SlotMarketSQL

Zack Horton, Virginia Maguire, Tanner Street, and Yutong Zhang

Group A26

Problem

- Models for stock price and volatility forecasts are crucial for day traders.
- However, many traders do not have the skill set to build and interface with these models.

Solution

- Forecast daily closing prices
- Forecast daily volatilities
- Chatbot for easy info access

Data:

- Historical closing prices (Yahoo Finance)

Scope:

- 2022-02-22 to 2024-02-21
- Stocks in the S&P 500

SlotMarketSQL helps day traders make informed decisions

SlotMarketSQL

Enter your question into the following textbox:

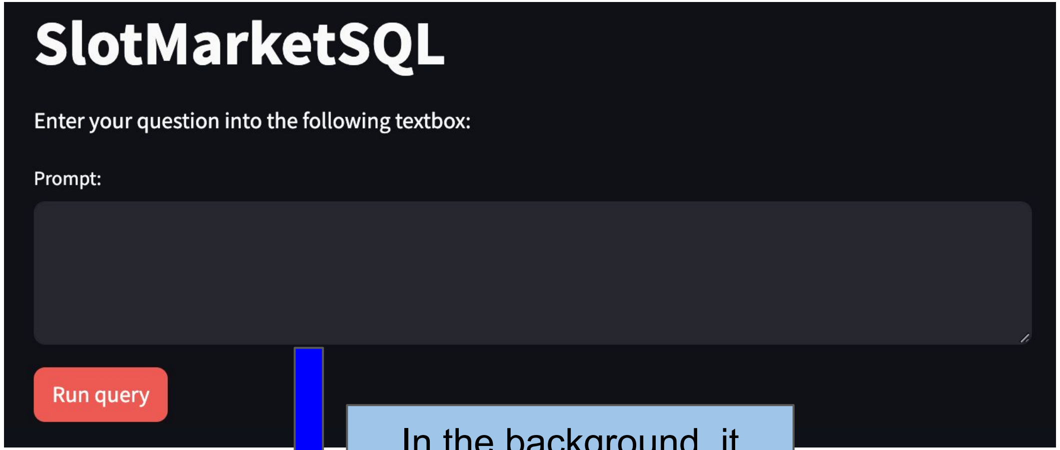
Prompt:

Run query

In the background, it references aggregated model outputs

Ticker	Stock	current_price	forecasted_price	forecasted_price_change	percent_change	current_volatility	forecasted_volatility	forecasted_volatility_change	model_error_last_week	model_mda_last_week
xyl	xylem inc.	124.099998	111.671234	-12.428764	-10.01512	0.007417	0.015811	113.160105	110.922851	42.857143
yum	yum! brands	133.949997	128.791183	-5.158813	-3.851298	0.004823	0.010888	125.77327	126.736209	57.142857
zbra	zebra technologies	271.929993	261.986816	-9.943176	-3.656521	0.011846	0.028328	139.128409	247.513789	57.142857
zbh	zimmer biomet	124.980003	120.568626	-4.411377	-3.529666	0.000231	0.014593	6228.137813	120.531616	42.857143
zts	zoetis	186.550003	181.352402	-5.197601	-2.786171	0.002227	0.017044	665.448184	182.485736	28.571429

