Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Γλωσσική Τεχνολογία

Ακαδημαϊκό Έτος 2010 – 2011

Άσκηση

Συγκομιδή και δεικτοδότηση ιστοσελίδων

Μανέττα Νικολίτσα	**ΑΜ: 3967**
Βάσσης Γεώργιος	**ΑΜ: 3896**

Αναφορά

Λίγα λόγια για την άσκηση …

Σκοπός της άσκησης είναι η υλοποίηση ενός ολοκληρωμένου συστήματος συγκομιδής και δεικτοδότησης ιστοσελίδων. Η αρχιτεκτονική του συστήματος αποτελείται από έναν προσκομιστή ιστοσελίδων ο οποίος διατρέχει το δίκτυο και κατεβάζει τα περιεχόμενα των νέων σελίδων που συναντά, καθώς και από ένα υποσύστημα επεξεργασίας του περιεχομένου των ιστοσελίδων το οποίο στόχο έχει την δεικτοδότηση σε ανεστραμμένο ευρετήριο που σχεδιάσαμε. Για αξιολόγηση της απόδοσης του ευρετηρίου υλοποιήσαμε έναν μηχανισμό υποβολής ερωτημάτων στα δεικτοδοτημένα κείμενα της συλλογής, μέσω του οποίου υποβάλουμε ερωτήματα προς το ευρετήριο και ανακτάμε τα κείμενα της συλλογής που σχετίζονται με αυτά (δηλ. περιέχουν τους όρους των ερωτημάτων).

Αποτελέσματα μετρήσεων:

- Αριθμός ιστοσελίδων: 1000
- Μέσο μέγεθος (σε χαρακτήρες) της συλλογής: 99367
- Μέσο μέγεθος καθαρού κειμένου (σε χαρακτήρες) της συλλογής: 10032
- Συνολικός χρόνος κατασκευής του ανεστραμμένου ευρετηρίου: 143 sec
- Μέσος χρόνος απόκρισης του ευρετηρίου: 0.713 sec

Υποσυστήματα που υλοποιήσαμε:

Για κάθε ένα από τα υποσυστήματα που υλοποιήσαμε περιγράφουμε τη βασική λειτουργία του και ένα σχόλιο για την είδος της υλοποίησης.

1ον Προσκομιστής Ιστοσελίδων

Ο προσκομιστής ιστοσελίδων (crawler) ξεκινά από μια λίστα 5 ιστοσελίδων, κατεβάζει το περιεχόμενο τους, το οποίο έχει html μορφοποίηση , θα πρέπει να έχει μέγεθος πάνω από 40.000 χαρακτήρες και ακολουθεί τα links που περιέχει κάθε ιστοσελίδα για να συνεχίσει την συγκομιδή. Στην εκφώνηση της άσκησης παρουσιάζονται και κάποιοι άλλοι περιορισμοί για τον crawler. Στην συνέχεια αποθηκεύει τα περιεχόμενα που κατέβασε σε τοπικά αντίγραφα που θα χρησιμοποιηθούν για την δημιουργία του ευρετηρίου.

Για την υλοποίηση του crawler χρησιμοποιήσαμε τα δυο modules και τις αντίστοιχες συναρτήσεις που φαίνονται παρακάτω:

module **urlparse** – διαβάζει ένα URL και το σπάει σε κομμάτια

Συναρτήσεις:

- `urlparse.urlparse(url)`

  Διατρέχει του URL και το σπάει σε 6 κομμάτια. Το URL θεωρείται ότι είναι της μορφής:
  scheme://netloc/path;parameters?query#fragment

module **urllib2** – βιβλιοθήκη για το άνοιγμα URLs

Συναρτήσεις:

- `urllib2.Request(url)`
  Δημιουργεί μια αίτηση (Request object) του τι θέλουμε να κατεβάσουμε
- `urllib2.build_opener()`
  Δημιουργεί έναν URL opener
- `opener.open()`
  Εκτελείται ο opener και κατεβαίνει η πληροφορία που θέλουμε

➢ 2ον Προεπεξεργασία

Το συγκεκριμένο υποσύστημα εξάγει το καθαρό κείμενο από τις ιστοσελίδες που συγκεντρώσαμε και εφαρμόζει tokenization ώστε στο tokenized κείμενο που προκύπτει να εμφανίζεται μια λέξη ανά γραμμή ή ένα σημείο στίξης ανά γραμμή.

Σε αυτό το υποσύστημα χρησιμοποιήσαμε το Natural Language Toolkit το οποίο είναι ένα πακέτο βιβλιοθηκών και εργαλείων για Natural Language Processing σε Python.

