ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ
Τμήμα Πληροφορικής

ΕΠΛ 133 – Αντικειμενοστρεφής Προγραμματισμός
ΑΣΚΗΣΗ 4 – Automatically Solving TOEFL Synonym Questions with Computational Linguistics

Διδάσκων: Καθ. Μάριος Δικαιάκος

Υπεύθυνος Άσκησης: Πύρρος Μπράτσκας

Ημερομηνία Ανάθεσης: 03 Απριλίου 2023
Ημερομηνία Παράδοσης: 27 Απριλίου 2018 13:00

(ο κώδικας να υποβληθεί στο Moodle)
http://www.cs.ucy.ac.cy/courses/EPL133

I. Στόχος

Σε αυτήν την εργασία θα ασχοληθούμε με αντικειμενοστραφή σχεδίαση προβλημάτων, σχεδίαση διεπαφών και αφαιρετικών κλάσεων, κληρονομικότητα και πολυμορφισμό, χρησιμοποίηση βιβλιοθηκών Java API, collections. Η άσκηση αποτελείται από πολλαπλά μέρη. Υλοποιήστε όλα τα μέρη σε αρχεία Java, όπου κάθε κλάση θα αντιπροσωπεύει ένα ξεχωριστό αντικείμενο.

II. Περιγραφή

Ένας είδος ερώτησης που αντιμετωπίζουν οι εξεταζόμενοι στα τεστ TOEFL είναι το "ερώτημα συνώνυμο" (Synonym Question), όπου οι μαθητές καλούνται να επιλέξουν το συνώνυμο μιας λέξης από μια λίστα επιλογών. Για παράδειγμα:

1. vexed
   a. annoyed
   b. amused
   c. frightened
   d. excited
   (Answer: (a) annoyed)

Σε αυτή την εργασία πρέπει να σχεδιάσετε ένα ευφυές σύστημα που μαθαίνει να απαντά σε τέτοιες ερωτήσεις. Η εκμάθηση/εκπαίδευση του συστήματος θα γίνει μέσω της επεξεργασίας ενός μεγάλου σώματος αγγλικού κειμένου (δηλαδή χωρίς τη χρήση λεξικών).
Για να γίνει αυτό, το σύστημα θα χρησιμοποιήσει τη σημασιολογική ομοιότητα οποιουδήποτε ζευγαριού λέξεων. Η σημασιολογική ομοιότητα μεταξύ δύο λέξεων είναι το μέτρο της εγγύτητας των εννοιών τους. Για παράδειγμα, η σημασιολογική ομοιότητα μεταξύ "αυτοκινήτου" και "οχήματος" είναι υψηλή, ενώ η σημασιολογική ομοιότητα μεταξύ "αυτοκινήτου" και "λουλουδιού" είναι χαμηλή.
Για να απαντήσει το σύστημα στην ερώτηση TOEFL, θα πρέπει να υπολογίσει τη σημασιολογική ομοιότητα μεταξύ της λέξης που σας δίνεται και όλων των πιθανών απαντήσεων και να επιλέξει την απάντηση με την υψηλότερη σημασιολογική ομοιότητα με την δεδομένη λέξη. Πιο συγκεκριμένα, δεδομένης μιας λέξης w και μιας λίστας δυνητικών συνωνύμων s_1, s_2, s_3, s_4, το σύστημα πρέπει να υπολογίζει τις ομοιότητες των ζευγαριών (w,s1),(w,s2),(w,s3),(w,s4) και να επιλέξει τη λέξη με την οποία η ομοιότητα της λέξης w είναι η μεγαλύτερη.
Το σύστημα θα μετρήσει τη σημασιολογική ομοιότητα των ζευγών λέξεων, αρχικά υπολογίζοντας ένα διάνυσμα σημασιολογικής περιγραφής (semantic descriptor vector) για κάθε μια από τις λέξεις, και στη συνέχεια θα υπολογίζει την ομοιότητα συνημίτονου μεταξύ δύο διανυσμάτων. Ο τύπος συνημίτονου υπολογίζει το συνημίτονο της γωνίας μεταξύ των δύο διανυσμάτων. Καθώς το συνημίτονο προσεγγίζει το "1", τα δύο διανύσματα συμπίπτουν, δηλαδή όσο πιο κοντά στο 1 είναι η τιμή τόσο μεγαλύτερη η ομοιότητα, ενώ όσο πιο κοντά στο 0, τόσο πιο ανεξάρτητα είναι τα κείμενα. Εάν τα δύο διανύσματα είναι τελείως

