Γλωσσικη τεχνολογια

Προεπεξεργασία Κειμένου

Στόχος Επεξεργασίας

- Γραπτό κείμενο:
  - Τρόπος επικοινωνίας
  - Φέρει σημασιολογικό περιεχόμενο
- Αναζητούμε τρόπο να:
  - Μετρήσουμε το πληροφοριακό περιεχόμενο
  - Ποσοτικοποιήσουμε το πληροφοριακό περιεχόμενο
  - Μετρήσουμε την ομοιότητα μεταξύ κειμένων
- Χρειάζεται φορμαλιστική αναπαράσταση
- Εφαρμογές:
  - Συστήματα δεικτοδότησης για Ανάκτηση Πληροφορίας
  - Κατηγοριοποίηση κειμένου
  - Σχεδόν ο,τιδήποτε έχει να κάνει με αυτόματη επεξεργασία κειμένου...

Βασικές Έννοιες

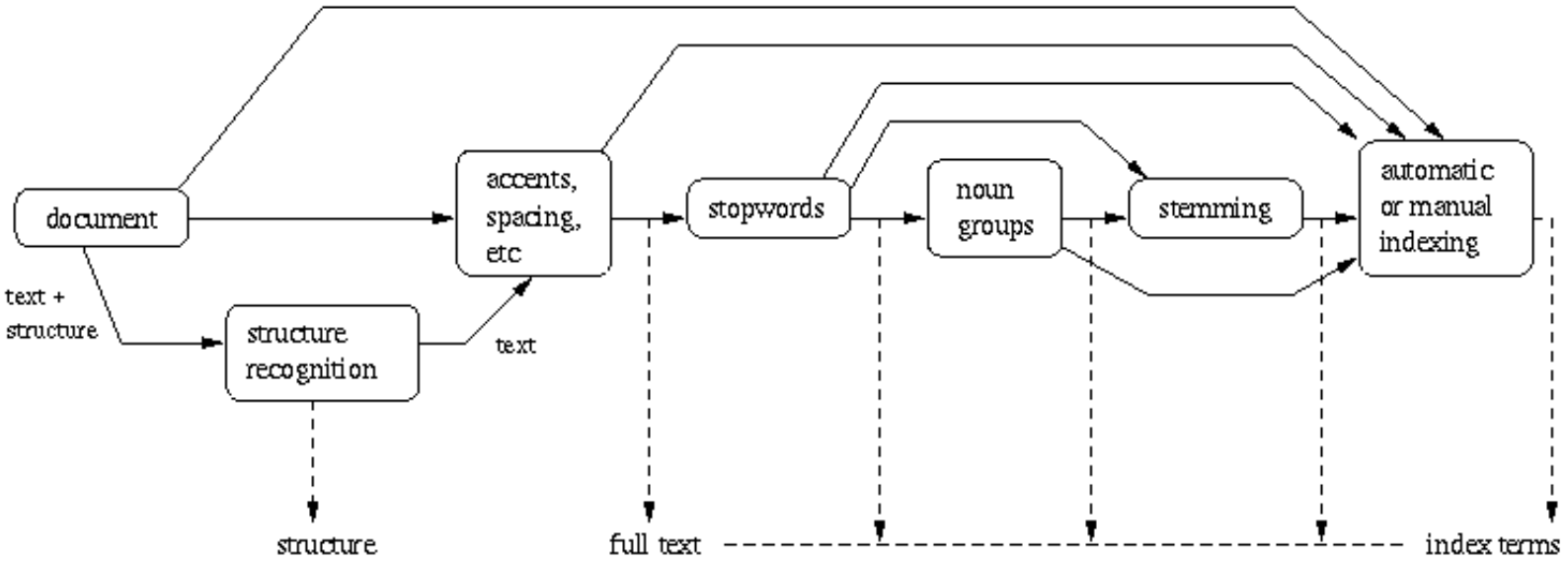
- Όροι
  - Κάθε κείμενο περιγράφεται από ένα σύνολο από αντιπροσωπευτικές λέξεις κλειδιά που ονομάζονται όροι.
- Λεξιλόγιο
  - Το σύνολο όλων των μοναδικών όρων που υπάρχουν στη συλλογή κειμένων
- Αναπαράσταση κειμένου
  - Η αντιστοίχιση του κειμένου (αδόμητη πληροφορία) σε δομημένη αναπαράσταση

