SIOP MACHINE LEARNING COMPETITION 2023

SENTIENT SENTENCE SENSE-AIs

THE TEAM

Ivan Hernandez
Virginia Tech

Andrew Cutler
Sama Therapeutics

Joe Meyer
Erudit & UCSC

Weiwen Nie
Hogan Assessments

Project Language: Python (numpy, pandas) via Jupyter Notebooks
Coding Environments: Google Colab, Virginia Tech Compute Cluster
Machine Learning Packages: Scikit-learn
Deep Learning Packages: Huggingface, Sentence-Transformers, Keras, KerasTuner
Communication/Coordination: Discord, Zoom

ONE SOLUTION TO THE PROBLEM

Multi-
Embedding
LSTM
Ensemble of
Ensembles

melee noun

me·lee ˈmā-ˌlā mā-ˈlā

variants *or less commonly* **mêlée**

Synonyms of *melee* ›

: a confused struggle

especially **:** a hand-to-hand fight among several people

CONCEPTUAL OVERVIEW OF MELEE

- Multi-Embedding LSTM Ensemble of Ensembles (MELEE)
  - Stage 1: Get Different Perspectives on the Exercise Responses
    - A single embedding model is attuned to specific aspects
    - Embed each response with a single embedding model
    - Use LSTM to figure out how embedding sequence relates to the outcomes
    - Repeat using different embedding models to pick up on some things other models do not
  - Stage 2: Weight Each Perspective Appropriately in Different Ways
    - Ensemble the predictions of the different LSTMs to predict the 7 outcomes even better
    - Vary:
      - The combination/grouping of LSTMs
      - Type of ensembling method (Ridge, Random Forest, etc.)
  - Stage 3: Aggregate the Ensembles to Optimize Specific Outcomes
    - Record Dev performance of ensembles from Step 2 for each Outcome
    - Average predictions of Best Performing Ensembles for each Outcome

CONCEPTUAL OVERVIEW OF MELEE

- Multi-Embedding LSTM Ensemble of Ensembles (MELEE)
  - Stage 1: Get Different Perspectives on the Exercise Responses
    - A single embedding model is attuned to specific aspects
    - Embed each response with a single embedding model
    - Use LSTM to figure out how embedding sequence relates to the outcomes
    - Repeat using different embedding models to pick up on some things other models do not
  - Stage 2: Weight Each Perspective Appropriately in Different Ways
    - Ensemble the predictions of the different LSTMs to predict the 7 outcomes even better
    - Vary:
      - The combination/grouping of LSTMs
      - Type of ensembling method (Ridge, Random Forest, etc.)
  - Stage 3: Aggregate the Ensembles to Optimize Specific Outcomes
    - Record Dev performance of ensembles from Step 2 for each Outcome
    - Average predictions of Best Performing Ensembles for each Outcome

CONCEPTUAL OVERVIEW OF MELEE

- Multi-Embedding LSTM Ensemble of Ensembles (MELEE)
  - Stage 1: Get Different Perspectives on the Exercise Responses
    - A single embedding model is attuned to specific aspects
    - Embed each response with a single embedding model
    - Use LSTM to figure out how embedding sequence relates to the outcomes
    - Repeat using different embedding models to pick up on some things other models do not
  - Stage 2: Weight Each Perspective Appropriately in Different Ways
    - Ensemble the predictions of the different LSTMs to predict the 7 outcomes even better
    - Vary:
      - The combination/grouping of LSTMs
      - Type of ensembling method (Ridge, Random Forest, etc.)
  - Stage 3: Aggregate the Ensembles to Optimize Specific Outcomes
    - Record Dev performance of ensembles from Step 2 for each Outcome
    - Average predictions of Best Performing Ensembles for each Outcome

CONCEPTUAL OVERVIEW OF MELEE

- Multi-Embedding LSTM Ensemble of Ensembles (MELEE)
  - Stage 1: Get Different Perspectives on the Exercise Responses
    - A single embedding model is attuned to specific aspects
    - Embed each response with a single embedding model
    - Use LSTM to figure out how embedding sequence relates to the outcomes
    - Repeat using different embedding models to pick up on some things other models do not
  - Stage 2: Weight Each Perspective Appropriately in Different Ways
    - Ensemble the predictions of the different LSTMs to predict the 7 outcomes even better
    - Vary:
      - The combination/grouping of LSTMs
      - Type of ensembling method (Ridge, Random Forest, etc.)
  - Stage 3: Aggregate the Ensembles to Optimize Specific Outcomes
    - Record Dev performance of ensembles from Step 2 for each Outcome
    - Average predictions of Best Performing Ensembles for each Outcome

WHY MELEE?