ανεξάρτητα, θα είναι ορθογώνια οπότε και η αξία του συνημίτονου είναι "0".
Η ομοιότητα συνημίτονου μεταξύ δύο διανυσμάτων u={$u_1, u_2, \ldots, u_N$} και v={$v_1, v_2, \ldots, v_N$} ορίζεται ως:

$$\text{sim}(u,v) = \frac{u \cdot v}{|u||v|} = \frac{\sum_{i=1}^{N} u_i v_i}{\sqrt{\left(\sum_{i=1}^{N} u_i^2\right)\left(\sum_{i=1}^{N} v_i^2\right)}}$$

Έτσι, η ομοιότητα συνημίτονου ορίζεται ως το άθροισμα του γινομένου των τιμών των συνιστωσών των διανυσμάτων u και v, δια το γινόμενο των τετραγωνικών ριζών του αθροίσματος του τετραγώνου των συνιστωσών του u και των συνιστωσών του v.

Δοθέντος ενός κειμένου n λέξεων ($w_1, w_2, \ldots, w_n$) και μιας λέξης w, ονομάζουμε $desc_w$ το διάνυσμα σημασιολογικής περιγραφής της λέξης w που υπολογίζεται χρησιμοποιώντας αυτό το κείμενο. Το $desc_w$ είναι ένα διάνυσμα n στοιχείων. Το i-οστό στοιχείο του $desc_w$ είναι το πλήθος των προτάσεων στις οποίες εμφανίζονται και οι δυο λέξεις w και w_i.
Για λόγους αποτελεσματικότητας, μπορείτε να αποθηκεύσετε το διάνυσμα $desc_w$ χωρίς τα μηδενικά που αντιστοιχούν σε λέξεις που δεν συνυπάρχουν με w. Για παράδειγμα, ας υποθέσουμε ότι μας δίνεται το ακόλουθο κείμενο:

> *I am a sick man. I am a spiteful man. I am an unattractive man. I believe my liver is diseased. However, I know nothing at all about my disease, and do not know for certain what ails me.*

Η λέξη "man" εμφανίζεται μόνο στις τρεις πρώτες προτάσεις. Το διάνυσμα σημασιολογικής περιγραφής της συγκεκριμένης λέξης είναι:
`{"i": 3, "am": 3, "a": 2, "sick": 1, "spiteful": 1, "an": 1, "unattractive": 1}`
Η λέξη "liver" εμφανίζεται μόνο στη τέταρτη πρόταση, οπότε το διάνυσμα σημασιολογικής περιγραφής είναι:
`{"i": 1, "believe": 1, "my": 1, "is": 1, "diseased": 1}`
Αποθηκεύουμε όλες τις λέξεις σε πεζά γράμματα, γιατί δεν θεωρούμε, για παράδειγμα τις λέξεις , "Man" και "man" σαν διαφορετικές λέξεις. Ωστόσο, θεωρούμε λέξεις όπως "believe" και "believes", ή "am" και "is" σαν διαφορετικές λέξεις. Επίσης, απορρίπτουμε κάθε σημείο στίξης.
Επειδή, όπως τονίστηκε πιο πάνω, δεν αποθηκεύουμε τα μηδενικά στοιχεία του διανύσματος $desc_w$, δεν μπορούμε να εφαρμόσουμε κατευθείαν τον τύπο υπολογισμού της ομοιότητας συνημίτονου. Ωστόσο, μπορούμε ακόμα να υπολογίσουμε την ομοιότητα μεταξύ των διανυσμάτων, χρησιμοποιώντας μόνο τα θετικά στοιχεία τους.
Για παράδειγμα, η ομοιότητα συνημίτονου των λέξεων "man" και "liver", δεδομένου των παραπάνω διανυσμάτων τους, μπορεί να υπολογιστεί ως ακολούθως:

