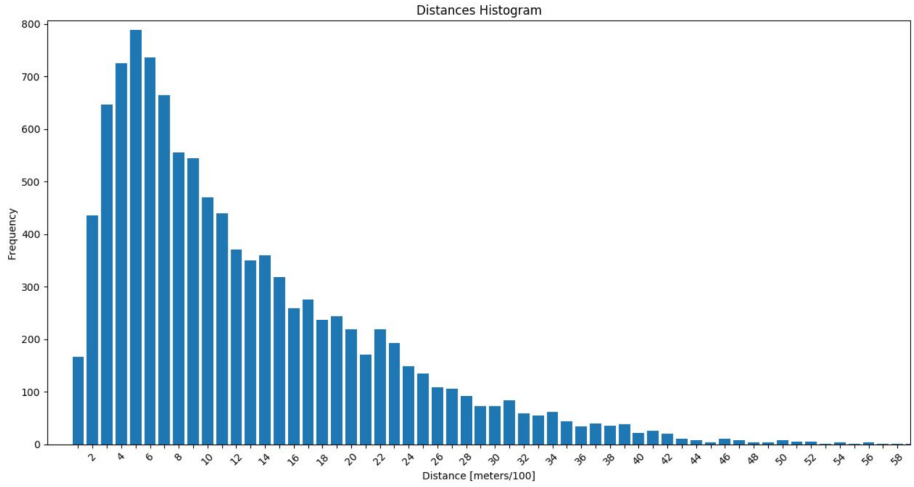


Figure 24. Regression distances histogram.

The figure shows how the distance distribution is decreasing exponentially and almost none of the values do not exceed more than 5 km. The cases falling further than 13 km are hotspots predicted in Barcelona rather than the airport or vice versa.

4.3. Balancing the data

The results of the model that have been seen so far ended up predicting predominant hotspots. This tendency may be attributed to the fact that the model has been trained with unbalanced data, meaning that the different classes to be predicted do not have the same number of data. Some classes had enough data but some others did not. What ends up happening is that the model predicts the most likely cases.

This is also observed in the confusion matrix of the model as a classifier seen in the previous section (Figure 21). In the confusion matrix, quite pronounced vertical lines are observed apart from the diagonal line. These belong precisely to the predominant hotspots. They indicate that the model instead of selecting the hotspot it had to predict, predicted precisely one of the predominant hotspots.

Different methods have been tried to balance the data. These have been upsampling, adding weights to the loss function or even changing the loss function that has been used so far for Focal loss [29]. Another available option could have been downsampling but it could not be used since there was not enough data.

The upsampling pretends to increase the number of samples in classes that had less data. This has been implemented by repeating the data of these classes to be commensurate to the predominant ones.

Weights have also been applied to the loss function to punish the predominant classes and highlight the lower populated ones. Its operation is equivalent to the upsampling.

Finally, an attempt has also been made to use Focal loss as a loss function. This method is used in image processing to differentiate the foreground from the background in an image. These two classes are very unbalanced and hence the need for this loss function. What is usually applied in binary classification can also be applied when there are more than two classes.

The graphic below shows the confusion matrix (figure 25) which has been generated after applying the weights to the loss function. The model has been configured as defined in section 4.1, only adding the weights to the loss function. At first glance, the model seems to be doing better because the diagonal is more pronounced and no vertical elements are observed. However, the results obtained are worse since the value of the accuracy is 16.9% of a total of 10925 predictions.

