ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΔΙΑΧΕΙΡΙΣΗ ΔΕΔΟΜΕΝΩΝ ΜΕΓΑΛΗΣ ΚΛΙΜΑΚΑΣ

Ακ. έτος 2019-2020

Εξαμηνιαία Εργασία

Η εργασία να γίνει σε ομάδες το **πολύ των 2** ατόμων.

Για την εργασία θα χρησιμοποιηθεί το **Apache Spark** για την επεξεργασία δεδομένων. Η εργασία αφορά τόσο την εξαγωγή πληροφορίας από δεδομένα (MapReduce, SparkSQL) όσο και την χρήση αλγορίθμων μηχανικής μάθησης.

Μέρος 1ο : Εξαγωγή Πληροφορίας - SQL

Περιγραφή Δεδομένων

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. Το αρχείο των TripData έχει 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.

Το δεύτερο αρχείο που σας δίνεται περιέχει πληροφορία για τις εταιρίες taxi. Η μορφή του φαίνεται στο παρακάτω παράδειγμα:

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).

1Α. Εξαγωγή πληροφορίας με διαφορετικούς τρόπους

Παρόλο που τα δεδομένα δίνονται σε μορφή απλού κειμένου (csv), 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.
Παρ'ότι 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: https://parquet.apache.org/.
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:
https://spark.apache.org/docs/latest/sql-data-sources-parquet.html

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

Πιο συγκεκριμένα, θα πρέπει να κάνετε τα εξής:

1. Φορτώστε τα csv αρχεία που σας δίνονται στο HDFS.
2. Υλοποιήστε και τρέξτε τα ερωτήματα του Πίνακα 1 με χρήση:
    a. MapReduce κώδικα. Η υλοποίηση θα πρέπει να τρέξει απευθείας πάνω στα αρχεία κειμένου.
    b. SparkSQL. Φορτώστε τα αρχεία κειμένου σε Dataframes και εκτελέστε τα ερωτήματα με χρήση της SparkSQL.
3. Μετατρέψτε τα αρχεία κειμένου σε αρχεία Parquet. Στη συνέχεια φορτώστε τα Parquet αρχεία ως Dataframes και εκτελέστε το υποερώτημα 2b. Πόσος χρόνος χρειάζεται για τη μετατροπή των αρχείων;
4. Για κάθε ερώτημα του Πίνακα 1, συγκρίνετε και φτιάξτε ένα διάγραμμα με τον χρόνο εκτέλεσης των παραπάνω περιπτώσεων, δηλαδή:
    a. MR πάνω σε csv αρχεία.
    b. SQL πάνω σε csv αρχεία.
    c. SQL πάνω σε Parquet αρχεία. (Διευκρίνιση: Τα SQL ερωτήματα θα πρέπει να τρέξουν σε Dataframes που έχουν δημιουργηθεί είτε από αρχεία κειμένου είτε από αρχεία Parquet).

    Σχολιάστε τα αποτελέσματά σας.

Πίνακας 1

<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>

Σημειώσεις-Υποδείξεις

1. 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'.
2. Για το Q2, η ταχύτητα μιας κούρσας ορίζεται ως η απόσταση που διανύθηκε προς τον χρόνο που χρειάστηκε.
3. 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 (6371m)

1Β. Μελέτη του βελτιστοποιητή για την συνένωση δεδομένων.

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: https://spark.apache.org/docs/latest/sql-performance-tuning.html.

Για την μελέτη της επίδρασης του βελτιστοποιητή στην εκτέλεση των ερωτημάτων συνένωσης, στη συνέχεια της εργασίας καλείστε να εκτελέσετε τα παρακάτω:

1. Απομονώστε τις 50 πρώτες γραμμές από το αρχείο με τις εταιρείες ταξί και εκτελέσετε με SparkSQL ένα join πάνω στα 2 parquet αρχεία και πάρτε πίσω όλα τα δεδομένα. Εντοπίστε ποια υλοποίηση join χρησιμοποίησε το Spark. Γιατί επιλέχθηκε η συγκεκριμένη υλοποίηση;
   Υπόδειξη: Μπορείτε να χρησιμοποιήσετε το 'explain' statement της SQL για να δείτε τις λεπτομέρειες του πλάνου εκτέλεσης. Αιτιολογήστε την απάντησή σας με βάση το πλάνο εκτέλεσης του ερωτήματος και τις προκαθορισμένες ρυθμίσεις του Spark. Για να απομονώσετε τις 50 εγγραφές μπορείτε να φτιάξετε ένα εμβόλιμο ερώτημα στο ερώτημα συνένωσης, το οποίο θα απομονώνει τις πρώτες 100 εγγραφές (limit).

