UNIVERSIDADE
DE TRÁS-OS-MONTES
E ALTO DOURO

Continental FOF
-
Previsão de Níveis de Enchimento em Estações de Abastecimento Industriais

---

Engenharia Informática

Laboratório de projeto em engenharia informática

Orientadores:

Prof. Vitor Filipe

Eng. José Ribeiro

Autores

José Luís Moreira Barros - al73804

Sérgio Emanuel da Silva Magalhães – al73623

Vila Real, 2023

(em branco)

Índice

1. Introdução 1
2. Enquadramento Teórico 2
3. Desenvolvimento 4
   3.1. Dataset 4
   3.2. Modelo de Deep Learning 4
   3.3. Resultados 4
4. Conclusão 4

1. Introdução

O presente projeto tem o objetivo de desenvolver um algoritmo de *deep learning* para prever os níveis de enchimento de estações de abastecimento industriais. Com o avanço da tecnologia, as técnicas de *deep learning* tem se mostrado cada vez mais capazes de resolver problemas de previsão e otimização a níveis industriais.

As estações de abastecimento têm um papel fulcral na logística de uma fábrica, fornecendo matéria-prima aos mais variados setores para as operações fabris. Por isso, é de extrema relevância que haja uma correta e eficiente gestão dos níveis de enchimento uma vez que podem ocorrer dois tipos de cenários desfavoráveis:

1) Materiais insuficientes – vão gerar atrasos na produção e consequentemente perdas financeiras;
2) Materiais em excesso – vão implicar despesas precoces para aquisição do produto, sem projeção para ser utilizado, ou seja, pode dar origem a perdas financeiras;

Com a implementação do algoritmo de *deep learning* integrado com os sistemas de abastecimento e outras áreas intervenientes no processo é esperado que se consiga antecipar possíveis esgotamentos de materiais nas estações, evitando assim paragens na produção bem como a otimização no que concerne a aquisição destes materiais.

2. Enquadramento Teórico

Neste capítulo vamos abordar um pouco dos conceitos que envolveram o desenvolvimento do algoritmo *deep learning*. Neste projeto desenvolvemos um modelo de *deep learning,* usando uma LSTM. Para o desenvolvimento usamos a plataforma *Google Colab,* e as respetivas bibliotecas já pré-instaladas, *TensorFlow* e *Keras*.

Deep Learning é uma subárea de *Machine Learning* que se baseia em redes neuronais artificiais de várias camadas que tem o propósito de aprender e extrair padrões de um certo conjunto de dados. Estas redes estão organizadas em camadas, que por sua vez são compostas por unidade de processamento interconectadas, também chamadas de neurónios.

Recurrent Neural Networks ou RNN's são um tipo especial de rede neuronal que permite lidar com dados sequenciais, como por exemplo series temporais. Diferentes de outro género de redes, este tipo de redes possui conexões recorrentes ($output_{[n-1]} \rightarrow input_{[n]}$) que lhe permite ter uma memória interna e dessa forma considerar informações passadas no processamento de informações futuras.

LSTM ou *Long Term Short Memory* é uma variante de RNN e surgiu com o objetivo de colmatar algumas deficiências das RNN tradicionais. Uma dessas deficiências é facto da sua memória interna ser de curta duração e ao longo do processamento essa informação é perdida. As LSTM por outro lado conseguem armazenar essa memória por períodos mais longos, ou seja, é capaz de detetar padrões mais complexos nos dados e de gerar precisões bastante precisas.

TensorFlow é uma biblioteca open-source desenvolvida pela Google que se tornou rapidamente uma escolha para implementação de modelos de *machine learning*. Esta biblioteca é composta por uma gama ampla de ferramentas e recursos para o desenvolvimento e implementação de modelos nesta área.

