IMDB Reviews Keyword Extraction Using Scala

Author: You-Jun Chen (yc7093)

1. Introduction

This Zeppelin notebook demonstrates the ingestion and processing of the IMDB Review Dataset (https://www.kaggle.com/datasets/ebiswas/imdb-review-dataset?resource=download), sourced from Kaggle.
The dataset contains user reviews of movies and TV shows. It includes fields such as `review_id`, `review_summary`, `review_detail`, `rating`, and more. The dataset is split into six JSON files, each approximately 1.5GB in size. This notebook processes a single file, `part-01.json`, as a demonstration. The methods shown can be applied to the remaining files.

The primary objective of the data ingestion process is to partition the dataset by rating and keywords. This allows efficient access to subsets of the data for targeted analysis. For instance, we can quickly retrieve records with a specific rating and keyword using the following partition path:

```
val specificPartitionPath = "/user/yc7093_nyu_edu/imdb_partitioned_by_rating_a
```

Challenges

Initally, I encountered issues loading the JSON files directly into Spark. The error was likely due to non-standard formatting in the JSON file.
I attempted debugging but could not resolve the issue before the deadline.

Workaround

To overcome this challenge, I used Pandas to preprocess the data:

1. Converted the problematic JSON file into parquet format.
2. Used the resulting parquet file (`part-01.parquet`) as input to the Spark pipeline in this notebook.

Here is the Python code used for preprocessing:

```
import pandas as pd

df = pd.read_json("part-01.json")

df.to_parquet("~/part-01.parquet", index=False)
```

I will investigate further to identify and resolve the issues with the JSON format for future scalability. For now, the CSV file serves as a clean and manageable input to proceed with data ingestion and transformation.

Took 0 sec. Last updated by yc7093_nyu_edu at December 10 2024, 1:58:09 PM.

2. Load Data

FINISHED

Took 0 sec. Last updated by yc7093_nyu_edu at December 10 2024, 1:58:13 PM.

SPARK JOB FINISHED

```
val basePath = "/user/yc7093_nyu_edu/imdb-reviews-w-emotion/part"
val fileSuffixes = List("-01-all") //, "-02-all", "-03-all", "-04-all")

val initialPath = s"$basePath${fileSuffixes.head}"
var rawDF = spark.read.parquet(initialPath)

for (suffix <- fileSuffixes.tail) {
  val fullPath = s"$basePath$suffix"
  val part_df = spark.read.parquet(fullPath)
  rawDF = rawDF.union(part_df)
}


rawDF.show(5)
```

```
+---------+-----------+--------------------+------+--------------------+------------------+-
----------+--------------------+--------+--------------------+--------------------+--------
-----------+--------------------+--------------------+--------------------+----------------
----+-------+
|review_id|   reviewer|               movie|rating|      review_summary|       review_date|s
poiler_tag|       review_detail| helpful|   predicted_emotion|             sadness|
joy|                love|               anger|                fear|            surprise|emo
tion|
+---------+-----------+--------------------+------+--------------------+-----------------+-
----------+--------------------+--------+--------------------+--------------------+--------
-----------+--------------------+--------------------+--------------------+----------------
----+-------+
|rw5552176|  FeastMode|It Chapter Two (2...|   2.0|      bad and BORING|    15 March 2020|
0|I was enjoying it...|  [0, 2]|[{score -> 0.1004...| 0.10041969269514084| 0.67338609695434
57|0.017813226208090782|  0.19233390688896180|0.011774818412959576|0.004272174555808306|
joy|
|rw6455111|rapadgettra|Perry Mason: The ...|   4.0|        Not feasible|   8 January 2021|
```

Took 46 sec. Last updated by yc7093_nyu_edu at December 10 2024, 1:58:03 PM.

```
rawDF.count
```

SPARK JOB FINISHED

```
res2: Long = 303907
```

Took 3 sec. Last updated by yc7093_nyu_edu at December 10 2024, 1:58:06 PM.

```
rawDF.select("movie").distinct().count()
```

SPARK JOB FINISHED

```
res3: Long = 75884
```

Took 7 sec. Last updated by yc7093_nyu_edu at December 10 2024, 1:58:13 PM.