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

2. Ρυθμίστε κατάλληλα το Spark χρησιμοποιώντας τις ρυθμίσεις του βελτιστοποιητή ώστε να μην επιλέγει την υλοποίηση join του προηγούμενου ερωτήματος. Σε πόσο χρόνο εκτελείται τώρα το ερώτημα και ποια υλοποίηση join χρησιμοποίησε το Spark αυτή τη φορά; Συγκρίνετε τον χρόνο εκτέλεσης των δύο υποερωτημάτων.

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.

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:

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

$$\mathrm{tf}(t,d) = 0.5 + 0.5 \cdot \frac{f_{t,d}}{\max\{f_{t',d} : t' \in d\}}$$

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

Περισσότερες πληροφορίες για την σημασία των όρων μπορείτε να βρείτε στο λινκ που παρατίθεται. Τα ζητούμενα της εργασίας είναι τα εξής:

1. Φορτώστε το csv στο hdfs.

Για τα ακόλουθα βήματα χρησιμοποιείστε το RDD API:

2. Καθαρίστε τα δεδομένα! Δυστυχώς δεδομένα που προέρχονται από τον πραγματικό κόσμο, δεν είναι πάντα έτοιμα για χρήση. Στο συγκεκριμένο dataset μπορούμε να αποκτήσουμε ένα καθαρό σύνολο δεδομένων κάνοντας τους ακόλουθους δύο μετασχηματισμούς:
   a. 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.
   b. Βρείτε ποιες γραμμές έχουν σχόλιο του χρήστη. Μπορείτε να εξετάσετε πότε η τρίτη στήλη του πίνακα έχει τιμή και δεν είναι κενό string.
3. 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.
   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.
4. Υπολογίστε την ζητούμενη μετρική με χρήση MapReduce. Κάθε γραμμη της εξόδου θα περιγράφει ένα κείμενο. Φροντίστε η κάθε γραμμή της εξόδου σας να έχει ένα στοιχείο με την ακόλουθη μορφή:

   (N, (ind1, ind2, …, indK), (tfidf1, tfidf2, tfidfK))

   Όπου N το πλήθος των αρχείων, ind<i> η θέση της συγκεκριμένης λέξης του κειμένου στη λίστα που βρήκατε στο βήμα 3, και tfidf<i> η τιμή της μετρικής για την λέξη αυτή στο δεδομένο κείμενο.
   Το RDD πρέπει να περιέχει επιπλέον ως στοιχείο κάθε γραμμή να έχει την πληροφορία του προϊόντος στο οποίο αναφέρεται το παράπονο.
   Απο 5 κείμενα της επιλογής σας, εισάγετε την παραπάνω έξοδο στην αναφορά.

5. Μετατρέψτε το παραπάνω RDD σε Spark Dataframe, για να χρησιμοποιηθεί σε εκπαίδευση μοντέλου, σύμφωνα με τα όσα δείξαμε στο εργαστήριο, στην κατάλληλη μορφή για εκπαίδευση ενός μοντέλου.

Στα υπόλοιπα βήματα θα χρησιμοποιήσουμε το Dataframe API.

6. 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 αναγράφεται 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.
7. Χρησιμοποιείστε τα παραπάνω δεδομένα για την εκπαίδευση ενός μοντέλου Perceptron. Πειραματιστείτε με τη δομή του δικτύου και χρησιμοποιείται αυτό με τον καλύτερο accuracy για να δώσετε αποτελέσματα. Για την αναφορά ζητούνται τα εξής:
    a. 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?
    b. Αναφέρετε το accuracy στο testset.

Παραδοτέα

- Η παράδοση θα γίνει στο mycourses site.
- Η παράδοση θα αποτελείται από:
    - Μια σύντομη αναφορά όπου θα περιγράφετε την μεθοδολογία που ακολουθήσατε (όχι κώδικας εδώ) και διαγράμματα ή σχολιασμούς που έχουν ζητηθεί στην εκφώνηση.
    - Ψευδοκώδικας για τα προγράμματα Map/Reduce που χρησιμοποιήσατε για κάθε κομμάτι της άσκησης. Ο ψευδοκώδικας θα δείχνει εποπτικά τα key/values που παίρνει η συνάρτηση map, την επεξεργασία που τους κάνει, τα key/values που κάνει emit στην συνάρτηση reduce, και την επεξεργασία που κάνει η reduce (σαν τον ψευδοκώδικα του wordcount).
    - Link στο hdfs site όπου έχετε βάλει τα datasets.
    - Ένα zip file με τον κώδικα.
    - Ένα zip file με τα τελικά αποτελέσματα.
    - Ένα zip file με τα log-files των εργασιών MapReduce από τις οποίες βγήκαν τα αποτελέσματα.