ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΕΞΑΜΗΝΙΑΙΑ ΕΡΓΑΣΙΑ

Δ.Π.Μ.Σ. ΕΠΙΣΤΗΜΗ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ
ΑΚ. ΕΤΟΣ 2020 - 2021
ΔΙΑΧΕΙΡΙΣΗ ΔΕΔΟΜΕΝΩΝ ΜΕΓΑΛΗΣ ΚΛΙΜΑΚΑΣ

Χρήση του Apache Spark σε SQL ερωτήματα και Μηχανική Μάθηση
Ημερομηνία Παράδοσης: Κυριακή 25/07/2021

Στην παρούσα εργασία, θα γίνει χρήση του Apache Spark για τον υπολογισμό αναλυτικών ερωτημάτων πάνω σε αρχεία που περιγράφουν σύνολα δεδομένων (μέσω MapReduce και SparkSQL) καθώς και για την εκπαίδευση μοντέλων μηχανικής μάθησης. Το Apache Spark προσφέρει δύο βασικά APIs για την υλοποίηση αναλυτικών ερωτημάτων:

- RDD API
  - ✓ https://spark.apache.org/docs/2.4.4/rdd-programming-guide.html
- Dataframe API / Spark SQL
  - ✓ https://spark.apache.org/docs/2.4.4/sql-programming-guide.html

Μέρος 1ο: Υπολογισμός Αναλυτικών Ερωτημάτων με τα APIs του Apache Spark

Τα δεδομένα που θα χρησιμοποιήσετε είναι πραγματικά και αφορούν σε διαδρομές taxi στην Νέα Υόρκη. Οι δοθείσες διαδρομές των taxi έγιναν από τον Ιανουάριο εώς το Ιούνιο του 2015 και υπάρχουν διαθέσιμες online στο παρακάτω link:

https://data.cityofnewyork.us/Transportation/2015-Yellow-Taxi-Trip-Data/ba8s-jw6u.

Λόγω των περιορισμένων πόρων που η κάθε ομάδα έχει στη διάθεσή της, θα επεξεργαστούμε μόνο ένα υποσύνολο μεγέθους 2 GB. Τα δεδομένα αυτά περιέχουν 13 εκατομμύρια διαδρομές, που πραγματοποιήθηκαν το Μάρτιο του 2015 και μπορείτε να τα κατεβάσετε από εδώ:

http://www.cslab.ntua.gr/courses/atds/yellow_trip_data.zip.

Στο συμπιεσμένο αρχείο που σας δίνουμε, περιλαμβάνονται δύο comma-delimited αρχεία κειμένου (.csv) που ονομάζονται: yellow_tripdata_1m.csv και yellow_tripvendors_1m.csv. Το πρώτο αρχείο περιλαμβάνει όλη την απαραίτητη πληροφορία για μια διαδρομή, ενώ το δεύτερο πληροφορία για τις εταιρείες ταξί. Τα αρχεία έχουν την εξής μορφή:

- yellow_tripdata_1m.csv

```
369367789289,2015-03-27 18:29:39,2015-03-27 19:08:28,-73.975051879882813,40.760562896728516,-73.847900390625,40.732685089111328,34.8
369367789290,2015-03-27 18:29:40,2015-03-27 18:38:35,-73.988876342773438,40.77423095703125,-73.985160827636719,40.763439178466797,11.16
```

Το πρώτο πεδίο αποτελεί το μοναδικό id μιας διαδρομής. Το δεύτερο (τρίτο) πεδίο την ημερομηνία και ώρα έναρξης (λήξης) της διαδρομής. Το τέταρτο και πέμπτο πεδίο το γεωγραφικό μήκος και πλάτος του σημείου επιβίβασης, ενώ το έκτο και έβδομο πεδίο περιλαμβάνουν το γεωγραφικό μήκος και πλάτος του σημείου αποβίβασης. Τέλος, το όγδοο πεδίο δείχνει το συνολικό κόστος της διαδρομής.

