sm11326-midterm

November 8, 2024

1 Spark initialization - spark template

[1]:
```
import os
import pyspark

conf = pyspark.SparkConf()
conf = conf.setAppName("<my-app-name>")
conf.set('spark.ui.proxyBase', '/user/' + os.environ['JUPYTERHUB_USER'] + '/
 ↪proxy/4040') ## to setup SPARK UI
conf = conf.set('spark.jars', os.environ['GRAPHFRAMES_PATH']) ## graphframes in␣
 ↪spark configuration
sc = pyspark.SparkContext(conf=conf)
sc
```

```
24/11/08 00:17:46 WARN NativeCodeLoader: Unable to load native-hadoop library
for your platform… using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
```

[1]:
```
<SparkContext master=local[*] appName=<my-app-name>>
```

1.0.1 Open Spark UI

https://csgy1-6513-fall.rcnyu.org/user/<USER_NETID>/proxy/4040/jobs/

[2]:
```
# QUESTION 2 - WITHOUT ANY FILTERING - I.E INCLUSIVE OF ALL PUNCTUATIONS [ @, .
 ↪, , , < , >, p ]

from pyspark.sql import SparkSession
from pyspark.sql.functions import col, split, explode, count, concat_ws, expr

spark = SparkSession.builder.appName("TrigramLanguageModel").getOrCreate()

input_path = "*.txt"
text_df = spark.read.text(input_path)
```

```
words_df = text_df.select(split(col("value"), "\s+").alias("words"))

words_df = words_df.filter("size(words) >= 3")

bigrams_df = words_df.select(
    explode(expr("transform(sequence(0, size(words) - 2), i -> array(words[i],␣
 ↪words[i + 1]))")).alias("bigram")
)
trigrams_df = words_df.select(
    explode(expr("transform(sequence(0, size(words) - 3), i -> array(words[i],␣
 ↪words[i + 1], words[i + 2]))")).alias("trigram")
)

bigram_counts = bigrams_df.groupBy("bigram").count().withColumnRenamed("count",␣
 ↪"bigram_count")
trigram_counts = trigrams_df.groupBy("trigram").count().
 ↪withColumnRenamed("count", "trigram_count")
trigram_counts = trigram_counts.withColumn("bigram_prefix", concat_ws(" ",␣
 ↪col("trigram")[0], col("trigram")[1]))
bigram_counts = bigram_counts.withColumn("bigram_str", concat_ws(" ",␣
 ↪col("bigram")[0], col("bigram")[1]))
conditional_df = trigram_counts.join(
    bigram_counts,
    trigram_counts.bigram_prefix == bigram_counts.bigram_str
).select(
    "trigram", "trigram_count", "bigram_count"
).withColumn(
    "conditional_probability", col("trigram_count") / col("bigram_count")
)


top_trigrams = conditional_df.orderBy(col("trigram_count").desc()).limit(10)
top_trigrams.select("trigram", "trigram_count", "conditional_probability").
 ↪show(truncate=False)
spark.stop()
```

```
24/11/08 00:18:00 WARN GarbageCollectionMetrics: To enable non-built-in garbage
collector(s) List(G1 Concurrent GC), users should configure it(them) to
spark.eventLog.gcMetrics.youngGenerationGarbageCollectors or
spark.eventLog.gcMetrics.oldGenerationGarbageCollectors

+----------------+-------------+----------------------+
|trigram         |trigram_count|conditional_probability|
+----------------+-------------+----------------------+
|[@, @, @]       |113349       |0.8886563021850084    |
|[., <p>, "]     |7959         |0.16716022934912733   |
|[., <p>, The]   |6287         |0.13204376955873395   |
```

```
|[., ", <p>]       |4810          |0.4664468580294802     |
|[said, ., <p>]    |2415          |0.5625436757512229     |
|[,, ", he]        |1487          |0.16955530216647663    |
|[<p>, ", We]      |1376          |0.15076147693656186    |
|[., <p>, In]      |1321          |0.02774452355449142    |
|[the, spread, of]|1180          |0.9161490683229814     |
|[,, ", said]      |1142          |0.13021664766248575    |
+-----------------+-------------+-----------------------+
```

