

Recipes Summarization using Large Language Model

DataSet:-

I have web-scraped 14,000 recipes from allrecipes.com using the Beautiful Soup library and generated corresponding summaries using the LAMA model. The following is the structure of the dataset.

	Title	Recipe	Generated Summary	Recipe Word Count	Summary Word Count
0	Midwest Salisbury Steak	### Midwest Salisbury Steak\n\nDescription: Delicious and easy-to-make recipe.\n\nIngredients: 1 pound ground sirloin; 0.5 cup panko bread crumbs; 1 egg, beaten; 2 tablespoons milk; 0.5 (1 ounce) package dry onion soup mix; 1 teaspoon Worcestershire sauce; 0.25 teaspoon ground black pepper; 3 tablespoons butter; 2 cups fresh mushrooms, sliced; 1 sweet onion, sliced; 3 tablespoons all-purpose flour; 0.5 (1 ounce) package dry onion soup mix; 1.5 cups beef stock; 1 cup water; salt and ground black pepper to taste.\n\nInstructions: Gather the ingredients. Make the patties: Mix ground sirloin, panko bread crumbs, egg, milk, 1/2 packet onion soup mix, Worcestershire sauce, and black pepper together in a large bowl; shape into 5 patties. Heat a skillet over medium heat. Cook patties in the hot skillet until browned, 3 to 5 minutes per side. Make the gravy: Melt butter in a separate skillet over medium-high heat. Sauté mushrooms and onion in melted butter until tender, about ...	This recipe for Midwest Salisbury Steak involves mixing ground sirloin with bread crumbs, egg, and seasonings to form patties, then browning them in a skillet and serving with a rich gravy made from sautéed mushrooms, onions, and beef stock.	230	39
1	Grilled Turkey Legs	### Grilled Turkey Legs\n\nDescription: Delicious and easy-to-make recipe.\n\nIngredients: 1 (2 liter) bottle lemon-lime flavored carbonated beverage; 2 tablespoons sugar; 2 tablespoons hot sauce; 1 tablespoon crushed red pepper flakes; 1 tablespoon black pepper; 1 large sweet onion, sliced; 4 turkey legs; 2 tablespoons honey; 1 tablespoon steak seasoning.\n\nInstructions: Heat an outdoor grill for high heat and lightly oil grates. In a large pot, mix the lemon-lime flavored carbonated beverage, sugar, hot sauce, red pepper, pepper, and onion. Place the turkey legs in the mixture, and bring to a boil. Cook 30 to 45 minutes, until the turkey has reached an internal temperature of 180 degrees F (80 degrees C). Remove onion slices from the mixture, and arrange on the prepared grill. Place turkey legs over the onions. Drizzle with honey, and season with steak seasoning. Cook, turning once, 20 minutes, or until a crisp browned crust has formed on the turkey legs.	This recipe for grilled turkey legs involves marinating turkey legs in a sweet and spicy mixture of lemon-lime soda, sugar, hot sauce, and spices, then grilling them over onions until a crispy browned crust forms.	155	35