- Για την εξαγωγή του καθαρού κειμένου από τις ιστοσελίδες που συγκεντρώσαμε χρησιμοποιήσαμε τη συνάρτηση:
  `nltk.clean_html(raw)`
- Για τη δημιουργία του tokenized κειμένου χρησιμοποιήσαμε τη συνάρτηση:

  `nltk.word_tokenize(text)`

Στη συνέχεια για κάθε λέξη και κάθε σύμβολο που προκύπτει από τη δημιουργία του tokenized κειμένου, το αποθηκεύουμε σε ένα αρχείο που το ονομάζουμε namei.txt όπου i το id του κάθε ιστοσελίδας της οποίας αποθηκεύουμε το tokenized κείμενο. Στο αρχείο namei.txt έχουμε μία λέξη ή ένα σύμβολο ανά γραμμή

➢ 3ον Μορφοσυντακτική Ανάλυση

Σε αυτό το υποσύστημα δεχόμαστε στην είσοδο το tokenized κείμενο και υλοποιούνται τα 2 παρακάτω γεγονότα:

- Text Normalization
  Κανονικοποίηση λέξεων δηλαδή η μετατροπή τους σε τύπους που μπορούν να ομαδοποιηθούν.

- Tagging
  Μορφοσυντακτικής σχολιασμός δηλαδή αναγνώριση του μέρους του λόγου κάθε λέξης ή συμβόλου του κειμένου.
  Για την υλοποίηση των 2 παραπάνω γεγονότων χρησιμοποιήσαμε τον Tree Tagger και την έτοιμη υλοποίηση που αυτός προσφέρει. Στο τέλος της μορφοσυντακτικής ανάλυσης κάθε κείμενο της συλλογής ιστοσελίδων διαθέτει μορφοσυντακτικό σχολιασμό (PoS-tags) για κάθε λέξη που περιέχει. Τα μορφοσυντακτικά σχολιασμένα κείμενα της συλλογής αποθηκεύονται σε βοηθητικό-ενδιάμεσο αρχείο το οποίο ονομάζουμε taggei.txt όπου i είναι το id του κειμένου του οποίου το μορφοσυντακτικά σχολιασμένο κείμενο περιέχεται σε αυτό το αρχείο. Έτσι κάθε γραμμή του αρχείου taggei.txt περιέχει τρία στοιχεία. Το πρώτο στοιχείο είναι η λέξη ή το σύμβολου που δώσαμε στην είσοδο, το δεύτερο στοιχείο είναι ένα PoS tag και το τρίτο στοιχείο είναι το λήμμα που αντιστοιχεί στην λέξη που δώσαμε στην είσοδο.

➢ 4ον Αναπαράσταση ιστοσελίδων στο Μοντέλο Διανυσματικού Χώρου

Σε αυτό το υποσύστημα αναπαριστούμε το περιεχόμενο κάθε κειμένου ως διάνυσμα. Χρησιμοποιούμε τα μορφοσυντακτικά σχολιασμένα κείμενα που έχουν προκύψει στο προηγούμενο ερώτημα και αφαιρούμε τους τερματικούς όρους (stop-words) από κάθε κείμενο. Οι τερματικοί όροι είναι λέξεις που δεν έχουν σημασιολογικό περιεχόμενο και εμφανίζονται σε όλα τα κείμενα, με αποτέλεσμα να μην αποτελούν χρήσιμους όρους δεικτοδότησης. Τους τερματικούς όρους τους αφαιρούμε χρησιμοποιώντας το PoS tag(δεύτερο στοιχείο κάθε γραμμής) που έχει προκύψει κατά το μορφοσυντακτικό σχολιασμό, σε συνδυασμό με τον πίνακα τον PoS tag για closed class category. Επιπλέον από κάθε κείμενο αφαιρούμε όρους που ναι μεν δεν ανήκουν στους τερματικούς όρους έχουν όμως σαν λήμμα τους τη λέξη <unknow>. Αυτό το κάνουμε για να μην αλλοιωθούν τα αποτελέσματά μας.

Αφού αφαιρέσουμε τους τερματικούς όρους από κάθε κείμενο της συλλογής, για κάθε μοναδικό λήμμα του κειμένου μετράμε τη συχνότητα

εμφάνισής του στο κείμενο. Στη συνέχεια όλα τα λήμματα του κάθε κειμένου μαζί με την συχνότητα εμφάνισής τους τα αποθηκεύουμε σε ένα λεξικό οπού τα κλειδιά του λεξικού αυτού αποτελούν τα λήμματα. Κάθε κείμενο αποτελείται από ένα τέτοιο λεξικό το οποίο αποθηκεύουμε σε μία λίστα σε κάθε θέση της οποίας υπάρχει το λεξικό του αντίστοιχου κειμένου.