- yellow_tripvendors_1m.csv

```
369367789289,1
369367789290,2
```

Το πρώτο πεδίο αποτελεί το μοναδικό id μιας διαδρομής και το δεύτερο πεδίο το μοναδικό αναγνωριστικό μιας εταιρείας taxi (vendor).

Το πρόβλημα που καλείστε να αντιμετωπίσετε είναι ο υπολογισμός των ερωτημάτων του Πίνακα 1 για την εξαγωγή πληροφορίας από τα δεδομένα με δύο διαφορετικούς τρόπους:

- Γράφοντας MapReduce κώδικα χρησιμοποιώντας το RDD API του Spark
- Χρησιμοποιώντας SparkSQL και το DataFrame API

Πίνακας 1: Τα ζητούμενα ερωτήματα για το πρώτο μέρος της άσκησης

<table>
<tr><td>ID</td><td>Query</td></tr>
<tr><td>Q1</td><td>Ποια είναι η μέση τιμή του γεωγραφικού μήκους και πλάτους επιβίβασης ανά ώρα έναρξης της διαδρομής; Ταξινομείστε το αποτέλεσμα με βάση την ώρα έναρξης σε αύξουσα σειρά και αγνοείστε dirty εγγραφές που προκαλούν προβληματικά αποτελέσματα (π.χ. Γεωγραφικό μήκος / πλάτος με μηδενική τιμή)
Ενδεικτικά αποτελέσματα:
<table><tr><td>HourOfDay</td><td>Latitude</td><td>Longitude</td></tr><tr><td>00</td><td>-73,...</td><td>40,...</td></tr><tr><td>01</td><td>-73,...</td><td>40,...</td></tr></table></td></tr>
<tr><td>Q2</td><td>Για κάθε vendor, θεωρώντας ως απόσταση την απόσταση Haversine, βρείτε τη μέγιστη απόσταση διαδρομής που αντιστοιχεί σε αυτόν, καθώς και τον χρόνο στον οποίο αυτή εκτελέστηκε.</td></tr>
</table>

Πιο συγκεκριμένα, θα πρέπει να κάνετε τα εξής:

Ζητούμενο 1 (0,5 μονάδες): Φορτώστε τα csv αρχεία που σας δίνονται στο HDFS.
Ζητούμενο 2 (0,5 μονάδες): Όπως αναφέρθηκε τα δεδομένα σας δίνονται σε μορφή απλού κειμένου (csv). Παρόλα αυτά, είναι γνωστό ότι ο υπολογισμός ερωτημάτων αναλυτικής επεξεργασίας απευθείας πάνω σε αρχεία csv δεν είναι αποδοτικός. Για να βελτιστοποιηθεί η πρόσβαση των δεδομένων, παραδοσιακά οι βάσεις δεδομένων φορτώνουν τα δεδομένα σε ειδικά σχεδιασμένα binary formats. Παρότι το Spark δεν είναι μια τυπική βάση δεδομένων, αλλά ένα σύστημα κατανεμημένης επεξεργασίας, για λόγους απόδοσης, υποστηρίζει κι αυτό μια παρόμοια λογική. Αντί να τρέξουμε τα ερωτήματά μας απευθείας πάνω στα csv αρχεία, μπορούμε να μετατρέψουμε πρώτα το dataset σε μια ειδική μορφή που:
- Έχει μικρότερο αποτύπωμα στη μνήμη και στον δίσκο και άρα βελτιστοποιεί το I/O (input/output) μειώνοντας τον χρόνο εκτέλεσης.
- Διατηρεί επιπλέον πληροφορία, όπως στατιστικά πάνω στο dataset, τα οποία βοηθούν στην πιο αποτελεσματική επεξεργασία του. Για παράδειγμα, αν ψάχνω σε ένα σύνολο δεδομένων τις τιμές που είναι μεγαλύτερες από 100 και σε κάθε block του dataset έχω πληροφορία για το ποια είναι η min και ποια η max τιμή, τότε μπορώ να παρακάμψω την επεξεργασία των blocks με max τιμή < 100 γλιτώνοντας έτσι χρόνο επεξεργασίας.

