Πρώτο παραδοτέο εξαμηνιαίας εργασίας στον Αντικειμενοστραφή Προγραμματισμό II (Deadline 9 Νοεμβρίου)

Υπηρεσία Συστάσεων Πόλεων με χρήση Open Data

Στην παρούσα εργασία θα αναπτύξουμε μια υπηρεσία που θα παράγει συστάσεις πόλεων με βάση τα γενικά κριτήρια που ενδιαφέρουν υποψήφιους ταξιδιώτες και θα χρησιμοποιεί ανοιχτά δεδομένα, δεδομένα που είναι διαθέσιμα στην Wikipedia και στο OpenWeatherMap.

Use case:
Η υπηρεσία συστάσεων πόλεων εξετάζει περισσότερες από δέκα πόλεις και έχει τρεις κατηγορίες με υποψήφιους ταξιδιώτες Young Traveller (ηλικία 16 έως 25), Middle Traveller (ηλικία 25 έως 60) και Elder Traveller (ηλικία 60 έως 115). Σε κάθε μια από τις τρεις διαφορετικές κατηγορίες υποψηφίων ταξιδιωτών θα προτείνει τις πόλεις που αποφασίζουν οι τρεις αντίστοιχοι Perceptron[1] [1].

Μπορείτε να ξεκινήσετε και να δοκιμάσετε τα ερωτήματα 1 εως 3 χρησιμοποιώντας dummy και hardcode data. Στο ερώτημα 4 θα γίνει η σύνδεση με τα Open Data. Το δεύτερο μισό της 3ης διάλεξης θα είναι εργαστηριακό για να επιλύσουμε απορίες που θα προκύψουν.

Ερώτημα 1. (lecture 1)
Ορίστε μια κλάση την City, που θα περιέχει ένα vector representation ως ένα **array** με 10 features (κριτήρια) της πόλης. Τα πρώτα 7 features θα αντιστοιχούν σε 7 terms (cafe, sea, museums, restaurant, stadium, etc) από το άρθρο της πόλης στην Wikipedia. Το 8ο έως και το 10ο features θα περιέχουν, την Kelvin θερμοκρασία (temp), το ποσοστό συννεφιάς (clouds:all) και την γεωδαισιακή απόσταση (coord:lat,lon) μεταξύ της πόλης και του τουριστικού γραφείου. Τα features υπολογίζονται με βάση τις ακόλουθες εξισώσεις και τα δεδομένα που ανακτώνται από τα Open Data.

Τα features από την wikipedia θα πρέπει να είναι κανονικοποιημένα με τον min-max scaler [2].

$$normalized\_feature(\#term_i) = \frac{\#term_i - min(\#term_i)}{max(\#term_i) - min(\#term_i)}$$

Όπου $\#term_i$ το πόσες φορές βρέθηκε το $term_i$ στο άρθρο της wikipedia με μέγιστο το 10, $min(\#term_i)$ θεωρήστε το ίσο με το μηδέν και $max(\#term_i)$ ίσο με δέκα.

[1] Στο μάθημα θα ασχοληθούμε μόνο με το inference stage του Perceptron.

Η κανονικοποιημένη θερμοκρασία επίσης, θα υπολογιστεί με τον min-max scaler όπου ενδεικτικά $max(temp)$ = 331 και $min(temp)$ = 184.

Το OpenWeatherMap δίνει το ποσοστό συννεφιάς εκφρασμένο ως επί τις 100 οπότε θα πρέπει απλά να διαιρεθεί με το 100.

Το geodesic distance μεταξύ της Αθήνας και της υποψήφιας πόλης υπολογίζεται με βάση το latitude, το longitude και χρησιμοποιώντας το GeoDataSource [3] [4].

$$normalized\_geodesic\_distance\ (Athens,\ city) = \frac{geodesic\_distance\ (Athens,city)}{maxdist}$$

Ως $maxdist$ θεωρήστε την μεγαλύτερη απόσταση μεταξύ δύο πόλεων που μπορεί να γίνει ένα ταξίδι (πχ Αθήνα Σίντνεϊ).