THE MELEE APPROACH OFFERS MANY ADVANTAGES

- **Good for multi-text datasets with a single outcome**
  - More memory efficient than joining texts together
  - LSTMs can figure out how short sequences combine to make final score
  - Combining embeddings gives optimal attention

- **Good for smaller sized datasets**
  - Do not need to train entire language model
  - All embeddings used a pre-trained
  - Only need to estimate LSTM regressor parameters

- **Good for multi-text problems with missing data**
  - LSTMs naturally handle missing elements in a sequence
  - Use masking to indicate missingness

- **Good for multi-outcome problems**
  - Have a choice of different ensembles
  - Can specialize in different outcome variables

WHY MELEE?

THE MELEE APPROACH OFFERS MANY ADVANTAGES

- **Good for multi-text datasets that have an overall score(s)**
  - More memory efficient than joining texts together
  - LSTMs can figure out how short sequences combine to make final score
  - Combining embeddings gives optimal attention

- **Good for smaller sized datasets**
  - Do not need to train entire language model
  - All embeddings used a pre-trained
  - Only need to estimate LSTM regressor parameters

- **Good for multi-text problems with missing data**
  - LSTMs naturally handle missing elements in a sequence
  - Use masking to indicate missingness

- **Good for multi-outcome problems**
  - Have a choice of different ensembles
  - Can specialize in different outcome variables

WHY MELEE?

THE MELEE APPROACH OFFERS MANY ADVANTAGES

- **Good for multi-text datasets with a single outcome**
  - More memory efficient than joining texts together
  - LSTMs can figure out how short sequences combine to make final score
  - Combining embeddings gives optimal attention

- **Good for smaller sized datasets**
  - Do not need to train entire language model
  - All embeddings used a pre-trained
  - Only need to estimate LSTM regressor parameters

- **Good for multi-text problems with missing data**
  - LSTMs naturally handle missing elements in a sequence
  - Use masking to indicate missingness

- **Good for multi-outcome problems**
  - Have a choice of different ensembles
  - Can specialize in different outcome variables

WHY MELEE?

THE MELEE APPROACH OFFERS MANY ADVANTAGES

- **Good for multi-text datasets with a single outcome**
  - More memory efficient than joining texts together
  - LSTMs can figure out how short sequences combine to make final score
  - Combining embeddings gives optimal attention

- **Good for smaller sized datasets**
  - Do not need to train entire language model
  - All embeddings used a pre-trained
  - Only need to estimate LSTM regressor parameters

- **Good for multi-text problems with missing data**
  - LSTMs naturally handle missing elements in a sequence
  - Use masking to indicate missingness

- **Good for multi-outcome problems**
  - Have a choice of different ensembles
  - Can specialize in different outcome variables

WHY MELEE?

THE MELEE APPROACH OFFERS MANY ADVANTAGES

- **Good for multi-text datasets with a single outcome**
  - More memory efficient than joining texts together
  - LSTMs can figure out how short sequences combine to make final score
  - Combining embeddings gives optimal attention

- **Good for smaller sized datasets**
  - Do not need to train entire language model
  - All embeddings used a pre-trained
  - Only need to estimate LSTM regressor parameters

