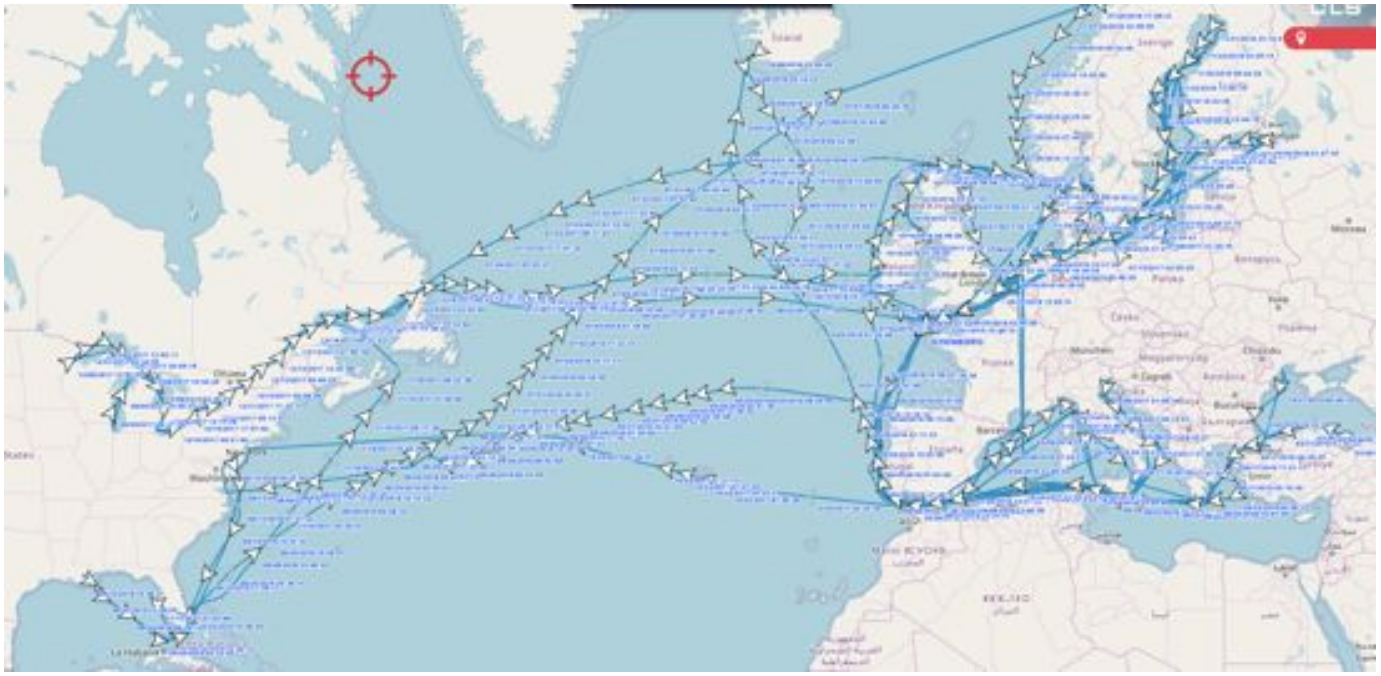
INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

RAPPORT DU PROJET

PRÉDICTION D'ESCALES POUR DES CARGOS



Réalisé par :
LAGRINI Youness
BENMADA Ayman
DZIANISAU Constantin
KABBAJ Fatine

Encadré par :
Marie-José HUGUET
Mohamed SIALA
Slim ABDELLATIF
Jean Michel Farenc

Soutenu le 25-01-2019

Table des matières :

I. Introduction :

A. Contexte :

Les déplacements de navires (cargos) sont collectés par différents systèmes (radio, radar, satellites) afin de pouvoir surveiller le trafic maritime que ce soit pour des activités spécifiques (comme la pêche) ou le suivi de flottes de compagnies maritimes.
En agrégeant ces diverses données, la société CLS construit un historique des escales de navires maritimes. Le cas d'utilisation envisagé pour ce projet consiste à prédire les futures escales de navires à deux niveaux : spatial (port) et temporel (durée de l'escale et des déplacements). Cela permettra à un organisme d'inspection de planifier des visites de navires.
L'historique des escales est composé de l'ensemble des escales enregistrées entre le 01/12/2015 et le 31/11/2018. Il fournit un identifiant de navire, un identifiant de port avec le nom ; une date de début d'escale dans le port et une date de fin d'escale.
Un second fichier fournit une liste de navires spécifiques (des cargos) pour lesquels une prédiction est attendue.

B. Objectifs :

L'objectif du projet est de prédire le jour t, les prochaines escales (identifiant de port et fenêtre temporelle) de l'ensemble des cargos dans une fenêtre allant de t+5 jours à t+20 jours. A cette prévision devra être associé un niveau de pertinence (valué entre 0 et 100). Il est important d'éviter les prédictions de type « faux positifs ».
Une extension de ce travail est de se placer dans un contexte « dynamique » où la prédiction effectuée le jour t sera affinée le jour suivant t+1 (prédiction sur l'horizon t+8 à t+21).
L'ensemble des données fournies devra être partitionné à un niveau temporel pour permettre de rejouer les situations passées. Pour évaluer les méthodes proposées, il est possible, en se plaçant à une date dans le passé, de faire des prédictions et de vérifier si elles ont bien eu lieu ou non.
La validation de la qualité / performance d'une méthode peut se faire selon différentes mesures :

- Qualité de la prédiction spatiale (l'escale prévue a eu lieu dans l'intervalle t+5, t+20).
- Qualité de la prédiction temporelle (écart entre fenêtre prévue et fenêtre réalisée).