$$\frac{3 \cdot 1 \text{ (for the word "i")}}{\sqrt{(3^2+3^2+2^2+1^2+1^2+1^2+1^2)(1^2+1^2+1^2+1^2+1^2)}} = 3/\sqrt{130} = 0.2631\,.$$

Ο ψευδοκώδικας υπολογισμού της ομοιότητας συνημίτονο, αγνοώντας τα μηδενικά στοιχεία, είναι ο παρακάτω:

```
double norm(vec)
sum_of_squares = 0.0
  for x in vec:
    sum_of_squares+=vec[x]*vec[x]

  return math.sqrt(sum_of_squares)
```

```
double cosine_similarity(vec1, vec2)
  dot_product = 0.0
  for x in vec1:
    if x in vec2:
       dot_product += vec1[x] * vec2[x]
  return dot_product/(norm(vec1)*norm(vec2))
```

III. Υλοποίηση

Στόχος του προγράμματος είναι η δημιουργία ενός ευφυές συστήματος που απαντά με όσο το δυνατόν μεγαλύτερη ευκρίνεια στις ερωτήσεις στα τεστ TOEFL. **Πριν από αυτό όμως, πρέπει το σύστημα να εκπαιδευτεί έτσι ώστε να μάθει να απαντά σε τέτοιες ερωτήσεις. Άρα, το πρώτο πράγμα που πρέπει να κάνει το πρόγραμμα σας, είναι η εκπαίδευση του αλγορίθμου έτσι ώστε να «μάθει».**

Η εκπαίδευση του συστήματος θα γίνει χρησιμοποιώντας αρχεία κειμένου αγγλικής γλώσσας όπως αυτά που σας δίνονται. Εκπαίδευση σημαίνει ότι, χρησιμοποιώντας ένα τέτοιο κείμενο, το πρόγραμμα πρέπει πρώτα να δημιουργήσει το διάνυσμα σημασιολογικής περιγραφής για κάθε λέξη. Αυτό μπορεί να γίνει υλοποιώντας τις παρακάτω λειτουργίες:

- **Λειτουργία 1 - getSentenceLists(text)**. Αυτή η λειτουργία πρέπει να υλοποιηθεί μέσω μεθόδων οι οποίες παίρνουν μια συμβολοσειρά text και επιστρέφουν μια λίστα που περιέχει λίστες συμβολοσειρών, μία λίστα για κάθε μία πρόταση (όπως ορίζεται παρακάτω) στη συμβολοσειρά text. Μια λίστα που αντιπροσωπεύει μια πρόταση είναι μια λίστα με τις λέξεις της πρότασης, η καθεμία με πεζά γράμματα.
  Για τους σκοπούς της εργασίας μας, θα θεωρήσουμε ότι οι προτάσεις διαχωρίζονται από μία από τις συμβολοσειρές ".", "?", ή "!". Αγνοούμε τη δυνατότητα άλλων σημείων στίξης που διαχωρίζουν τις προτάσεις και υποθέτουμε επίσης ότι κάθε περίοδος χωρίζει προτάσεις. Για παράδειγμα, η συμβολοσειρά: `"The St. Bernard is a friendly dog!"` θεωρείται ότι αποτελείται από δύο προτάσεις, `"The St"` και `"Bernard is a friendly dog".`
  Οι λέξεις στη λίστα που αντιπροσωπεύει μια πρόταση πρέπει να είναι στη σειρά με την οποία εμφανίζονται στη πρόταση αφήνοντας εκτός λίστας τα σημεία στίξης.
  Υποθέστε ότι υπάρχουν μόνο τα ακόλουθα σημεία στίξης στο αρχείο κειμένου: ",", "-", "--", ":", ";", "!", "?", ".", τα μονά διαλυτικά ή η απόστροφος (‘) και τα διπλά διαλυτικά (“). Οι λέξεις που περιέχουν απόστροφο θα υπολογίζονται ως δύο διαφορετικές λέξεις. Για παράδειγμα το «don’t» αποτελείται από τις λέξεις don και t, ενώ το «School’s» αποτελείται από τις λέξεις School και s .