3. Drop redundant columns

FINISHED

Took 0 sec. Last updated by yc7093_nyu_edu at December 10 2024, 1:58:46 PM.

```
val df = rawDF.drop("spoiler_tag", "helpful", "predicted_emotion")

df.show(5)
```

SPARK JOB (http://nyu-dataproc-w-0.c.hpc-dataproc-19b8.internal:45771/jobs/job?id=7) FINISHED

```
|review_id|   reviewer|               movie|rating|      review_summary|      review_date|
review_detail|             sadness|                 joy|                love|
anger|                fear|            surprise|emotion|
+---------+-----------+--------------------+------+--------------------+-----------------+-
-------------------+--------------------+--------------------+--------------------+--------
-----------+--------------------+--------------------+-------+
|rw5552176|  FeastMode|It Chapter Two (2...|   2.0|      bad and BORING|    15 March 2020|I
was enjoying it...| 0.10041969269514084| 0.6733860969543457|0.017813226208090782|  0.192333
90688896180|0.011774818412959576|0.004272174555808306|    joy|
|rw6455111|rapadgettra|Perry Mason: The ...|   4.0|        Not feasible|   8 January 2021|I
t's hard when gu...| 0.04410260170698166| 0.6805945634841919|0.013725311495363712| 0.246736
16886138916|0.010360205546021461|0.004481049720197916|    joy|
|rw5178379| MehnaJain2|     Fixerr (2019– )|  10.0|              Fixerr|  11 October 2019|E
veryone plays th...|0.003539941739290...| 0.9815665483474731|0.007583207450807095|0.0025564
30874392...|0.002453002380207181|0.002300753956660...|    joy|
|rw2486692|  boblipton|Doctor Who: The G...|   9.0|Saving Time in a ...|10 September 2011|T
```

```
he essence of a ...| 0.04708291217684746| 0.04350305721163750|0.011013857088983059|  0.83472
```

Took 1 sec. Last updated by yc7093_nyu_edu at December 10 2024, 1:59:21 PM.

4. Process the ratings

FINISHED

Remove records with invalid ratings

Took 0 sec. Last updated by yc7093_nyu_edu at December 10 2024, 1:58:22 PM.

SPARK JOB FINISHED

```
val distinctRatings = df.select("rating").distinct()

distinctRatings.show()
```

```
+------+
|rating|
+------+
|   1.0|
|   6.0|
|   5.0|
|   2.0|
|   4.0|
|  10.0|
|   8.0|
|   7.0|
|   3.0|
|   9.0|
+------+

distinctRatings: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [rating: double]
```

Took 3 sec. Last updated by yc7093_nyu_edu at December 10 2024, 1:59:26 PM.

SPARK JOB (http://nyu-dataproc-w-0.c.hpc-dataproc-19b8.internal:45771/jobs/job?id=10) FINISHED

```
val ratingsDf = df.filter(col("rating").between(0, 10.0))

ratingsDf.show()
```

```
2020|Rebecca\n2020\n12...|0.003995738457888365|  0.9808672070503235|0.008512324653565884|0.
002620950341224...|0.001256136922165...|0.002747666323557496|     joy|
|rw6091617|zondaar-887-733837|  The Turning (2020)|   1.0|         Utterly crap|13 September
2020|What in the world...|  0.6855226755142212|0.007001281715929508|0.002699430100619793|
0.2886124253273010|0.013867603614926338|0.002296684077009...| sadness|
|rw6426522|morrison-dylan-fan|Zai na he pan qin...|   7.0|The grass is alwa...| 31 December
2020|After viewing Bat...|0.002465607132762...|  0.9881367683410645|0.005373251158744097|0.
00192202976904 8...|8.612305391579866E-4|0.001241155783645...|     joy|
|rw6261298|tom24601-84-418324|Agents of S.H.I.E...|   3.0|             Sub par| 12 November
2020|All the 80s refer...|0.006499762181192...|  0.7685708403587341| 0.06828149408102036|0.
04038564488291740 4|0.017089199274778366| 0.09917303919792175|     joy|
|rw5184180|   fireshead-70473|          小丑 (2019)|  10.0|In a nutshell ......|  13 October
```

```
2019|Both the setting,...|0.004886488895863295|0.001548342406749...|0.001674780040048...|0.
009280906990170479|  0.9791361689567566|0.003473345655947...|    fear|
|rw6443478|         blackoutH|        Tenet (2020)|   9.0|Nolan cannot disa...|   4 January
2021|The greatest dire...| 0.03710252046585083|  0.5483881235122681| 0.12739445269107819|
0.2654239237308502|0.018496770411729813|0.003194198710843...|     joy|
|rw2970942|    kunalkhandwala|      RoboCop (2014)|   7.0|A more human rema...| 28 February
```