C. Motivations :

Les problèmes de prédiction de séries chronologiques (aussi appelées séries temporelles) constituent un défi dans de nombreux domaines. En finance, les experts prévoient des cours de bourse ou des indices boursiers; les spécialistes du traitement des données prévoient le flux d'informations sur leurs réseaux; les producteurs d'électricité prévoient la charge du lendemain. Le point commun à leurs problèmes est le suivant : comment analyser et utiliser le passé pour essayer de prédire l'avenir?
Contrairement à ce que nous avons vu durant les cours et les travaux pratiques d'apprentissage supervisé et non supervisé, les séries chronologiques ajoutent également la complexité de dépendance de séquence entre les variables d'entrée.

II. Solutions envisagées :

A. Les chaînes de Markov :

Une chaîne de Markov est un système mathématique qui subit des transitions d'un état à un autre en fonction d'un ensemble de règles probabilistes. Les chaînes de Markov sont des processus stochastiques, mais elles se différencient par le fait de ne pas avoir de "mémoire": quel que soit le processus parvenu à l'état actuel, les états futurs possibles sont fixés. En d'autres termes, la probabilité de passer à un état particulier dépend uniquement de l'état actuel et non de la séquence des états précédents.
Une chaîne de Markov est définie par les deux propriétés suivantes :

- Un espace d'état : un ensemble de valeurs ou d'états dans lequel un processus pourrait exister.
- Un opérateur de transition : définit la probabilité de passer d'un état à un autre.

B. Les réseaux de neurones récurrents :

L'idée derrière les RNN est d'utiliser des informations séquentielles. Dans un réseau de neurones traditionnel, nous supposons que toutes les entrées et sorties sont indépendantes les unes des autres. Mais pour de nombreuses tâches, c'est une très mauvaise idée. Si vous voulez prédire le mot suivant dans une phrase, vous devez savoir quels mots l'ont précédé. Les RNN sont appelés récurrents car ils effectuent la même tâche pour chaque élément d'une séquence, la sortie étant dépendante des calculs précédents. Une autre façon de penser aux RNN est qu'ils ont une «mémoire» qui capture des informations sur ce qui a été calculé jusqu'à présent. En théorie, les RNN peuvent utiliser les informations dans des séquences arbitrairement longues. Voici à quoi ressemble un RNN typique:

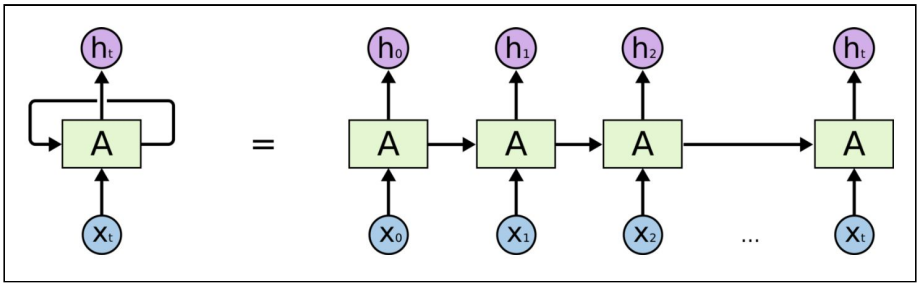


Figure : un réseau de neurones récurrent

Il existe de nombreuses variantes de réseaux de neurones récurrents, telles que la LSTM (Long-Short Term Memory) que nous utiliserons dans ce projet et la GRU (Gated Recurrent Unit), un peu plus simple.

1. Long-Short Term Memory - LSTM :

Un type puissant de réseau neuronal conçu pour gérer la dépendance de séquence est appelé réseau de mémoire à long terme ou à court terme LSTM.
Les LSTM permettent aux RNN de se souvenir de leurs entrées sur une longue période. En effet, les LSTM stockent leurs informations dans une mémoire, ce qui ressemble beaucoup à

la mémoire d'un ordinateur car le LSTM peut lire, écrire et supprimer des informations de sa mémoire. Cette mémoire peut être vue comme une cellule fermée avec des portes, où chaque porte signifie que la cellule décide ou non de stocker ou de supprimer des informations, en fonction de l'importance qu'elle attribue à l'information. L'attribution d'importance se fait par le biais de pondérations, également apprises par l'algorithme. Cela signifie simplement qu'il apprend avec le temps quelles informations sont importantes et lesquelles ne le sont pas.
Dans un LSTM, on a trois portes:

a. Une porte d'entrée (input gate) : détermine s'il faut garder ou non une nouvelle entrée.
b. Une porte d'oubli (forget gate) : décide quelles informations à supprimer de l'état de la cellule car elles n'ont plus d'importance.
c. Une porte de sortie (output gate) : décide ce que la cellule va produire. La sortie dépend à la fois des anciennes informations et de l'entrée actuelle.

2. Gated Recurrent Unit - GRU :

Une des variantes les plus populaires du LSTM est GRU. Il combine les portes d'oubli et d'entrée dans une seule porte de mise à jour (update gate). Il fusionne également l'état de la cellule et l'état caché, et apporte d'autres modifications. Le modèle résultant est plus simple que les modèles LSTM standard et connaît une popularité croissante.
Un GRU a deux portes:

a. Une porte de mise à jour : détermine comment combiner la nouvelle entrée avec la mémoire précédente et définit la quantité de mémoire précédente à conserver.
b. Une porte de sortie : décide de ce que la cellule va produire. La sortie dépend à la fois des anciennes informations et de l'entrée actuelle.