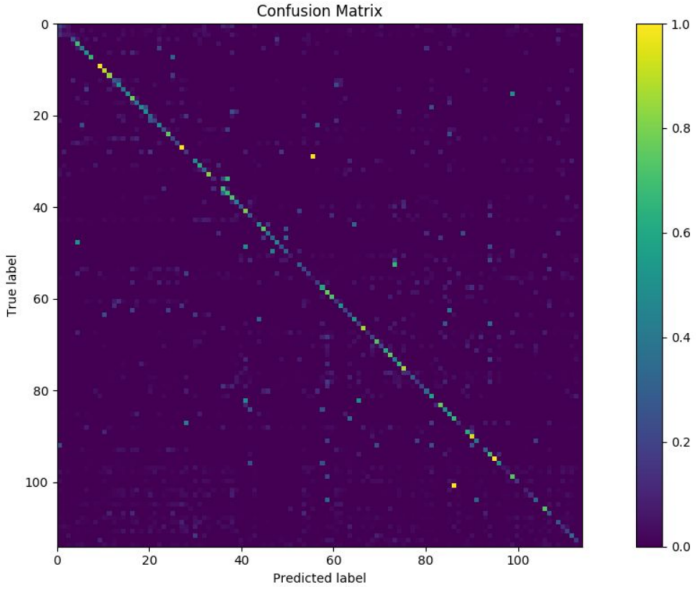


Figure 25. Classification with balanced data.

After applying all the methods separately to balance the data, the obtained results have been worse than the unbalanced ones.

4.4. Comparison with decision trees

The decision tree [30] is a machine learning algorithm. Unlike models based on neural networks, the internal decisions can be observed. The time to train these models is very fast.

A classifier based on decision trees has been implemented. The results obtained have been compared with those obtained with the model when it operates as a classifier. The results obtained (accuracy of 15%) are approximately 20% worse than those obtained using the deep learning model, which had an accuracy of 36%. It is also important to take into account that the data with which the decision tree has been trained was unbalanced. This affects the model, causing the decisions to be biased.

4.5. Other tests

Different tests have been done on the model. These tests include the one that does not have the attention model and the one that freezes the weights of the temporal cluster embedding. Tests have also been made by changing the entry data in the model and adding the data of the taxi races during the period 2010-2013 apart from those of 2014-2015.

Testing the model without the attention model does not bring any improvement. When the model works as a classifier, the results are slightly worse both in training and in validation.

Leaving the pre-trained weights of the temporal cluster embedding unfrozen results in a slight improvement in regression results at the beginning. Under the same conditions there is less error. When doing a complete training, the results obtained are quite similar.

A test was performed by changing the input pick-up and drop-off sequence so that all values in the sequence were hotspots. The results improve by lowering the average prediction distance to approximately 1200 meters. Doing so would really not make sense because in some sequences there may be the same hotspots consecutively.

Taxi data from the 2010-2013 period has been added to see if this will improve results. By doing so, there has been a slight improvement. For example, the error decreased below the obtained under the same conditions using only the 2014-2015 data by a 3%.

5. Budget

Dedicated GPUs are required to train deep learning models. These can be purchased, or contract them as a service that companies such as Google, Amazon or Microsoft offer, through which you can access the servers where they are installed.

When contracting servers with GPUs, different users share the same resources as which are limited, but not all users use them at the same time. In times of high demand you may not be able to access it if you have a basic plan, then you could also hire another plan that does not have this limitation, but the price may be very high.

In the case of physically buying one, they are usually expensive and the electricity consumption is high as well. This is not very profitable if the total number of hours required to train the model is not very high.

Deep learning models usually need to run for many hours to get results. The model used in this project, as it does not have a very large amount of data, the computation time needed is not very high, so the best option is hiring a server.

In order to train the model, the image processing group (GPI) servers were used free of charge. In order to simulate the actual price that the project would have had if it had run on a server, the hourly prices of servers such as AWS, Google Cloud, and Azure have been used as a reference. As the project has been developed in 19 weeks, the price per hour selected is part of a non-annual plan.

Employees					
Type	Quantity	Hourly price	Weekly hours	Weeks	Total
Senior engineer	1	25 €	2	19	950 €
Junior engineer	1	10 €	25	19	4750 €

Components			
Type	Hourly price	Total hours	Total
Cloud server with GPUs	0,7 €	50	35 €

TOTAL	5735 €

Table 3. Project budget.

6. Conclusions and future development:

A model capable of predicting hotspots has been developed over the course of this project by using deep learning. The data has been pre-processed so that it can be entered correctly into the model. This model consists of embedding layers, an attention model and a LSTM layer.