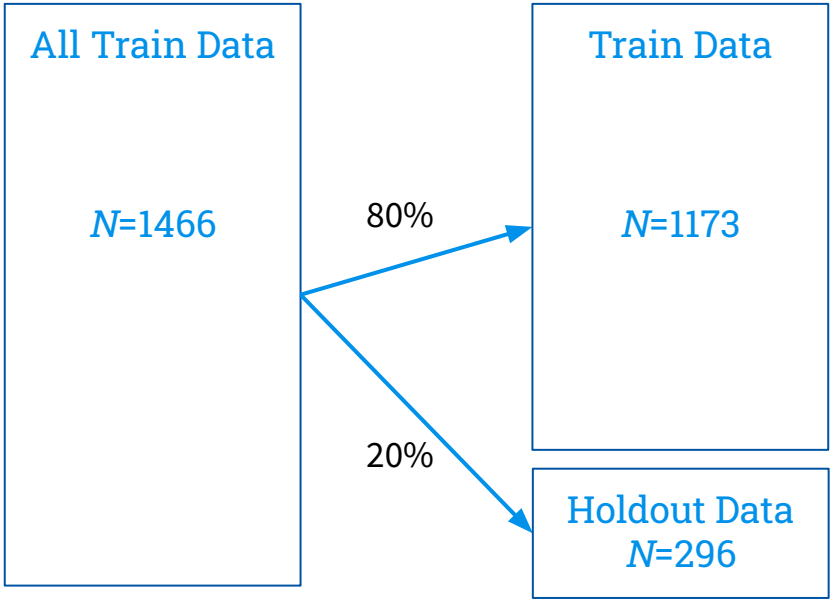
- **Good for multi-text problems with missing data**
  - LSTMs naturally handle missing elements in a sequence
  - Use masking to indicate missingness

- **Good for multi-outcome problems**
  - Have a choice of different ensembles
  - Can specialize in different outcome variables

