---------- Search-Engine Program details ----------

Goal: building a search engine for the plot summaries that are available in the file “plot_summaries.txt”.
Data: Carnegie Movie Summary Corpus (http://www.cs.cmu.edu/~ark/personas/)

- movie.metadata.tsv: contains movie ID, name, and other metadata
- plot_summaries.txt: movie plot summaries

Steps:

1. Remove stopwords from plot_summaries
2. Compute tf-idf using MapReduce
3. Show movie results by search terms

Note: data is uploaded to databricks

```
// import libraries
import org.apache.spark.ml.feature.StopWordsRemover


// load movie metadata
val movie_meta = sc.textFile("/FileStore/tables/movie_metadata-ab497.tsv")

movie_meta: org.apache.spark.rdd.RDD[String] = /FileStore/tables/movie_metadata-ab497.tsv MapPartitionsRDD[1] at textFile at command-3234923437497423:1

// extract movie ID and names
val names = movie_meta.map(x => (x.split("\t")(0), x.split("\t")(2)))
names.take(10)

names: org.apache.spark.rdd.RDD[(String, String)] = MapPartitionsRDD[2] at map at command-3234923437497468:2
res3: Array[(String, String)] = Array((975900,Ghosts of Mars), (3196793,Getting Away with Murder: The JonBenét Ramsey Mystery), (28463795,Brun bitter), (9363483,White Of The Eye), (26123
6,A Woman in Flames), (13696889,The Gangsters), (18998739,The Sorcerer's Apprentice), (10408933,Alexander's Ragtime Band), (9997961,Contigo y aquí), (2345652,City of the Dead))

// get stop words
val stop_word_set = StopWordsRemover.loadDefaultStopWords("english").toSet

import org.apache.spark.ml.feature.StopWordsRemover
stop_word_set: scala.collection.immutable.Set[String] = Set(down, it's, ourselves, that's, for, s, further, she'll, any, there's, this, haven't, in, ought, myself, have, your, off, once,
i'll, are, is, his, why, too, why's, am, than, isn't, didn't, himself, but, you're, below, what, would, i'd, if, you'll, own, they'll, up, we're, they'd, so, our, t, do, all, him, had, no
r, before, just, it, a, she's, as, hadn't, because, has, she, yours, or, above, yourself, herself, she'd, such, they, each, can't, don't, i, until, that, out, he's, cannot, to, we've, no
w, hers, you, did, let's, most, here, these, hasn't, was, there, when's, shan't, doing, at, through, been, over, i've, can, on, being, same, how, whom, my, after, who, itself, me, them, b
y, then, couldn't, he, should, will, few, wasn't, again, while, their, not, with, from, you've, they've, what's, wouldn't, both, could, its, under, which, you'd, an, be, here's, into, whe
re, he'll, her, themselves, were, more, we'd, where's, they're, who's, between, aren't, ours, about, doesn't, how's, against, during, no, very, we, don, having, mustn't, some, does, when,
shouldn't, yourselves, he'd, other, of, weren't, and, won't, theirs, i'm, we'll, the, those, only)

// load plot summaries and remove stop words
val movie_summary = sc.textFile("/FileStore/tables/plot_summaries.txt").map(x => (x.split("\t")(0), x.split("\t")(1).toLowerCase.split("""\W+""").filter(x => x.length > 4).filter(x =>
!stop_word_set.contains(x))))
movie_summary.take(1)

movie_summary: org.apache.spark.rdd.RDD[(String, Array[String])] = MapPartitionsRDD[14] at map at command-3234923437497426:2
res8: Array[(String, Array[String])] = Array((23890098,Array(shlykov, working, driver, lyosha, saxophonist, develop, bizarre, relationship, despite, prejudices, realize, different)))
```

```
// total number of movies
val movie_count = movie_summary.count()

movie_count: Long = 42306

// merge movie_summary with names
val movie_full = movie_summary.join(names)
movie_full.take(1)

movie_full: org.apache.spark.rdd.RDD[(String, (Array[String], String))] = MapPartitionsRDD[20] at join at command-3234923437497429:2
res9: Array[(String, (Array[String], String))] = Array((4039693,(Array(deals, young, farmer, named, billy, appreciate, terrestrial, instead, wishes, explore, outer, space, story, develop