2	Chicken Stir-Fry	### Chicken Stir-Fry\n\nDescription: Delicious and easy-to-make recipe.\n\nIngredients: 1/4 cups water; 2 cups white rice; 0.6666666666666666 cup soy sauce; 0.25 cup brown sugar; 1 tablespoon cornstarch; 1 tablespoon minced fresh ginger; 1 tablespoon minced garlic; 0.25 teaspoon red pepper flakes; 3 skinless, boneless chicken breast halves, thinly sliced; 2 tablespoons vegetable oil; 1 head broccoli, broken into florets; 1 onion, cut into large chunks; 1 cup sliced carrots; 1 (8 ounce) can sliced water chestnuts, drained; 1 green bell pepper, cut into matchsticks.\n\nInstructions: Bring water and rice to a boil in a saucepan over high heat. Reduce heat to medium-low, cover, and simmer until rice is tender, and liquid has been absorbed, 20 to 25 minutes. Meanwhile, combine soy sauce, brown sugar, and cornstarch in a medium glass or ceramic bowl; stir until smooth. Stir in ginger, garlic, and red pepper flakes; add chicken and stir to coat. Cover and marinate in the ref...	This chicken stir-fry recipe combines marinated chicken breast with a mixture of soy sauce, brown sugar, and spices, then cooks it with a variety of vegetables, including broccoli, carrots, and bell peppers, and serves it over cooked white rice.	263	39
3	Hong Kong-Style Chicken Chow Mein	### Hong Kong-Style Chicken Chow Mein\n\nDescription: Delicious and easy-to-make recipe.\n\nIngredients: 1/4 ounces skinless, boneless chicken breast, thinly sliced; 1 egg white, beaten; 2 teaspoons cornstarch; 1 teaspoon sesame oil; 1 (8 ounce) package Chinese egg noodles; 2 tablespoons vegetable oil, or as needed; 0.5 cup chicken broth; 3 spring onions, chopped, or to taste; 1.5 tablespoons light soy sauce; 1 tablespoon rice wine (sake); 0.5 teaspoon ground white pepper; 0.5 teaspoon ground black pepper; 1 tablespoon cornstarch; 2 tablespoons water; 2 tablespoons oyster sauce; 1 cup fresh bean sprouts, or to taste.\n\nInstructions: Mix chicken with egg white, 2 teaspoons cornstarch, and sesame oil in a bowl; bring a large pot of water to a boil. Add egg noodles; cook until soft, about 4 minutes. Drain. Spread out on paper towels to remove excess moisture. Heat vegetable oil in a wok over medium heat. Cook and stir noodles in the hot oil until golden brown, 3 to 5 minutes.	This Hong Kong-Style Chicken Chow Mein recipe involves stir-frying chicken with vegetables and a savory sauce, served over cooked Chinese egg noodles, using a combination of techniques including marinating, boiling, and pan-frying.	250	32
4	Simple Hamburger Stroganoff	### Simple Hamburger Stroganoff\n\nDescription: Delicious and easy-to-make recipe.\n\nIngredients: 1 (16 ounce) package egg noodles; 1 pound lean ground beef; 1 (8 ounce) package cream cheese, cut into pieces; 1 (6 ounce) can chopped mushrooms, with liquid; 1 (.75 ounce) packet dry brown gravy mix; 2 (10.5 ounce) cans condensed cream of mushroom soup; 1 (8 ounce) container sour cream; 0.5 cup milk.\n\nInstructions: Fill a large pot with lightly salted water and bring to a rapid boil. Cook egg noodles at a boil until tender yet firm to the bite, 7 to 9 minutes. Drain. Meanwhile, cook ground beef in a large skillet over medium-high heat, stirring occasionally, until browned and crumbly, 5 to 7 minutes; drain and discard grease. Stir in cream cheese, mushrooms with liquid, and gravy mix; cook and stir over medium heat until cream cheese melts, 2 to 3 minutes. Add condensed soup, sour cream, and milk; cook, stirring occasionally, until smooth and creamy, 3 to 5 minutes. Dr...	This Simple Hamburger Stroganoff recipe combines cooked egg noodles with a creamy sauce made from ground beef, cream cheese, mushrooms, brown gravy mix, cream of mushroom soup, sour cream, and milk, cooked in a skillet over medium heat.	180	38