MELEE IN ACTION

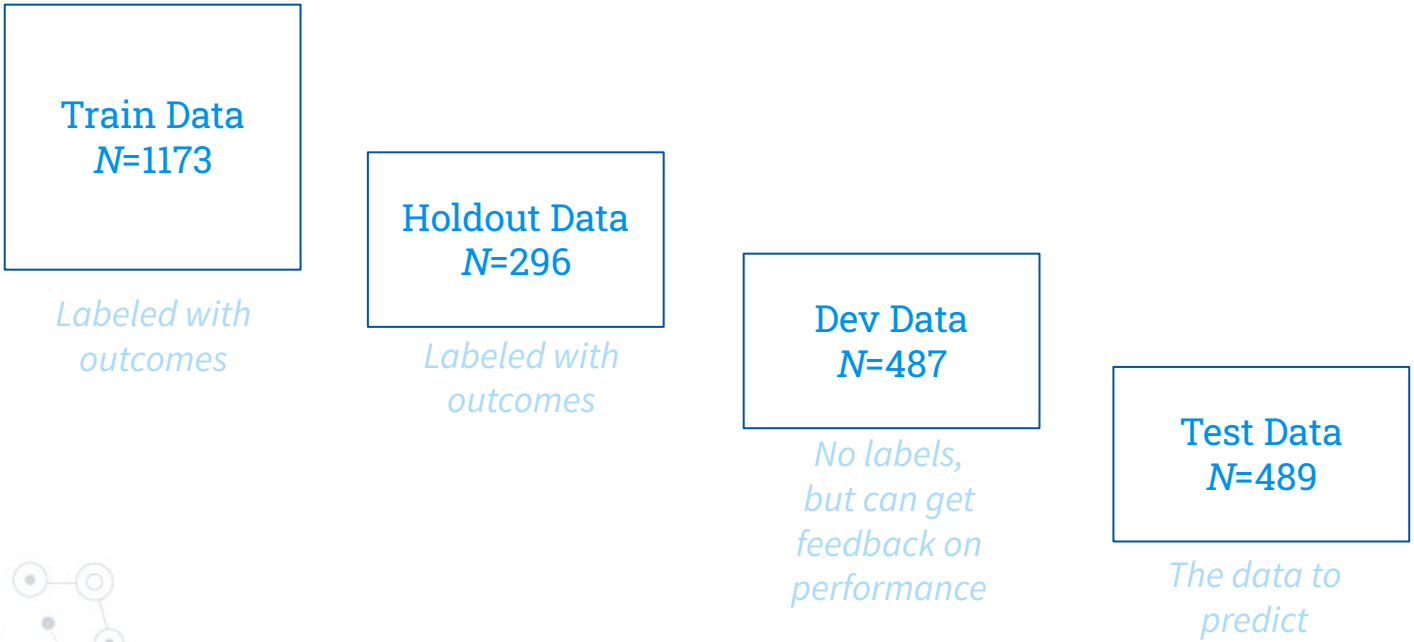
Let’s take a closer look at the process

SPLIT TRAIN DATA INTO TWO SETS



STAGE 0: SPLIT DATA

THERE SHOULD BE FOUR SETS OF DATA

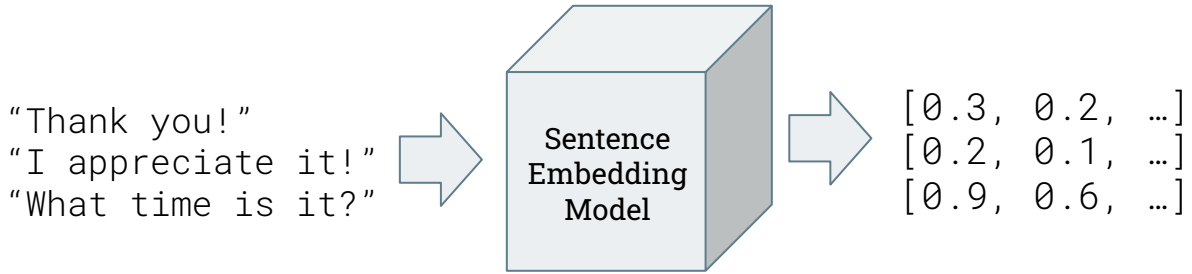


STAGE 1

Get Different Perspectives on Problem

STAGE 1: GET DIFFERENT PERSPECTIVES

- Embed each separate response using a pre-trained embedding model
  - Pick a random sentence embedding model (t5, BERT, LaBSE, roBERTa, etc.)



STAGE 1: GET DIFFERENT PERSPECTIVES

- Embed each response using a pre-trained embedding model

Chooses	Commits	...	Decision Making	Exercise 4	Exercise 5	Exercise 6	...	Exercise 17
2	3		5	Dear Tracy, can you..	Hi J.J., I was hoping that..	<MISSING>		Overall, I tried to get

Using a your selected Embedding Model …

Create Embeddings for Each Exercise Response →

.25	.34	0	-.52
-1.2	.77	0	1.7
.37	1.8	0	.44
…	…	…	…
.88	-1.9	0	-1.3

USING TRAIN DATASET

STAGE 1: GET DIFFERENT PERSPECTIVES

- Embed each response using a pre-trained embedding model

What the data look like (if using a 384 dimension embedding model):

Predictor Data (X) = 3 dimensional matrix with a shape of 1100 x 17 x 384
Each person (*N* = 1100) has a sequence of 17 embeddings that are each 384 numbers long

Outcome data (y) = 2-dimensional matrix with a shape of 1100 x 7
Each person (*N* = 1100) has 7 outcome values
Missing values imputed with iterativemodel-based imputation