Keras é uma biblioteca de alto nível, desenvolvida em *Python*, que é usada como uma interface simplificada para *TensorFlow*. Esta biblioteca simplifica o processo de implementação de *machine learning* uma vez que abstrai detalhes complexos de baixo nível. Oferece, ainda, uma vasta variedade de funções e de optimizadores que facilita muito o processo de desenvolvimento.

Google Colab é uma plataforma *cloud-based* que oferece um ambiente de desenvolvimento de código *Python*. Apesar de ter muitos usos, normalmente é usada por desenvolvedores de código de *machine learning* e por analistas de dados devido à sua facilidade na utilização e ao seu grande poder computacional.

3. Desenvolvimento

No desenvolvimento do nosso projeto usamos a plataforma *Google Colab* para podermos executar, treinar e testar o nosso modelo. Para tal usamos as bibliotecas *TensorFlow* e *Keras* para criar o nosso modelo de *deep learning* e, em conjunto, com a linguagem *Python* que serviu para processar e criar o modo geral.

Inicialmente foram criados 2 modelos, um *univariate* e outro *multivariate*. O *univariate* é caracterizado por só considerar uma única variável, que no caso deste projeto, considera apenas o conjunto único dos dados de uma estação de abastecimento e como tal, treina, valida e por fim prevê valores apenas dessa estação. Este modelo foi eventualmente descartado por razões que iremos enumerar mais à frente.

O modelo *multivariate*, por outro lado, considera todos os dados de todas as estações de abastecimento.

Nos próximos capítulos iremos aprofundar e demonstrar mais em concreto todos os intervenientes do algoritmo e como este foi construído.

3.1 Dataset

O *dataset* fornecido pela empresa é constituído por 19 colunas:

- ➔ A primeira corresponde á marca temporal na qual foi obtida a percentagem de enchimento de cada estação;
- ➔ As 18 restantes colunas correspondem ás estações de abastecimento e contem o nível de enchimentos das mesmas, valor compreendido entre 0 e 100 (percentagem);

Time	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13
05_12_2022_09_05_31	93.624	64.262	64.138	74.275333	0.0	0.0	35.686666	0.0046666	49.976	0.0913333	0.0	0.0	0.0
05_12_2022_09_21_59	89.998666	65.0	64.523333	71.158	0.0393333	20.972	38.0	0.0	46.0	0.0046666	12.974	0.0393333	0.046666
05_12_2022_09_27_23	89.638	64.776	64.783333	70.810666	0.0	20.958	37.924	0.0	45.896666	0.0	12.934666	0.0	0.0
05_12_2022_09_32_47	89.938	64.96133	65.0	71.0	0.0	20.99933	38.0	0.0	45.920666	0.0	12.822	0.0	0.0
05_12_2022_09_38_11	89.138666	64.391333	64.263333	71.410666	0.0593333	20.94733	37.898666	0.0226666	43.673333	0.0	12.708	0.032	0.020666
05_12_2022_10_57_17	88.406666	65.692	60.766666	78.894666	0.0	39.35333	40.0	0.0	73.962	18.0	98.0	0.1013333	67.83733
05_12_2022_11_02_41	88.762666	65.868	60.998666	79.0	0.0	39.344	40.0	0.0	73.97	18.065333	98.0	0.02	67.82866
05_12_2022_11_08_05	89.0	66.0	60.552	77.398	0.1393333	39.33133	40.0	0.0	73.86	18.418	97.618666	0.524	67.226
05_12_2022_11_13_29	87.932	61.556666	60.520666	75.848	0.0	39.374666	40.0	0.0	73.866666	18.0	98.0	0.0	67.82466
05_12_2022_11_18_53	88.940666	52.964666	60.123333	76.0	0.0	39.686	40.0	0.0	73.818	18.218	97.804	0.1253333	67.74533
05_12_2022_11_24_17	88.940666	52.964666	59.993333	76.0	0.0	39.80933	40.0	0.0	73.814	18.086	97.542666	0.0453333	23.934
05_12_2022_11_29_41	87.995333	52.152	59.457333	77.108666	0.1826666	39.30133	38.746666	0.11	73.538	19.722	97.412	0.4746666	0.498666
05_12_2022_11_35_05	89.0	53.0	61.002666	79.0	0.0	39.69533	40.0	0.0	73.809333	18.0	98.0	0.0	0.0
05_12_2022_11_40_29	84.256666	51.99	60.273333	78.141333	0.0933333	39.628666	39.866666	0.0926666	72.804	23.714	95.68	0.288	0.589333
05_12_2022_11_45_53	79.946666	52.929333	60.959333	78.0	0.0	39.726	42.074666	0.514	81.656	36.854	97.0	0.0	0.0
05_12_2022_11_51_17	79.752	52.732	60.172	77.584	0.052	39.960666	49.0	0.0	91.004	39.32	97.0	0.0	0.0
05_12_2022_11_56_41	78.773333	51.975333	53.904	77.928	0.0593333	39.915333	48.622666	3.412	89.676	44.98	97.0	0.0	0.0