Methodology:

(a). So, initially, I cleaned the recipe data to remove information that is not important for model training, to avoid the model becoming noisy. For example, symbols like ###, instructions, etc., were considered unwanted prefixes and needed to be removed and replaced with an empty string for the complete dataset. After splitting, we got the following samples:--

training data size = 9800 samples

validationdatasize=1400samples

test data size =2800 sample

why validation data is important??

During model training I need to track validation performance for to detect model will be underfitting or overfitting ,model generalization,Hyperparamater Tuning

(b). The model and its tokenizer were loaded from Transformers, and the prepared data was fed into the model along with the labels.

(c). Now, my goal is to convert the train, validation, and test datasets into a tokenized form because the model cannot understand non-numeric data. I applied the tokenizer to each word of every sample in the train, validation, and test datasets, with a maximum sequence length of 1024 for the recipe data. For the target or labels, I applied the tokenizer with a maximum length of 128 and set truncation to True to ensure that if the input exceeds the maximum length, the model ignores those extra words.

Then, I mapped the input IDs of the labels as keys with each token in the model's input directory. This is crucial for sequence-to-sequence tasks, where the model learns to predict a sequence of token IDs corresponding to the

target IDs. The tokenizer returns a dictionary where each token corresponds to its labels for training, validation, and testing, as follows:--

Preprocessed Training Dataset:

```
Dataset({  
  features: ['input_ids', 'attention_mask', 'labels'],  
  num_rows: 9800  
})
```

Preprocessed Test Dataset:

```
Dataset({  
  features: ['input_ids', 'attention_mask', 'labels'],  
  num_rows: 1400  
})
```

Preprocessed Validation Dataset:

```
Dataset({  
  features: ['input_ids', 'attention_mask', 'labels'],  
  num_rows: 2800  
})
```

+ Code

+ Markdown

```
998, 19, 6740, 8, 909, 10702, 10116, 7772, 14983, 681, 11, 10, 786, 19650, 35243, 81,
4761, 2859, 4, 30371, 112, 73, 176, 21161, 9050, 11, 2131, 681, 4, 4287, 16639, 1954
3, 8, 37570, 15783, 131, 7142, 8, 14351, 454, 19543, 3772, 7, 356, 42026, 6, 59, 195,
728, 10116, 38053, 2859, 7, 4761, 12, 5481, 4, 23726, 19543, 12652, 7, 2380, 9, 5, 35
243, 4, 30371, 2405, 112, 73, 176, 21161, 9, 9050, 11, 5, 35243, 4, 6067, 19976, 111
5, 493, 11, 5, 1312, 9, 5, 35243, 8, 1719, 19, 19543, 12652, 4, 18652, 35243, 8, 714
2, 19976, 1115, 493, 454, 24, 2012, 7, 1004, 9030, 6, 59, 195, 728, 4, 23726, 19543,
12652, 7, 5, 2380, 456, 8, 11113, 19976, 1115, 493, 4, 18652, 8, 7142, 454, 200, 526,
16, 9030, 8, 37517, 2773, 19, 10, 20935, 6, 59, 195, 728, 55, 10116, 27336, 35243, 3
1, 2859, 4, 3107, 19976, 1115, 493, 19, 4435, 1070, 30471, 28656, 7134, 6, 1719, 6,
8, 905, 1413, 454, 7134, 16, 24136, 6, 132, 7, 155, 728, 4, 2]
```

attention_mask:

```
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1]
```

sample:

```
[0, 713, 10324, 13, 5302, 12, 15536, 12146, 20937, 1115, 493, 19, 22059, 8, 1211, 416
38, 6890, 5730, 12, 506, 15975, 10, 19976, 1115, 493, 14242, 2716, 19, 2241, 1182, 11
40, 196, 21568, 8, 15783, 6, 8319, 19, 30471, 28656, 7134, 6, 8, 1550, 19, 10, 15020,
9, 14099, 10580, 6, 634, 10, 4069, 9, 9050, 8, 14983, 681, 13, 6836, 4, 2]
```

((d) Now, my goal is to evaluate the performance of the model on the training, validation, and test data. I have defined the evaluation metric as the ROUGE score, which compares the model's predictions to the ground truth in evaluation predictions.

Initially, the predictions and labels are decoded by converting the predicted token IDs into their corresponding textual representations using the tokenizer's Batch decode method, while skipping special characters to ensure the output is clean.