[3]:
```
#QUESTION 2 - STRICT VERSION - WHERE I HAVE REGEX FILTERED EVERYTHING APART␣
 ↪FROM THE ALPHA-NUMERIC

from pyspark.sql import SparkSession
from pyspark.sql.functions import col, split, explode, count, concat_ws, expr

spark = SparkSession.builder.appName("TrigramLanguageModel").getOrCreate()
input_path = "*.txt"
text_df = spark.read.text(input_path)
words_df = text_df.select(split(col("value"), "\s+").alias("words"))
words_df = words_df.filter("size(words) >= 3")
bigrams_df = words_df.select(
    explode(expr("transform(sequence(0, size(words) - 2), i -> array(words[i],␣
 ↪words[i + 1]))")).alias("bigram")
).filter(
    (col("bigram")[0].rlike("^[A-Za-z0-9]+$")) & (col("bigram")[1].
 ↪rlike("^[A-Za-z0-9]+$"))
)

trigrams_df = words_df.select(
    explode(expr("transform(sequence(0, size(words) - 3), i -> array(words[i],␣
 ↪words[i + 1], words[i + 2]))")).alias("trigram")
).filter(
    (col("trigram")[0].rlike("^[A-Za-z0-9]+$")) &
    (col("trigram")[1].rlike("^[A-Za-z0-9]+$")) &
    (col("trigram")[2].rlike("^[A-Za-z0-9]+$"))
)

bigram_counts = bigrams_df.groupBy("bigram").count().withColumnRenamed("count",␣
 ↪"bigram_count")
trigram_counts = trigrams_df.groupBy("trigram").count().
 ↪withColumnRenamed("count", "trigram_count")
trigram_counts = trigram_counts.withColumn("bigram_prefix", concat_ws(" ",␣
 ↪col("trigram")[0], col("trigram")[1]))
bigram_counts = bigram_counts.withColumn("bigram_str", concat_ws(" ",␣
 ↪col("bigram")[0], col("bigram")[1]))
```

```
conditional_df = trigram_counts.join(
    bigram_counts,
    trigram_counts.bigram_prefix == bigram_counts.bigram_str
).select(
    "trigram", "trigram_count", "bigram_count"
).withColumn(
    "conditional_probability", col("trigram_count") / col("bigram_count")
)

top_trigrams = conditional_df.orderBy(col("trigram_count").desc()).limit(10)
top_trigrams.select("trigram", "trigram_count", "conditional_probability").
 ↪show(truncate=False)
spark.stop()
```

```
+---------------------------+-------------+----------------------+
|trigram                    |trigram_count|conditional_probability|
+---------------------------+-------------+----------------------+
|[the, spread, of]          |1180         |0.9161490683229814    |
|[of, the, coronavirus]     |854          |0.04978430686720298   |
|[as, well, as]             |824          |0.7285587975243147    |
|[the, number, of]          |819          |0.9457274826789839    |
|[one, of, the]             |791          |0.6233254531126872    |
|[spread, of, the]          |772          |0.5482954545454546    |
|[due, to, the]             |719          |0.42244418331374856   |
|[the, coronavirus, pandemic]|711          |0.20887191539365452   |
|[of, the, virus]           |691          |0.04028214993587501   |
|[the, end, of]             |615          |0.9057437407952872    |
+---------------------------+-------------+----------------------+
```

