Desarrollo de una Aplicación Web para la Predicción y Visualización de Acciones Utilizando Modelos LSTM

GS Desarrollo de Aplicaciones Multiplataforma

Jorge Esteban Gil

Junio 2024

1. **Introducción**
    - Contexto y justificación del proyecto
    - Objetivos del TFG

2. **Modelos de Predicción**
    - Introducción a los modelos LSTM
    - Implementación de los modelos LSTM
    - Diferencias entre el modelos 1 y 2

3. **Desarrollo de la Aplicación Web**
    - Introducción a Streamlit
    - Arquitectura de la aplicación

4. **Páginas de la Aplicación Web**
    - Página de Inicio de Sesión
    - Página de Bienvenida
    - Página de Información de Acciones
    - Página de Predicciones (Modelos LSTM)
    - Página de Comparación de Acciones
    - Página de Noticias
    - Página de Análisis de Twitter

5. **Conclusiones y Trabajos Futuros**
    - Resumen de resultados obtenidos
    - Limitaciones del proyecto
    - Posibles mejoras y desarrollos futuros

6. **Referencias**

1. Introducción

En este Trabajo de Fin de Grado (TFG), se busca alcanzar dos objetivos principales. El primero es aprender a utilizar modelos de inteligencia artificial, específicamente modelos LSTM, en casos reales donde se pueda verificar la eficacia de su entrenamiento a través de la predicción de acciones. El segundo objetivo es adquirir habilidades en la visualización y programación con tecnologías no cubiertas durante el curso, como Streamlit para la creación de aplicaciones web y Python para el desarrollo de scripts y análisis de datos.

Contexto y Justificación del Proyecto

La predicción de acciones es un área de gran interés en el campo financiero y tecnológico, ya que permite a los inversores tomar decisiones informadas basadas en análisis de datos históricos y actuales. La incorporación de modelos LSTM, una variante de redes neuronales recurrentes, permite capturar patrones temporales en series de tiempo, mejorando la precisión de las predicciones.

Objetivos del TFG

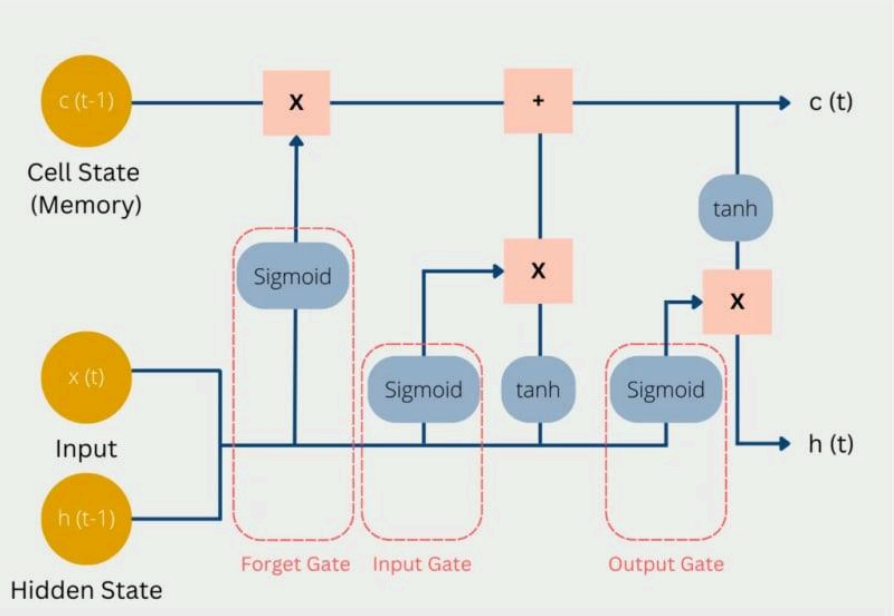
- **Aprender y aplicar modelos de inteligencia artificial:** Implementar y evaluar modelos LSTM en la predicción de acciones para comprobar su eficacia.
- **Desarrollo de habilidades en nuevas tecnologías:** Utilizar Streamlit para crear una aplicación web interactiva y programar en Python, herramientas no abordadas durante el curso.

2. Modelos de Predicción

Introducción a los Modelos LSTM

Una red neuronal de LSTM es un tipo de red neuronal recurrente (RNN) que puede aprender dependencias a largo plazo entre unidades de tiempo de datos secuenciales (MATHWORKS).

Las LSTM son capaces de aprender dependencias a largo plazo en secuencias de datos, lo que las hace particularmente adecuadas para tareas de predicción en series temporales, como la predicción de precios de acciones.



Fundamentos de las LSTM

A diferencia de las RNN estándar, las LSTM utilizan unidades de memoria especiales, conocidas como celdas de memoria, que pueden retener información durante largos períodos de tiempo. Cada celda de memoria tiene tres compuertas principales:

1. **Compuerta de Olvido (Forget Gate):** Decide qué información de la celda de memoria debe ser olvidada.
2. **Compuerta de Entrada (Input Gate):** Determina qué nueva información debe ser almacenada en la celda de memoria.