Took 1 sec. Last updated by yc7093_nyu_edu at December 10 2024, 2:00:15 PM. (outdated)

```
ratingsDf.count()
```

SPARK JOB FINISHED

```
res10: Long = 303907
```

Took 2 sec. Last updated by yc7093_nyu_edu at December 10 2024, 2:00:21 PM.

```
ratingsDf.select("rating").distinct().show()
```

SPARK JOB FINISHED

```
+------+
|rating|
+------+
|   1.0|
|   6.0|
|   5.0|
|   2.0|
|   4.0|
|  10.0|
|   8.0|
|   7.0|
|   3.0|
|   9.0|
+------+
```

Took 3 sec. Last updated by yc7093_nyu_edu at December 10 2024, 2:00:26 PM.

5. Process the keywords in the reviews

FINISHED

Unigrams analysis

Objective: Identify the most common words in `review_summary` and `review_detail` while removing stop words.

Took 3 sec. Last updated by yc7093_nyu_edu at December 10 2024, 1:57:52 PM.

FINISHED

```
import org.apache.spark.sql.functions._

val stopWords = Set("a", "an", "the", "and", "or", "of", "to", "in", "is", "on", "with", "
    "then", "so", "no", "yes", "not", "am", "are", "as", "do", "does", "did", "my", "your"
    "all", "have", "his", "her", "just", "more", "very", "t", "s", "story", "show", "out",
    "only", "still", "movies", "into", "characters", "review", "make", "seen", "plot", "ch
    "watch", "has", "there", "here", "some", "made", "where", "him", "tv", "could", "many"

val broadcastStopWords = spark.sparkContext.broadcast(stopWords)


val tokenizeAndFilter = udf { (text: String) =>
  if (text == null) Array.empty[String]
  else {
    text.toLowerCase
      .split("\\W+") // Split by non-word characters
      .filter(word => word.nonEmpty && !broadcastStopWords.value.contains(word)) // Remove
  }
}
```

```
import org.apache.spark.sql.functions._
stopWords: scala.collection.immutable.Set[String] = Set(for, s, series, review, this, in, h
ave, your, are, is, his, why, too, show, seen, watching, am, than, plot, yes, but, what, wo
uld, another, if, so, our, t, do, all, him, just, us, it, watch, a, movie, as, because, ha
s, she, m, tv, man, or, they, characters, way, i, films, that, out, to, you, did, movies, h
ere, was, there, drama, at, 1, been, over, also, can, on, how, my, after, who, me, them, v
e, by, then, he, even, should, story, will, much, their, not, character, with, from, still,
2, episode, could, make, end, its, which, an, be, into, where, get, her, time, were, more,
about, many, see, 3, made, no, very, we, don, some, does, when, film, of, and, one, ever, t
he, ...
```

Took 1 sec. Last updated by yc7093_nyu_edu at December 10 2024, 1:59:40 PM.

Show top 100 words in `review_summary`

FINISHED

Took 0 sec. Last updated by yc7093_nyu_edu at December 10 2024, 1:59:47 PM.

SPARK JOB FINISHED

```
val tokenizedSummaryDf = ratingsDf
  .withColumn("summary_tokens", explode(tokenizeAndFilter(col("review_summary")))) // Toke

val summaryWordCounts = tokenizedSummaryDf
  .groupBy("summary_tokens")
  .count()
  .orderBy(desc("count"))
  .withColumnRenamed("summary_tokens", "word")

println("Top Frequent Words in review_summary:")
summaryWordCounts.show(100, truncate = false)
```

```
|decent       |1507 |
|never        |1483 |
|family       |1475 |
|original     |1457 |
|slow         |1457 |
|done         |1406 |
|again        |1386 |
|wonderful    |1382 |
|dark         |1375 |
|poor         |1372 |
|true         |1318 |
|long         |1311 |
|absolutely   |1301 |
|old          |1279 |
|horrible     |1238 |
|far          |1212 |
|off          |1185 |
|had          |1174 |
```

Took 5 sec. Last updated by yc7093_nyu_edu at December 10 2024, 2:00:44 PM. (outdated)

Show top 100 words in `review_detail`

FINISHED

Took 0 sec. Last updated by yc7093_nyu_edu at December 10 2024, 2:13:08 PM.