[4]:
```
#QUESTION 2 : BROADCASTING VERSION

from pyspark.sql import SparkSession
from pyspark.sql.functions import col, split, explode, count, concat_ws, expr,␣
 ↪broadcast

spark = SparkSession.builder.appName("TrigramLanguageModel").getOrCreate()
input_path = "*.txt"
text_df = spark.read.text(input_path)
words_df = text_df.select(split(col("value"), "\s+").alias("words"))
words_df = words_df.filter("size(words) >= 3")

bigrams_df = words_df.select(
    explode(expr("transform(sequence(0, size(words) - 2), i -> array(words[i],␣
 ↪words[i + 1]))")).alias("bigram")
```

```
).filter(
    (col("bigram")[0].rlike("^[A-Za-z0-9]+$")) & (col("bigram")[1].
 ↪rlike("^[A-Za-z0-9]+$"))
)

trigrams_df = words_df.select(
    explode(expr("transform(sequence(0, size(words) - 3), i -> array(words[i],␣
 ↪words[i + 1], words[i + 2]))")).alias("trigram")
).filter(
    (col("trigram")[0].rlike("^[A-Za-z0-9]+$")) &
    (col("trigram")[1].rlike("^[A-Za-z0-9]+$")) &
    (col("trigram")[2].rlike("^[A-Za-z0-9]+$"))
)

bigram_counts = bigrams_df.groupBy("bigram").count().withColumnRenamed("count",␣
 ↪"bigram_count")
trigram_counts = trigrams_df.groupBy("trigram").count().
 ↪withColumnRenamed("count", "trigram_count")
trigram_counts = trigram_counts.withColumn("bigram_prefix", concat_ws(" ",␣
 ↪col("trigram")[0], col("trigram")[1]))
bigram_counts = bigram_counts.withColumn("bigram_str", concat_ws(" ",␣
 ↪col("bigram")[0], col("bigram")[1]))

conditional_df = trigram_counts.join(
    broadcast(bigram_counts),
    trigram_counts.bigram_prefix == bigram_counts.bigram_str
).select(
    "trigram", "trigram_count", "bigram_count"
).withColumn(
    "conditional_probability", col("trigram_count") / col("bigram_count")
)

top_trigrams = conditional_df.orderBy(col("trigram_count").desc()).limit(10)
top_trigrams.select("trigram", "trigram_count", "conditional_probability").
 ↪show(truncate=False)
spark.stop()
```

```
+---------------------------+-------------+-----------------------+
|trigram                    |trigram_count|conditional_probability|
+---------------------------+-------------+-----------------------+
|[the, spread, of]          |1180         |0.9161490683229814     |
|[of, the, coronavirus]     |854          |0.04978430686720298    |
|[as, well, as]             |824          |0.7285587975243147     |
|[the, number, of]          |819          |0.9457274826789839     |
|[one, of, the]             |791          |0.6233254531126872     |
```

```
|[spread, of, the]           |772          |0.5482954545454546     |
|[due, to, the]              |719          |0.42244418331374856    |
|[the, coronavirus, pandemic]|711          |0.20887191539365452    |
|[of, the, virus]            |691          |0.04028214993587501    |
|[the, end, of]              |615          |0.9057437407952872     |
+----------------------------+-------------+-----------------------+
```

[5]:
```
###### QUESTION 3

from pyspark.sql import SparkSession
from pyspark.sql.functions import col, hour, when, count, rank, collect_list
from pyspark.sql.window import Window

spark = SparkSession.builder \
    .appName("Top 3 Items Per Daypart") \
    .getOrCreate()

df = spark.read.option("header", "true").csv("shared/data/Bakery.csv")
df = df.withColumn("Time", col("Time").cast("timestamp"))
df = df.withColumn("Daypart",
                   when((hour(col("Time")) >= 6) & (hour(col("Time")) < 11),␣
 ↪"morning")
                   .when((hour(col("Time")) >= 11) & (hour(col("Time")) < 14),␣
 ↪"noon")
                   .when((hour(col("Time")) >= 14) & (hour(col("Time")) < 17),␣
 ↪"afternoon")
                   .when((hour(col("Time")) >= 17) | (hour(col("Time")) < 6),␣
 ↪"evening")
                   )

item_counts = df.groupBy("Daypart", "Item").agg(count("Item").alias("count"))
window_spec = Window.partitionBy("Daypart").orderBy(col("count").desc())

ranked_items = item_counts.withColumn("rank", rank().over(window_spec)) \
    .filter(col("rank") <= 3)

top_items_per_daypart = ranked_items \
    .groupBy("Daypart") \
    .agg(collect_list("Item").alias("TopItems"))

top_items_per_daypart.show(truncate=False)

spark.stop()
```

```
+---------+-----------------------+
```

```
|Daypart  |TopItems                |
+---------+------------------------+
|afternoon|[Coffee, Bread, Tea]    |
|evening  |[Coffee, Bread, Tea]    |
|morning  |[Coffee, Bread, Pastry]|
|noon     |[Coffee, Bread, Tea]    |
+---------+------------------------+
```

