

# FinBERT for Sentiment Analysis on 10-K Filings

Pranathi Alluri



# Background

## 10-K filings

- Companies legally required annual reports
- Focus on 1A, 7, & 7A - risk factors & management's discussion and analysis
- Sentiment analysis used to gauge future prospects → impacting stock prices

## FinBERT

- Built off of BERT:
  - Encoder architecture of transformer model = self-attention heads
  - Consider words that come before and after
  - Masked language Model
- Fine-tuned on financial data
- Predict sentiments of financial texts as negative, neutral, or positive



# Methodology

## Fine-tune

## FinBERT



# Dataset & Preprocessing

- 10-K SEC filings sourced from Hugging Face
- Filtered for post 2016 filings for specified sections
- Labels derived by tapping into historical stock prices and returns within a day from filing date

```
# Sepearting out the returns dict for only 1d to be able to understand immediate market chnages due to SEC fillings  
concatenated_df['1D_return']=concatenated_df['returns'].apply(lambda x: x.get('1d', None))
```

```
# get sentimental labels on a per filling bases from the market reaction for 1 day after the filling date  
positive_threshold = 0.02  
negative_threshold = -0.02
```

```
def assign_sentiment_label(row):  
    if row['1D_return']['ret'] > positive_threshold:  
        return 'positive'  
    elif row['1D_return']['ret'] < negative_threshold:  
        return 'negative'  
    else:  
        return 'neutral'
```

```
concatenated_df['sentiment']=concatenated_df.apply(assign_sentiment_label, axis=1)
```

# Dataset

	docID	section	cik	sentence	filingDate	name	tickers	exchanges	entityType	sic	stateOfIncorporation	acceptanceDateTime	reportDate	sentiment
0	0000001750_10-K_2016	1	0000001750	ITEM 1A. RISK FACTORS The following is a descr...	2016-07-13	AAR CORP	[AIR]	[NYSE]	operating	3720	DE	2016-07-13T15:13:34.000Z	2016-05-31	negative
1	0000001750_10-K_2016	8	0000001750	ITEM 7. MANAGEMENT'S DISCUSSION AND ANALYSIS O...	2016-07-13	AAR CORP	[AIR]	[NYSE]	operating	3720	DE	2016-07-13T15:13:34.000Z	2016-05-31	negative
2	0000001750_10-K_2016	9	0000001750	ITEM 7A. QUANTITATIVE AND QUALITATIVE DISCLOSU...	2016-07-13	AAR CORP	[AIR]	[NYSE]	operating	3720	DE	2016-07-13T15:13:34.000Z	2016-05-31	negative
3	0000001750_10-K_2017	1	0000001750	ITEM 1A. RISK FACTORS The following is a descr...	2017-07-12	AAR CORP	[AIR]	[NYSE]	operating	3720	DE	2017-07-12T15:26:07.000Z	2017-05-31	positive
4	0000001750_10-K_2017	8	0000001750	ITEM 7. MANAGEMENT'S DISCUSSION AND ANALYSIS O...	2017-07-12	AAR CORP	[AIR]	[NYSE]	operating	3720	DE	2017-07-12T15:26:07.000Z	2017-05-31	positive
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
221	0001699150_10-K_2019	8	0001699150	ITEM 7. MANAGEMENT'S DISCUSSION AND ANALYSIS O...	2020-02-26	Ingersoll Rand Inc.	[IR]	[NYSE]	operating	3560	DE	2020-02-26T16:04:02.000Z	2019-12-31	negative
222	0001699150_10-K_2019	9	0001699150	ITEM 7A. QUANTITATIVE AND QUALITATIVE DISCLOSU...	2020-02-26	Ingersoll Rand Inc.	[IR]	[NYSE]	operating	3560	DE	2020-02-26T16:04:02.000Z	2019-12-31	negative
223	0001699150_10-K_2020	1	0001699150	ITEM 1A. RISK FACTORS The following risk facto...	2021-02-26	Ingersoll Rand Inc.	[IR]	[NYSE]	operating	3560	DE	2021-02-26T16:03:05.000Z	2020-12-31	positive
224	0001699150_10-K_2020	8	0001699150	ITEM 7. MANAGEMENT'S DISCUSSION AND ANALYSIS O...	2021-02-26	Ingersoll Rand Inc.	[IR]	[NYSE]	operating	3560	DE	2021-02-26T16:03:05.000Z	2020-12-31	positive
225	0001699150_10-K_2020	9	0001699150	ITEM 7A. QUANTITATIVE AND QUALITATIVE DISCLOSU...	2021-02-26	Ingersoll Rand Inc.	[IR]	[NYSE]	operating	3560	DE	2021-02-26T16:03:05.000Z	2020-12-31	positive