➢ 5ον Δημιουργία του ευρετηρίου

Σε αυτό το υποσύστημα ολοκληρώνεται η κατασκευή του ανεστραμμένου ευρετηρίου. Η δημιουργία του ανεστραμμένου ευρετηρίου πραγματοποιείται εφόσον έχουν υλοποιηθεί όλα τα παραπάνω βήματα για όλα τα κείμενα της συλλογής μας. Στην προκειμένη περίπτωση για κάθε ένα από τα λήμματα κάθε κειμένου ελέγχουμε κατά πόσο το λήμμα αυτό εμφανίζεται και στα υπόλοιπα κείμενα της συλλογής μας. Έτσι δημιουργούμε μία λίστα για κάθε λήμμα της συλλογής μας που στις ζυγές του θέσεις περιέχει την τιμή id του κειμένου που εμφανίζεται το λήμμα αυτό και στην αμέσως επόμενη θέση εμφανίζεται το βάρος του λήμματος στο κείμενο αυτό με βάση τη μετρική TD-IDF. Οπότε για ολόκληρη τη συλλογή ιστοσελίδων που έχουμε συγκεντρώσει, εντοπίζουμε τα μοναδικά λήμματα, καθώς και τα κείμενα στα οποία εμφανίζεται το κάθε λήμμα. Δημιουργούμε την αντίστοιχη εγγραφή στο ανεστραμμένο ευρετήριο και υπολογίζουμε τα αντίστοιχα βάρη. Το βάρος του κάθε λήμματος για ένα κείμενο αντιπροσωπεύει το βαθμό σπουδαιότητας του λήμματος για το συγκεκριμένο κείμενο και το υπολογίζουμε χρησιμοποιώντας τη μετρική TD-IDF.

➢ 6ον Αποθήκευση και επαναφόρτωση του ευρετηρίου

Σε αυτό το υποσύστημα υλοποιούμε τις απαραίτητες συναρτήσεις έτσι ώστε να είναι δυνατή η αποθήκευση του ευρετηρίου στον δίσκο σε xml μορφή και η επαναφόρτωση του στη μνήμη.

Ο τρόπος που γίνετε αυτό είναι χρησιμοποιώντας την βιβλιοθήκη ElementTree η οποία παρέχει ένα Element type, το οποίο είναι ένα απλό και ευέλικτο container object, σχεδιασμένο για την αποθήκευση ιεραρχικών δομών δεδομένων. Η βιβλιοθήκη παρέχει υποστήριξη για την ανάγνωση και τη εγγραφή Element structures ως XML.

Οι συναρτήσεις τις οποίες χρησιμοποιήσαμε για την ανάγνωση και τη εγγραφή Element structures ως XML είναι οι εξής:

<u>Για την εγγραφή-αποθήκευση:</u>

ElementTree methods

- `xml.etree.ElementTree.Element(tag)`

Δημιουργεί μια δομή δέντρου

- `xml.etree.ElementTree.SubElement(parent, tag)`
  Δημιουργεί ένα αντικείμενο element και το προσθέτει στο υπάρχον element.

dictionary-like methods

- `set()`
  Θέτει το attribute με όνομα key ενός element στην τιμή value

ElementTree Objects

- `write(file)`
  Γράφει το element tree στο αρχείο file

Η εγγραφή του ανεστραμμένου ευρετηρίου στο αρχείο γίνεται στο αρχείο tokens.py. Στο αρχείο αυτό για κάθε λήμμα που περιέχεται στο ανεστραμμένο ευρετήριο του προηγούμενου ερωτήματος εισάγουμε τα κείμενα στα οποία υπάρχει το λήμμα αυτό καθώς και το αντίστοιχο βάρος τους. Το ανεστραμμένο ευρετήριο το αποθηκεύουμε στο φάκελο C:\\project\\page1.xhtml ενώ έχει τη δομή που ζητείται στην εκφώνηση της άσκησης.