USING TRAIN DATASET

STAGE 1: GET DIFFERENT PERSPECTIVES

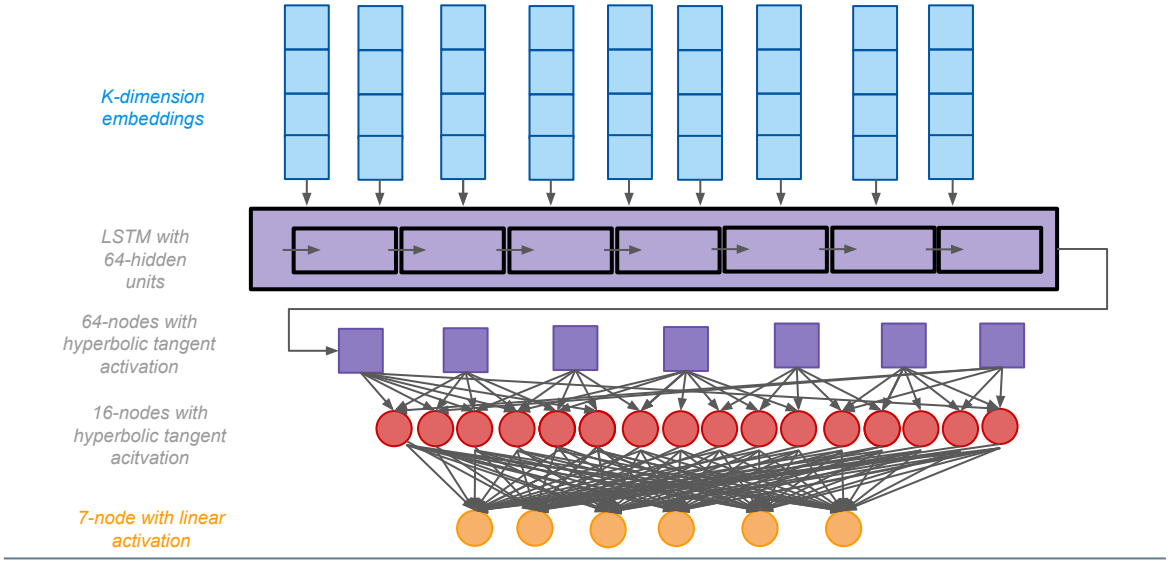
- TRAIN SIMPLE LSTM TO PREDICT OUTCOMES

Input: A person's sequence of 17 embeddings

Hidden Layers: Determine how to combine sequences to replicate rater scoring

Output: Predicted values for the person's 7 outcomes

Use trained model to make predictions on all other datasets



TRAIN USING TRAIN DATASET

STAGE 1: GET DIFFERENT PERSPECTIVES

Custom Loss Function to Make LSTM predictions less correlated and more informative across different outcomes

New Goal of Model:

Minimize the MSE

Reproduce the pattern of correlations between outcomes

	Commits	Chooses	Gathers	Identifies	Interprets	Involves	Overall
Commits	-						
Chooses	.52	-					
Gathers	.22	.25	-				
Identifies	.25	.17	.13	-			
Interprets	.39	.25	.15	.31	-		
Involves	.24	.31	.22	.16	.20	-	
Overall	.75	.54	.35	.38	.68	.36	-

Outcome correlations from train dataset

Adjust the weights in the LSTM Model to minimize the:

mean squared error difference between each predicted outcome and the actual outcome

as well as the

average absolute difference between the observed and actual inter-outcome correlation matrix

```
mse = tf.keras.losses.MeanSquaredError()
def customLoss(y_true,y_pred):
    c_true = 0
    for i,j, in itertools.combinations(range(7),2):
        c = correlation_coefficient(y_pred[:,i],y_pred[:,j])
        c_y= y_corr[i,j]
        error =  tf.math.abs(tf.subtract(c,c_y))
        c_true += error
    return mse(y_true,y_pred) + tf.math.divide(c_true,21)
```

TRAIN USING TRAIN DATASET

STAGE 1: GET DIFFERENT PERSPECTIVES

- REPEAT PROCESS USING DIFFERENT **EMBEDDING MODELS**

- all-MiniLM-L6-v2
- all-mpnet-base-v2
- all_roberta_large
- bart-large-cnn
- bart-large-instructiongen-w-inputs
- contriever-msmarco
- distilbert-base-multilingual-cased-ner-hrl
- e5-large
- electra-base-discriminator
- emotion-english-distilroberta-base
- flan-t5-base
- gtr-t5-xxl
- LaBSE
- mDeBERTa-v3-base-mnli-xnli
- msmarco-bert-co-condenser
- opt-125m-email-generation
- Paraphrase-multilingual-mpnet-base-v2
- Sentence-t5-xxl
- sgpt-27-weightedmean
- sgpt-bloom
- sn-xlm-roberta
- t5-3b-ssm
- t5-one-line-summary
- unsup-simcse-bert-base-uncased
- xlm-roberta-large

TRAIN USING TRAIN DATASET

STAGE 1: GET DIFFERENT PERSPECTIVES

- REPEAT PROCESS USING DIFFERENT **LSTM ARCHITECTURES**

  - Vary
    - Bidirectionality of LSTM
    - Number of Layer Units
    - Dropout

TRAIN USING TRAIN DATASET

Dev Leaderboard performance varies wildly (.30-.47)

No single model is perfect.

Each captures a different perspective.