SlotMarketSQL helps day traders make informed decisions



In the background, it references aggregated model outputs

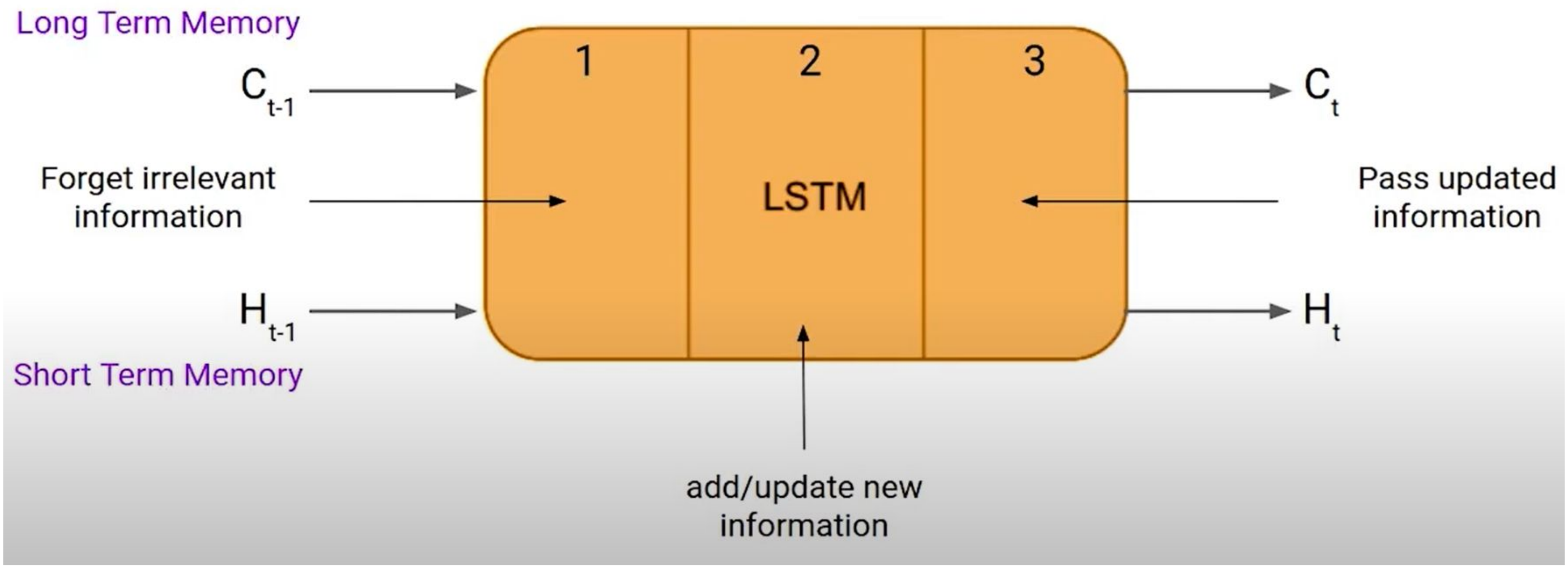
LSTM model outputs

Ticker	Stock	current_price	forecasted_price	forecasted_price_change	percent_change	current_volatility	forecasted_volatility	forecasted_volatility_change	model_error_last_week	model_mda_last_week
xyl	xylem inc.	124.099998	111.671234	-12.428764	-10.01512	0.007417	0.015811	113.160105	110.922851	42.857143
yum	yum! brands	133.949997	128.791183	-5.158813	-3.851298	0.004823	0.010888	125.77327	126.736209	57.142857
zbra	zebra technologies	271.929993	261.986816	-9.943176	-3.656521	0.011846	0.028328	139.128409	247.513789	57.142857
zbh	zimmer biomet	124.980003	120.568626	-4.411377	-3.529666	0.000231	0.014593	6228.137813	120.531616	42.857143
zts	zoetis	186.550003	181.352402	-5.197601	-2.786171	0.002227	0.017044	665.448184	182.485736	28.571429

Garch model outputs

LSTM performance metrics

LSTM Overview



Source: https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/

LSTM Architecture Experimentation

Parameters Tested:

1. Number of epochs
2. Input sequence length
3. Number of LSTM Units
4. Dropout Rate
5. Number of LSTM Stacks

Procedure:

1. Randomly select three stocks
2. Perform walk-forward validation with varying parameters
3. Parameters that achieve lowest MSE are chosen

LSTM Final Architecture

Final Parameter Results:

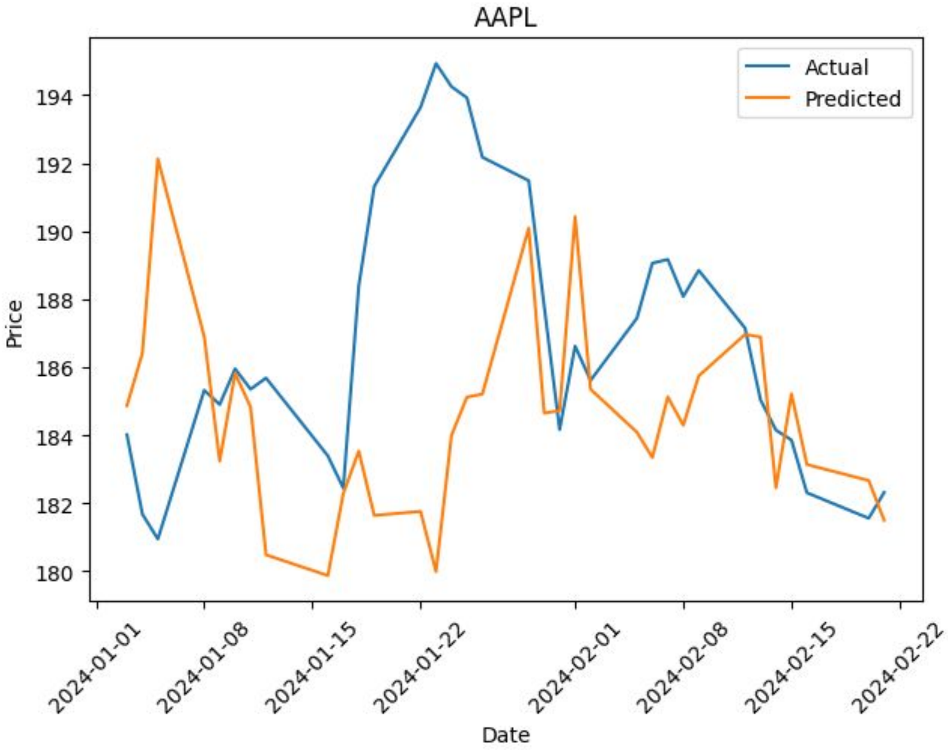
1. **Epochs** = 5
2. **Input Length** = 7
3. **LSTM Units** = 15 per stack
4. **Dropout Rate** = 15%
5. **LSTM Stacks** = 6

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 7, 1)]	0
lstm (LSTM)	(None, 7, 15)	1020
dropout (Dropout)	(None, 7, 15)	0
lstm_1 (LSTM)	(None, 7, 15)	1860
dropout_1 (Dropout)	(None, 7, 15)	0
lstm_2 (LSTM)	(None, 7, 15)	1860
dropout_2 (Dropout)	(None, 7, 15)	0
lstm_3 (LSTM)	(None, 7, 15)	1860
dropout_3 (Dropout)	(None, 7, 15)	0
lstm_4 (LSTM)	(None, 7, 15)	1860
dropout_4 (Dropout)	(None, 7, 15)	0
lstm_5 (LSTM)	(None, 15)	1860
dropout_5 (Dropout)	(None, 15)	0
dense (Dense)	(None, 1)	16

Total params: 10336 (40.38 KB)
Trainable params: 10336 (40.38 KB)
Non-trainable params: 0 (0.00 Byte)

LSTM: Predicting Closing Prices

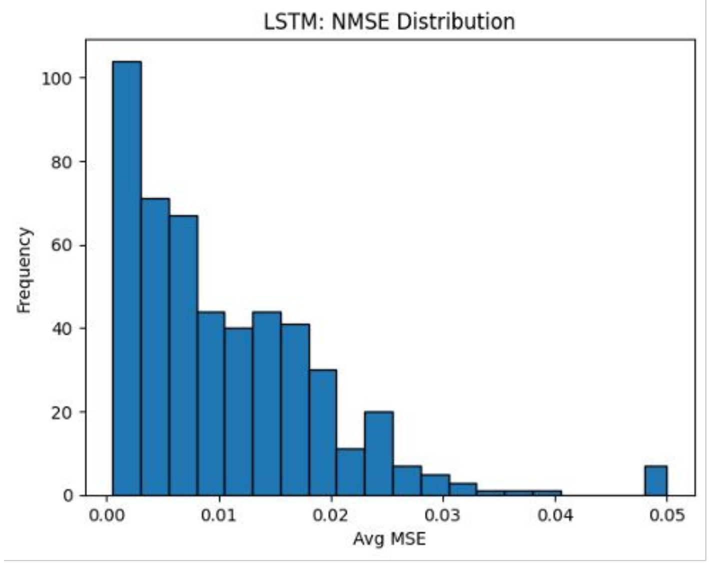
- Model trained to predict first day of 2024 (1/2/24)
- Evaluate each day in 2024:
    1. Update model with the previous day's information (1/2/24)
    2. Predict the next day's closing price (1/3/24)
- Provides more long-term view of performance