SPARK JOB FINISHED

```
// Tokenize and flatten `review_detail`
val tokenizedDetailDf = ratingsDf
  .withColumn("detail_tokens", explode(tokenizeAndFilter(col("review_detail"))))


val detailWordCounts = tokenizedDetailDf
  .groupBy("detail_tokens")
  .count()
  .orderBy(desc("count"))
  .withColumnRenamed("detail_tokens", "word")


println("Top Frequent Words in review_detail:")
detailWordCounts.show(100, truncate = false)
```

```
|going      |36968 |
|scene      |36702 |
|every      |36671 |
|real       |36131 |
|doesn      |35123 |
|season     |35020 |
|feel       |34924 |
|makes      |34519 |
|through    |34178 |
|things     |33866 |
|though     |33793 |
|those      |33551 |
|world      |33315 |
|actors     |32840 |
|same       |32680 |
|cast       |32557 |
|again      |32255 |
Twork         152054 1
```

Took 13 sec. Last updated by yc7093_nyu_edu at December 10 2024, 2:00:57 PM.

FINISHED

```
val unigramTargetWords = List(
    "good", "great", "best", "love", "bad", "funny", "fun", "amazing", "worst", "comedy", '
    "beautiful", "masterpiece", "brilliant", "classic", "interesting", "awesome", "terribl
    "disappointing", "underrated", "family"
)


unigramTargetWords.size
```

```
unigramTargetWords: List[String] = List(good, great, best, love, bad, funny, fun, amazing,
worst, comedy, excellent, boring, horror, entertaining, beautiful, masterpiece, brilliant,
classic, interesting, awesome, terrible, perfect, enjoyable, original, fantastic, wonderfu
l, horrible, disappointing, underrated, family)
res28: Int = 30
```

Took 0 sec. Last updated by yc7093_nyu_edu at December 10 2024, 2:25:46 PM.

Bigrams Analysis

FINISHED

In my initial exploration, I observed that analyzing individual words (unigrams) did not yield sufficient context or meaningful insights about the dataset. Many words appeared frequently but lacked the ability to convey the relationships or patterns within the reviews.

To address this, I utilized the **NGram** model to generate **bigrams** (two-word sequences). This approach captures relationships between adjacent words and provides richer insights into common phrases used in the reviews. For example, phrases like “great acting” or “bad movie” offer more actionable information than the individual words “great” or “bad.”

Took 0 sec. Last updated by yc7093_nyu_edu at December 10 2024, 2:00:47 PM.

Show top 100 bigrams in `review_summary`

FINISHED

Took 0 sec. Last updated by yc7093_nyu_edu at December 10 2024, 2:00:49 PM.

SPARK JOB FINISHED

```
import org.apache.spark.ml.feature.NGram
import org.apache.spark.sql.functions._

// Define stop words
val stopWords = Set("a", "an", "the", "and", "or", "of", "to", "in", "is", "on", "with", "
    "then", "no", "yes", "not", "am", "are", "as", "do", "does", "did", "my", "your", "our
    like", "feels like", "m", "has", "look like", "seems like", "could ve")

val broadcastStopWords = sc.broadcast(stopWords)
val tokenizedSummaryDf = ratingsDf.withColumn("summary_tokens", tokenizeAndFilter(col("rev

val nGramSummary = new NGram()
  .setN(2)
  .setInputCol("summary_tokens")
  .setOutputCol("summary_bigrams")

val bigramSummaryDf = nGramSummary.transform(tokenizedSummaryDf)

val explodedSummaryBigrams = bigramSummaryDf.withColumn("summary_bigram", explode(col("sum

val filteredSummaryBigrams = explodedSummaryBigrams.filter { row =>
  val bigram = row.getString(row.fieldIndex("summary_bigram"))
  val words = bigram.split(" ")
  words.forall(word => !broadcastStopWords.value.contains(word))
}

val summaryBigramCounts = filteredSummaryBigrams
  .groupBy("summary_bigram")
  .count()
  .orderBy(desc("count"))

println("Top Frequent Bigrams in review_summary:")
summaryBigramCounts.show(100, truncate = false)
```

```
Top Frequent Bigrams in review_summary:
+----------------------+-----+
|summary_bigram        |count|
+----------------------+-----+
|sci fi                |1017 |
|star wars             |668  |
|pretty good           |634  |
|well done             |618  |
|really good           |541  |
|10 10                 |527  |
|better expected       |439  |
|low budget            |394  |
|bad reviews           |365  |
|feel good             |359  |
|great acting          |306  |
|thought provoking     |295  |
|really bad            |294  |
|mind blowing          |277  |
```