3. **Compuerta de Salida (Output Gate):** Decide qué parte de la información almacenada en la celda de memoria debe ser utilizada para la salida en el siguiente paso temporal.

Estas compuertas permiten que las LSTM controlen el flujo de información de manera efectiva, manteniendo y utilizando contextos relevantes a lo largo de largas secuencias.

Implementación de los Modelos LSTM

En este proyecto, se han implementado modelos LSTM para predecir los precios de acciones utilizando datos históricos. El proceso de implementación incluye:

1. **Preprocesamiento de Datos:** Los datos de precios de acciones se recopilan y normalizan para ser utilizados como entradas en el modelo LSTM.
2. **Construcción del Modelo:** Se define la arquitectura del modelo LSTM utilizando bibliotecas como Tensor Flow y Keras.
3. **Entrenamiento del Modelo:** El modelo se entrena con los datos históricos, ajustando los parámetros para minimizar el error de predicción.
4. **Evaluación del Modelo:** Se evalúa el rendimiento del modelo en datos de prueba para comprobar su precisión y generalización.

Integración en la Web App

Una de las mejores formas de crear un modelo de forma ordenada es a través de un Júpiter Notebook, donde podemos detallar ejecutando paso a paso y con comentarios como se crea el modelo. Pero en cambio, es muy ineficiente cargar todo este proceso en una aplicación streamlit. Por ello primero contamos con los ficheros *.ipynb* donde se crea el modelo, se guarda en un fichero *.h5* y posteriormente se carga en la web app para poder ejecutarlo y mostrar los resultados.

A la hora de implementar el modelo previamente generado con TensorFlow y Keras solo tendremos que declarar los datos de entrenamiento y test del modelo. Y ejecutando la función *model.predict(x)* donde 'x' es el set de datos (train / test) que queremos predecir.

Diferencias entre el modelos 1 y 2

Como se puede observar en el repositorio existen dos modelos creados. Una vez en la aplicación en “Slow and scatter” (Lento y disperso) tenemos la primera versión del modelo, y en “Fast and accurate tenemos” una versión mejorada.

La principal diferencia entre los dos modelos es el *dropout*. Resumiendo mucho se podría decir que es una técnica para hacer las redes neuronales más pequeñas con el tamaño justo para que aprendan lo que tienen que aprender sin a aprender demasiado. Es decir que aplicando el dropout adecuado nuestra red aprenderá lo justo para que no haya sobreentrenamiento.

En el primer caso utilizaremos un *dropout* de un 20% después de la primera capa LSTM, un 30% tras la siguiente, un 40% en la tercera y un 50% en la última. Mientras que en el segundo modelo no se utiliza ningún tipo de *dropout*. Lógicamente esto afecta el coste computacional de cada modelo. Y hace que, por lo general, a más *dropout* más tarde en llegar a una conclusión. Suele ser un sacrificio de precisión y velocidad por rendimiento.

index.py

```
model = Sequential()
model.add(LSTM(units=50, activation='relu', return_sequences=True, input_shape=((x_train.shape[1],1))))
model.add(Dropout(0.2))

model.add(LSTM(units=60, activation='relu', return_sequences=True))
model.add(Dropout(0.3))

model.add(LSTM(units=80, activation='relu', return_sequences=True))
model.add(Dropout(0.4))

model.add(LSTM(units=120, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(units=1))
```

VS

index.py

```
model = Sequential()
model.add(LSTM(units=50, activation='relu', return_sequences=True, input_shape=((x_train.shape[1],1))))
model.add(Dropout(0.2))

model.add(LSTM(units=60, activation='relu', return_sequences=True))
model.add(Dropout(0.3))

model.add(LSTM(units=80, activation='relu', return_sequences=True))
model.add(Dropout(0.4))

model.add(LSTM(units=120, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(units=1))
```

3. Desarrollo de la Aplicación Web

Introducción a Streamlit

Streamlit es una herramienta de código abierto en Python que permite crear aplicaciones web de manera rápida y sencilla. Su principal objetivo es la visualización de proyectos de ciencia de datos y machine learning programando únicamente con Python sin necesidad de usar HTML, CSS o JavaScript. Además otra de sus ventajas es la creación de componentes personalizados que se comparten de forma libre entre los usuarios. A veces son herramientas simples y otras, complejas integraciones de tecnologías como Snowflake o DataBricks. Por último, la gran ventaja de Streamlit es la posibilidad de desplegar la aplicación creada en su propia nube para que todo el mundo tenga acceso a ella, facilitando la colaboración e inspiración entre proyectos de distintos usuarios.

Streamlit

Arquitectura de la Aplicación

Streamlit tiene su propia arquitectura. Tal y como se puede observar en el repositorio existe una página principal, "*Log-In*" en este caso, y una carpeta "*pages/*" donde existen las páginas "*Hello", "Info", "Prediction", "Comparison", "News" y "Twitter".* A la hora de ejecutar la página por línea de comandos durante el desarrollo solo es necesarios ejecutar el comando *streamlit run mainPage.py* sobre la página principal y la propia página tendrá en cuenta el resto.
A la hora de desarrollar esta aplicación se han usado todas las librerías declaradas en el documento "*requirements.txt*". Es fundamental adherirse a las versiones declaradas en el documento para que la aplicación funcione correctamente.