Once decoded, the function addresses any masked tokens in the labels—typically marked with -100 to denote irrelevance in loss calculations—by replacing them with the tokenizer's padding token ID, thus avoiding decoding errors. Both the decoded predictions and labels are then post-processed using the NLTK library to tokenize the texts into sentences and join them with newlines.

The Compute Rouge Matrices function calculates ROUGE scores, focusing on word-level matches rather than surface forms. The ROUGE computation results are refined by extracting the mid-value of the precision-recall-F1 triplet. Additionally, the function calculates the average length of the generated predictions by counting non-padding tokens. This metric provides insights into the verbosity or conciseness of the generated text relative to the true labels.

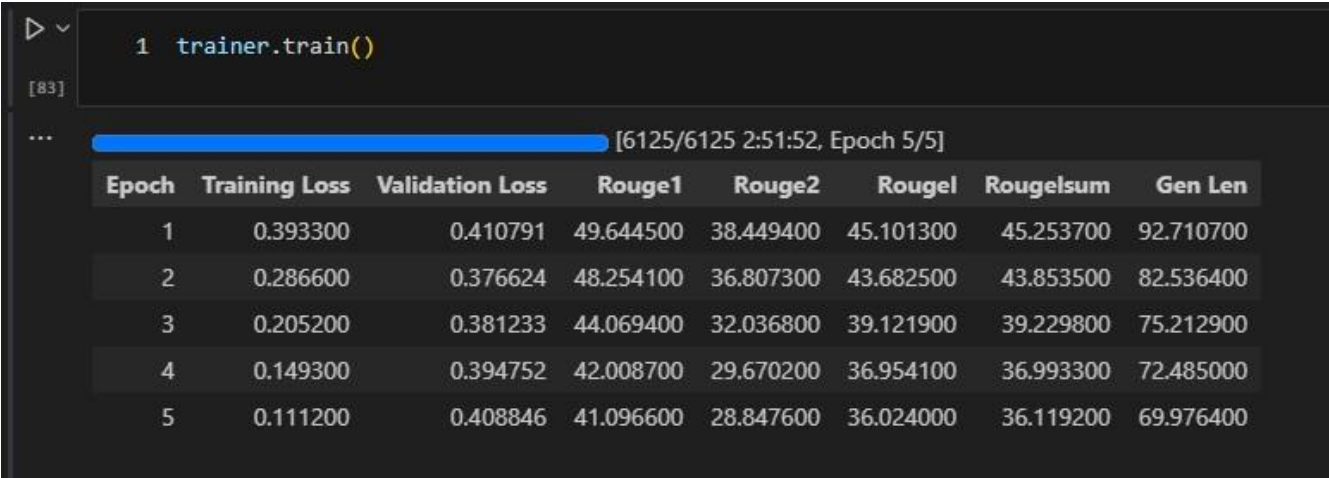
(e) Now, my goal is to build a sequence-to-sequence trainer function. This function takes the following as input: the model, training arguments (such as the number of epochs, evaluation strategy, learning rate, and batch size for training and validation), checkpoints, metrics to evaluate the best model, tokenized training and validation data, and a data collator.

The training pipeline ensures that individual training examples are correctly batched and padded, creating uniform input batches for the model. It handles tasks like padding, managing attention masks, and constructing batches, which are crucial for efficient model training and evaluation. The tokenizer and evaluation matrices are also included. Finally, the model is trained using this setup.