226 rows × 14 columns

- Further processed for text to be lowercase, remove punctuations, and map sentiment to a numeric label (0: neutral, 1: positive, 2: negative)
- Split into training and testing data randomly - 80/20

### 3 finBERT Models




#### Baseline finBERT

finBERT directly applies on text data without any fine-tuning.


#### Baseline FinBERT w/ training

finBERT model is applied to text data in conjunction with training using cross entropy loss and AdamW optimization in order to better update the pre-set weights of model for our specific context.



#### Custom Sentiment Model

Builds a whole new neural network using finBERT as the initial layer, that is then concatenated with other dataset features. To create a more robust sentiment analysis model





# Custom Sentiment Model

```
class CustomSentimentModel(nn.Module):
    def __init__(self, finbert, hidden_size, num_classes = 3, dropout = 0.1):
        super(CustomSentimentModel, self).__init__()
        self.finbert = finbert
        # dropout layer
        self.dropout = nn.Dropout(dropout)

        # relu activation function
        self.relu = nn.ReLU()

        # Batch normalization layers
        self.bn1 = nn.BatchNorm1d(768 + 3)
        self.bn2 = nn.BatchNorm1d(hidden_size)

        # dense layer 1
        self.fc1 = nn.Linear(768 + 3, hidden_size)

        # dense layer 2 (Output layer)
        self.fc2 = nn.Linear(hidden_size, num_classes)

        #softmax activation function
        self.softmax = nn.LogSoftmax(dim=1)

    def forward(self, input_ids, attention_mask, features):
        pooled_output = self.finbert(input_ids=input_ids, attention_mask=attention_mask).pooler_output
        combined_features = torch.cat((pooled_output, features), dim=1)
        x = self.bn1(combined_features)
        x = self.fc1(x)
        x = self.relu(x)
        x = self.dropout(x)
        x = self.bn2(x)
        x = self.fc2(x)
        x = self.softmax(x)
        return x
```





# Sentence Encoding





```
# Tokenize and encode the training data
train_encodings = tokenizer(train_texts, truncation=True, padding=True, return_tensors='pt', max_length=512)
train_labels = torch.tensor(train_labels)

# Other features (section, state, time) need to be encoded and concatenated
train_time_features = torch.tensor([timestamp.timestamp() for timestamp in train_times]).unsqueeze(1)
label_encoder_state = LabelEncoder()
label_encoder_section = LabelEncoder()
train_state_labels = label_encoder_state.fit_transform(train_states)
train_section_labels = label_encoder_section.fit_transform(train_sections)

train_state_features = torch.tensor(train_state_labels, dtype=torch.float32).unsqueeze(1)
train_section_features = torch.tensor(train_section_labels, dtype=torch.float32).unsqueeze(1)

# Combine all features horizontally
train_additional_features = torch.cat((train_section_features, train_state_features, train_time_features), dim=1)

# Create data loaders
train_dataset = torch.utils.data.TensorDataset(train_encodings['input_ids'].clone().detach(),
                                                train_encodings['attention_mask'].clone().detach(),
                                                train_additional_features,
                                                train_labels.clone().detach())
train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
```







# Baseline FinBERT







```
# Load model with specified architecture  
model = BertForSequenceClassification.from_pretrained("yiyanghkust/finbert-tone", num_labels=3)
```