  Για παράδειγμα, αν το αρχείο κειμένου περιέχει τα ακόλουθα (και τίποτα άλλο εκτός αυτών):

  Hello, Jack. How is it going? Not bad; pretty good, actually... Very very good, in fact.

  τότε η λειτουργία πρέπει να δημιουργήσει την ακόλουθη λίστα:

  ```
  [[’hello’, ’jack’],
  [’how’, ’is’, ’it’, ’going’],
  [’not’, ’bad’, ’pretty’, ’good’, ’actually’],
  [’very’, ’very’, ’good’, ’in’, ’fact’]]
  ```

- **Λειτουργία 2 - get_sentence_lists_from_files(filenames).** Για αυτήν την λειτουργία πρέπει να υλοποιήσετε μεθόδους οι οποίες παίρνουν μια λίστα αποτελούμενη από ονόματα αρχείων κειμένου, και επιστρέφει τη λίστα για κάθε πρόταση των αρχείων κειμένου. Η λίστα πρέπει να είναι στην ίδια μορφή με τη λίστα που περιγράφεται στη Λειτουργία 1.

- **Λειτουργία 3 - build_semantic_descriptors(sentences)**. Για αυτήν την λειτουργία πρέπει να υλοποιήσετε μεθόδους οι οποίες παίρνουν μια λίστα που περιέχει λίστες που αντιπροσωπεύουν προτάσεις (η λίστα που δημιουργείται με τη Λειτουργία 2), και επιστρέφει μια δομή δεδομένων d στην οποία, για κάθε λέξη w η οποία εμφανίζεται σε τουλάχιστον μια πρόταση, το στοιχείο d[w] είναι από μόνο του μια δομή δεδομένων που αναπαριστά το διάνυσμα σημασιολογικής περιγραφής της λέξης w. Η δομή αυτή μπορεί να είναι ένα HashMap όπου κάθε στοιχείο του είναι ένα HashMap. Για το παράδειγμα της σελίδας 2, ένα μέρος της δομής θα ήταν:

```
{'man':{'i':3, 'am':3, 'a':2, 'sick':1, 'spiteful':1, 'an':1, 'unattractive':1},

'liver': {'i':1, 'believe':1, 'my':1, 'is':1, 'diseased':1},
...
}
```

- **Λειτουργία 4 - most_similar_word(word, choices, semantic_descriptors)**. Με αυτήν την λειτουργία, βάσει μιας λέξης word, μιας λίστας με λέξεις choices, και του HashMap semantic_descriptors που δημιουργείται με την Λειτουργία 3, πρέπει να βρεθεί η λέξη από τη λίστα choices που έχει τη μεγαλύτερη σημασιολογική ομοιότητα με τη λέξη word. Η σημασιολογική ομοιότητα υπολογίζεται χρησιμοποιώντας τα δεδομένα στο HashMap semantic_descriptors.
  Εάν η σημασιολογική ομοιότητα μεταξύ δύο λέξεων δεν μπορεί να υπολογιστεί θεωρείται ότι είναι -1.
  Σε περίπτωση ισότητας μεταξύ πολλών στοιχείων στη λίστα choices, πρέπει να επιστραφεί η λέξη με τον μικρότερο δείκτη στη λίστα choices (π.χ. αν υπάρχει ισοπαλία τιμών μεταξύ choices[5] και choices[7], πρέπει να επιστραφεί η τιμή choices[5]).