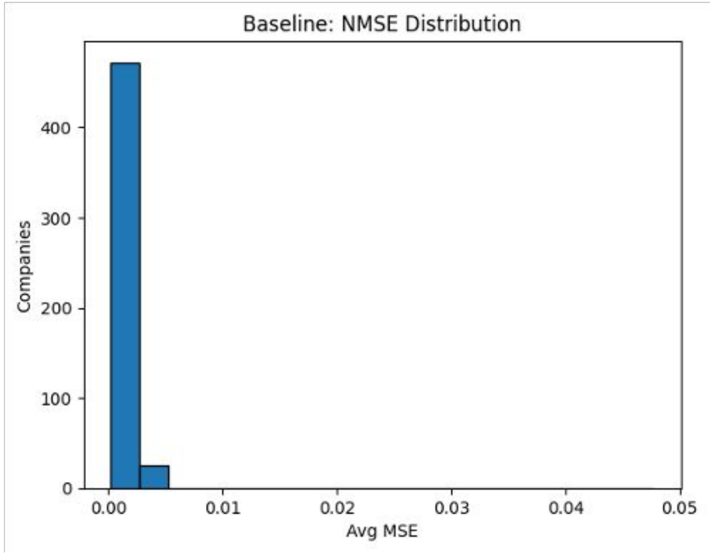


Actual vs Predicted Closing Stock Price for Apple

LSTM: Comparing the Model to the Baseline

The baseline model of using previous day's closing price generally performs better than the LSTM models.





Why?

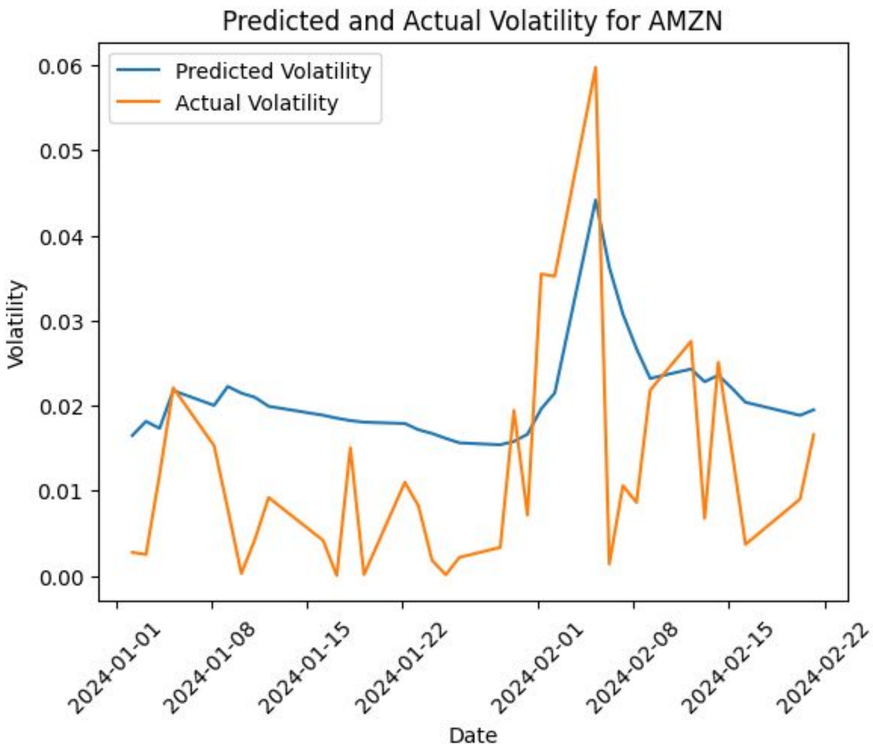
Did not use exogenous variables

Used the same architecture for all stocks

Fine-tuned parameters on only 3 stocks

GARCH: Predicting Volatility

- **Volatility** – measure of the amount of uncertainty involved in the size of changes in a stock's value
- **GARCH** – Generalized Autoregressive Conditional Heteroskedasticity
- GARCH(1,1): look at yesterday's data
- Models not fine-tuned or evaluated for scope purposes



Actual vs Predicted Closing Stock Price for Amazon

Slot Filling and Parsing

"What 10 stocks are expected to have the highest increase in price for tomorrow?"

Transformer Encoder*

"o select-stock o o o o o order-by-forecasted_price_change-desc select-forecasted_price o o o o"

SlotParser**

```
SELECT stock, forecasted_price, forecasted_price_change
FROM stock_data
ORDER BY forecasted_price_change DESC
LIMIT 10
```

* no pre-trained embeddings or transformers were used for slot filling
** SlotParser is a custom function we developed explicitly for this project

Training Dataset

- **32** starter questions
- **25** permutations per starter from ChatGPT
- **2,365** examples in total with regex help

Transformer Encoder

- **8,747,163** weights

Model Evaluation

- **94%** output token accuracy
- **93%** slot-filling accuracy

Future Work

Forecasts: LSTM & Garch

Use Exogenous Variables
Include data such as previous day high/low, market trends, and trading volume.