The hotspot prediction has been approached in two different ways: as a regression and as a classification. When the model operates as a classifier, it predicts a hotspot among a group of hotspots. Meanwhile, when the model operates as a regression, it predicts the latitude and longitude of the hotspot.

The metrics of the predictions obtained, such as the accuracy value, are low values, but these must be interpreted. The values at which the model is not wrong are not a problem, and in the case of the values at which it is wrong, as it is predicting hotspots, they cannot be considered serious errors either, since the algorithm is directing the taxi or car sharing to a high demand hotspot. This analysis was possible by obtaining the confusion matrices when the classifier was adopted. Histograms of distances also brought some lights to understand the results. Hotspots, as discussed above, vary depending on the time and day of the week, but the values of the hotspots that the model predicts are precisely hotspots that are present every day of the week and at any time.

A future development could be to do transfer learning by first training the model with a dataset with a lot of data and then adapting the model to the city of Barcelona, training it with the datasets from the cabs from Barcelona that do not have as much data. An example of a dataset with a lot of data that would be great for transfer learning would be the New York City taxi dataset, which has approximately over 100 million taxi rides a year.

The current model could include an event predictor based on the analysis of web page content, such as the one mentioned in the related work section, [11]. This one would be able to predict one-off events, such as concerts and football matches. In this way, the hotspots that are now activated and deactivated by the cyclical behavior of the taxi data, would also be activated when there were specific events.

The current model predicts the hotspot from a sequence of fixed-length pick-ups and drop-offs. Therefore, an improvement that could be implemented to the model would be that it could predict from a variable length, so that the model could make predictions without the need of having any ride.

Bibliography:

[1] De Brébisson A., Simon E., Auvolat A., Vincent P., Bengio Y. “Artificial Neural Networks Applied to Taxi Destination Prediction”, arXiv:1508.00021, September 2015.
[2] Observatori del Taxi - Àrea Metropolitana de Barcelona. Provided by Miquel Estrada.
[3] AEMET - Meteorological data. [Online] Available: https://datosclima.es/
[4] Cython. [Online] Available: https://cython.org/
[5] QGIS. [Online] Available: https://www.qgis.org/es/site/
[6] Xavi Roig Aznar, “Destination prediction of car sharing and other vehicles using deep learning”. Degree thesis, Faculty of the Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona, Universitat Politècnica de Catalunya, Barcelona, 2014.
[7] Alberto Rossi, Gianni Barlacchi, Monica Bianchini, Bruno Lepri. “Modeling Taxi Drivers’ Behaviour for the Next Destination Prediction”, arXiv:1807.08173, January 2019.
[8] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, Zhenhui Li. “Deep Multi-View Spatial-Temporal Network for Taxi Demand Prediction”, arXiv:1802.08714, February 2018.
[9] Jintao Ke, Hongyu Zheng, Hai Yang, Xiqun (Michael)Chen. “Short-Term Forecasting of Passenger Demand under On-Demand Ride Services: A Spatio-Temporal Deep Learning Approach”, arXiv:1706.0627, June 2017.
[10] Amin Sadri, Flora D. Salim, Yongli Ren, Wei Shao, John C. Krumm, and Cecilia Mascolo. 2018. “What Will You Do for the Rest of the Day? An Approach to Continuous Trajectory Prediction” Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 2, 4, Article 186 (December 2018), 26 pages. https://doi.org/10.1145/3287064
[11] Filipe Rodrigues, Ioulia Markou, Francisco Pereira. “Combining time-series and textual data for taxi demand prediction in event areas: a deep learning approach”, arXiv:1808.05535, August 2018.
[12] “Calculate distance, bearing and more between Latitude/Longitude points”. *Movable Type Scripts*. [Online] Available: https://www.movable-type.co.uk/scripts/latlong.html. [Accessed: June 2020].
[13] Daniel Lenton. "Part II: Projective Transformations in 2D". *Medium*, 2019. [Online] Available: https://medium.com/@daniel.j.lenton/part-ii-projective-transformations-in-2d-2e99ac9c7e9f. [Accessed: June 2020].
[14] Richard Hartley, Andrew Zisserman. “Multiple View Geometry in Computer Vision”, 2nd ed. Cambridge University Press. ISBN: 9780521540513.
[15] "How HDBSCAN Works". HDBSCAN *Reference - Read the Docs*. [Online] Available: https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html. [Accessed: June 2020].
[16] "Comparing Python Clustering Algorithms". *Notebooks - Jupyter NBViewer*. [Online] Available: https://nbviewer.jupyter.org/github/scikit-learn-contrib/hdbscan/blob/master/notebooks/Comparing%20Clustering%20Algorithms.ipynb [Accessed: June 2020].
[17] "Clustering geolocation coordinates". *Datascience - Stackexchange*, 2017. [Online] Available: https://datascience.stackexchange.com/a/848. [Accessed: June 2020].
[18] Jason Brownlee. "What Are Word Embeddings for Text?". *Machine Learning Mastery*, 2017. [Online] Available: https://machinelearningmastery.com/what-are-word-embeddings. [Accessed: June 2020].
[19] Neeraj Singh Sarwan "An Intuitive Understanding of Word Embeddings". Analytics Vidhya, 2017. Available: https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/. [Online] [Accessed: June 2020].
[20] McCormick, C. "Word2Vec Tutorial - The Skip-Gram Model". Chris McCormick, 2016. [Online] Available: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/. [Accessed: June 2020].
[21] Colyer A. “The Amazing Power of Word Vectors”. *kdnuggets, 2016* [Online] Available: *https://www.kdnuggets.com/2016/05/amazing-power-word-vectors.html.* [Accessed: June 2020].
[22] Purnasai Gudikandula. "Recurrent Neural Networks and LSTM explained". *Medium*, 2019. [Online] https://medium.com/@purnasaigudikandula/recurrent-neural-networks-and-lstm-explained-7f51c7f6bbb9. [Accessed: June 2020].
[23] Aditi Mittal. "Understanding RNN and LSTM". *Towards Data Science*, 2019. [Online] Available: https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e. [Accessed: June 2020].