Το ειδικό format που χρησιμοποιούμε για να επιτύχουμε τα παραπάνω είναι το Apache Parquet. Όταν φορτώνουμε έναν πίνακα σε Parquet, αυτός μετατρέπεται κι αποθηκεύεται σε ένα columnar format που βελτιστοποιεί το I/O και τη χρήση της μνήμης κι έχει τα χαρακτηριστικά που αναφέραμε. Περισσότερες πληροφορίες σχετικά με το Parquet μπορείτε να βρείτε εδώ. Από άποψη κώδικα, η μετατροπή ενός dataset σε Parquet είναι ιδιαίτερα απλή. Παραδείγματα και πληροφορίες για το πώς διαβάζω και γράφω Parquet αρχεία μπορείτε να βρείτε εδώ.
Στο συγκεκριμένο ερώτημα, ζητείται να μετατρέψετε κάθε ένα csv που υπάρχει στο hdfs σε Parquet μορφή, διαβάζοντας κάθε CSV σε dataframe και αποθηκεύοντας το στη συνέχεια σε parquet μορφή πίσω στο hdfs (συμβουλευτείτε και τις παραπάνω οδηγίες). Τελικά θα πρέπει να υπάρχουν 4 αρχεία στο hdfs, 2 CSV και 2 parquet. Πόσος χρόνος χρειάστηκε για την μετατροπή των αρχείων;
Ζητούμενο 3 (4 μονάδες): Για κάθε ερώτημα του Πίνακα 1 υλοποιείστε μία λύση με το RDD API και μία με Spark SQL, η οποία θα μπορεί να διαβάζει είτε αρχεία CSV χρησιμοποιώντας το option inferSchema είτε αρχεία Parquet (1 μονάδα ανά υλοποίηση).
Ζητούμενο 4 (0,5 Μονάδες): Να εκτελεστούν οι υλοποιήσεις του ζητούμενου 3 για κάθε query. Συγκεκριμένα, θέλουμε τα αποτελέσματα και τους χρόνους εκτέλεσης από τις 3 ακόλουθες περιπτώσεις:

1. Map Reduce Queries – RDD API
2. Spark SQL με είσοδο το csv αρχείο (συμπεριλάβετε infer schema)
3. Spark SQL με είσοδο το parquet αρχείο

Δώστε τους χρόνους εκτέλεσης σε ένα ραβδόγραμμα, ομαδοποιημένους ανά Ερώτημα Σχολιάστε τα αποτελέσματα σε κάθε query. Τι παρατηρείται με τη χρήση του parquet ? Γιατί δεν χρησιμοποιείται το infer schema?
Ζητούμενο 5 (0,5 μονάδες): Το SparkSQL έχει υλοποιημένα και τα δύο είδη ερωτημάτων συνένωσης στο DataFrame API. Συγκεκριμένα, με βάση τη δομή των δεδομένων και των υπολογισμών που θέλουμε καθώς και τις ρυθμίσεις του χρήστη, πραγματοποιεί από μόνο του κάποιες βελτιστοποιήσεις στην εκτέλεση του ερωτήματος χρησιμοποιώντας έναν βελτιστοποιητή ερωτημάτων (query optimizer), κάτι που όλες οι βάσεις δεδομένων έχουν. Μια τέτοια βελτιστοποίηση είναι ότι επιλέγει αυτόματα την υλοποίηση που θα χρησιμοποιήσει για ένα ερώτημα join λαμβάνοντας υπόψη το μέγεθος των δεδομένων

και πολλές φορές αλλάζει και την σειρά ορισμένων τελεστών προσπαθώντας να μειώσει τον συνολικό χρόνο εκτέλεσης του ερωτήματος. Αν ο ένας πίνακας είναι αρκετά μικρός (με βάση ένα όριο που ρυθμίζει ο χρήστης) θα χρησιμοποιήσει το broadcast join, αλλιώς θα κάνει ένα repartition join. Περισσότερες πληροφορίες για τις ρυθμίσεις βελτιστοποίησης του SparkSQL υπάρχουν εδώ.