Optimize Architecture for All Stocks Independently
Not all stocks will have the same patterns and therefore different model structures may improve performance

Hypertune Garch Model

Automated data pulls & model updates

SlotMarketSQL Application

Incorporating Filtering Capability
Allow for the application to return filtered tables based on user conditions.

Test pre-trained embedding models (GloVe, BERT)

Consider Using LLMs
Remove the dependency to hand-label queries for training & be able to handle more complex user requests.

Questions?

Backup Slides

Backup Slides

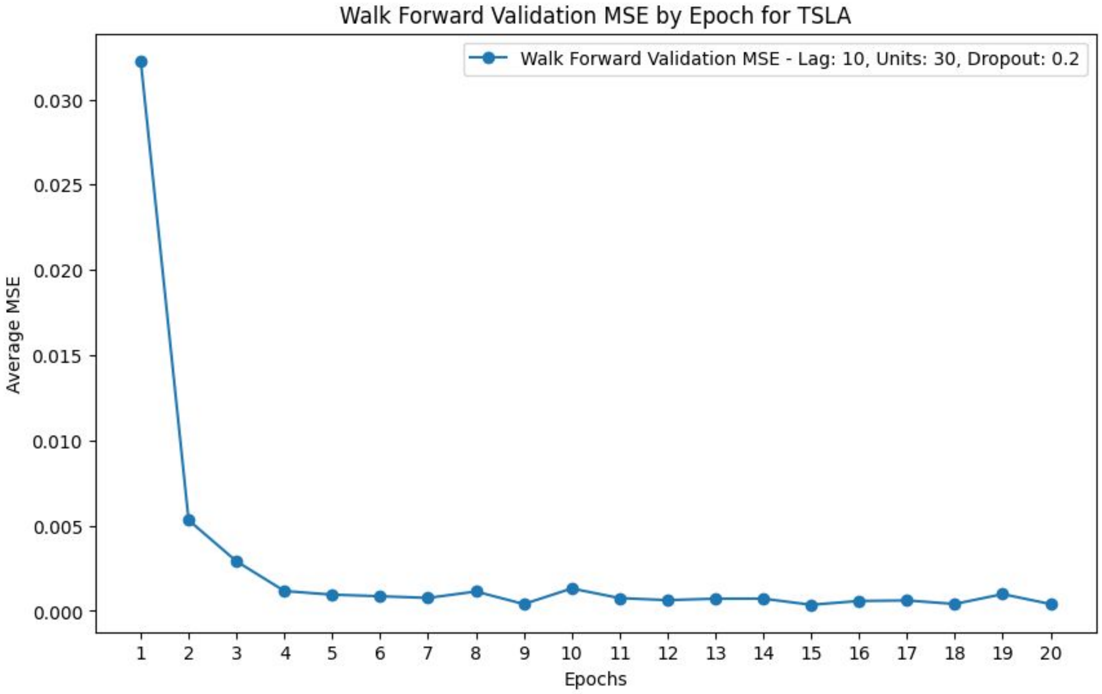


Figure 1: Average mean squared error from walk-forward validation across varying numbers of epochs for TSLA