III. Implémentation :

A. Prétraitement des données :

Une phase préliminaire et nécessaire dans notre cas est la phase de prétraitement des données. Les données chronologiques doivent être préparées avant de pouvoir être utilisées pour former un modèle d'apprentissage.
Le jeu de données original contient des redondances au niveau des escales effectuées par les bateaux, un bateau fait deux escales consécutives dans le même port dans une durée minimale à cause du fait qu'une escale est notée au moment où la vitesse du bateau est nulle. Ainsi, un simple mouvement du bateau dans le même port peut générer deux escales consécutives. Un job map reduce est utilisé pour fusionner ces redondances afin d'éliminer les duplications et ne garder que la première date d'entrée au port et la dernière date de sortie.
Le jeu de données résultant est divisé en un jeu de données contenant toutes les escales effectuées avant le premier juin 2018 pour l'apprentissage, et un jeu de données contenant tous les événements après cette date pour le test.
Finalement, afin d'adapter la base d'apprentissage à notre modèle d'apprentissage choisi, nous allons construire une base de séquences des codes ports associés aux escales

effectuées à partir des historiques de bateaux. Pour chaque bateau du jeu de données d'apprentissage, on trie son historique selon la date d'entrée puis on transmet son historique des escales en une phrase dont les mots représentent les codes ports.

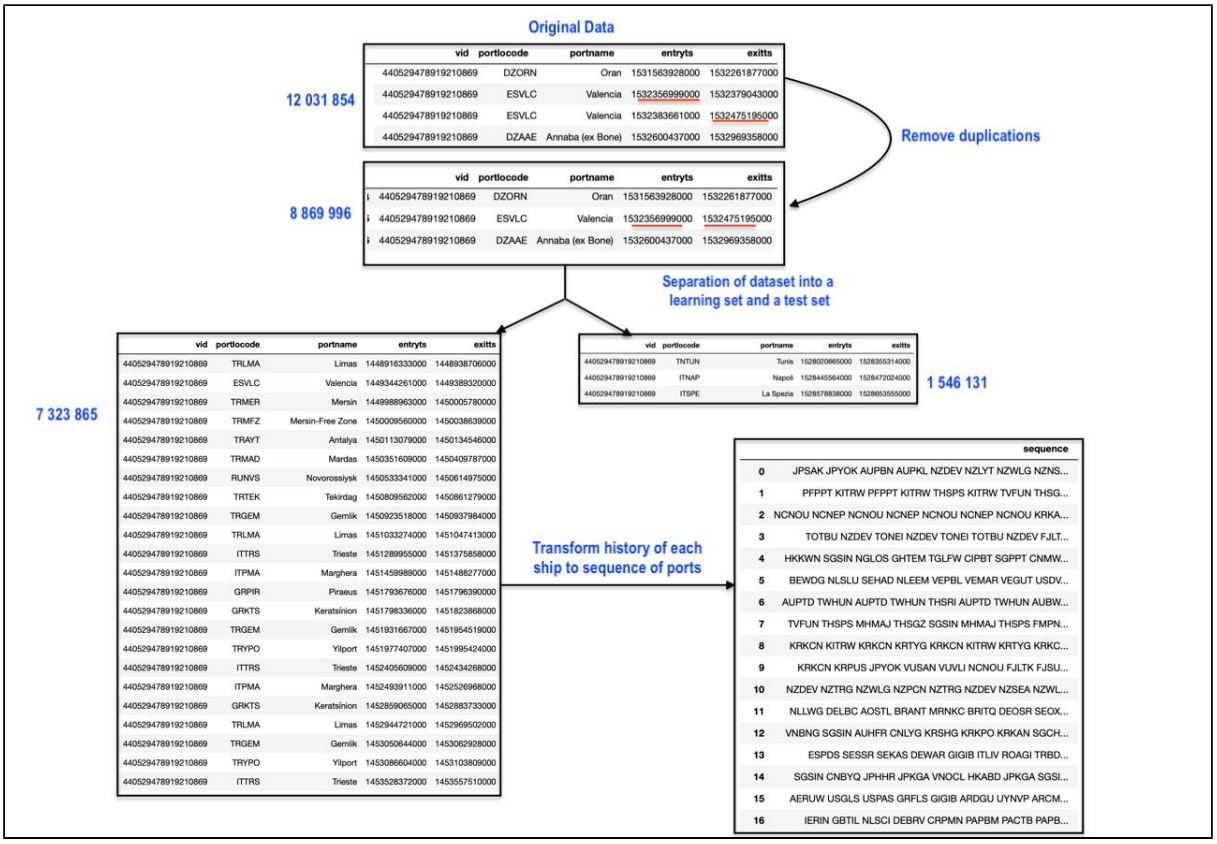


Figure : prétraitement des données

B. Apprentissage :

Pour réaliser cette phase, nous avons besoin de plusieurs étapes de traitement. Premièrement, nous devons convertir les mots-textes bruts en jetons, qui sont des valeurs entières. Ces jetons ne sont en réalité que des indices dans une liste de tout le vocabulaire. Ensuite, nous convertissons ces jetons entiers en vecteurs à valeurs réelles, dont le mapping sera réalisé en même temps que le réseau de neurones afin de mapper des mots ayant une signification similaire à des vecteurs similaires. Finalement, nous entrons ces vecteurs d'incorporation dans un réseau de neurones récurrents qui peut prendre. Tout ce processus nous permet de connaître les probabilités associées à chaque port d'être le prochain port d'escale.