Took 11 sec. Last updated by yc7093_nyu_edu at December 10 2024, 2:01:08 PM. (outdated)

Show top 100 bigrams in `review_detail`

FINISHED

Took 0 sec. Last updated by yc7093_nyu_edu at December 10 2024, 2:01:48 PM.

```
import org.apache.spark.ml.feature.NGram
import org.apache.spark.sql.functions._

val tokenizedDetailDf = ratingsDf.withColumn("detail_tokens", tokenizeAndFilter(col("revie

// Generate bigrams for `review_detail`
val nGramDetail = new NGram()
  .setN(2)
  .setInputCol("detail_tokens")
  .setOutputCol("detail_bigrams")

val bigramDetailDf = nGramDetail.transform(tokenizedDetailDf)

val explodedDetailBigrams = bigramDetailDf.withColumn("detail_bigram", explode(col("detail_

val filteredDetailBigrams = explodedDetailBigrams.filter { row =>
  val bigram = row.getString(row.fieldIndex("detail_bigram"))
  val words = bigram.split(" ")
  words.forall(word => !broadcastStopWords.value.contains(word))
}

val detailBigramCounts = filteredDetailBigrams
  .groupBy("detail_bigram")
  .count()
  .orderBy(desc("count"))

println("Top Frequent Bigrams in review_detail:")
detailBigramCounts.show(100, truncate = false)
```

SPARK JOB FINISHED

```
Top Frequent Bigrams in review_detail:
+--------------------+-----+
|detail_bigram       |count|
+--------------------+-----+
|special effects     |6222 |
|well done           |6124 |
|feel like           |6073 |
|star wars           |6055 |
|really good         |6036 |
|sci fi              |6033 |
|real life           |5508 |
|felt like           |5083 |
|feels like          |5001 |
|year old            |4946 |
|10 10               |4860 |
|pretty good         |4634 |
|looks like          |4512 |
|low budget          |3932 |
```

Took 7 min 55 sec. Last updated by yc7093_nyu_edu at December 10 2024, 2:09:45 PM.

```
val bigramTargetWords = List(
  "sci fi", "well done", "really good", "better expected", "low budget", "feel good", "sur
      money", "well worth", "science fiction",
```

FINISHED

```
  "well acted", "action packed", "mind blowing", "romantic comedy", "great cast", "special
  "action flick", "good idea", "rip off", "wow wow", "best horror", "rom com", "cult class
  "hear warming", "top notch", "definitely worth", "visually stunning", "best action", "ho
  "absolutely amazing", "hidden gem", "great family", "highly recommend"
)



bigramTargetWords.size
```

```
bigramTargetWords: List[String] = List(sci fi, well done, really good, better expected, low
budget, feel good, surprisingly good, thought provoking, really bad, let down, good acting,
bad acting, worth seeing, waste money, well worth, science fiction, well acted, action pack
ed, mind blowing, romantic comedy, great cast, special effects, good fun, nothing special,
really enjoyed, action flick, good idea, rip off, wow wow, best horror, rom com, cult class
ic, nothing new, above average, soap opera, high school, hear warming, top notch, definitel
y worth, visually stunning, best action, horror flick, die hard, pleasantly surprised, abso
lutely amazing, hidden gem, great family, highly recommend)
res27: Int = 48
```

Took 1 sec. Last updated by yc7093_nyu_edu at December 10 2024, 2:25:40 PM.

Trigrams

FINISHED

Upon analyzing the trigrams, I found that they do not provide additional meaningful keywords beyond what is already captured by bigrams. Most of the significant phrases are sufficiently represented in the bigrams, making trigrams redundant for this analysis. Therefore, I chose to focus on bigrams for extracting useful insights.