(f) After training the model, its performance is evaluated on the test dataset, meaning I have fine-tuned the pre-trained model on my data

MODEL PERFORMANCE BENCHMARKING:-

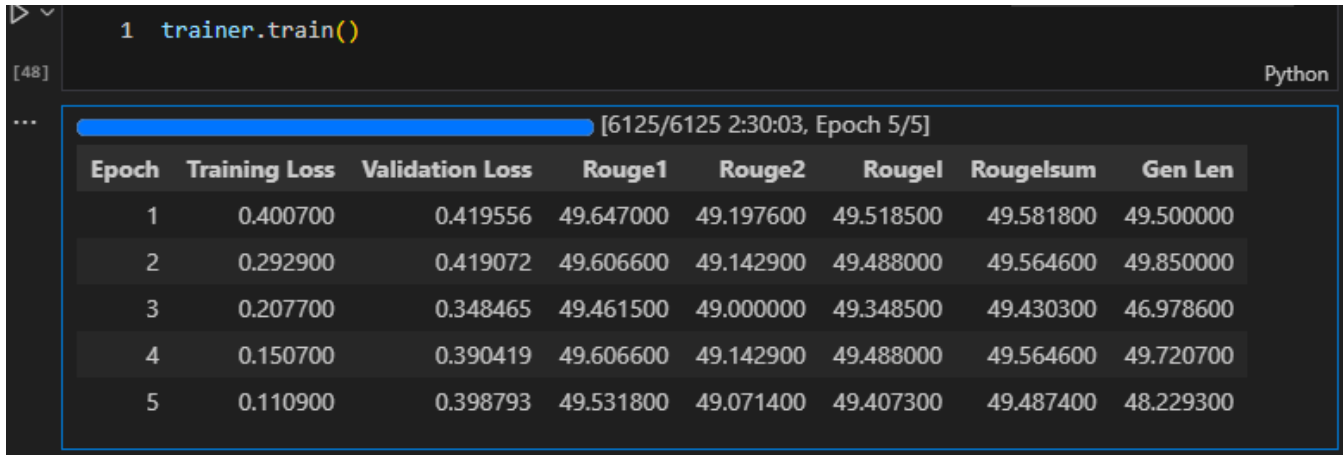
1.BART MODEL:- Base Model



Performance onTest [Data](#)

Metric	Value
eval_loss	0.3606
eval_rouge1	47.0159
eval_rouge2	36.2706
eval_rougeL	42.8237
eval_rougeLsum	42.9244
eval_gen_len	82.8196
eval_runtime	2155.5378
eval_samples_per_second	1.299
eval_steps_per_second	0.325
epoch	5.0

2.LLaMA-3BMODEL:-



Performance on TEST data:-

Metric	Value
eval_loss	0.3332
eval_rouge1	52.7019
eval_rouge2	52.1071
eval_rougeL	52.6515
eval_rougeLsum	52.5825
eval_gen_len	62.0
eval_runtime	1484.9311
eval_samples_per_second	1.886
eval_steps_per_second	0.471
epoch	5.0

3.pegasus-x-baseModel:--

[6125/6125 42:45, Epoch 5/5]

Epoch	Training Loss	Validation Loss	Rouge1	Rouge2	RougeL	RougeLsum	Gen Len
1	0.585800	0.473311	28.953100	27.683000	28.916300	28.911000	20.000000
2	0.484200	0.422108	28.953100	27.683000	28.916300	28.911000	20.000000
3	0.411500	0.398261	29.051900	27.758400	29.017000	29.011800	20.000000
4	0.371500	0.388832	28.953100	27.683000	28.916300	28.911000	20.000000
5	0.351300	0.383732	28.953100	27.683000	28.916300	28.911000	20.000000

PERFORMANCEONTTESTDATA:

Metric	Value
eval_loss	0.3647
eval_rouge1	30.6303
eval_rouge2	29.25
eval_rougeL	30.6092
eval_rougeLsum	30.5462
eval_gen_len	20.0
eval_runtime	281.5677 seconds
eval_samples_per_second	9.944
eval_steps_per_second	2.486
epoch	5.0