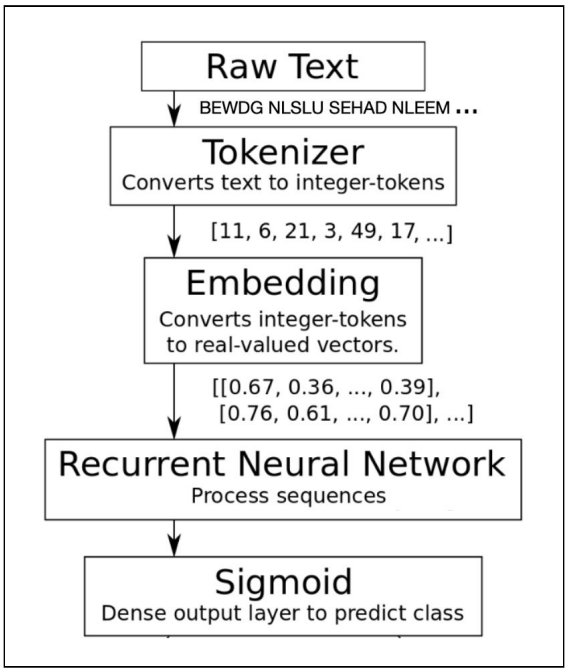


Figure : Organigramme résumant les étapes de la phase d'apprentissage

1. Transformation en jetons (tokens) :

Un réseau de neurones ne peut travailler directement sur des chaînes de texte, nous devons donc les convertir. Deux étapes sont nécessaires pour cette conversion : la première étape est appelée le "tokenizer" ou transformation en jetons qui convertit les mots en entiers, elle est effectuée sur le jeu de données avant qu'il ne soit entré dans le réseau de neurones; la seconde étape est une partie intégrée du réseau de neurones lui-même et est appelée la couche "embedding" ou couche d'intégration, décrite plus en détail ci-dessous.
Le tokenizer construit un vocabulaire de tous les mots uniques, ainsi à chaque code du port sera associé un entier.

2. Rembourrage (padding) et troncation des données :

Le réseau de neurones récurrents peut prendre en entrée des séquences de longueur arbitraire, mais pour utiliser un lot complet de données, les séquences doivent avoir la même longueur. La solution est donc de s'assurer que toutes les séquences de l'ensemble des données ont la même longueur, mais si nous utilisons la longueur de la séquence la plus longue dans le jeu de données, beaucoup de mémoire sera perdue. Ceci est particulièrement important pour de grands ensembles de données.
Pour ces raisons, durant la phase de prétraitement des données, nous avons construit les séquences de sorte qu'elles aient au maximum 282 codes ports. Ce nombre représente la moyenne des longueurs des historiques de bateaux auquel s'ajoute un écart type. Cela couvre environ 90% de l'ensemble des données. Afin de ne pas perdre les informations à partir des séquences dont la longueur est strictement supérieure à 282, nous avons utilisé une stratégie de découpage de ces séquences en sous séquences de longueur 282.
Nous devons ensuite déterminer si le remplissage sera effectué en "pré" ou "post". Si une séquence est complétée, cela signifie que des zéros sont ajoutés à la séquence.

Par conséquent, le choix de “pré” ou “post” peut être important car il détermine s’il faut ajouter des zéros au début ou à la fin de la séquence lors du remplissage. Cela peut perturber le réseau de neurones récurrents. Le choix de “pré” est plus raisonnable ici, car le réseau de neurones récurrents commence par beaucoup de zéros et tombe sur la séquence réelle.

3. Création du réseau de neurones récurrent :

On est maintenant prêt à créer le réseau de neurones récurrents (RNN). On a utilisé l'API Keras pour cela grâce à sa documentation et sa simplicité.
La première couche du RNN est une couche dite “embedding” ou couche d'intégration qui convertit chaque jeton entier en un vecteur de valeurs. Cela est nécessaire car les entiers-jetons peuvent prendre des valeurs comprises entre 0 et 6491 pour notre vocabulaire de mots. Le RNN ne peut pas travailler sur des valeurs dans une plage aussi large. La couche d'intégration fait partie du RNN et apprendra à mapper des mots ayant une signification sémantique similaire à des vecteurs d’incorporation similaires.
Nous allons utiliser deux couches cachées LSTM avec 100 cellules de mémoire chacune. Plus de cellules de mémoire et un réseau plus profond nous ont donné de meilleurs résultats.
Finalement, une couche dense fully connected avec 100 neurones se connecte aux couches cachées de LSTM pour interpréter les caractéristiques extraites de la séquence. La couche de sortie prédit le mot suivant sous forme d’un vecteur unique dont la longueur est égale à la taille du vocabulaire avec une probabilité pour chaque mot du vocabulaire. Une fonction d'activation softmax est utilisée pour s'assurer que les sorties ont les caractéristiques de probabilités normalisées.

4. Apprentissage du réseau de neurones récurrents :

Ensuite, le modèle est compilé en spécifiant les paramètres nécessaires des fonctions d'optimisation et de calcul de perte pour améliorer les poids utilisés dans le réseau.
Enfin, le modèle est ajusté sur les données avec 150 époques d’entraînement et une taille de lot modeste de 256 unités pour accélérer les choses.
L’apprentissage prend des heures sur du matériel moderne sans GPU.