[6]:
```
###### QUESTION 3 - VERIFICATION

from pyspark.sql import SparkSession
from pyspark.sql.functions import col, hour, when, count, rank, collect_list,␣
 ↪desc
from pyspark.sql.window import Window

spark = SparkSession.builder \
    .appName("Top 3 Items Per Daypart with Verification") \
    .getOrCreate()

df = spark.read.option("header", "true").csv("shared/data/Bakery.csv")
df = df.withColumn("Time", col("Time").cast("timestamp"))
df = df.withColumn("Daypart",
                   when((hour(col("Time")) >= 6) & (hour(col("Time")) < 11),␣
 ↪"morning")
                   .when((hour(col("Time")) >= 11) & (hour(col("Time")) < 14),␣
 ↪"noon")
                   .when((hour(col("Time")) >= 14) & (hour(col("Time")) < 17),␣
 ↪"afternoon")
                   .when((hour(col("Time")) >= 17) | (hour(col("Time")) < 6),␣
 ↪"evening")
                   )

item_counts = df.groupBy("Daypart", "Item").agg(count("Item").alias("count"))
print("Raw item counts per daypart:")
item_counts.show(truncate=False)
window_spec = Window.partitionBy("Daypart").orderBy(col("count").desc())
ranked_items = item_counts.withColumn("rank", rank().over(window_spec)) \
    .filter(col("rank") <= 3)

top_items_per_daypart = ranked_items \
    .groupBy("Daypart") \
    .agg(collect_list("Item").alias("TopItems"))

print("\nTop 3 Items per Daypart (Ranked):")
top_items_per_daypart.show(truncate=False)
```

```
print("\nVerification: Raw counts for the top 3 items per daypart:")
verification = ranked_items.select("Daypart", "Item", "count").
 ↪orderBy("Daypart", "rank")
verification.show(truncate=False)

spark.stop()
```

```
Raw item counts per daypart:
+---------+-------------------------+-----+
|Daypart  |Item                     |count|
+---------+-------------------------+-----+
|noon     |Bare Popcorn             |1    |
|noon     |My-5 Fruit Shoot         |7    |
|morning  |Jammie Dodgers           |22   |
|noon     |Christmas common         |5    |
|evening  |Focaccia                 |3    |
|morning  |Chocolates               |2    |
|noon     |Drinking chocolate spoons|2    |
|afternoon|Empanadas                |3    |
|afternoon|Cherry me Dried fruit    |1    |
|afternoon|Cake                     |480  |
|afternoon|Extra Salami or Feta     |15   |
|afternoon|Scone                    |127  |
|morning  |Muffin                   |79   |
|morning  |NONE                     |201  |
|evening  |Cookies                  |21   |
|afternoon|Bowl Nic Pitt            |1    |
|evening  |Juice                    |13   |
|morning  |Truffles                 |16   |
|morning  |Empanadas                |1    |
|noon     |Tacos/Fajita             |6    |
+---------+-------------------------+-----+
only showing top 20 rows


Top 3 Items per Daypart (Ranked):
+---------+----------------------+
|Daypart  |TopItems              |
+---------+----------------------+
|afternoon|[Coffee, Bread, Tea]  |
|evening  |[Coffee, Bread, Tea]  |
|morning  |[Coffee, Bread, Pastry]|
|noon     |[Coffee, Bread, Tea]  |
+---------+----------------------+


Verification: Raw counts for the top 3 items per daypart:
```

```
+---------+------+-----+
|Daypart  |Item  |count|
+---------+------+-----+
|afternoon|Coffee|1476 |
|afternoon|Bread |847  |
|afternoon|Tea   |566  |
|evening  |Coffee|87   |
|evening  |Bread |55   |
|evening  |Tea   |49   |
|morning  |Coffee|1615 |
|morning  |Bread |1081 |
|morning  |Pastry|453  |
|noon     |Coffee|2293 |
|noon     |Bread |1342 |
|noon     |Tea   |540  |
+---------+------+-----+
```