[24] Gabriel Loye. "Long Short-Term Memory: From Zero to Hero with PyTorch". *Floydhub*, 2020. [Online] https://blog.floydhub.com/long-short-term-memory-from-zero-to-hero-with-pytorch/. [Accessed: June 2020].

[25] Anusha Lihala. "Attention and its Different Forms". *Towards Data Science*, 2019. [Online] Available: https://towardsdatascience.com/attention-and-its-different-forms-7fc3674d14dc. [Accessed: June 2020].

[26] Guillaume Genthial, Lucas Liu, Barak Oshri, Kushal Ranjan "Lecture Notes: Part VI Neural Machine Translation, Seq2seq and Attention". *Course CS224n - Stanford*, winter 2019. [Online] Available: http://web.stanford.edu/class/cs224n/readings/cs224n-2019-notes06-NMT_seq2seq_attention.pdf. [Accessed: June 2020].

[27] Gensim. [Online] Available: https://radimrehurek.com/gensim/

[28] LJ Miranda. "Understanding softmax and the negative log-likelihood". *Lj Miranda*, 2017. [Online] https://ljvmiranda921.github.io/notebook/2017/08/13/softmax-and-the-negative-log-likelihood/. [Accessed: June 2020].

[29] Lin T., Goyal P., Girshick R., He K., Dollár P. “Focal Loss for Dense Object Detection”, arXiv:1708.02002, February 2018.

[30] Avinash Navlani. “Decision Tree Classification in Python”. Datacamp, 2018. [Online] Available: https://www.datacamp.com/community/tutorials/decision-tree-classification-python. [Accessed: June 2020].

Glossary

RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
AM	Attention Model
NLP	Natural Language Processing
GPI	Image Processing Group
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
HDBSCAN	Hierarchical DBSCAN
CBOW	Common Bag of Words
CRS	Coordinate Reference System
NLLLoss	Negative Log-Likelihood Loss function
SGD	Stochastic Gradient Descent
ML	Machine Learning
DL	Deep Learning