Για την ανάγνωση-επαναφόρτωση:

ElementTree methods

- `xml.etree.ElementTree.parse(source)`
  Φορτώνει ένα XML από το αρχείο source στην μνήμη σε μορφή element tree

dictionary-like methods

- `get(key)`
  Επιστρέφει την τιμή του attribute με όνομα key

ElementTree Objects

- `getroot()`
  Επιστρέφει το root element του δέντρου

Η επαναφόρτωση του ανεστραμμένου ευρετηρίου από το αρχείο C:\\project\\page1.xhtml γίνεται μέσω του αρχείου xml_to_dic.py. Στο αρχείο αυτό για την επαναφόρτωση του ανεστραμμένου ευρετηρίου αρχικά αναζητούμε τα λήμματα τα οποία είναι αποθηκευμένα σε αυτό και στη

συνέχεια για κάθε λήμμα κρατάμε σε μία λίστα τα id όλων των κειμένων στο οποίο περιέχεται και τα αντίστοιχα βάρη τους. Έτσι δημιουργείται ένα λεξικό στο οποίο αποθηκεύουμε το ανεστραμμένο ευρετήριο όπου κλειδιά του είναι τα διάφορα λήμματα και τιμές των λημμάτων αυτών αποτελούν η λίστα που περιέχει τα id των κειμένων και τα αντίστοιχα βάρη. Παρατηρούμε ότι η δομή αυτή που προκύπτει για την επαναφόρτωση του ανεστραμμένου ευρετηρίου είναι ίδια με αυτή που χρησιμοποιήθηκε για την αποθήκευση του.

➢ 7ον Αξιολόγηση ευρετηρίου

Στο τελευταίο υποσύστημα υλοποιούμε έναν απλό μηχανισμό υποβολής ερωτημάτων στο ευρετήριο. Ο μηχανισμός δέχεται input από τον χρήστη ένα ερώτημα που περιέχει έναν ή περισσότερους όρους. Κάθε ένα από αυτούς τους όρους, τους ταυτοποιούμε με τα λήμματα του ευρετηρίου και θα επιστρέφει τα urls των ιστοσελίδων τα οποία περιέχουν το λήμμα ή τα λήμματα του ερωτήματος. Πιο συγκεκριμένα για να υλοποιήσουμε τον μηχανισμό αυτόν αρχικά για κάθε ένα από τους όρους του ερωτήματος ελέγχουμε αν υπάρχει στο ευρετήριό μας. Στην περίπτωση που υπάρχει κρατάμε σε μία προσωρινή λίστα τα id των url στα οποία αυτός περιέχεται και τα αντίστοιχα βάρη του σε αυτά τα url. Στην περίπτωση που κάποιος όρος δεν υπάρχει στο ευρετήριό μας κρατάμε μία λίστα με το στοιχείο -1.

Στη συνέχεια στην περίπτωση που μας έχει δοθεί ένας όρος κατά την υποβολή του ερωτήματος εμφανίζουμε στον χρήστη τα url και τα αντίστοιχα βάρη στα οποία αυτός ο όρος εμφανίζεται κατά αύξουσα σειρά με βάση το TF-IDF βάρος. Αν ο όρος δεν υπάρχει στο ευρετήριό μας εμφανίζουμε ένα αντίστοιχο μήνυμα στο χρήστη.

Αν κατά την υποβολή του ερωτήματός μας έχουν δοθεί παραπάνω από ένας όροι τότε εφόσον για κάθε ένα από αυτούς γνωρίζουμε τα id των κειμένων στα οποία εμφανίζεται ελέγχουμε αν υπάρχει ένα ή περισσότερα id στα οποία εμφανίζονται όλοι οι όροι του ερωτήματος εισόδου. Αν ναι όμοια με παραπάνω εμφανίζουμε στον χρήστη τα αντίστοιχα uls ταξινομημένα σε αύξουσα σειρά με βάση το TF-IDF. Όμως σε αντίθεση με την προηγούμενη περίπτωση, τώρα το βάρος TF-IDF προκύπτει από το άθροισμα των βαρών των επιμέρους όρων που δόθηκαν στο κείμενο. Αν δεν υπάρχει κάποιο url στο οποίο όλοι οι όροι του ερωτήματος του χρήστη να εμφανίζονται ενημερώνουμε το χρήστη ότι οι όροι είσοδοι δεν υπάρχουν όλοι μαζί σε κανένα από τα δοσμένα url. Στη συνέχεια εμφανίζουμε στον χρήστη τα url που εμφανίζεται κάθε ένας από τους επιμέρους όρους της εισόδου με την ίδια διαδικασία που ακολουθήσαμε όταν μας δίνεται ένας όρος στην είσοδο.

Σε όλες τις παραπάνω περιπτώσεις μπορούμε να διαβάσουμε το url του κειμένου στο οποίο εμφανίστηκε ένας όρος από το αρχείο url.txt στο οποίο αποθηκεύουμε όλα τα url που έχουμε προσπελάσει.

Στο ερώτημα αυτό δεν έχουμε χρησιμοποιήσει κάποια έτοιμη υλοποίηση.