Figura 1 - Amostra do Dataset sem refinação

Alguns problemas foram detetados ao longo do desenvolvimento que explicamos a seguir:

- ➔ O *dataset*, tal como explicamos anteriormente, tem valores de 0 a 100 e por vezes nos cálculos de métricas gera erros;
- ➔ Devido a vários erros de captação os intervalos de tempo não são constantes e portanto, tanto o treino como a previsão têm dificuldade em ajustar-se a intervalos de tempo bem definidos;

3.2 Modelo de Deep Learning

Inicialmente desenvolvemos um modelo de *deep learning* que apenas previa uma coluna de cada vez, ou seja, *univariate*. Este modelo serviu para compreendermos melhor como funcionavam as bibliotecas bem como percebermos de que maneira podíamos melhorar a performance dos modelos alterando certos parâmetros.

De seguida introduzimos os dados relativamente inalterados no modelo e verificamos que o modelo não respondia bem. Portanto inserimos um novo passo de pré-processamento que basicamente alterava a escala dos valores para um intervalo de 0 a 1. Apesar de melhor a performance do modelo, surgiu ainda um problema que eventualmente nos levou a desconsiderar como solução.

Esse problema era o facto de na otimização do modelo, os parâmetros eram alterados tendo em conta a performance de um modelo para uma certa coluna, mas quando testávamos para outras colunas gerava muitos erros associados e portanto desconsideramos esse tipo de modelo. Num contexto empresarial não fazia muito sentido haver 18 modelos de *deep learning* para gerir cada estação de abastecimento e para além do custo de produção e manutenção desses modelos, ia ser necessário um investimento em poder computacional grande.

O modelo que desenvolvemos de seguida considera todas as colunas e consegue prevê-las todas de uma vez. Neste caso conseguimos treinar, testar e modificar os parâmetros de forma a melhorar a performance do modelo. Usamos os dois tipos de RNN's, a LSTM e a GRU, uma RNN que com memória interna porem menos complexa que a LSTM.

A previsão é realizada com base numa janela temporal, ou seja, num certo período o modelo considera as percentagens de enchimento e a partir das relações que extrai daí tenta prever a percentagem a seguir. Por exemplo, se ele considerar 5 percentagens com intervalos de 30 minutos, ele irá prever qual a percentagem dos próximos 30 minutos em relação à última percentagem considerada.

Por fim, calculamos três medidas de erro para podermos avaliar a performance de cada alteração que fomos fazendo. As três medidas que escolhemos são:

- SMAPE - Symmetric mean absolute percentage error;
- MAE - Mean Absolute Error;
- RMSE - Root Mean Square Error;

3.3 Resultados

Todos os treinos executados nos nosso modelos foram de 15 *epochs*. Testamos também diferentes tipos de RNN's que no caso foram LSTM e GRU. No caso da LSTM temos:

LSTM			
	SMAPE	MAE	RMSE
RF1	7,22675	6,88171	9,34792
RF2	5,2337	7,1144	10,3737
RF3	30,539	6,82423	9,90126
RF6	13,64	12,0026	16,3134
RF8	10,031	8,40887	12,1463
RF9	60,671	10,521	14,1217
RF10	61,689	7,77337	12,329
RF13	5,8015	9,58558	13,2496
RF14	3,413	4,95173	8,27474
RF15	3,2462	4,65407	7,69851
RF16	8,6883	5,74208	11,2027
RF17	9,2499	6,9139	12,2361
RF18	5,3839	8,37279	13,6445

Figura 2 - SMAPE, MAE e RMSE associado a cada estação

Na tabela seguinte apresentamos os resultados correspondentes a cada tabela recorrendo ao modelo GRU:

GRU			
	SMAPE	MAE	RMSE
RF1	8,97337	8,08758	10,3378
RF2	7,4184	10,6239	13,4251
RF3	31,225	7,53363	10,5472
RF6	12,591	9,95453	14,675
RF8	7,8617	5,15014	10,2078
RF9	60,124	10,8597	14,2796
RF10	62,775	8,25633	12,8923
RF13	3,9654	6,34942	11,221
RF14	2,9038	3,97746	7,47327
RF15	2,8264	3,91025	7,0308
RF16	11,247	9,48487	13,6912
RF17	9,5705	7,18969	12,2389
RF18	6,7114	10,8171	15,1645

Figura 3 - SMAPE, MAE e RMSE associados a cada estação

Por fim, apresentamos os resultados gerais de algumas métricas relativas ao uso de LSTM e GRU:

LSTM			
Métricas	SMAPE	MAE	RMSE
Média	17,2933	7,67279	11,603

GRU			
Métricas	SMAPE	MAE	RMSE
Média	17,5533	7,86113	11,7834

SMAPE - Symmetric mean absolute percentage error

MAE - Mean Absolute Error

RMSE - Root Mean Square Error

Figura 4 e 5 - Erros associados ao uso de LSTM e GRU, respetivamente

Tendo em vista dos resultados presentes, podemos concluir que a LSTM tem um erro menor que a GRU e, portanto, uma performance melhor.

Desta forma, e com base em todos os testes que executamos no modelo percebemos o seguinte:

- ➔ O modelo é bom a generalizar;
- ➔ Quando os valores de teste são constantes ou relativamente próximos, o modelo tem dificuldade em detetar o padrão e gera erros bastante elevados;

A performance do modelo é o que esperávamos, e tendo em conta os erros do modelo que apresentamos bem como os erros do *dataset*, consideramos que atingimos o objetivo.

4. Conclusão

Em suma, consideramos que atingimos o objetivo do projeto mesmo que ainda haja um erro considerável associados aos resultados da previsão do nosso modelo. Segundo a nossa avaliação os erros surgiram primeiramente no *dataset*, devido a erros de processamento dos dados e falhas em geral. Depois, a previsão do nosso modelo não é muito uniforme, uma vez que os intervalos dos dados também não são constantes é expectável que o modelo não consiga prever num certo intervalo de tempo.

As principais dificuldades que sentimos foi inicialmente compreender as RNN's e como funcionavam as previsões juntamente com o contexto temporal. Também não estávamos familiarizados com as bibliotecas de *Keras* e *TensorFlow* e inicialmente tivemos um pouco de dificuldade em habituarmo-nos à linguagem.

No entanto, e apesar de todas as dificuldades, este projeto foi um projeto muito enriquecedor e serviu-nos imenso tanto a nível de conhecimento acerca de *deep learning* bem como experiência de trabalhar num projeto mais complexo e exigente. Gostávamos ainda de deixar uma mensagem de apreço aos nossos orientadores, ao professor Vítor Filipe e ao José Ribeiro que foram incansáveis para connosco e que para além de nos guiarem estimularam-nos sempre a ir além do básico e a pesquisarmos sempre sobre matéria da área para adicionarmos novas componentes ao nosso trabalho.