Sequential Modelling of Typing Performance Logs

3. Preprocessing

The preprocessing steps are performed in Spark running in pseudo-distributed mode. This setup simulates a real world ETL pipeline for log like data while still running on a single node. In addition to Spark based preprocessing, we also carried out two small Hadoop MapReduce jobs on the processed data to demonstrate classic batch analytics.

We begin by launching the required cluster daemons: YARN (NodeManager, ResourceManager), HDFS (DataNode, NameNode), and Spark (Master, Worker). The running status of these processes can be checked at any time with the jps command, which lists Java processes.

For preprocessing itself, we used a Python PySpark script. The script starts by loading the two raw user CSV files into Spark DataFrames. These files contain session level typing performance statistics. We then filter the sessions to include only the 30 second typing tests. Restricting to 30 second tests ensures the sessions are directly comparable between users, mixing durations would confound analysis since speed and error rates depend on test length.

Once filtered, we harmonized data types across the columns to guarantee consistent schemas. This step is essential for later comparisons and for merging user data into a combined DataFrame. For User 1, we intentionally capped the DataFrame to 645 entries using the head() method. This was done deliberately as part of our domain choices and documented accordingly.

Next, we examined unique values across columns to identify those with constant or near constant values. Such non informative or configuration related fields were dropped from the DataFrames. These include: _id, mode, mode2, quoteLength, punctuation, numbers, language, funbox, difficulty, lazyMode, tags, blindMode, bailedOut, and isPb. Columns of this type either never vary

and thus provide no learning signal or are purely UI settings irrelevant for downstream analysis.

We then enriched the temporal information. The raw timestamp column, which records Unix epoch time in milliseconds (from January 1st 1970), was converted to a proper datetime type. From this datetime we derived a categorical time_of_day feature with values morning, afternoon, evening, or night based on the hour of day. Because Monkeytype logs are stored in UTC, we did not convert to local timezone, keeping UTC as a neutral reference. After this, the raw millisecond timestamps were removed.

The charStats column, originally stored as a semicolon delimited string (e.g., "250;3;1;2"), was parsed into four separate numeric columns: correct_characters, incorrect_characters, extra_characters, and missed_characters. This transformation makes error counts explicit and avoids repeated string parsing during modeling.

We also added a user_id column (1 for User 1, 2 for User 2) to each DataFrame. This enables combined analysis across both users while still allowing per user splits. A consistent ordered schema was then established to ensure the two DataFrames aligned correctly. Finally, we used Spark's unionByName to merge them into a combined DataFrame, ensuring that columns matched by name rather than position, preventing silent misalignments.

Regarding the downsampling of User 1: using head() to cap the DataFrame does bias the sample toward earlier rows. However, in this case the total number of sessions was not much larger than 645 (around 800 originally), and the change in WPM trend across those extra rows was not large enough to distort the analysis. This tradeoff was acceptable and explicitly noted as a design decision.

The entire preprocessing job was executed in Spark, with raw inputs and curated outputs stored on HDFS. Spark read the raw CSV files directly from HDFS and wrote the cleaned outputs back as Parquet files. Parquet was chosen for its efficiency, it is compressed, columnar, and well supported by both Spark and pandas.

In addition, we performed two small MapReduce analyses on the processed data:

- Average WPM by User ID
- Average WPM by Time of Day

Because Hadoop MapReduce jobs operate on line based text input, we converted the curated Parquet datasets to CSV and stored them on HDFS. Each job consisted of a Mapper, a Reducer (which also functioned as a Combiner in this case), and a Driver class. The jobs were compiled into a single JAR archive (monkeytype-mr.jar). We then ran this JAR on YARN, each job producing aggregated summaries that validated our Spark preprocessing and provided quick analytical insights.