C. Prédiction des durées :

La prédiction des durées est divisée en deux parties. Dans un premier temps, nous allons effectuer des agrégats sur le jeu de données d’apprentissage pour la prédiction de la durée d’escale dans un port pour chaque bateau. Dans un deuxième temps, nous allons effectuer des agrégats sur un jeu de données construit à partir du jeu de données d’apprentissage pour la prédiction de la durée entre deux ports pour chaque bateau.

1. Prédiction de la durée d’escale dans un port :

D’abord, on calcule la différence entre la date de sortie et la date d’entrée dans un port pour toutes les lignes du jeu de données. Ensuite, on fait appel à des méthodes de la librairie Pandas de Python pour calculer la médiane et la moyenne de ces différences, qui

représentent la durée dans le port, pour chaque bateau et chaque port. Finalement, on obtient un jeu de données contenant comme indices l'id du bateau et le code du port et comme champs "médiane" et "moyenne" qui représentent respectivement la médiane et la moyenne calculées en agrégeant les différentes escales du jeu de données.

2. Prédiction de la durée entre deux ports :

D'abord, un prétraitement sur le jeu de données d'apprentissage est effectué afin d'obtenir un jeu de données contenant l'id du port, le port de départ, le port d'arrivée, la date de sortie du port de départ et la date d'entrée au port d'arrivée. Par la suite, on calcule la différence entre les dates pour chaque ligne du jeu de données obtenu afin d'obtenir la durée entre les deux ports. Finalement, comme dans la partie précédente, nous faisons appel à des méthodes de la librairie Pandas de Python pour calculer la médiane et la moyenne de ces différences, qui représentent la durée entre les deux ports, pour chaque bateau et chaque couple "port de départ-port d'arrivée".

D. Tests et résultats:

1. Tests :

vid	portlocode	portname	entryts	exitts
440529478919210869	TNTUN	Tunis	1528020665000	1528355314000
440529478919210869	ITNAP	Napoli	1528445564000	1528472024000
440529478919210869	ITSPE	La Spezia	1528578838000	1528653555000
440529478919210869	FRPSL	Port-Saint-Louis-du-Rhône	1528772150000	1528802138000
440529478919210869	ESBCN	Barcelona	1528864741000	1528903831000
...	...	...	...	...

Figure : Jeu de données de tests contenant les escales à partir 01-06-2018

Pour effectuer les tests, on considère chaque ligne du jeu de données de tests comme le jour j+1 qu'on doit prédire en fonction des escales effectuées précédemment. Ainsi pour chaque ligne du jeu de données on effectue la procédure suivante :