STAGE 2

Weight Each Perspective Appropriately in Different Ways

STAGE 2: ENSEMBLE THE DIFFERENT PERSPECTIVES

ID	Chooses	Commits	...	Overall
AAA	1.5	3.6		6.2
BBB	2.2	2.5		3.4
CCC	3.2	1.1		4.5

You now have predicted outcomes on the holdout data from multiple different LSTMs

- BERT Shallow LSTM
- LaBSE Bidirectional LSTM
- Sentence-T5 Deep LSTM

ID	Chooses	Commits	...	Overall
AAA	1.2	3.6		6.2
BBB	2.9	2.5		3.4
CCC	4.1	1.1		4.5

ID	Chooses	Commits	...	Overall
AAA	3.5	3.6		6.2
BBB	2.3	2.5		3.4
CCC	4.2	1.1		4.5

STAGE 2: ENSEMBLE THE DIFFERENT PERSPECTIVES

Holdout Predictions from LSTM model 1

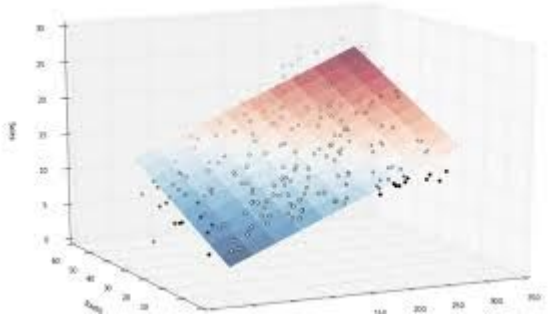
Holdout Predictions from LSTM model 20

ID	Chooses1	Commits1	...	Overall1	...	Chooses20	Commits20	...	Overall20
AAA	1.5	3.6	...	6.2	...	1.5	3.6	...	6.2
BBB	2.2	2.5	...	3.4	...	2.2	2.5	...	3.4
CCC	3.2	1.1	...	4.5	...	3.2	1.1	...	4.5

Horizontally Concatenate the Holdout Predictions into a Single Matrix
Then do the exact same for the Dev and Test Datastes

USING DIFFERENT HOLDOUT DATASETS WITH LSTM PREDICTIONS

STAGE 2: ENSEMBLE THE DIFFERENT PERSPECTIVES

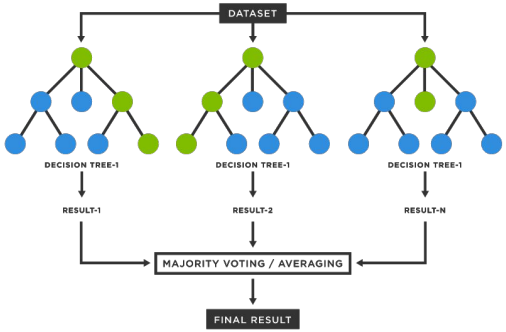


Ridge or Lasso

Train 7 different supervised learning models to predict the outcomes in the holdout dataset

X = The matrix of concatenated LSTM predictions

y = one of the outcomes (e.g., chooses, commits)



Random Forests

Use trained model to make predictions on the Dev and Test datasets using the concatenated data…you'll use these predictions in Stage 3

USING DIFFERENT HOLDOUT DATASETS WITH LSTM PREDICTIONS

STAGE 2: ENSEMBLE THE DIFFERENT PERSPECTIVES

- REPEAT PROCESS VARYING…
  - Number of SELECTED LSTMs: 15, 20, 25
  - Ensembler: Ridge, Lasso, Random Forest

- Keep track of Dev Performance of Each Ensemble

We see an improvement when combining different perspectives

Dev Leaderboard performance varies between .42-.49 (better and more consistent than individual LSTMs), but vary between specific outcomes…