## Custom Sentiment Model

```
for num_epochs in epochs_values:  
    for batch_size in batch_size_values:  
        for hidden_size in hidden_size_values:  
            for dropout_prob in dropout_prob_values:  
                print(f"Ablation Study on training data: epochs={num_epochs}, batch_size={batch_size}, "  
                    f"hidden_size={hidden_size}, dropout_prob={dropout_prob}")  
  
            # Load model with specified architecture  
            custom_model = CustomSentimentModel(finbert, hidden_size=hidden_size, num_classes=3, dropout=dropout_prob)  
  
            # The hyperparameters to tune  
            epochs_values = [5, 10]  
            batch_size_values = [16, 32]  
            hidden_size_values = [256, 512]  
            dropout_prob_values = [0.1, 0.2]
```





```
# Define k-fold cross-validation
num_folds = 5
kf = KFold(n_splits=num_folds, shuffle=True, random_state=42)

fold_accuracies = []

for fold, (train_indices, val_indices) in enumerate(kf.split(texts)):
    print(f"Fold {fold + 1}/{num_folds}")

    train_texts = [texts[i] for i in train_indices]
    train_labels = [labels[i] for i in train_indices]
    val_texts = [texts[i] for i in val_indices]
    val_labels = [labels[i] for i in val_indices]
```



# K-fold cross validation



- Ran with 5 splits on all three model
- Average Validation accuracy across the fold was used to compare the models

# Training Models

```
# Define loss function and optimizer
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.AdamW(model.parameters(), lr=1e-3)
```

```
num_epochs = 5

# Train the model
for epoch in range(num_epochs):
    model.train() # Set the model to training model
    train_loss = 0.0
    for batch in train_loader:
        optimizer.zero_grad()
        inputs, masks, targets = batch
        inputs, masks, targets = inputs.to(device), masks.to(device), targets.to(device)

        outputs = model(input_ids=inputs, attention_mask=masks, labels=targets)
        loss = outputs.loss
        loss.backward()
        optimizer.step()
        train_loss += loss.item()
```

```
# Evaluate on validation set
model.eval() # Set the model to evaluation mode
with torch.no_grad():
    val_loss = 0.0
    val_predictions = []
    for batch in val_loader:
        inputs, masks, targets = batch
        inputs, masks, targets = inputs.to(device), masks.to(device), targets.to(device)

        outputs = model(input_ids=inputs, attention_mask=masks, labels=targets)
        loss = outputs.loss
        val_loss += loss.item()
    val_predictions.extend(torch.argmax(outputs.logits, dim=1).tolist())
```