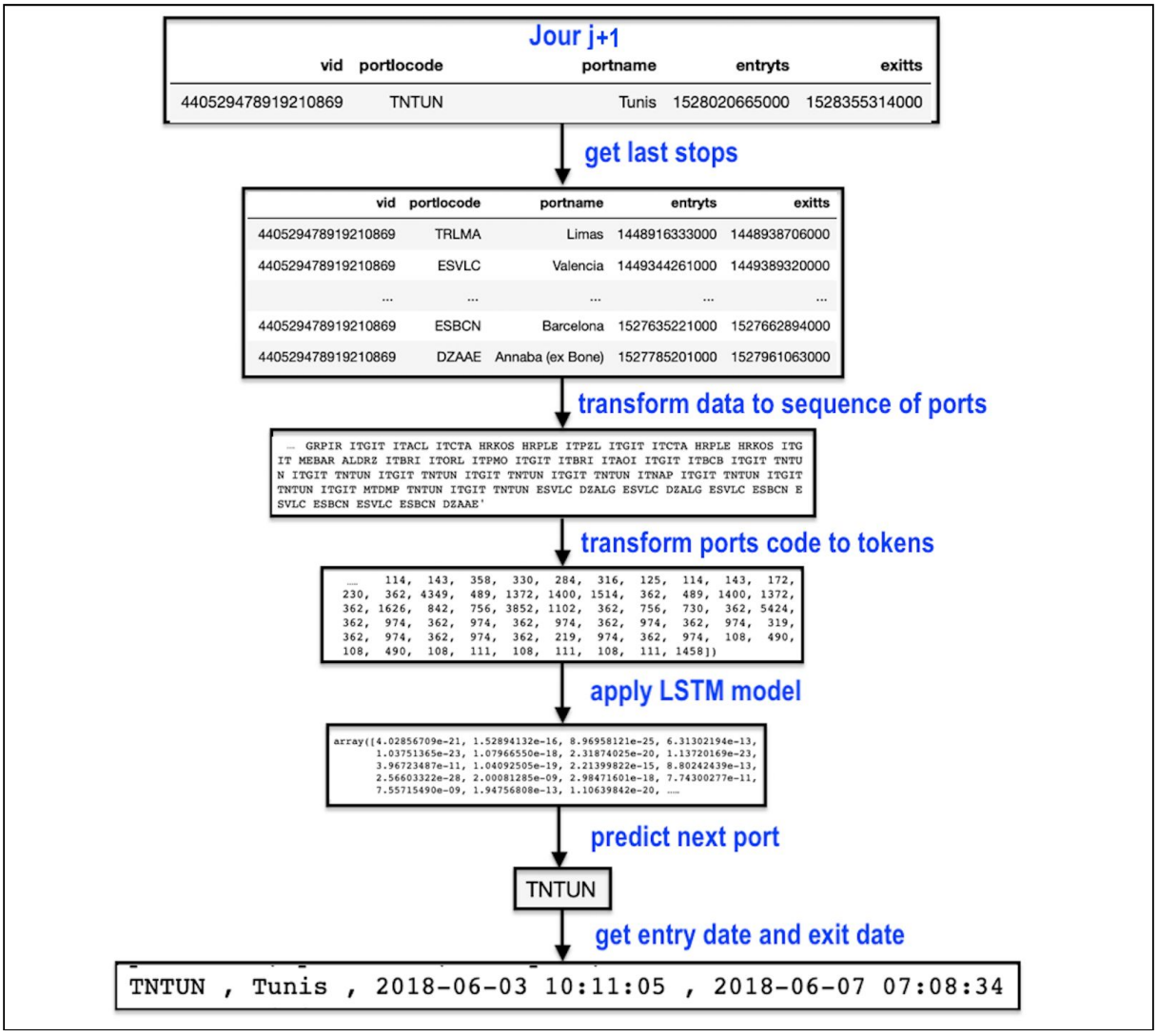


Figure : Processus de test pour chaque ligne du jeu de données

Pour prédire le prochain port d'escale, nous avons adopté quatre approches différentes :

a. 1ère approche : Modèle LSTM :

En se basant simplement sur les probabilités obtenues avec le modèle LSTM présenté précédemment, on obtient le code du port de la prochaine escale en convertissant le token ayant la plus grande probabilité en string.

b. 2ème approche : Modèle LSTM + Historique du bateau :

Comme son nom l'indique, on élimine tous les ports qui ne figurent pas dans l'historique du bateau et on prend le token ayant la plus grande probabilité. On convertit par la suite le token en un code port.

c. 3ème approche : chaînes de Markov :

Le principe des chaînes de Markov est expliqué dans la partie des solutions envisagées. Dans notre cas, les états sont représentés par le port d'escale et la transition représente la probabilité de partir du port actuel à n'importe quel autre port pour un bateau donné. En se

basant sur ces probabilités, on prend le code du port ayant la plus grande probabilité et on le considère comme le prochain port d'escale.

d. 4ème approche : Modèle LSTM + Historique du bateau + chaînes de Markov :

Dans cette approche, on ne prend que les probabilités résultantes du modèle LSTM pour les ports figurant dans l'historique du bateau auxquelles on ajoute les probabilités obtenues par le modèle des chaînes de Markov. Par la suite, le token ayant la plus grande probabilité est considéré comme le token du port de la prochaine escale. On convertit ce token en string pour obtenir le code du port.

Après l'obtention du code du port de la prochaine escale, on détermine la date d'entrée et la date de sortie au port à partir des jeux de données obtenues dans la partie de la prédiction des durées.

2. Résultats :

Nous avons effectué les tests dans un premier temps sur toute la base de tests, et dans un second temps sur les bateaux d'intérêts de la base de tests uniquement.

a. Prédiction de la date d'entrée et la date de sortie :

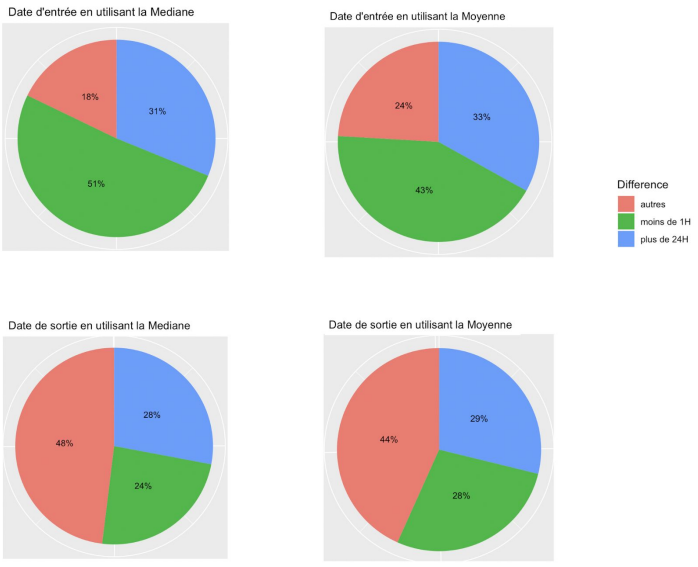


Figure : Comparaison des valeurs réelles avec les valeurs prédites sur toute la base de tests