4. ETL and MR Execution

a. Viewing active daemons with jps (Java Processes)

```
nogi@LAPTOP-8JQG56KU:/mnt/d/Projects/Deep Learning/Projects/Monkeytype-Trend-Detection$ jps
1521 ResourceManager
4066 ExecutorLauncher
4195 YarnCoarseGrainedExecutorBackend
856 NameNode
1000 DataNode
4506 Jps
2203 Worker
1821 Master
4141 YarnCoarseGrainedExecutorBackend
1262 SecondaryNameNode
3855 SparkSubmit
1663 NodeManager
nogi@LAPTOP-8JQG56KU:/mnt/d/Projects/Deep Learning/Projects/Monkeytype-Trend-Detection$
```

b. Running the PySpark preprocessing job with spark-submit

```
nogi@LAPTOP-8JQG56KU:~$ spark-submit --master yarn --deploy-mode client \
> ~/Preprocessing.py \
> --user1_csv hdfs:///user/$USER/monkeytype/raw/user1.csv \
> --user2_csv hdfs:///user/$USER/monkeytype/raw/user2.csv \
> --output_base hdfs:///user/$USER/monkeytype/curated \
> --head_n 645 --single_file
```

```
Combined count: 1290
Wrote:
 - hdfs:///user/nogi/monkeytype/curated/filtered_user1.parquet
 - hdfs:///user/nogi/monkeytype/curated/filtered_user2.parquet
 - hdfs:///user/nogi/monkeytype/curated/combined_df.parquet
```