- **Λειτουργία 5 - run_similarity_test(filename, semantic_descriptors)**. Με αυτήν την λειτουργία, βάσει ενός αρχείου κειμένου, έστω filename, επιστρέφει το ποσοστό των ερωτήσεων όπου η λειτουργία most_similar_word() μαντεύει την σωστή απάντηση χρησιμοποιώντας τα δεδομένα στο HashMap semantic_descriptors.
  Η δομή του αρχείου filename είναι ως ακολούθως: Σε κάθε γραμμή, υπάρχει μια λέξη (με όλα τα γράμματα πεζά), η σωστή απάντηση και οι επιλογές. Για παράδειγμα, η γραμμή:

  ```
  feline cat dog cat horse
  ```

  αντιπροσωπεύει την ερώτηση:

  ```
  feline:
        (a) dog
        (b) cat
        (c) horse
  ```

  και υποδεικνύει ότι η σωστή απάντηση είναι "cat".
  Το πρόγραμμα σας πρέπει να δημιουργήσει ένα αρχείο εξόδου με όλες τις απαντήσεις και να υπολογίσει το ποσοστό επιτυχίας (εύρεσης σωστών απαντήσεων).
- Μπορείτε να προσθέσετε άλλες λειτουργίες στην εργασία, π.χ. άλλες μεθόδους μέτρησης ομοιότητας. Χρησιμοποιήστε τη φαντασία σας.
- Είναι σημαντικό ο χρόνος εκτέλεσης του προγράμματος να είναι μέσα σε λογικά πλαίσια. Αυτό προϋποθέτει τη σωστή δημιουργία και χρήση των δομών. Αυτό θα επηρεάσει επίσης και την βαθμολόγηση σας.

IV. Γενικές Οδηγίες

Δομή του προγράμματος

Για το πρόγραμμα σας **πρέπει** να χρησιμοποιήστε το **github** δίνοντας πρόσβαση στο αποθετήριο (repository) σας και στους διδάσκοντες του μαθήματος.

Είναι απαραίτητο να ακολουθήσετε την παρακάτω οργάνωση του προγράμματος:

- Package name: hw4
- Class names: Όπως σας ζητείται από την εκφώνηση

- Javadoc documentation για όλες τις κλάσεις και μεθόδους.

Το πρόγραμμα σας θα πρέπει να περιλαμβάνει εύστοχα και περιεκτικά σχόλια, να έχει καλή στοίχιση και το όνομα κάθε μεταβλητής, σταθεράς, ή συνάρτησης να είναι ενδεικτικό του ρόλου της. **Να χρησιμοποιήσετε το λογισμικό τεκμηρίωσης javadoc** έτσι ώστε να μπορούμε να μετατρέψουμε τα σχόλια του προγράμματός σας σε HTML αρχεία και να τα δούμε με ένα browser. Η συστηματική αντιμετώπιση της λύσης ενός προβλήματος περιλαμβάνει στο παρόν στάδιο τη διάσπαση του προβλήματος σε μικρότερα ανεξάρτητα προβλήματα που κατά κανόνα κωδικοποιούμε σε ξεχωριστές συναρτήσεις. Για αυτό τον λόγο σας καλούμε να κάνετε χρήση μεθόδων και άλλων τεχνικών αντικειμενοστραφής προγραμματισμού. Επίσης, σας θυμίζουμε ότι κατά την διάρκεια της εκτέλεσης του προγράμματος σας, αυτό θα πρέπει να δίνει τα κατάλληλα μηνύματα σε περίπτωση λάθους. Τέλος το πρόγραμμα σας θα πρέπει να μεταγλωττίζεται στις μηχανές του εργαστηρίου.

Καλή επιτυχία