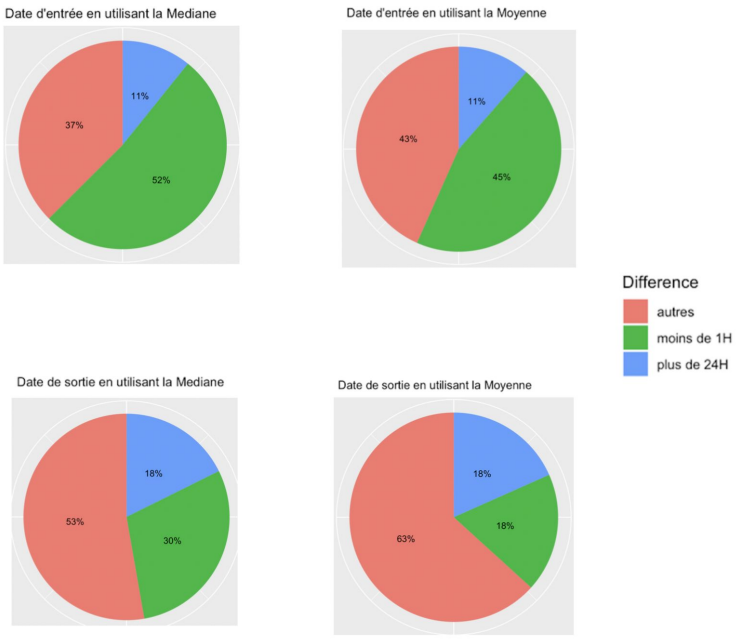
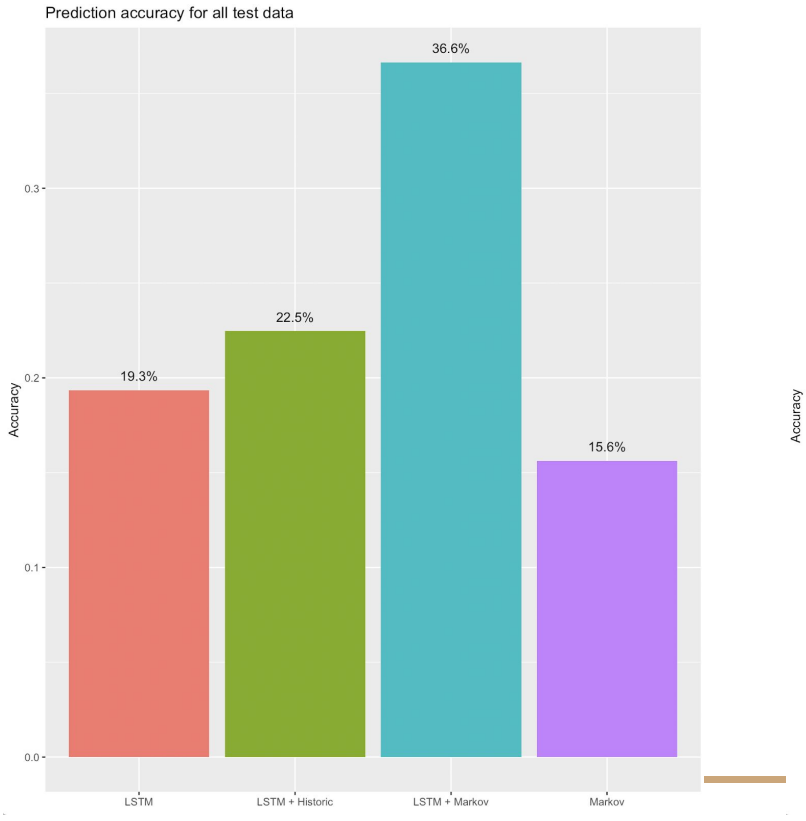
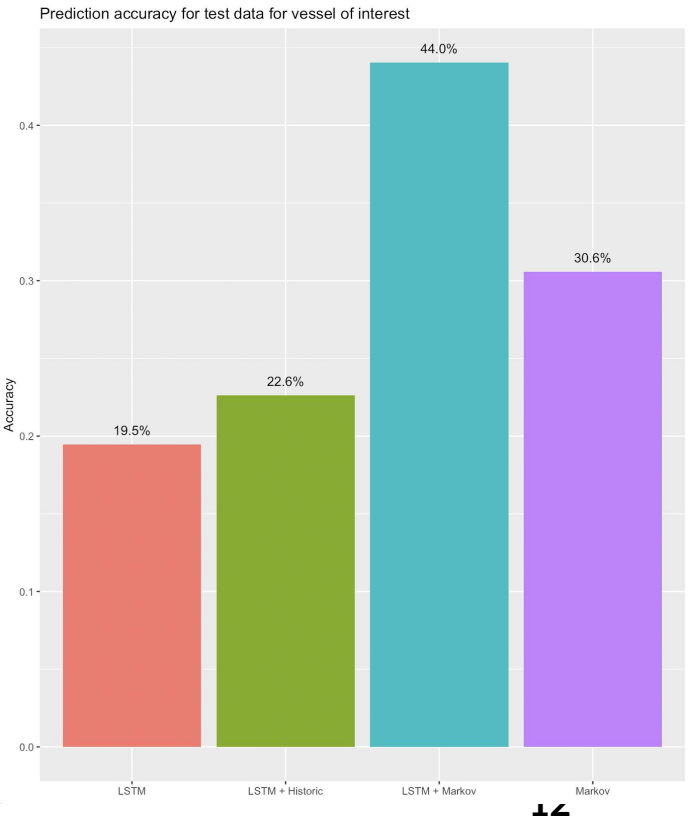


Figure : Comparaison des valeurs réelles avec les valeurs prédites sur les bateaux d'intérêts dans la base de tests

Presque 50% des dates d'entrée prédites ont une différence de moins d'une heure avec les dates réelles, soit en utilisant la médiane soit en utilisant la moyenne. En ce qui concerne les dates de sortie, nous avons environ 30% des dates prédites ayant une différence de moins d'une heure avec les dates de sorties réelles. Le calcul de la date de sortie est effectué en faisant une sommation de la date d'entrée prédite et de la durée d'escale prédite, ainsi la valeur de la date de sortie est affectée à la fois par les différences de la date d'entrée et les différences de la durée d'escale.

b. Prédiction du prochain port d'escale :