Χρησιμοποιώντας το παρακάτω script, συμπληρώστε τις <> ώστε να μπορείτε να απενεργοποιήσετε την επιλογή του join από το βελτιστοποιητή. Εκτελέστε το παρακάτω query για την συνένωση των πρώτων 100 γραμμών του πίνακα με τις εταιρείες ταξί με τον πίνακα των διαδρομών με και χωρίς βελτιστοποιητή και παρουσιάστε τα αποτελέσματα με την μορφή ενός ραβδογράμματος και το πλάνο εκτέλεσης που παράγει ο βελτιστοποιητής στην κάθε περίπτωση. Τι παρατηρείτε? Εξηγείστε.

```
from pyspark.sql import SparkSession
import sys
import time

disabled = sys.argv[1]

spark = SparkSession.builder.appName('query1-sql').getOrCreate()

if disabled == "Y":
        spark.conf.set(<Όνομα_Ιδιότητας>, <Τιμή Ιδιότητας>)
elif disabled == 'N':
        pass
else:
        raise Exception ("This setting is not available.")

df = spark.read.format("parquet")

df1 = df.load(<Path στο hdfs για tripdata>)
df2 = df.load(<Path στο hdfs για tripvendors>)

df1.registerTempTable("tripdata")
df2.registerTempTable("tripvendors")

sqlString = \
"SELECT * " + \
"FROM " + \
"       (SELECT * FROM tripvendors LIMIT 100) as v, " + \
"       tripdata as d " + \
"WHERE " + \
"       v._c0 = d._c0"

t1 = time.time()
spark.sql(sqlString).collect()
t2 = time.time()

spark.sql(sqlString).explain()

print("Time with choosing join type %s is %.4f sec."%("enabled" if disabled
== 'N' else "disabled", t2-t1))
```

Σημειώσεις-Υποδείξεις

- Κάθε γραμμή του αρχείου που διαβάζουμε με το RDD API φορτώνεται στη μνήμη ως string. Αφού εξάγουμε τις επιθυμητές στήλες από τη γραμμή, για να κάνουμε πράξεις με κάποιες στήλες θα πρέπει οι τιμές να μετατραπούν από string στον κατάλληλο τύπο πρώτα. Τις ημερομηνίες π.χ. μπορούμε να τις μετατρέψουμε κατάλληλα χρησιμοποιώντας τη μορφή
  '%Y-%m-%d %H:%M:%S'.
- Υπολογισμός απόστασης (Haversine[1]). Αν φ είναι το γεωγραφικό πλάτος και λ το γεωγραφικό μήκος, τότε η απόσταση δύο σημείων δίνεται από τους τύπους¨
  $a = \sin^2(\Delta\varphi/2) + \cos\varphi 1 \cdot \cos\varphi 2 \cdot \sin^2(\Delta\lambda/2)$
  $c = 2 \cdot \text{atan2}(\sqrt{a}, \sqrt{(1-a)})$
  $d = R \cdot c$, όπου R είναι η ακτίνα της Γης (6371km)

Μέρος 2ο: Machine Learning - Κατηγοριοποίηση κειμένων

Στο δεύτερο μέρος της εργασίας θα ασχοληθούμε με τη χρήση του Apache Spark για την κατηγοριοποίηση κειμένων. Για τις ανάγκες της εργασίας θα χρησιμοποιήσουμε ένα πραγματικό σύνολο δεδομένων, το οποίο περιγράφει παράπονα πελατών σε σχέση με οικονομικά προϊόντα και υπηρεσίες. Κάθε ένα παράπονο έχει επισημανθεί με το σε ποια γενική κατηγορία προϊόντος αναφέρεται και έτσι μπορεί να χρησιμοποιηθεί. Το παραπάνω σύνολο δεδομένων αφορά δεδομένα που έχουν συλλεχθεί από το 2011 μέχρι σήμερα και βρίσκονται διαθέσιμα εδώ:

https://catalog.data.gov/dataset/consumer-complaint-database

Για τις ανάγκες τις εργασίες, θα δουλέψουμε με ένα υποσύνολο των δεδομένων το οποίο βρίσκεται στην συγκεκριμένη τοποθεσία:

http://www.cslab.ntua.gr/courses/atds/customer_complains.tar.gz

Το συμπιεσμένο αρχείο που σας δίνουμε, περιλαμβάνει ένα comma-delimited αρχείο κειμένου (.csv) που ονομάζεται customer_complaints.csv περιλαμβάνει όλη την παραπάνω πληροφορία και έχει την εξής μορφή:

```
2019-09-24,Debt collection,transworld systems inc.  is trying to collect a
  debt that is not mine not owed and is inaccurate.
2019-09-19,Credit reporting credit repair services or other personal
  consumer reports,
```

Το πρώτο πεδίο αποτελεί την ημερομηνία που κατατέθηκε το σχόλιο, το δεύτερο την κατηγορία προϊόντος ή υπηρεσίας που αναφέρεται, ενώ το τρίτο είναι το σχόλιο.

[1] https://en.wikipedia.org/wiki/Haversine_formula

Για την εξαγωγή χαρακτηριστικών με στόχο την εκπαίδευση μοντέλων μηχανικής μάθησης, θα χρησιμοποιήσουμε την τεχνική TF-IDF. [2] Η μετρική είναι μια ένδειξη της σημαντικότητας μιας λέξης μέσα σε ένα κείμενο. Ο μαθηματικός τύπος για τον υπολογισμό της μετρικής είναι:

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D)$$

Όπου t, d, D ο όρος, το κείμενο και η συλλογή κειμένων στα οποία γίνεται ο υπολογισμός αντίστοιχα. Οι όροι tf, idf υπολογίζονται σύμφωνα με τους τύπους:

$$\mathrm{tf}(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

και εκφράζουν την συχνότητα εμφάνισης της λέξης t στο κείμενο d και τον λογάριθμο του πλήθους των κειμένων στο D προς τον αριθμό των κειμένων που περιέχουν τη λέξη t. Περισσότερες πληροφορίες για την σημασία των όρων μπορείτε να βρείτε στο λινκ που παρατίθεται, ενώ παρατίθεται και ένα παράδειγμα υπολογισμού του tfidf, το οποίο θα βρείτε στο παράρτημα στο τέλος της εκφώνησης.

Τα ζητούμενα της εργασίας είναι τα εξής:

Ζητούμενο 1 (0,5 μονάδες): Φορτώστε το csv στο hdfs.

Για τα ακόλουθα ζητούμενα χρησιμοποιείστε το RDD API:

Ζητούμενο 2 (0,5 μονάδες): Καθαρίστε τα δεδομένα! Δυστυχώς δεδομένα που προέρχονται από τον πραγματικό κόσμο, δεν είναι πάντα έτοιμα για χρήση. Στο συγκεκριμένο dataset μπορούμε να αποκτήσουμε ένα σχετικά καθαρό σύνολο δεδομένων κάνοντας τους ακόλουθους δύο μετασχηματισμούς:

- Κρατήστε μόνο τις γραμμές που ξεκινάνε από "201". Θα κρατήσετε έτσι μόνο τις έγκυρες γραμμές που έχουν στην αρχή ημερομηνίες όπως προδιαγράφεται στο σύνολο. **Υπόδειξη:** *Χρησιμοποιείστε την* startswith *της python σε περιβάλλον MapReduce.*
- Βρείτε ποιες γραμμές έχουν σχόλιο του χρήστη και ετικέτα. Μπορείτε να εξετάσετε πότε η δεύτερη και η τρίτη στήλη του πίνακα έχει τιμή και δεν είναι κενό string.