c. Processed data written to HDFS (filtered_user1, filtered_user2, combined_df)

```
>>> df = spark.read.parquet("hdfs:///user/nogi/monkeytype/curated/filtered_user1.parquet")
>>> df.printSchema()
root
 |-- user_id: integer (nullable = true)
 |-- datetime: timestamp (nullable = true)
 |-- time_of_day: string (nullable = true)
 |-- wpm: double (nullable = true)
 |-- rawWpm: double (nullable = true)
 |-- acc: double (nullable = true)
 |-- consistency: double (nullable = true)
 |-- restartCount: integer (nullable = true)
 |-- testDuration: double (nullable = true)
 |-- afkDuration: integer (nullable = true)
 |-- incompleteTestSeconds: double (nullable = true)
 |-- correct_characters: integer (nullable = true)
 |-- incorrect_characters: integer (nullable = true)
 |-- extra_characters: integer (nullable = true)
 |-- missed_characters: integer (nullable = true)

>>> df.show(5, truncate=False)
+-------+-------------------+-----------+-----+------+-----+-----------+------------+------------+-----------+---------------------+------------------+--------------------+----------------+-----------------+
|user_id|datetime           |time_of_day|wpm  |rawWpm|acc  |consistency|restartCount|testDuration|afkDuration|incompleteTestSeconds|correct_characters|incorrect_characters|extra_characters|missed_characters|
+-------+-------------------+-----------+-----+------+-----+-----------+------------+------------+-----------+---------------------+------------------+--------------------+----------------+-----------------+
|1      |2025-08-14 06:22:23|morning    |80.8 |83.2  |97.16|75.24      |0           |30.0        |0          |0.0                  |202               |1                   |0               |1                |
|1      |2025-08-14 05:58:05|morning    |89.99|91.59 |98.26|78.88      |0           |30.0        |0          |0.0                  |225               |2                   |0               |0                |
|1      |2025-08-14 05:56:35|morning    |75.6 |82.79 |96.63|75.6       |1           |30.0        |0          |4.45                 |189               |3                   |0               |1                |
|1      |2025-08-14 05:54:46|morning    |89.99|89.99 |98.68|75.83      |1           |30.0        |0          |7.3                  |225               |0                   |0               |0                |
|1      |2025-08-13 17:40:48|evening    |74.0 |76.4  |96.45|74.92      |1           |30.0        |0          |17.58                |185               |1                   |0               |0                |
+-------+-------------------+-----------+-----+------+-----+-----------+------------+------------+-----------+---------------------+------------------+--------------------+----------------+-----------------+
only showing top 5 rows

>>> df2 = spark.read.parquet("hdfs:///user/nogi/monkeytype/curated/filtered_user2.parquet")
>>> df2.printSchema()
root
 |-- user_id: integer (nullable = true)
 |-- datetime: timestamp (nullable = true)
 |-- time_of_day: string (nullable = true)
 |-- wpm: double (nullable = true)
 |-- rawWpm: double (nullable = true)
 |-- acc: double (nullable = true)
 |-- consistency: double (nullable = true)
 |-- restartCount: integer (nullable = true)
 |-- testDuration: double (nullable = true)
 |-- afkDuration: integer (nullable = true)
 |-- incompleteTestSeconds: double (nullable = true)
 |-- correct_characters: integer (nullable = true)
 |-- incorrect_characters: integer (nullable = true)
 |-- extra_characters: integer (nullable = true)
 |-- missed_characters: integer (nullable = true)

>>> df2.show(5, truncate=False)
+-------+-------------------+-----------+------+------+-----+-----------+------------+------------+-----------+---------------------+------------------+--------------------+----------------+-----------------+
|user_id|datetime           |time_of_day|wpm   |rawWpm|acc  |consistency|restartCount|testDuration|afkDuration|incompleteTestSeconds|correct_characters|incorrect_characters|extra_characters|missed_characters|
+-------+-------------------+-----------+------+------+-----+-----------+------------+------------+-----------+---------------------+------------------+--------------------+----------------+-----------------+
|2      |2024-08-21 15:23:59|afternoon  |113.18|115.98|96.96|71.37      |0           |30.0        |0          |0.0                  |283               |2                   |1               |0                |
|2      |2024-05-08 18:17:32|evening    |64.4  |64.4  |94.25|62.4       |0           |30.0        |0          |0.0                  |161               |0                   |0               |0                |
|2      |2024-05-08 18:16:57|evening    |58.8  |58.8  |89.76|50.12      |1           |30.0        |0          |1.31                 |147               |0                   |0               |0                |
|2      |2024-05-08 18:16:24|evening    |58.37 |58.37 |92.5 |56.38      |0           |30.02       |0          |0.0                  |146               |0                   |0               |0                |
|2      |2024-05-08 18:15:51|evening    |53.2  |55.6  |90.2 |58.48      |0           |30.0        |0          |0.0                  |133               |1                   |0               |0                |
+-------+-------------------+-----------+------+------+-----+-----------+------------+------------+-----------+---------------------+------------------+--------------------+----------------+-----------------+
only showing top 5 rows

>>> combined = spark.read.parquet("hdfs:///user/nogi/monkeytype/curated/combined_df.parquet")
>>> combined.printSchema()
root
 |-- user_id: integer (nullable = true)
 |-- datetime: timestamp (nullable = true)
 |-- time_of_day: string (nullable = true)
 |-- wpm: double (nullable = true)
 |-- rawWpm: double (nullable = true)
 |-- acc: double (nullable = true)
 |-- consistency: double (nullable = true)
 |-- restartCount: integer (nullable = true)
 |-- testDuration: double (nullable = true)
 |-- afkDuration: integer (nullable = true)
 |-- incompleteTestSeconds: double (nullable = true)
 |-- correct_characters: integer (nullable = true)
 |-- incorrect_characters: integer (nullable = true)
 |-- extra_characters: integer (nullable = true)
 |-- missed_characters: integer (nullable = true)

>>> combined.show(5, truncate = False)
+-------+-------------------+-----------+------+------+-----+-----------+------------+------------+-----------+---------------------+------------------+--------------------+----------------+-----------------+
|user_id|datetime           |time_of_day|wpm   |rawWpm|acc  |consistency|restartCount|testDuration|afkDuration|incompleteTestSeconds|correct_characters|incorrect_characters|extra_characters|missed_characters|
+-------+-------------------+-----------+------+------+-----+-----------+------------+------------+-----------+---------------------+------------------+--------------------+----------------+-----------------+
|2      |2024-08-21 15:23:59|afternoon  |113.18|115.98|96.96|71.37      |0           |30.0        |0          |0.0                  |283               |2                   |1               |0                |
|2      |2024-05-08 18:17:32|evening    |64.4  |64.4  |94.25|62.4       |0           |30.0        |0          |0.0                  |161               |0                   |0               |0                |
|2      |2024-05-08 18:16:57|evening    |58.8  |58.8  |89.76|50.12      |1           |30.0        |0          |1.31                 |147               |0                   |0               |0                |
|2      |2024-05-08 18:16:24|evening    |58.37 |58.37 |92.5 |56.38      |0           |30.02       |0          |0.0                  |146               |0                   |0               |0                |
|2      |2024-05-08 18:15:51|evening    |53.2  |55.6  |90.2 |58.48      |0           |30.0        |0          |0.0                  |133               |1                   |0               |0                |
+-------+-------------------+-----------+------+------+-----+-----------+------------+------------+-----------+---------------------+------------------+--------------------+----------------+-----------------+
only showing top 5 rows
```