s, billy, struggles, homosexuality, changing, relationships, around, black, magic, zealand, saying, fabulous),50 Ways of Saying Fabulous)))

// calculate tf ((token, (id, name)), 1)
val mapped_token = movie_full.flatMap(x => x._2._1.map(y => ((y, (x._1, x._2._2)), 1)))
mapped_token.take(1)

mapped_token: org.apache.spark.rdd.RDD[((String, (String, String)), Int)] = MapPartitionsRDD[21] at flatMap at command-3234923437497431:2
res10: Array[((String, (String, String)), Int)] = Array(((deals,(4039693,50 Ways of Saying Fabulous)),1))

val tf = mapped_token.reduceByKey(_+_).map(x => (x._1._1, (x._1._2, x._2)))
tf.take(1)

tf: org.apache.spark.rdd.RDD[(String, ((String, String), Int))] = MapPartitionsRDD[23] at map at command-3234923437497433:1
res11: Array[(String, ((String, String), Int))] = Array((philharmonic,((744783,Kolya),1)))

// calculate df
val df = tf.map(x => (x._1, 1)).reduceByKey(_+_)
df.sortBy(-_._2).take(10)

df: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[25] at reduceByKey at command-3234923437497435:2
res12: Array[(String, Int)] = Array((however,10437), (finds,10299), (takes,9947), (later,9928), (father,9848), (tells,9505), (young,9322), (first,9067), (family,8588), (tries,8324))

// calculate idf
val idf = df.map(x => (x._1, Math.log((movie_count + 1)/(x._2 + 1)) + 1))
idf.take(10)

idf: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[31] at map at command-3234923437497437:2
res13: Array[(String, Double)] = Array((reimi,10.95953701838715), (glorifying,9.706655856361559), (zootie,10.95953701838715), (nothin,9.573195381531523), (bresac,10.95953701838715), (myst
eroid,10.95953701838715), (doralee,10.95953701838715), (bruns,10.95953701838715), (kwong,9.349957272040324), (guoxiang,10.95953701838715))

// calculate tf-idf: (token, ((id, name), tf-idf))
val tf_idf = tf.join(idf).map(x => (x._1, (x._2._1._1, x._2._1._2 * x._2._2)))
tf_idf.take(10)

tf_idf: org.apache.spark.rdd.RDD[(String, ((String, String), Double))] = MapPartitionsRDD[35] at map at command-3234923437497438:2
res14: Array[(String, ((String, String), Double))] = Array((reimi,((1974703,Burn Up!),10.95953701838715)), (glorifying,((5574594,Borat: Cultural Learnings of America for Make Benefit Glor
ious Nation of Kazakhstan),9.706655856361559)), (glorifying,((5443971,The Room),9.706655856361559)), (glorifying,((5729,Chariots of Fire),9.706655856361559)), (glorifying,((1017172,Red Ni
ghtmare),9.706655856361559)), (glorifying,((2983983,Paradise Now),9.706655856361559)), (glorifying,((32998223,A Tale of the Australian Bush),9.706655856361559)), (zootie,((573288,Babe: Pi
g in the City),43.8381480735486)), (nothin,((6004372,Grace of My Heart),9.573195381531523)), (nothin,((101899,Deliverance),9.573195381531523)))

// ############### single term search #################


val single_term = sc.parallelize(Array("COMEDY")).map(_.toLowerCase).first

single_term: String = comedy
```

```
val single_res = tf_idf.filter(x => x._1 == single_term)

single_res: org.apache.spark.rdd.RDD[(String, ((String, String), Double))] = MapPartitionsRDD[38] at filter at command-3234923437497441:1

val sorted_single_res = single_res.sortBy(-_._2._2).map(x => (x._2._1._2, x._2._2))

sorted_single_res: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[44] at map at command-3234923437497443:1

spark.createDataFrame(sorted_single_res.take(10)).toDF("movie_name", "tf-idf").show(false)

+-----------------------------------+------------------+
|movie_name                         |tf-idf            |
+-----------------------------------+------------------+
|Talkin' Dirty After Dark           |21.21626037281668 |
|General Motors 50th Anniversary Show|21.21626037281668 |
|Where the Truth Lies               |21.21626037281668 |
|Hollywood Outlaw Movie             |21.21626037281668 |
|Man on the Moon                    |21.21626037281668 |
|Mahaul Theek Hai                   |15.912195279612511|