Ζητούμενο 3 (1 μονάδα): Στο βήμα αυτό πρέπει να έχετε ένα καθαρό dataset έτοιμο για εξαγωγή χαρακτηριστικών. Για τον υπολογισμό των TFIDF για κάθε λέξη κάθε κειμένου, βρείτε αρχικά όλες τις διαφορετικές λέξεις που υπάρχουν στα σχόλια των χρηστών, αφού έχετε κρατήσει μόνο χαρακτήρες της αλφαβήτου και το κενό ώστε να χωρίσετε μετά τις λέξεις. Αφαιρέστε stopwords ή λέξεις που δεν έχουν σημασία όπως άρθρα κλπ. Από αυτές κρατείστε ως λεξικό τις Κ πιο συχνές (διαλέξτε την τιμή του Κ που στα επόμενα βήματα θα δίνει το καλύτερο δυνατό accuracy στο test set)

Υποδείξεις: *Για την διατήρηση μόνο χαρακτήρων της αλφαβήτου και των κενών μπορείτε να χρησιμοποιήσετε την βιβλιοθήκη re της python σε περιβάλλον MapReduce. Για την αφαίρεση των stopwords και των άρθρων διερευνήστε την βιβλιοθήκη NLTK της python ή οποιαδήποτε άλλη της επιλογής σας.*

[2] https://en.wikipedia.org/wiki/Tf%E2%80%93idf

Ζητούμενο 4 (1 μονάδα): Υπολογίστε την ζητούμενη μετρική με χρήση MapReduce. Κάθε γραμμή της εξόδου θα περιγράφει ένα κείμενο. Φροντίστε η κάθε γραμμή της εξόδου σας να έχει ένα στοιχείο με την ακόλουθη μορφή:

(K, (ind1, ind2, …, indM), (tfidf1, tfidf2, tfidfM))

Όπου K το μέγεθος του λεξικού, ind<i> η θέση της i-οστής λέξης του κειμένου στη λίστα που βρήκατε στο βήμα 3, και tfidf<i> η τιμή της μετρικής για την λέξη αυτή στο δεδομένο κείμενο. Η παραπάνω μορφή παριστάνει ένα SparseVector στο Spark. Το RDD πρέπει να περιέχει επιπλέον ως στοιχείο κάθε γραμμή να έχει την πληροφορία του προϊόντος στο οποίο αναφέρεται το παράπονο. Τελικά θα έχετε ένα RDD το οποίο θα έχει δύο στοιχεία, την ετικέτα που περιγράφει το παράπονο και το αντίστοιχο SparseVector. **Απο 5 κείμενα της επιλογής σας, εισάγετε την παραπάνω έξοδο στην αναφορά.**
Μετατρέψτε το παραπάνω RDD σε Spark Dataframe, για να χρησιμοποιηθεί έπειτα στην εκπαίδευση μοντέλου και αποθηκεύστε το στο hdfs σε μορφή parquet.

Στα υπόλοιπα βήματα θα χρησιμοποιήσουμε το Dataframe API.
Ζητούμενο 5 (0,5 μονάδες): Χωρίστε τα δεδομένα σε train και test set για να χρησιμοποιηθούν για την εκπαίδευση ενός μοντέλου. Κάντε stratified split ώστε κάθε set να έχει στοιχεία από κάθε κατηγορία. **Στην αναφορά να δώσετε πόσες γραμμές έχει το κάθε set, καθώς και το πλήθος των γραμμών της κάθε κατηγορίας για το κάθε κομμάτι.**
Υπόδειξη: *Θα σας φανούν χρήσιμες οι συναρτήσεις* sampleby *και* subtract *του Dataframe API.*
Ζητούμενο 6 (0.5 μονάδες): Χρησιμοποιείστε τα παραπάνω δεδομένα για την εκπαίδευση ενός μοντέλου Perceptron. Πειραματιστείτε με τη δομή του δικτύου και χρησιμοποιείται αυτό με τον καλύτερο accuracy στο test set για να δώσετε αποτελέσματα. Για την αναφορά ζητούνται τα εξής:

- Κάντε train ένα μοντέλο αρχικά χωρίς να κάνετε cache το trainset και έπειτα επαναλάβετε τη διαδικασία χρησιμοποιώντας τον μηχανισμό cache. Δώστε ένα barchart με τους χρόνους. Τι παρατηρείται? Είναι χρήσιμο το caching και γιατί?
- Αναφέρετε το accuracy στο testset.