d. Converting Parquet outputs to CSV for MapReduce jobs

```
>>> base_out = "hdfs:///user/nogi/monkeytype/csv"
>>> base_in = "hdfs:///user/nogi/monkeytype/curated"
>>> u1 = spark.read.parquet(f"{base_in}/filtered_user1.parquet/")
>>> u2 = spark.read.parquet(f"{base_in}/filtered_user2.parquet")
>>> cmb = spark.read.parquet(f"{base_in}/combined_df.parquet")
>>>
>>>
>>>
>>> u1.write.mode("overwrite").option("header", "true").csv(f"{base_out}/filtered_user1.csv")
>>> u2.write.mode("overwrite").option("header", "true").csv(f"{base_out}/filtered_user2.csv")
>>> cmb.write.mode("overwrite").option("header", "true").csv(f"{base_out}/combined_df.csv")
```

e. Compiling Java classes and creating the JAR for MR jobs

```
nogi@LAPTOP-8JQG56KU:/mnt/d/Projects/Deep Learning/Projects/Monkeytype-Trend-Detection$ mkdir -p build/classes
nogi@LAPTOP-8JQG56KU:/mnt/d/Projects/Deep Learning/Projects/Monkeytype-Trend-Detection$ javac -cp "$(hadoop classpath)" -d build/classes $(find . -name "*.java")
nogi@LAPTOP-8JQG56KU:/mnt/d/Projects/Deep Learning/Projects/Monkeytype-Trend-Detection$ jar -cvf monkeytype-mr.jar -C build/classes .
added manifest
adding: distributed/(in = 0) (out= 0)(stored 0%)
adding: distributed/hadoop/(in = 0) (out= 0)(stored 0%)
adding: distributed/hadoop/job1/(in = 0) (out= 0)(stored 0%)
adding: distributed/hadoop/job1/AvgWpmByUserDriver.class(in = 1846) (out= 961)(deflated 47%)
adding: distributed/hadoop/job1/AvgWpmByUserMapper.class(in = 2538) (out= 1135)(deflated 55%)
adding: distributed/hadoop/job1/AvgWpmByUserReducer.class(in = 1785) (out= 769)(deflated 56%)
adding: distributed/hadoop/job2/(in = 0) (out= 0)(stored 0%)
adding: distributed/hadoop/job2/AvgWpmByTodDriver.class(in = 1848) (out= 970)(deflated 47%)
adding: distributed/hadoop/job2/AvgWpmByTodMapper.class(in = 2227) (out= 977)(deflated 56%)
adding: distributed/hadoop/job2/AvgWpmByTodReducer.class(in = 1783) (out= 769)(deflated 56%)
nogi@LAPTOP-8JQG56KU:/mnt/d/Projects/Deep Learning/Projects/Monkeytype-Trend-Detection$ ls
Dataset  Documentation  Notebook  Results  build  distributed  monkeytype-mr.jar
```

f. Executing MapReduce Job 1 and Job 2 using the JAR

```
nogi@LAPTOP-8JQG56KU:~$ hdfs dfs -rm -r -f /user/$USER/monkeytype/out/job1_avg_wpm_by_user
hadoop jar ~/monkeytype-mr.jar distributed.hadoop.job1.AvgWpmByUserDriver \
  hdfs:///user/$USER/monkeytype/csv/combined_df.csv \
  hdfs:///user/$USER/monkeytype/out/job1_avg_wpm_by_user
nogi@LAPTOP-8JQG56KU:~$ hdfs dfs -rm -r -f /user/$USER/monkeytype/out/job2_avg_wpm_by_tod
hadoop jar ~/monkeytype-mr.jar distributed.hadoop.job2.AvgWpmByTodDriver \
  hdfs:///user/$USER/monkeytype/csv/combined_df.csv \
  hdfs:///user/$USER/monkeytype/out/job2_avg_wpm_by_tod
```

g. Hadoop MapReduce job outputs (aggregated results)

```
nogi@LAPTOP-8JQG56KU:~$ hdfs dfs -cat /user/$USER/monkeytype/out/job1_avg_wpm_by_user/part-* | sed -n '1,20p'
1       70.45229457364336
2       98.71119379844959
nogi@LAPTOP-8JQG56KU:~$ hdfs dfs -cat /user/$USER/monkeytype/out/job2_avg_wpm_by_tod/part-*  | sed -n '1,20p'
afternoon       80.4313616071429
evening 76.6786390532544
morning 86.43919626168227
night   100.5328985507246
nogi@LAPTOP-8JQG56KU:~$
```