# Results



Fold 1/5

Epoch 1/5 - Training Loss: 7.2165637546115455

Epoch 2/5 - Training Loss: 2.957081927193536

Epoch 3/5 - Training Loss: 1.4190496471193101

Epoch 4/5 - Training Loss: 1.4233010742399428

Epoch 5/5 - Training Loss: 1.4151165352927313

Fold 2/5

Epoch 1/5 - Training Loss: 1.1988754206233554

Epoch 2/5 - Training Loss: 1.206785135798984

Epoch 3/5 - Training Loss: 1.1664423280292087

Epoch 4/5 - Training Loss: 1.2569943004184299

Epoch 5/5 - Training Loss: 1.3299087948269315

Fold 3/5

Epoch 1/5 - Training Loss: 1.2098090516196356

Epoch 2/5 - Training Loss: 1.2810563378863864

Epoch 3/5 - Training Loss: 1.2024999459584553

Epoch 4/5 - Training Loss: 1.1654037104712591

Epoch 5/5 - Training Loss: 1.1441598070992365

Fold 4/5

Epoch 1/5 - Training Loss: 1.2060816884040833

Epoch 2/5 - Training Loss: 1.2904456588957045

Epoch 3/5 - Training Loss: 1.1821470790439181

Epoch 4/5 - Training Loss: 1.1365932358635797

Epoch 5/5 - Training Loss: 1.2146156364017062

Fold 5/5

Epoch 1/5 - Training Loss: 1.236109905772739

Epoch 2/5 - Training Loss: 1.2745632198121812

Epoch 3/5 - Training Loss: 1.213795820871989

Epoch 4/5 - Training Loss: 1.2761911021338568

Epoch 5/5 - Training Loss: 1.289529283841451

Avg Validation Loss: 1.1724252382914226

Avg Training Loss: 1.2580378664864433

Avg Validation Accuracy: 0.4277777777777778

=====

Fold 1/5

Fold 2/5

Fold 3/5

Fold 4/5

Fold 5/5

Avg Validation Accuracy: 0.3611111111111111

Ablation Study on training data: epochs=5, batch\_size=16, hidden\_size=256, dropout\_prob=0.2

Fold 1/5

Epoch 1/5 - Training Loss: 1.146749946806166  
Epoch 2/5 - Training Loss: 1.102505670653449  
Epoch 3/5 - Training Loss: 1.1166202757093642  
Epoch 4/5 - Training Loss: 1.1856345865461562  
Epoch 5/5 - Training Loss: 1.1726685762405396

Fold 2/5

Epoch 1/5 - Training Loss: 1.167617744869656  
Epoch 2/5 - Training Loss: 1.0950721899668376  
Epoch 3/5 - Training Loss: 1.1155158546235826  
Epoch 4/5 - Training Loss: 1.1442075901561313  
Epoch 5/5 - Training Loss: 1.1304906341764662

Fold 3/5

Epoch 1/5 - Training Loss: 1.1634618308809068  
Epoch 2/5 - Training Loss: 1.1279379394319322  
Epoch 3/5 - Training Loss: 1.1263916028870478  
Epoch 4/5 - Training Loss: 1.100187619527181  
Epoch 5/5 - Training Loss: 1.0963945455021329

Fold 4/5

Epoch 1/5 - Training Loss: 1.140145738919576  
Epoch 2/5 - Training Loss: 1.1443119313981798  
Epoch 3/5 - Training Loss: 1.1950476434495714  
Epoch 4/5 - Training Loss: 1.142042292488946  
Epoch 5/5 - Training Loss: 1.1136419508192275

Fold 5/5

Epoch 1/5 - Training Loss: 1.1260288159052532  
Epoch 2/5 - Training Loss: 1.1380067931281195  
Epoch 3/5 - Training Loss: 1.2425146102905273  
Epoch 4/5 - Training Loss: 1.181582464112176  
Epoch 5/5 - Training Loss: 1.1573274268044367