Υπόδειξη: Εάν επιθυμείτε να κατασκευάσετε μοντέλα μεγαλύτερης ακρίβειας, παρότι δεν είναι απαραίτητο στα πλαίσια της συγκεκριμένης εργασίας (θα θεωρούμε αποδεκτό ένα οποιοδήποτε μοντέλο έχει accuracy στο test set >= 50%), μπορείτε να επιχειρήσετε περαιτέρω καθαρισμό του συνόλου δεδομένων. Συγκεκριμένα μπορείτε να επιχειρήσετε να αφαιρέσετε κλάσεις στα αρχικά βήματα που εισάγουν θόρυβο στο μοντέλο, πχ κλάσης που έχουν υπερβολικά λίγα παραδείγματα παραπόνων. Επιπλέον, μπορείτε να μετατρέψετε όλες τις λέξεις ώστε να έχουν μόνο μικρά γράμματα του αλφαβήτου ή οποιονδήποτε άλλο μηχανισμό σκεφτείτε/βρείτε από τη βιβλιογραφία.

Παραδοτέα

- Η παράδοση θα γίνει στο mycourses site.
- Η παράδοση θα αποτελείται από ένα zip με τα εξής:
  - Μια σύντομη αναφορά όπου θα περιγράφετε την μεθοδολογία που ακολουθήσατε (όχι κώδικας εδώ) και διαγράμματα ή σχολιασμούς που έχουν ζητηθεί στην εκφώνηση. Επιπλέον, να περιέχετε ψευδοκώδικας για τα προγράμματα Map/Reduce που χρησιμοποιήσατε για την υλοποίηση των 2 queries στο πρώτο μέρος και για τον υπολογισμό του tfidf στο δεύτερο.

- Ένα αρχείο με όνομα ip, με την public ip του master
- Ένα zip file με όνομα code με τον κώδικα που γράψατε.
- Ένα zip file με όνομα results με τα τελικά αποτελέσματα.
- Ένα zip file με όνομα logs με τα log-files από την εκτέλεση των εργασιών MapReduce από τις οποίες βγήκαν τα αποτελέσματα.

Για οποιαδήποτε απορία σχετικά με την εργασία επικοινωνήστε μαζί μου στο email:

nprov@cslab.ece.ntua.gr

Παράρτημα: Παράδειγμα υπολογισμού tfidf

Το Tf-Idf ορίζεται ως το γινόμενο το γινόμενο των μετρικών tf, idf οι οποίες ορίζονται ως εξής:
TF → Λόγος των εμφανίσεων μιας λέξης σε ένα κείμενο προς το συνολικό πλήθος λέξεων στο κείμενο.
IDF → Λογάριθμος του λόγου του πλήθους των κειμένων προς το πλήθος των κειμένων που περιέχουν τη λέξη.
Για καλύτερη κατανόηση της μετρικής που πρέπει να υπολογίσετε ως σύνολο χαρακτηριστικών για την εκπαίδευση του μοντέλου, δίνεται ένα παράδειγμα υπολογισμού, θεωρώντας ως σύνολο κειμένων τρεις προτάσεις, κάθε μία από τις οποίες αντιστοιχεί σε ένα κείμενο.
Κείμενο 1: Θα βρέξει σήμερα. (3 λέξεις)
Κείμενο 2: Σήμερα δεν θα βγω έξω. (5 λέξεις)
Κείμενο 3: Θα πάω έξω αν δεν θα βρέξει. (7 λέξεις)
Σύνολο λέξεων: [θα, βρέξει, σήμερα, δεν, βγω, έξω, πάω, αν] (8 λέξεις – θέσεις 0 εώς 7 αντίστοιχα)

Πίνακας υπολογισμού TF:

	Κείμενο 1		Κείμενο 2		Κείμενο 3	
Λέξη	#Εμφανίσεων	TF	#Εμφανίσεων	TF	#Εμφανίσεων	TF
θα	1	1/3 = 0,33	1	1/5 = 0,2	2	2/7 = 0,29
βρέξει	1	1/3 = 0,33	0	0	1	1/7 = 0,14
σήμερα	1	1/3 = 0,33	1	1/5 = 0,2	0	0
δεν	0	0	1	1/5 = 0,2	1	1/7 = 0,14
βγω	0	0	1	1/5 = 0,2	0	0
έξω	0	0	1	1/5 = 0,2	1	1/7 = 0,14
πάω	0	0	0	0	1	1/7 = 0,14
αν	0	0	0	0	1	1/7 = 0,14

Πίνακας Υπολογισμού IDF

Λέξη	Πλήθος Κειμένων που περιέχουν τη λέξη	Λόγος #κειμένων προς #κειμένων που περιέχουν τη λέξη	IDF
θα	3	3/3 = 1	0
βρέξει	2	3/2 = 1.5	0.176
σήμερα	2	3/2 = 1.5	0.176
δεν	2	3/2 = 1.5	0.176
βγω	1	3/1 = 3	0.477
έξω	2	3/2 = 1.5	0.176
πάω	1	3/1 = 3	0.477
αν	1	3/1 = 3	0.477

Πίνακας TF-IDF για κάθε λέξη σε κάθε πρόταση

Index	Λέξη	IDF	Πρόταση 1		Πρόταση 2		Πρόταση 3	
			TF	TFIDF	TF	TFIDF	TF	TFIDF
0	θα	0	0,33	0	0,2	0	2/7 = 0,29	0
1	βρέξει	0.176	0,33	0.058	0	0	1/7 = 0,14	0,025
2	σήμερα	0.176	0,33	0.058	0,2	0,035	0	0
3	δεν	0.176	0	0	0,2	0,035	1/7 = 0,14	0,025
4	βγω	0.477	0	0	0,2	0,095	0	0
5	έξω	0.176	0	0	0,2	0,035	1/7 = 0,14	0,025
6	πάω	0.477	0	0	0	0	1/7 = 0,14	0,067
7	αν	0.477	0	0	0	0	1/7 = 0,14	0,067

Η αναπαράσταση αραιού διανύσματος για ένα κείμενο χρησιμοποιώντας τις μετρικές tfidf ορίζεται ως:

(N, (index1, index2, …, indexK), (tfidf1, tfidf2, …, tfidfK))

όπου:

- N : το μέγεθος του διανύσματος που περιγράφει ένα κείμενο ως προς το λεξικό, και επομένως το μέγεθος του λεξικού
- indexi : η θέση της i-οστής λέξης του κειμένου στη λίστα που είναι το λεξικό
- tfidfi : Η τιμή της μετρικής tfidf για την i-οστή λέξη στο δεδομένο κείμενο.

Θεωρώντας κάθε πρόταση ως αναπαράσταση αραιού διανύσματος με τις tf idf τιμές, όπως ζητείται και στην εκφώνηση της άσκησης, τα 3 κείμενα μπορούν να παρασταθούν ως:

Κείμενο 1: Θα βρέξει σήμερα. (3 λέξεις)

- [θα, βρέξει, σήμερα]
- (8, (0, 1, 2), (0, 0.058, 0.058))
- ή αγνοώντας όπου έχει tfidf 0
- (8, (1, 2), (0.058, 0.058))

Κείμενο 2: Σήμερα δεν θα βγω έξω. (5 λέξεις)

- [θα, σήμερα, δεν, βγω, έξω]
- (8, (0, 2,3,4,5), (0, 0.035, 0.035, 0.095, 0.035))
- ή αγνοώντας όπου έχει tfidf 0
- (8, (2,3,4,5), (0.035, 0.035, 0.095, 0.035))

Κείμενο 3: Θα πάω έξω αν δεν θα βρέξει. (7 λέξεις)

- [θα, βρέξει, δεν, έξω, πάω, αν]
- (8, (0, 1, 3, 5, 6, 7), (0, 0.025, 0.025, 0.025, 0.067, 0.067))
- ή αγνοώντας όπου έχει tfidf 0
- (8, (1, 3, 5, 6, 7), (0.025, 0.025, 0.025, 0.067, 0.067))