Took 0 sec. Last updated by yc7093_nyu_edu at December 10 2024, 2:04:16 PM.

SPARK JOB FINISHED

```
val stopWords = Set("a", "an", "the", "and", "or", "of", "to", "in", "is", "it", "with", "
    "then", "no", "yes", "not", "am", "are", "as", "do", "does", "did", "my", "your", "our
    like", "feels like", "m", "has", "look like", "seems like", "could ve")

val broadcastStopWords = spark.sparkContext.broadcast(stopWords)

val tokenizedDf = ratingsDf.withColumn("tokens", tokenizeAndFilter(col("review_detail")))

val nGram = new NGram()
  .setN(3) // Set n=3 for trigrams
  .setInputCol("tokens")
  .setOutputCol("trigrams")

val trigramDf = nGram.transform(tokenizedDf)

val explodedTrigrams = trigramDf.withColumn("trigram", explode(col("trigrams")))

val filteredTrigrams = explodedTrigrams.filter { row =>
  val trigram = row.getString(row.fieldIndex("trigram"))
  val words = trigram.split(" ")
  words.forall(word => !broadcastStopWords.value.contains(word)) // All words must not be s
}

val trigramCounts = filteredTrigrams
  .groupBy("trigram")
  .count()
  .orderBy(desc("count"))
```

```
+----------------------+-----+
|trigram               |count|
+----------------------+-----+
|new york city         |695  |
|world war ii          |662  |
|robert de niro        |537  |
|samuel l jackson      |453  |
|sushant singh rajput  |449  |
|batman v superman     |432  |
|breath fresh air      |414  |
|really looking forward|398  |
|first few episodes    |373  |
|had high hopes        |371  |
|7 5 10                |356  |
|really well done      |350  |
|well put together     |347  |
|chemistry between two |345  |
|star wars fan         |337  |
```

Took 8 min 0 sec. Last updated by yc7093_nyu_edu at December 10 2024, 2:17:45 PM.

6. Save data as Parquet Partitioned by Rating and Word FINISHED

From the analysis above, I identified a set of keywords that are frequently used in the reviews. These keywords represent various sentiments or aspects of the movies and TV shows, such as positive adjectives (e.g., “good”, “great”, “amazing”), negative adjectives (e.g., “bad”, “terrible”, “awful”), and thematic phrases (e.g., “special effects”, “sci-fi”, “roller coaster ride”).

Target Keywords

The following list of target keywords was used to flag reviews based on their content.

Took 0 sec. Last updated by yc7093_nyu_edu at December 10 2024, 2:03:10 PM.

FINISHED

```
val targetWords = unigramTargetWords ++ bigramTargetWords

targetWords
```

```
targetWords: List[String] = List(good, great, best, love, bad, funny, fun, amazing, worst,
comedy, excellent, boring, horror, entertaining, beautiful, masterpiece, brilliant, classi
c, interesting, awesome, terrible, perfect, enjoyable, original, fantastic, wonderful, horr
ible, disappointing, underrated, family, sci fi, well done, really good, better expected, l
ow budget, feel good, surprisingly good, thought provoking, really bad, let down, good acti
ng, bad acting, worth seeing, waste money, well worth, science fiction, well acted, action
packed, mind blowing, romantic comedy, great cast, special effects, good fun, nothing speci
al, really enjoyed, action flick, good idea, rip off, wow wow, best horror, rom com, cult c
lassic, nothing new, above average, soap ope...
```

Took 0 sec. Last updated by yc7093_nyu_edu at December 10 2024, 2:25:53 PM.

```
targetWords.size
```

FINISHED

```
res30: Int = 78
```

Took 0 sec. Last updated by yc7093_nyu_edu at December 10 2024, 2:25:55 PM.

FINISHED

Each review was checked for the presence of the keywords in both review_summary and review_detail.
If a keyword was found, a flag was added to indicate its presence, and the keyword was formatted to replace spaces with underscores for ease of partitioning.

Took 0 sec. Last updated by yc7093_nyu_edu at December 10 2024, 2:03:17 PM.

```
import org.apache.spark.sql.functions._

val dfWithWordFlags = targetWords.foldLeft(df) { (tempDf, word) =>
  tempDf.withColumn(word.replaceAll(" ", "_"),
    lower(col("review_detail")).contains(word) || lower(col("review_summary")).contains(wo
  )
}
```

FINISHED

```
import org.apache.spark.sql.functions._
dfWithWordFlags: org.apache.spark.sql.DataFrame = [review_id: string, reviewer: string ...
89 more fields]
```

Took 1 sec. Last updated by yc7093_nyu_edu at December 10 2024, 2:26:02 PM.