Some ensembles are better for some outcomes than others

STAGE 3

COMBINE THE BEST PERFORMING ENSEMBLES

STAGE 3: COMBINE THE BEST PERFORMING ENSEMBLES

Ensemble Name	Ensembler	Embed Models	LSTM Architecture	Commits Dev Performance	Chooses Dev Performance	...	Overall Dev Performance
Ensemble 1	Random Forest	BERT, T5, XLM	Shallow	.42	.41	...	.50
Ensemble 2	Ridge	BERT, T5, XLM	Shallow	.45	.43	...	.55
Ensemble 3	Lasso	BERT, T5, XLM	Shallow	.43	.49	...	.61
Ensemble 4	Random Forest	LaBSE, T5, BLOOM	Bidirectional	.48	.44	...	.49
....	...	...	...	...	...	...	...
Ensemble 60	Lasso	FLAN, T5, BART	Deep	.46	.41	...	.55

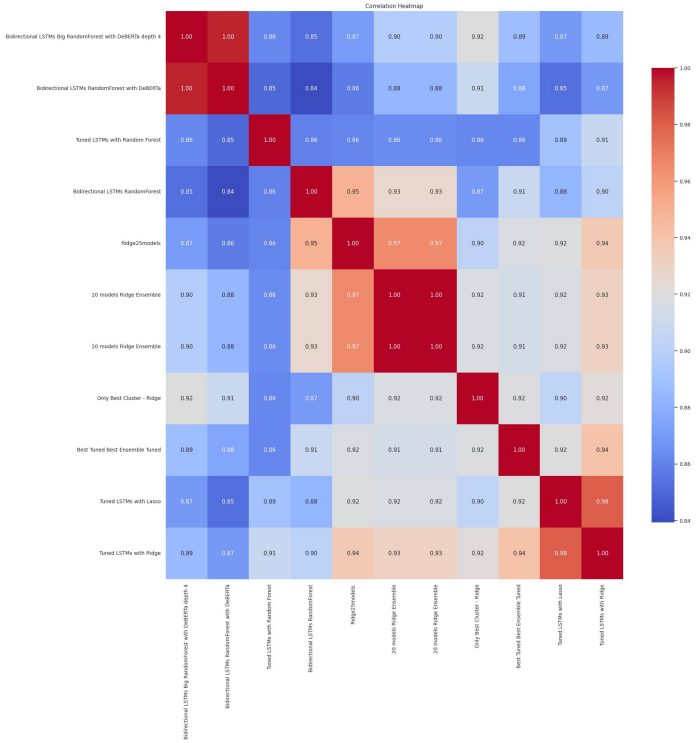
USING DEV BOARD PERFORMANCE

STAGE 3: COMBINE THE BEST PERFORMING ENSEMBLES

- Using the Dev Performance Scores
  - Pick an outcome (e.g., Commits)
  - Identify the best performing models for that outcome based on their dev score

STAGE 3: COMBINE THE BEST PERFORMING ENSEMBLES

- Examined correlation between best models' predictions
- Favored selecting ensemble models that were non-overlapping



USING DEV BOARD PERFORMANCE
USING DIFFERENT TEST DATASETS WITH ENSEMBLE PREDICTIONS

STAGE 3: COMBINE THE BEST PERFORMING ENSEMBLES

Best Chooses Models
Ensemble 1
Ensemble 5
Ensemble 22
Ensemble 28

Avg These Predictions

Best Commits Models
Ensemble 4
Ensemble 2
Ensemble 29
Ensemble 37

Avg These Predictions

Best Gathers Models
Ensemble 18
Ensemble 19
Ensemble 31
Ensemble 53

Avg These Predictions

We can incorporate other models that performed well during Dev phase too

Best Gathers Models
DeBERTa
Ensemble 29
Ensemble 46
Ensemble 33

Avg These Predictions

Best Identifies Models
Ensemble 4
Ensemble 20
Ensemble 29
Ensemble 52

Avg These Predictions

Best Involves Models
Ensemble 4
Ensemble 20
Ensemble 28
Ensemble 43

Avg These Predictions

Best Overall Models
Ensemble 17
Ensemble 23
Ensemble 33
Ensemble 47

Avg These Predictions

USING DIFFERENT TEST DATASETS WITH ENSEMBLE PREDICTIONS

We see further improvement when combining ensembles

Dev Leaderboard performance varied between .49-.52 (higher and more consistent than single ensemble models)

Test Leaderboard performance varied between .49-.52

Thanks!

Andrew Cutler

Ivan Hernandez

Joe Meyer

Weiwen Nie