[7]:
```
# QUESTION 4 : HASHINGTF, MINHASH-LSH - APPROX NEAREST NEIGHBORS - JACCARD␣
 ↪SIMILARITY - 50 points
# COMPUTING THE TOP 5 NEAREST URLS

from pyspark.sql import SparkSession
from pyspark.ml.feature import Tokenizer, HashingTF, MinHashLSH
from pyspark.sql.functions import col


spark = SparkSession.builder.appName("MinhashLSH").getOrCreate()
data_path = 'shared/data/Huffpost.json'
df = spark.read.json(data_path)
base_description = "Kitten Born With Twisted Arms And Legs Finds A Mom Who␣
 ↪Knows She's Perfect"
tokenizer = Tokenizer(inputCol="short_description", outputCol="words")
words_df = tokenizer.transform(df)
hashingTF = HashingTF(inputCol="words", outputCol="features", numFeatures=10000)
featurized_df = hashingTF.transform(words_df)
minhash = MinHashLSH(inputCol="features", outputCol="hashes", numHashTables=5)
model = minhash.fit(featurized_df)
transformed_df = model.transform(featurized_df)
base_df = spark.createDataFrame([(base_description,)], ["short_description"])
base_words_df = tokenizer.transform(base_df)
base_features_df = hashingTF.transform(base_words_df)
similar_items = model.approxNearestNeighbors(transformed_df, base_features_df.
 ↪select("features").first()["features"], numNearestNeighbors=5)
similar_items.select("link", "headline", "category", "short_description").
 ↪show(truncate=False)
```

```
[Stage 3:============================>                            (1 + 1) / 2]

+--------------------------------------------------------------------------------
--------------------------------------------------------------+----------------
--------------------------------------------------------------+-------------+---
--------------------------------------------------------------------------------
-----------------------------------+
|link
|headline
|category     |short_description
|
+--------------------------------------------------------------------------------
--------------------------------------------------------------+----------------
--------------------------------------------------------------+-------------+---
--------------------------------------------------------------------------------
-----------------------------------+
|https://www.huffingtonpost.com/entry/mom-hilariously-nails-what-getting-ready-
looks-like-for-mothers_us_595666f0e4b0da2c73230b40              |Mom Hilariously
Nails What Getting Ready Looks Like For Mothers               |PARENTS
|"Maybe she’s born with it … Maybe she’s a tired mom who doesn’t have time for
this."                                    |
|https://www.huffingtonpost.com/entry/andrew-garfield-lip-syncs-whitney-houston-
in-epic-drag-show-act_us_592ff5d1e4b0540ffc84b79b              |Andrew Garfield
Lip-Syncs Whitney Houston In Epic Drag Show Act
|ENTERTAINMENT|With a back flip and everything.
|
|https://www.huffingtonpost.com/entry/linked-by-their-choice-to-become-single-
mothers-two-women-share-their-stories_us_59b91b63e4b0edff9717de69|Linked By
Their Choice To Become Single Mothers, Two Women Share Their Stories|PARENTS
|The stories of an Idaho mom who gave birth and a California mom who adopted.
|
|https://www.huffingtonpost.com/entry/carol-brady_us_5b9c5c92e4b03a1dcc7e15be
|What Carol Brady Is Really Saying (INFOGRAPHIC)
|PARENTING    |Here's the story of a lovely lady -- who somehow kept six kids, a
husband, a housekeeper and a dog in perfect order with|
|https://www.huffingtonpost.com/entry/britney-spears-new-
album_us_563a0b2fe4b0307f2cab4995
|Britney Spears' Comeback Is About To Be Complete
|ENTERTAINMENT|She’s back, b***hes, with a brand new album.
|
+--------------------------------------------------------------------------------
--------------------------------------------------------------+----------------
--------------------------------------------------------------+-------------+---
--------------------------------------------------------------------------------
-----------------------------------+
```