|Cinta Kura Kura                    |15.912195279612511|
|Punchline                          |15.912195279612511|
|Funny Bones                        |15.912195279612511|
|When Stand Up Stood Out            |15.912195279612511|
+-----------------------------------+------------------+


// ############### multiple terms ################


val query_terms = "Action movie with scary scenes"

query_terms: String = Action movie with scary scenes

val multiple_terms = sc.parallelize(query_terms.split("""\W+""")).map(_.toLowerCase).filter(x => !stop_word_set.contains(x))

multiple_terms: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[47] at filter at command-3234923437497447:1

multiple_terms.collect()

res17: Array[String] = Array(action, movie, scary, scenes)

val reduced_terms = multiple_terms.map(x => (x, 1)).reduceByKey(_+_)

reduced_terms: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[49] at reduceByKey at command-3234923437497449:1

reduced_terms.take(10)

res18: Array[(String, Int)] = Array((scary,1), (movie,1), (action,1), (scenes,1))

val terms_cnt = multiple_terms.count()
val terms_tf = reduced_terms.map(x => (x._1, x._2/terms_cnt.toDouble))

terms_cnt: Long = 4
terms_tf: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[50] at map at command-3234923437497451:2
```

```
// tf-idf: (term, (tf, idf))
val terms_tf_idf = terms_tf.join(idf).map(x => (x._1, x._2._1 * x._2._2))

terms_tf_idf: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[54] at map at command-3234923437497452:2

terms_tf_idf.collect()

res19: Array[(String, Double)] = Array((scary,1.8301920735626456), (movie,0.7993061443340549), (action,1.1084968011212866), (scenes,1.1658904115324116))

val norm_terms_tfidf = Math.sqrt(terms_tf_idf.map(x => x._2 * x._2).sum())

norm_terms_tfidf: Double = 2.564480249154017

// (token, (term_tfidf, ((id, name), tf_idf)))
val doc_terms = terms_tf_idf.join(tf_idf)

doc_terms: org.apache.spark.rdd.RDD[(String, (Double, ((String, String), Double)))] = MapPartitionsRDD[58] at join at command-3234923437497455:2

// ((id, name), |norm|)
val norm_doc_terms = doc_terms.map(x => (x._2._2._1, x._2._2._2 * x._2._2._2)).reduceByKey(_+_).map(x => (x._1, Math.sqrt(x._2)))

norm_doc_terms: org.apache.spark.rdd.RDD[((String, String), Double)] = MapPartitionsRDD[61] at map at command-3234923437497457:2

// (token, (term_tfidf, ((id, name), tf_idf)))
val doc_terms_dot_product = doc_terms.map(x => (x._2._2._1, x._2._1 * x._2._2._2)).reduceByKey(_+_)

doc_terms_dot_product: org.apache.spark.rdd.RDD[((String, String), Double)] = ShuffledRDD[63] at reduceByKey at command-3234923437497459:2

// ((id, name), similarity)
val cosine_doc_term = doc_terms_dot_product.join(norm_doc_terms).map(x => (x._1, x._2._1 / (x._2._2 * norm_terms_tfidf)))

cosine_doc_term: org.apache.spark.rdd.RDD[((String, String), Double)] = MapPartitionsRDD[67] at map at command-3234923437497461:2

val multiple_term_result = cosine_doc_term.sortBy(-_._2).map(x => (x._1._2, x._2))

multiple_term_result: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[73] at map at command-3234923437497462:1

+--------------------------------+------------------+
|movie_name                      |similarity        |
+--------------------------------+------------------+
|The Victim                      |0.9017537721315425|
|Larry-Boy and the Rumor Weed    |0.8906802429059406|
|Grotesque                       |0.8906802429059406|
|Unstable Fables: 3 Pigs & a Baby|0.8461756732422155|
|Azumi                           |0.8343648479750548|
|The Hills Run Red               |0.7988786852773097|
|Whore                           |0.7988786852773097|
|Full Eclipse                    |0.7982617477798161|
|Alice in Wonderland             |0.7787625829843468|
|Scream                          |0.7787625829843468|
+--------------------------------+------------------+
```