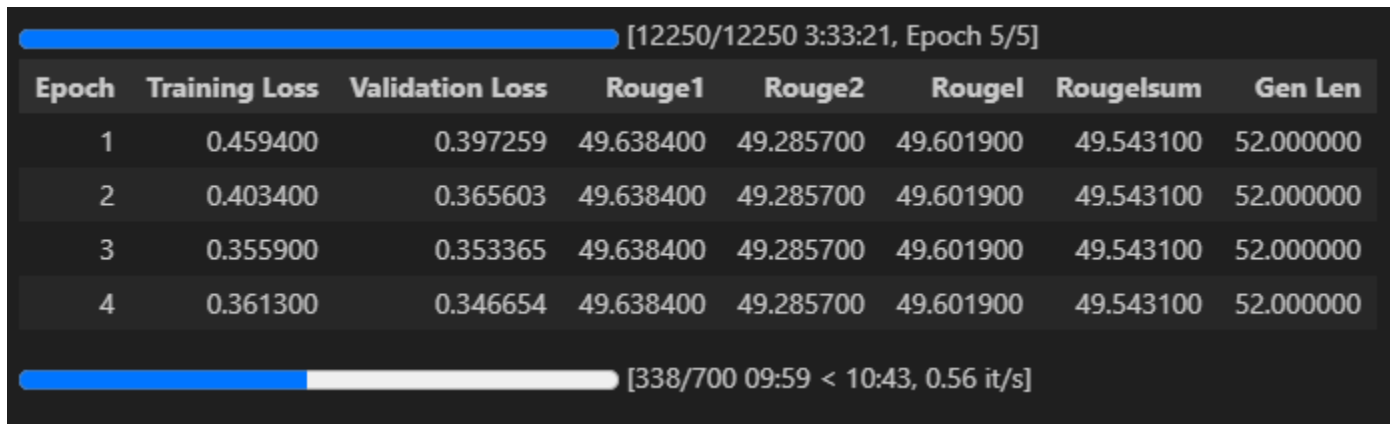
4. Facebook/Bart-large-xsum (340 million parameters

[6125/6125 42:45, Epoch 5/5]

Epoch	Training Loss	Validation Loss	Rouge1	Rouge2	RougeL	RougeLsum	Gen Len
1	0.585800	0.473311	28.953100	27.683000	28.916300	28.911000	20.000000
2	0.484200	0.422108	28.953100	27.683000	28.916300	28.911000	20.000000
3	0.411500	0.398261	29.051900	27.758400	29.017000	29.011800	20.000000
4	0.371500	0.388832	28.953100	27.683000	28.916300	28.911000	20.000000
5	0.351300	0.383732	28.953100	27.683000	28.916300	28.911000	20.000000

{'eval_loss': 0.36465972661972046, 'eval_rouge1': 30.6303, 'eval_rouge2': 29.25, 'eval_rougeL': 30.6092, 'eval_rougeLsum': 30.5462, 'eval_gen_len': 20.0, 'eval_runtime': 281.5677, 'eval_samples_per_second': 9.944, 'eval_steps_per_second': 2.486, 'epoch': 5.0}

5. Google Pegasus-Large Model



Epoch	Training Loss	Validation Loss	Rouge1	Rouge2	RougeL	Rougelsum	Gen Len
1	0.459400	0.397259	49.638400	49.285700	49.601900	49.543100	52.000000
2	0.403400	0.365603	49.638400	49.285700	49.601900	49.543100	52.000000
3	0.355900	0.353365	49.638400	49.285700	49.601900	49.543100	52.000000
4	0.361300	0.346654	49.638400	49.285700	49.601900	49.543100	52.000000

During model training, the notebook crashed due to the large size of the model and insufficient GPU memory, even when the batch size was set to 1.

CONCLUSION:--

LAMA 3B Paramater model has get best rouge 1,2,L in all among all open source LLM model because **It utilizes multiple transformer layers with self-attention mechanisms to capture contextual relationships between words in a sequence. With 3 billion parameters, it balances scalability and efficiency, offering strong performance for tasks**

TEXT SUMMARIZATION TASK GLOBAL LEADERBOARD:---