SPARK JOB (http://nyu-dataproc-w-0.c.hpc-dataproc-19b8.internal:45771/jobs/job?id=45) FINISHED

```
val explodedDf = dfWithWordFlags.selectExpr(
  "review_id",
  "movie",
  "review_summary",
  "review_detail",
  "emotion",
  "rating",
  "stack(" + targetWords.length + ", " +
  targetWords.map(word => s"'$word', ${word.replaceAll(" ", "_")}").mkString(", ") +
  ") as (word, is_present)"
).filter(col("is_present"))

explodedDf.show(5)
```

```
+---------+--------------------+--------------------+--------------------+-------+------+--
-----------+----------+
|review_id|               movie|      review_summary|       review_detail|emotion|rating|
word|is_present|
+---------+--------------------+--------------------+--------------------+-------+------+--
-----------+----------+
|rw5552176|It Chapter Two (2...|      bad and BORING|I was enjoying it...|    joy|   2.0|
bad|      true|
|rw5552176|It Chapter Two (2...|      bad and BORING|I was enjoying it...|    joy|   2.0|
boring|      true|
|rw6455111|Perry Mason: The ...|        Not feasible|It's hard when gu...|    joy|   4.0|di
sappointing|      true|
|rw5178379|     Fixerr (2019- )|              Fixerr|Everyone plays th...|    joy|  10.0|
perfect|      true|
|rw2486692|Doctor Who: The G...|Saving Time in a ...|The essence of a ...|  anger|   9.0|
great|      true|
+---------+--------------------+--------------------+--------------------+-------+------+--
-----------+----------+
```

Took 2 sec. Last updated by yc7093_nyu_edu at December 10 2024, 2:26:11 PM.

```
// explodedDf.write.mode("overwrite").parquet("/user/yc7093_nyu_edu/imdb-all-w-emotions-keyw
```

FINISHED

Took 0 sec. Last updated by yc7093_nyu_edu at December 10 2024, 2:27:08 PM.

Saving the data in parquet format

FINISHED

Took 0 sec. Last updated by yc7093_nyu_edu at December 10 2024, 2:03:31 PM.

ERROR

```
// Save the DataFrame partitioned by `rating` and modified `word` as Parquet
explodedDf
    .withColumn("word", regexp_replace(col("word"), " ", "_")) // Replace spaces with under
    .write
    .mode("overwrite") // Overwrite existing data
    .partitionBy("emotion", "rating", "word") //
    .parquet("/user/yc7093_nyu_edu/imdb_partitioned_by_emotion_rating_word")
```

Took 0 sec. Last updated by yc7093_nyu_edu at December 10 2024, 2:17:46 PM. (outdated)

6. Load the partition for testing

FINISHED

After partitioning the dataset by `rating` and `keyword`, the partitions can be loaded selectively for further analysis.

Took 0 sec. Last updated by yc7093_nyu_edu at December 10 2024, 2:03:36 PM.