Backup Slides

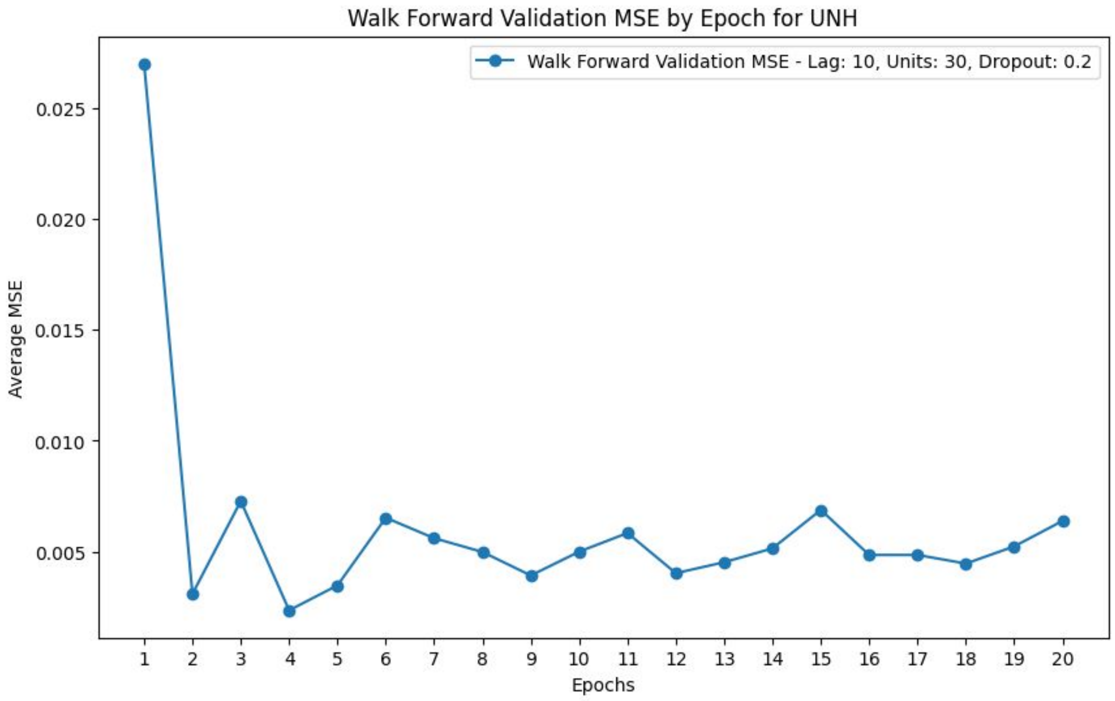


Figure 2: Average mean squared error from walk-forward validation across varying numbers of epochs for UNH

Backup Slides

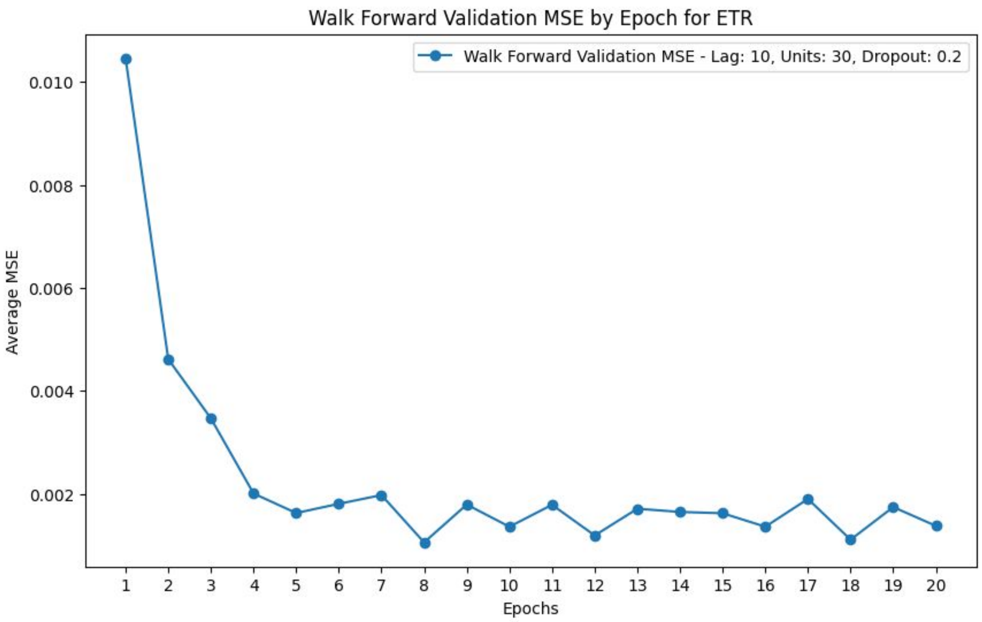


Figure 3: Average mean squared error from walk-forward validation across varying numbers of epochs for ETR

Backup Slides

Input Sequence Length	Average MSE [10^{-3}]
7	2.57
14	3.88
28	3.84

Table 1: Average mean squared error from walk-forward validation across varying input sequence lengths.

Stock	Number of LSTM Units	Dropout Rate
TSLA	90	10%
UNH	100	0%
ETR	50	15%

Table 2: Number of LSTM units and dropout rate selected from Keras hyperparameter searches for each stock.