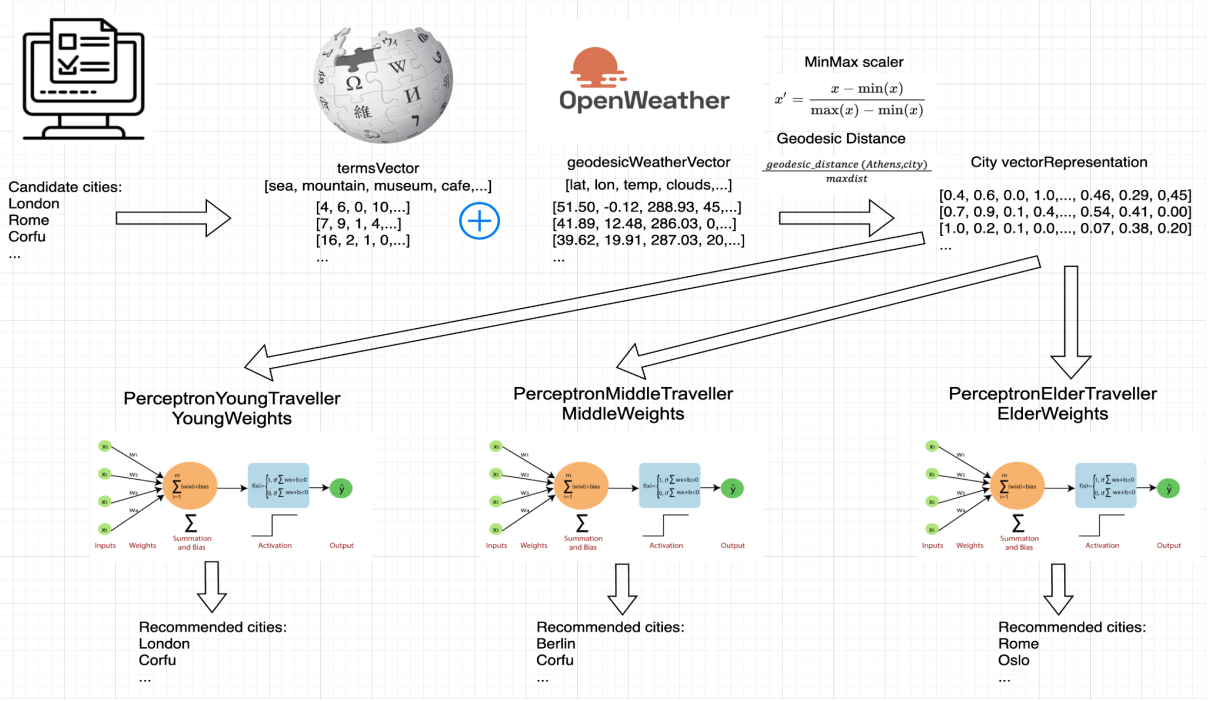
Ερώτημα 2. (lecture 2)
Ορίστε το **Interface** PerceptronTraveller που θα περιέχει μία μέθοδο recommend() που κάνει return ένα arraylist με τις πόλεις που επιλέγει ο perceptron.

Θα κάνετε implements το interface PerceptronTraveller στις τρεις concrete κλάσεις PerceptronYoungTraveller, PerceptronMiddleTraveller, και PerceptronElderTraveller. Θα ορίσετε hardcoded το διάνυσμα weightsBias και θα υλοποιήσετε την Heaviside step [5] activation function μέσα στην συνάρτηση recommend(). Τα weights θα τα επιλέξετε για κάθε concrete κλάση εσείς, θα είναι διαφορετικά και κάθε weight θα είναι ένας πραγματικός αριθμός στο range [-1,1]

Ερώτημα 3. (lecture 1)
Οι κλάσεις τύπου PerceptronTraveller και City θα περιέχουν **constructors**. Εφαρμόστε την προγραμματιστική τεχνική του **Encapsulation**, υλοποιώντας **setters** και **getters** και το σωστό **visibility** στις μεταβλητές και τις μεθόδους.

Ερώτημα 4. (lecture 2 & 3)
Εφαρμόστε **υπερφόρτωση** στη συνάρτηση recommend() υλοποιήστε μια δεύτερη recommend() σε κάθε concrete κλάση του Traveller που θα δέχεται ως argument ένα Boolean όπου αν είναι true θα επιστρέφει ένα ArrayList με τα ονόματα των city uppercased ενώ αν είναι false θα επιστρέφει τα ονόματα των city lowercased.

Εφαρμόστε την τεχνική του **πολυμορφισμού** και δημιουργήστε μια συνάρτηση που δέχεται ως όρισμα έναν PerceptronTraveller και επιστρέφει την πόλη που βρίσκεται πιο κοντά στο τουριστικό γραφείο.



Ερώτημα 5. (lecture 4)
Φτιάξτε μια μέθοδο στην City όπου καλεί το MediaWiki API της Wikipedia και του OpenWeatherMap API και εκχωρεί δεδομένα μέσω των setters στα αντίστοιχα πεδία του object City. Η τεχνική που θα εφαρμόσουμε για να καλέσουμε τα APIs και να ανακτήσουμε τα JSON files μέσα από την Java εφαρμογή μας θα παρουσιαστεί στις διαλέξεις και υπάρχει στο GitHub [6] [7].

Η τεχνική με την οποία από το String που έχετε από το άρθρο της Wikipedia θα υπολογίσετε τους όρους για κάθε termsVector υπάρχει επίσης στο GitHub [8]. Όλα τα υπόλοιπα δεδομένα για την δημιουργία των αντικειμένων PerceptronTraveller και City μπορείτε να τα έχετε hard coded ή να τα περνάτε από το πληκτρολόγιο (πχ scanner)

Υπάρχουν πολλά σημεία στην παραπάνω διαδικασία που κάτι μπορεί να μην πάει όπως αναμένουμε. Δημιουργήστε έναν νέο τύπο εξαίρεσης κάντε **throw** τις **εξαιρέσεις** εκεί που αρμόζει και διαχειριστείτε τες. Παράδειγμα, αν ο χρήστης αναφέρει μια πόλη για την οποία το MediaWiki API δεν επιστρέφει κάποιο λήμμα.

[1] https://en.wikipedia.org/wiki/Perceptron
[2] https://en.wikipedia.org/wiki/Feature_scaling#Rescaling_(min-max_normalization)
[3] https://www.geodatasource.com/developers/java
[4]https://stackoverflow.com/questions/3694380/calculating-distance-between-two-points-using-latitude-longitude
[5] https://en.wikipedia.org/wiki/Heaviside_step_function
[6] https://github.com/johnviolos/OpenData
[7] https://github.com/johnviolos/OpenDataREST
[8]https://github.com/johnviolos/OOPII/blob/master/src/gr/hua/dit/oopii/exercise/CountWords.java