FINISHED

```
import org.apache.hadoop.fs.{FileSystem, Path}

// Specify the path you want to list
val hdfsPath = new Path("/user/yc7093_nyu_edu/imdb_partitioned_by_emotion_rating_word/emot

// Get the FileSystem object
val fs = FileSystem.get(spark.sparkContext.hadoopConfiguration)

// List files and directories in the specified path
val files = fs.listStatus(hdfsPath)
files.foreach(file => println(file.getPath.toString))
```

```
hdfs://nyu-dataproc-m/user/yc7093_nyu_edu/imdb_partitioned_by_emotion_rating_word/emotion=a
nger/rating=9.0/word=above_average
hdfs://nyu-dataproc-m/user/yc7093_nyu_edu/imdb_partitioned_by_emotion_rating_word/emotion=a
nger/rating=9.0/word=absolutely_amazing
hdfs://nyu-dataproc-m/user/yc7093_nyu_edu/imdb_partitioned_by_emotion_rating_word/emotion=a
nger/rating=9.0/word=action_flick
hdfs://nyu-dataproc-m/user/yc7093_nyu_edu/imdb_partitioned_by_emotion_rating_word/emotion=a
nger/rating=9.0/word=action_packed
hdfs://nyu-dataproc-m/user/yc7093_nyu_edu/imdb_partitioned_by_emotion_rating_word/emotion=a
nger/rating=9.0/word=amazing
hdfs://nyu-dataproc-m/user/yc7093_nyu_edu/imdb_partitioned_by_emotion_rating_word/emotion=a
nger/rating=9.0/word=awesome
hdfs://nyu-dataproc-m/user/yc7093_nyu_edu/imdb_partitioned_by_emotion_rating_word/emotion=a
nger/rating=9.0/word=bad
hdfs://nyu-dataproc-m/user/yc7093_nyu_edu/imdb_partitioned_by_emotion_rating_word/emotion=a
nger/rating=9.0/word=bad_acting
hdfs://nyu-dataproc-m/user/yc7093_nyu_edu/imdb_partitioned_by_emotion_rating_word/emotion=a
nger/rating=9.0/word=beautiful
```

Took 1 sec. Last updated by yc7093_nyu_edu at December 10 2024, 2:26:36 PM.

Below is an example of loading a specific partition based on `rating=9.0` and `word=fun_watch`.

FINISHED

Took 0 sec. Last updated by yc7093_nyu_edu at December 10 2024, 2:03:39 PM.

SPARK JOB FINISHED

```
val specificPartitionPath = "/user/yc7093_nyu_edu/imdb_partitioned_by_emotion_rating_word/
val specificPartitionDf = spark.read.parquet(specificPartitionPath)

// Show the data
specificPartitionDf.show() // Use truncate = false to see full text
```

```
+---------+--------------------+--------------------+--------------------+----------+
|review_id|               movie|      review_summary|       review_detail|is_present|
+---------+--------------------+--------------------+--------------------+----------+
|rw5970623|Mathu Vadalara (2...|Hidden Gem of Tol...|Mathu vadalara is...|      true|
|rw3481314|     Triple 9 (2016)|like a classic be...|Movies like this ...|      true|
|rw2935873|  Black Death (2010)|A very underrated...|I watched this be...|      true|
|rw6141847|End of the Centur...|Count the number ...|I could talk abou...|      true|
|rw5517287|  Ghost World (2001)|Refreshingly and ...|Having seen this ...|      true|
|rw3472345|He Never Died (2015)|You get only one ...|Every once in a w...|      true|
|rw6127454|Kuon (2004 Video ...|Hidden Gem of the...|This is one of th...|      true|
|rw6149667|Ninja Resurrectio...|Stupid reviewers,...|I honestly don't ...|      true|
|rw5711957|The Rhythm Sectio...|LOVED THIS FILM. ...|This movie was a ...|      true|
|rw5389293|The Chargesheet: ...|          Hidden gem|Satish kaushik is...|      true|
|rw2868396|SAGA - Curse of t...|I loved this hidd...|For the life of m...|      true|
|rw5343829|   以黑死之名 (2010)|Way better than e...|Never heard of th...|      true|
|rw6175045|Ginny Weds Sunny ...|Growing idol with...|Vikrant as impres...|      true|
|rw5978255|     Riphagen (2016)|Hidden Gem!!! A T...|Can't say anythin...|      true|
+---------+--------------------+--------------------+--------------------+----------+
```

Took 2 sec. Last updated by yc7093_nyu_edu at December 10 2024, 2:17:48 PM.

SPARK JOB (http://nyu-dataproc-w-0.c.hpc-dataproc-19b8.internal:40771/jobs/job?id=28) FINISHED

```
val outputPath = "/user/yc7093_nyu_edu/rating_9_fun_watch_partition"

// Write the DataFrame to HDFS as Parquet
specificPartitionDf.write
  .mode("overwrite") // Overwrite if the path already exists
  .parquet(outputPath)

println(s"Partition saved successfully to HDFS at $outputPath")
```

```
Partition saved successfully to HDFS at /user/yc7093_nyu_edu/rating_9_fun_watch_partition
outputPath: String = /user/yc7093_nyu_edu/rating_9_fun_watch_partition
```

Took 0 sec. Last updated by yc7093_nyu_edu at December 10 2024, 2:17:48 PM.