Backup Slides

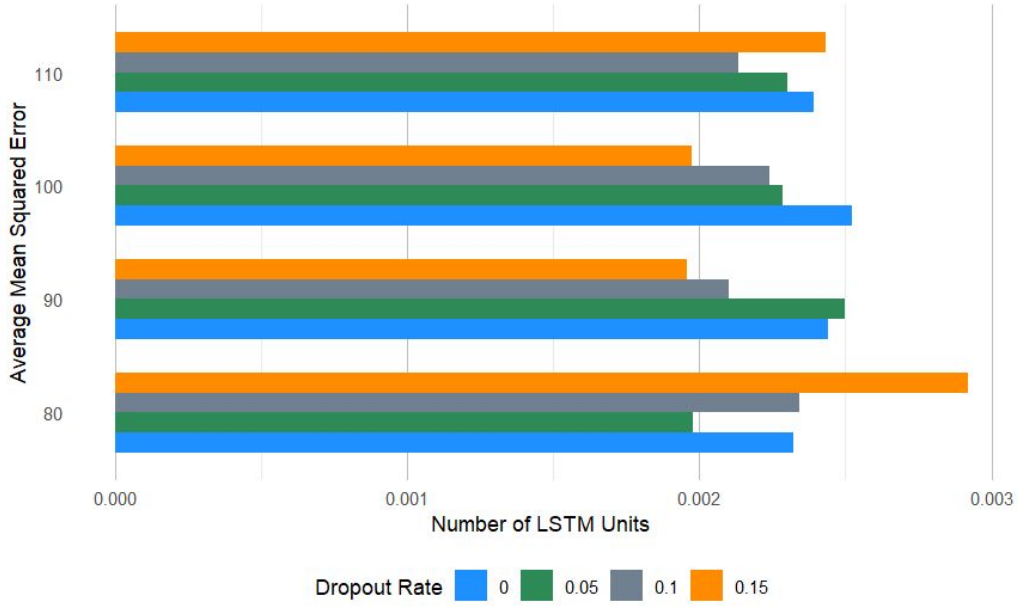


Figure 4: Average mean squared error from walk-forward validation across varying numbers of LSTM units and dropout rates.

Backup Slides

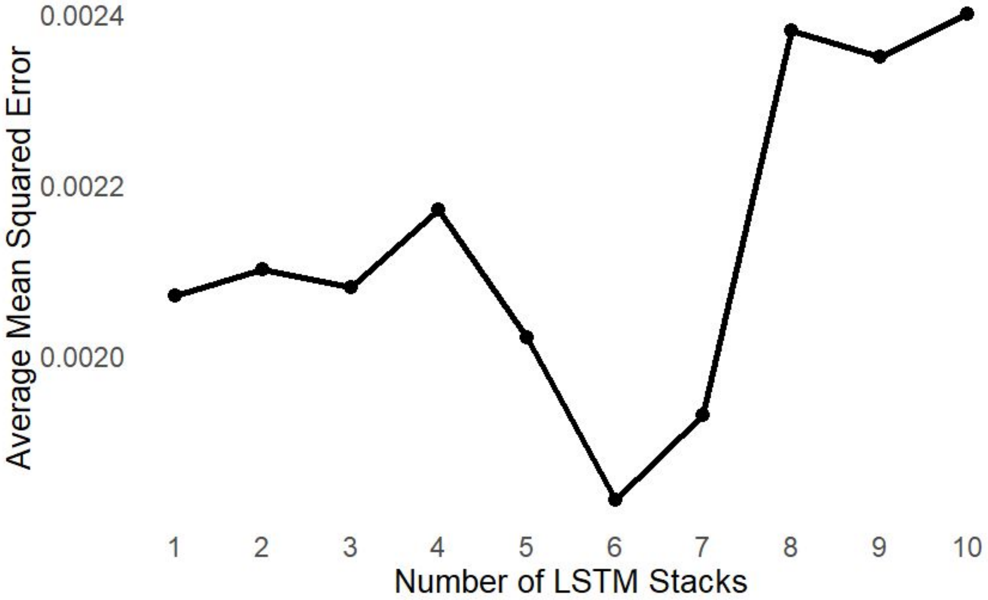


Figure 5: Average mean squared error from walk-forward validation across varying numbers of LSTM stacks.