4. Páginas de la Aplicación Web

Página de Inicio de Sesión

La página de inicio de sesión es un componente importado de otra aplicación Streamlit. Su principales funcionalidades son la de registrar usuarios e iniciar sesión. Al registrar un usuario guarda sus datos en un *.json* que encripta la contraseña como medida de seguridad. Además tiene la posibilidad de recuperar la contraseña en caso de pérdida a través de un correo automatizado gracias a la api Courier que enviará un correo con una contraseña provisional para poder cambiarla. La api key necesaria para su funcionamiento se puede guardar en un elementos de streamlit llamado *st.secrets* que permite esconder información sensible del código abierto de la aplicación escondiéndolo en el cloud de Streamlit.

Página de Bienvenida

Esta página proporciona una introducción general a la aplicación, incluye enlaces a las principales funcionalidades y un resumen de los datos disponibles.

Página de Información de Acciones

En esta página se puede consultar toda la información genérica de cada empresa (Lugar, Web, Descripción...). Existe la posibilidad de consultar las principales instituciones con capital invertido en la empresa. Muy útil para saber las posiciones de los mejores fondos de inversión a largo plazo. Después se puede consultar el equipo ejecutivo, su puesto, edad y salario. (No siempre se muestran todos los datos ya que no siempre están declarados.) Finalmente se muestra una tabla con el último balance presentado por la empresa para conocer su situación actual.

Página de Predicciones (Modelos LSTM)

Tal y como se define anteriormente los modelos LSTM cargados en el proyecto se pueden visualizar en esta página. Aquí podemos consultar los datos históricos en el rango de fechas seleccionado y su comparación respecto a las Moving Averages a 100 y 200 días. Esta medida es muy común en finanzas e inversión para medir la capacidad de predicción y estabilidad de una acción. Finalmente existe un desplegable en el que seleccionar el modelo que se muestra en la gráfica.

Página de Comparación de Acciones

En esta página se permite comparar dos empresas basándonos en distintos aspectos. A nivel backend se ha trabajado en distintos dataframes dependiendo el tiempo de métricas que se desea mostrar y cómo nos las presenta la librería *yfinance*. Luego mostramos estos dataframes de métricas en las tablas “Basic”, “Technicals”, “Statistics” y “Recommendation”. Algunas medida solo hay que recogerlas de la api y otras hay que calcularlas en base a datos temporales.

Página de Noticias

La propia api *yfinance* nos permite recoger noticias de la web Yahoo Finance. Al llamar a las noticias respectos un ticker de una acción nos devuelve un *.json* del que extraemos el titular, el autor o periódico original, una imagen, un link a la noticia completa y algunos ticker de empresas relacionadas con la noticia en los que poder pinchar para ver más noticias acerca de ellas.

Página de Análisis de Twitter

Utilizando la capacidad de streamlit de insertar partes de código HTML hacemos uso de las herramientas para desarrolladores de X importando un componente v1 de streamlit. A través de RapidApi obtenemos los link de los tweets que luego mostramos. Esta página es útil para juzgar la opinión generalizada sobre una acción.

5. Conclusiones y Trabajos Futuros

Resumen de Resultados Obtenidos

El desarrollo de esta aplicación web ha permitido explorar y aplicar modelos de inteligencia artificial en un contexto real, demostrando la viabilidad de las LSTM para la predicción y valoración de acciones. Además, se ha adquirido experiencia en el uso de Streamlit y otras tecnologías de Python.

Limitaciones del Proyecto

Algunas limitaciones encontradas incluyen la necesidad de un mayor volumen de datos para mejorar la precisión de los modelos y la dependencia de la calidad de los datos de entrada. Por otro lado desde el punto de vista del desarrollo de la aplicación las limitaciones las marcaba Streamlit ya que tiene un estilo muy determinado y poco modificable. Además algunas integraciones pueden arrojar errores una vez desplegada la aplicación en el cloud. Son errores propios de la plataforma.

Posibles Mejoras y Desarrollos Futuros

Para futuras versiones, se podrían incorporar más fuentes de datos, mejorar los modelos de predicción con técnicas avanzadas de machine learning, y ampliar las funcionalidades de la aplicación web para incluir más tipos de análisis.

6. Referencias

MATHWORKS. “Redes neuronales de memoria de corto-largo plazo - MATLAB & Simulink.” *MathWorks*, Link. Accessed 10 June 2024.

Nitish Srivastava, et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting.” *Computer Science*, Yoshua Bengio, 6 2014, Link. Accessed 10 June 2024.

Saraswat, Abhinav. “Introduction to Streamlit. Pros and cons of using the framework... | by Abhinav Saraswat.” *Medium*, 12 March 2023, Link.

GEEKS FOR GEEKS. *Build a Stock Trend Prediction Web App in Python | GeeksforGeeks*. *YouTube*, 25 Junio 2021,Link.