Il est clair d'après les résultats obtenus que la 4ème approche en utilisant le modèle LSTM, l'historique du bateau et les probabilités des chaînes de Markov donne de meilleures prédictions que ce soit pour la base de tests entière ou pour les bateaux d'intérêts dans la base de tests uniquement.
Les résultats présentés dans les figures précédentes sont obtenus à partir de la comparaison entre les résultats réels et les résultats prédits pour la prochaine escale uniquement.
Le résultat final du projet permet de prédire les escales qui auront lieu dans la fenêtre [j+1, j+25] à partir de l'escale actuelle en se basant sur l'historique précédent sous la forme suivante :

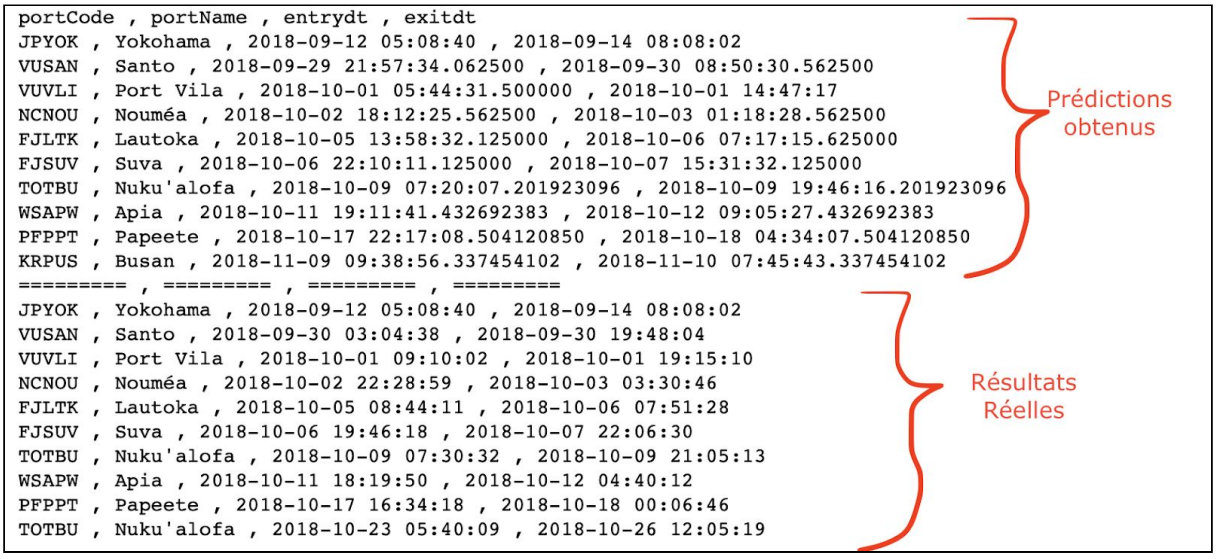


Figure : Comparaison des prédictions dans la fenêtre [j+1,j+25] aux résultats réels

E. Limites et perspectives:

1. Limites :

Plusieurs facteurs ont une grande influence sur les résultats de la prédiction obtenue :

- Le bateau n'a jamais fait d'escale dans le port du jour j :
  - **17.17%** dans la base de tests pour les navires d'intérêts.
  - **12.31%** dans toute la base de tests.
- La longueur de séquence j, j-1, j-2, …., j-n à prendre en considération.
- Le bateau n'a jamais fait un passage du port j vers un port j+1 dans son historique.
- La relation entre les probabilités du modèle LSTM et les probabilités des chaînes de Markov.
- Ports ne figurant pas dans la base d'apprentissage : **0.02%.**

Nous n'avons malheureusement pas pu étudier tous ces paramètres à cause de contraintes de temps, et nous n'avons donc pas pu définir un seuil permettant d'éviter les prédictions de type « faux positifs ».

2. Perspectives :

Par la suite pour améliorer la précision des prédictions et pour éviter les prédictions de types "faux positifs", nous envisageons de :

- Étudier les probabilités données par les modèles pour définir un threshold afin d'être sûr que les ports prédits dans la fenêtre [j+1,j+25] sont vraiment les prochains ports d'escale.
- Améliorer les paramètres ayant une influence sur la prédiction du prochain port :
    - Longueur de séquence à prendre en considération.
    - Relation entre les probabilités du modèle LSTM et les probabilités des chaînes de Markov.

F. Conclusion:

Dans ce projet, nous étions principalement concernés par le problème de prédiction pour des données chronologiques. Ce projet nous a permis de mieux comprendre ce type de données, comment adapter ces données à un modèle de réseau de neurones récurrents, la puissance des réseaux de neurones récurrents et les différentes phases de développement de ce genre de projet qui nécessite une bonne compréhension des données.
La démarche utilisée pour résoudre notre problème a tout de même quelques limites qui malheureusement demandent plus de temps et de moyens afin d'être abordées. Dans les versions futures, nous aimerions étudier les différents paramètres d'influence sur la précision de la prédiction afin de garantir des résultats corrects.

G. Bibliographie :

Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras
Crash Course in Recurrent Neural Networks for Deep Learning
How to Use the TimeseriesGenerator for Time Series Forecasting in Keras
Understanding LSTM Networks
Markov chain
Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM)
Keras Documentation For recurrent layers
Next Place Prediction using Mobility Markov Chains
LSTM Neural Networks for Language Modeling
Paper-DeepMove: Predicting Human Mobility with Attentional Recurrent Network