[8]:
```
#VALIDATION USING MANUAL JACCARD SIMILARITY - FORMULA

from pyspark.sql import SparkSession
from pyspark.ml.feature import Tokenizer, HashingTF, MinHashLSH
from pyspark.sql.functions import col, array_intersect, array_union, size

spark = SparkSession.builder.appName("MinhashLSH_ManualJaccard").getOrCreate()
data_path = 'shared/data/Huffpost.json'
df = spark.read.json(data_path)
base_description = "Kitten Born With Twisted Arms And Legs Finds A Mom Who␣
 ↪Knows She's Perfect"
tokenizer = Tokenizer(inputCol="short_description", outputCol="words")
words_df = tokenizer.transform(df)
base_df = spark.createDataFrame([(base_description,)], ["short_description"])
base_words_df = tokenizer.transform(base_df)
base_words = base_words_df.select("words").first()[0]  # Collect words as a list

hashingTF = HashingTF(inputCol="words", outputCol="features", numFeatures=10000)
featurized_df = hashingTF.transform(words_df)
minhash = MinHashLSH(inputCol="features", outputCol="hashes", numHashTables=5)
model = minhash.fit(featurized_df)
transformed_df = model.transform(featurized_df)
base_features_df = hashingTF.transform(base_words_df)
base_features = base_features_df.select("features").first()[0]  # Get the␣
 ↪feature vector
similar_items_minhash = model.approxNearestNeighbors(transformed_df,␣
 ↪base_features, numNearestNeighbors=5)

# VALIDATION
base_words_broadcast = spark.sparkContext.broadcast(set(base_words))

# JACCARD SIMILARITY FORMULA
def jaccard_similarity(words):
    words_set = set(words)
    intersection = words_set.intersection(base_words_broadcast.value)
    union = words_set.union(base_words_broadcast.value)
    return float(len(intersection)) / float(len(union)) if len(union) != 0 else␣
 ↪0.0

from pyspark.sql.functions import udf
from pyspark.sql.types import DoubleType

jaccard_udf = udf(jaccard_similarity, DoubleType())
jaccard_df = words_df.withColumn("jaccard_similarity",␣
 ↪jaccard_udf(col("words")))
top_jaccard_items = jaccard_df.orderBy(col("jaccard_similarity").desc()).
 ↪limit(5)
```

```
print("Top 5 items using MinHashLSH:")
similar_items_minhash.select("link", "headline", "category",␣
 ↪"short_description").show(truncate=False)

print("Top 5 items using Manual Jaccard Similarity:")
top_jaccard_items.select("link", "headline", "category", "short_description",␣
 ↪"jaccard_similarity").show(truncate=False)
spark.stop()
```

```
24/11/08 00:20:30 WARN SparkSession: Using an existing Spark session; only
runtime SQL configurations will take effect.
```

```
Top 5 items using MinHashLSH:
```

```
+--------------------------------------------------------------------------------
-------------------------------------------------------------+----------------
---------------------------------------------------------------+-------------+---
--------------------------------------------------------------------------------
-----------------------------------+
|link
|headline
|category     |short_description
|
+--------------------------------------------------------------------------------
-------------------------------------------------------------+----------------
---------------------------------------------------------------+-------------+---
--------------------------------------------------------------------------------
-----------------------------------+
|https://www.huffingtonpost.com/entry/mom-hilariously-nails-what-getting-ready-
looks-like-for-mothers_us_595666f0e4b0da2c73230b40              |Mom Hilariously
Nails What Getting Ready Looks Like For Mothers                |PARENTS
|"Maybe she’s born with it … Maybe she’s a tired mom who doesn’t have time for
this."                                    |
|https://www.huffingtonpost.com/entry/andrew-garfield-lip-syncs-whitney-houston-
in-epic-drag-show-act_us_592ff5d1e4b0540ffc84b79b              |Andrew Garfield
Lip-Syncs Whitney Houston In Epic Drag Show Act
|ENTERTAINMENT|With a back flip and everything.
|