Backup Slides

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 7, 1)]	0
lstm (LSTM)	(None, 7, 15)	1020
dropout (Dropout)	(None, 7, 15)	0
lstm_1 (LSTM)	(None, 7, 15)	1860
dropout_1 (Dropout)	(None, 7, 15)	0
lstm_2 (LSTM)	(None, 7, 15)	1860
dropout_2 (Dropout)	(None, 7, 15)	0
lstm_3 (LSTM)	(None, 7, 15)	1860
dropout_3 (Dropout)	(None, 7, 15)	0
lstm_4 (LSTM)	(None, 7, 15)	1860
dropout_4 (Dropout)	(None, 7, 15)	0
lstm_5 (LSTM)	(None, 15)	1860
dropout_5 (Dropout)	(None, 15)	0
dense (Dense)	(None, 1)	16

Total params: 10336 (40.38 KB)
Trainable params: 10336 (40.38 KB)
Non-trainable params: 0 (0.00 Byte)

Figure 6: Architecture of final models

Backup Slides

HODL-Project - NLP_to_SQL.py

```
MAX_QUERY_LENGTH = 50  #size of input
EMBED_DIM = 512  #dimension of embeddings
DENSE_DIM = 128
NUM_HEADS = 8  #number of multi-attention heads
DENSE_UNITS = 128  #num nodes in hidden layer
BATCH_SIZE = 64  #batch size for training transformer
EPOCHS = 10  #epochs for training transformer
```

Figure 7: Hyperparameters for transformer model

Backup Slides

HODL-Project - NLP_to_SQL.py

```python
# CREATE VECTORIZER (QUERY & SLOTS)
vectorize_query_text = keras.layers.TextVectorization(
    max_tokens=None,  #no maximum vocabulary
    output_sequence_length=MAX_QUERY_LENGTH,  #pad or truncate output to value
    output_mode="int",  #vector has index of vocabulary
    standardize="lower_and_strip_punctuation",  #convert input to lowercase and rmv punctuation
    split="whitespace",  #split values based on whitespace
    ngrams=1  #only look at whole words
)
vectorize_slot_text = keras.layers.TextVectorization(
    max_tokens=None,  #no maximum vocabulary
    output_sequence_length=MAX_QUERY_LENGTH,
    output_mode="int",  #vector has index of vocabulary
    standardize="lower",  #convert input to lowercase [can't do punctuation b/c of dashes]
    split="whitespace",  #split values based on whitespace
    ngrams=1  #only look at whole words
)

# CREATE VOCABULARY AND VECTORIZED TRAINING DATA
vectorize_query_text.adapt(train_query)  #build vocabulary
query_train = vectorize_query_text(train_query)  #vectorized training queries
query_test = vectorize_query_text(test_query)  #vectorized testing queries
QUERY_VOCAB_SIZE = vectorize_query_text.vocabulary_size() #total vocabulary of queries

vectorize_slot_text.adapt(train_slotfilling)  #build slot vocabulary
slots_train = vectorize_slot_text(train_slotfilling)  #vectorized training slots
slots_test = vectorize_slot_text(test_slotfilling)  #vectorized testing slots
SLOT_VOCAB_SIZE = vectorize_slot_text.vocabulary_size()  #total vocabulary of slots
```

Figure 8: Tokenization Layers

Backup Slides

HODL-Project - NLP_to_SQL.py

```python
# BUILD KERAS MODEL
inputs = keras.Input(shape=(MAX_QUERY_LENGTH,))
embedding = PositionalEmbedding(MAX_QUERY_LENGTH,
                                QUERY_VOCAB_SIZE,
                                EMBED_DIM)
x = embedding(inputs)
encoder_out = TransformerEncoder(EMBED_DIM,
                                 DENSE_DIM,
                                 NUM_HEADS)(x)
x = keras.layers.Dense(DENSE_UNITS, activation="relu", name="Dense_Layer")(encoder_out)
x = keras.layers.Dropout(0.25, name="Dropout_Layer")(x)
outputs = keras.layers.Dense(SLOT_VOCAB_SIZE, activation="softmax", name="Softmax_Layer")(x)

model = keras.Model(inputs, outputs)
print()
print(model.summary())
print()

# TRAIN KERAS MODEL
model.compile(optimizer="adam",
              loss="sparse_categorical_crossentropy",
              metrics=["sparse_categorical_accuracy"])
history = model.fit(query_train, slots_train,
                    batch_size=BATCH_SIZE,
                    epochs=EPOCHS)

# OUT-OF-SAMPLE TESTING
model.evaluate(query_test, slots_test)

# SAVE MODEL
filename = 'sql_transformer.keras'
model.save(filename)
ZipFile('model_save.zip', mode='w').write(filename)
```

Figure 9: Building keras model