Avg Validation Loss: 1.0891265551249187

Avg Training Loss: 1.1690920220481025

Avg Validation Accuracy: 0.47777777777777775



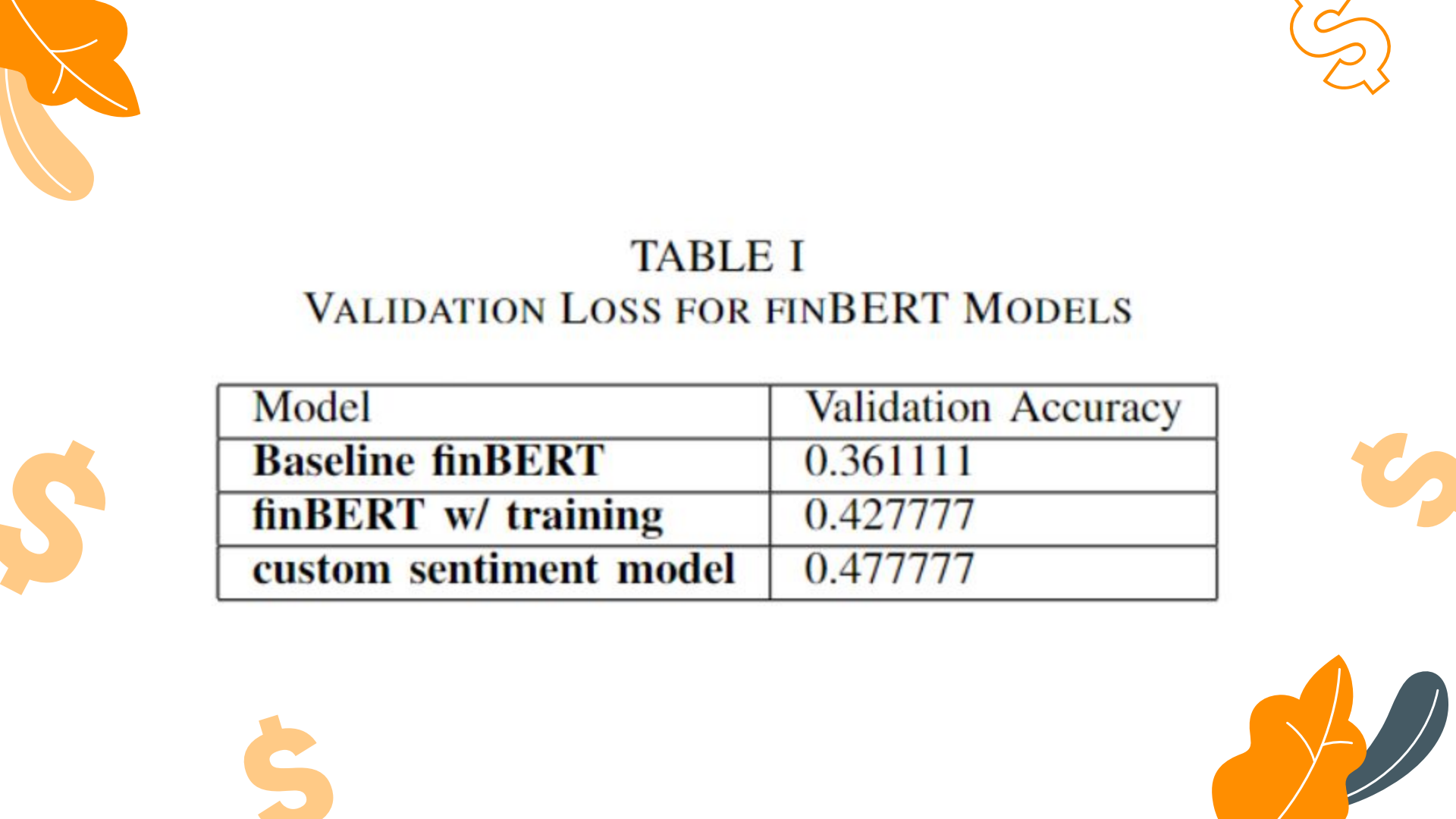


TABLE I  
VALIDATION LOSS FOR FINBERT MODELS

Model	Validation Accuracy
<b>Baseline finBERT</b>	0.361111
<b>finBERT w/ training</b>	0.427777
<b>custom sentiment model</b>	0.477777

# Testing Results

Test Accuracy: 0.5217

	precision	recall	f1-score	support
class 0	0.52	1.00	0.69	24
class 1	0.00	0.00	0.00	14
class 2	0.00	0.00	0.00	8
accuracy			0.52	46
macro avg	0.17	0.33	0.23	46
weighted avg	0.27	0.52	0.36	46

Test Accuracy: 0.3478

	precision	recall	f1-score	support
class 0	0.52	0.58	0.55	24
class 1	0.00	0.00	0.00	14
class 2	0.13	0.25	0.17	8
accuracy			0.35	46
macro avg	0.22	0.28	0.24	46
weighted avg	0.29	0.35	0.32	46



# Misclassified Word Cloud - Neg



# Misclassified Word Cloud - Pos





# Conclusions

- By enhancing the FinBERT model with tailored architectures and fine-tuning, we have achieved notable improvements in sentiment analysis performance.
- Discovered the importance of a holistic approach when analyzing the intricacies of 10-K filings
- Future work can include developing summarization models tailored to financial text data

# Thanks !

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, infographics & images by **Freepik** and illustrations by **Storyset**