|https://www.huffingtonpost.com/entry/linked-by-their-choice-to-become-single-
mothers-two-women-share-their-stories_us_59b91b63e4b0edff9717de69|Linked By
Their Choice To Become Single Mothers, Two Women Share Their Stories|PARENTS
|The stories of an Idaho mom who gave birth and a California mom who adopted.
|
|https://www.huffingtonpost.com/entry/carol-brady_us_5b9c5c92e4b03a1dcc7e15be
|What Carol Brady Is Really Saying (INFOGRAPHIC)
|PARENTING    |Here's the story of a lovely lady -- who somehow kept six kids, a
husband, a housekeeper and a dog in perfect order with|
```

```
|https://www.huffingtonpost.com/entry/britney-spears-new-
album_us_563a0b2fe4b0307f2cab4995
|Britney Spears' Comeback Is About To Be Complete
|ENTERTAINMENT|She’s back, b***hes, with a brand new album.
|
+-------------------------------------------------------------------------------
-----------------------------------------------------------------+---------------
-------------------------------------------------------------+-------------+---
-------------------------------------------------------------------------------
-----------------------------------+

Top 5 items using Manual Jaccard Similarity:


+-------------------------------------------------------------------------------
-----------------------------------------------------------------+---------------
-------------------------------------------------------------+-------------+---
-------------------------------------------------------------------------------
-----------------------------------+-------------------+
|link
|headline
|category     |short_description
|jaccard_similarity |
+-------------------------------------------------------------------------------
-----------------------------------------------------------------+---------------
-------------------------------------------------------------+-------------+---
-------------------------------------------------------------------------------
-----------------------------------+-------------------+
|https://www.huffingtonpost.com/entry/mom-hilariously-nails-what-getting-ready-
looks-like-for-mothers_us_595666f0e4b0da2c73230b40              |Mom Hilariously
Nails What Getting Ready Looks Like For Mothers               |PARENTS
|"Maybe she’s born with it … Maybe she’s a tired mom who doesn’t have time for
this."                                  |0.25               |
|https://www.huffingtonpost.com/entry/how-to-make-the-perfect-
c_us_5b9dc24ee4b03a1dcc8c8503
|How to Make the Perfect Chocolate Chip Cookie
|FOOD & DRINK |A cookie with the perfect combination of fat, flavor, and
comfort. Who needs detox?
|0.21739130434782608|
|https://www.huffingtonpost.com/entry/andrew-garfield-lip-syncs-whitney-houston-
in-epic-drag-show-act_us_592ff5d1e4b0540ffc84b79b              |Andrew Garfield
Lip-Syncs Whitney Houston In Epic Drag Show Act
|ENTERTAINMENT|With a back flip and everything.
|0.17647058823529413|
|https://www.huffingtonpost.com/entry/linked-by-their-choice-to-become-single-
mothers-two-women-share-their-stories_us_59b91b63e4b0edff9717de69|Linked By
Their Choice To Become Single Mothers, Two Women Share Their Stories|PARENTS
|The stories of an Idaho mom who gave birth and a California mom who adopted.
```

```
|0.17391304347826086|
|https://www.huffingtonpost.com/entry/carol-brady_us_5b9c5c92e4b03a1dcc7e15be
|What Carol Brady Is Really Saying (INFOGRAPHIC)
|PARENTING    |Here's the story of a lovely lady -- who somehow kept six kids, a
husband, a housekeeper and a dog in perfect order with|0.16666666666666666|
+-------------------------------------------------------------------------------
-----------------------------------------------------------------+---------------
---------------------------------------------------------------+-------------+---
-------------------------------------------------------------------------------
-----------------------------------+-------------------+
```