Πίνακας όρων-κειμένων

	d1	d2
term 1	$f_{1,1}$	$f_{1,2}$
term 2	$f_{2,1}$	$f_{2,2}$
term 3	$f_{3,1}$	$f_{3,2}$

- Η ύπαρξη ενός όρου σε ένα κείμενο δημιουργεί μια σχέση μεταξύ τους
- Αυτή η σχέση ποσοτικοποιείται από τη συνάρτηση *f*(i,j), διαφορετική σε κάθε μοντέλο αναπαράστασης
  - Πόσες f ξέρετε?
- Οι αντιπροσωπευτικοί όροι παρέχουν μια *λογική αναπαράσταση του κειμένου*.

Αναπαράσταση Κειμένου



- Λογική αναπαράσταση κειμένου
  - Από το πλήρες κείμενο σε σύνολο αντιπροσωπευτικών όρων

Βασικά σημεία προεπεξεργασίας

- Επεξεργασία της δομής
  - Εξαγωγή του κειμένου με επεξεργασία των μεταδεδομένων και των στοιχείων δομής, αν υπάρχουν
- Λεξική ανάλυση
  - Μετατροπή του κειμένου σε ακολουθία λέξεων
- Αφαίρεση των τερματικών όρων (stopwords)
  - Αφαίρεση των λέξεων που δεν φέρουν σημασιολογικό περιεχόμενο
- Κανονικοποίηση των λέξεων
  - Αναγωγή όλων των μορφολογικών τύπων μιας λέξης σε μια ενιαία αναπαράσταση
- Επιλογή των αντιπροσωπευτικών όρων
  - Κατασκευή της λογικής αναπαράστασης του κειμένου

Επεξεργασία της δομής ιστοσελίδων

Αφαίρεση του html markup με το NLTK:

```
>>> from urllib import urlopen
>>> url="http://www.diveintopython.net/"
>>> html=urlopen(url).read()
>>> len(html)
14599
>>> html[1100:1300]
'ntainer">\r\n                <h1 id="logo">Dive Into Python</h1>\r\n
   <p id="tagline">Python from novice to pro</p>\r\n            </td>\r\n
     <td colspan="3" align="right">\r\n            '
>>> raw=nltk.clean_html(html)
>>> len(raw)
3492
>>> raw[500:700]
' a variety of formats. It is also available in multiple languages .\r\n    \r\n
    \r\n    \r\n    \r\n     \r\n      \r\n       Read Dive Into Python \r\n
  \r\n     \r\n     \r\n    \r\n    This book is still being writ'
```

- Για πλήρη επεξεργασία html δείτε:
  - Built-in python support
    - "Dive into Python": Chapter 8. HTML Processing
  - Beautiful Soup package

Encodings

```
>>> import os
>>> path=os.path.expanduser('~/greek')
>>> f=open(path,'r')
>>> line=f.readline().strip()
>>> line
'\xce\x91\xcf\x85\xcf\x84\xcf\x8c \xcf\x84\xce\xbf \xce\xba\xce\xb5\xce\xaf\xce\
xbc\xce\xb5\xce\xbd\xce\xbf \xcf\x80\xce\xb5\xcf\x81\xce\xb9\xce\xad\xcf\x87\xce
\xb5\xce\xb9 \xce\xb5\xce\xbb\xce\xbb\xce\xb7\xce\xbd\xce\xb9\xce\xba\xce\xac.'
>>> print line.encode('utf8')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
UnicodeDecodeError: 'ascii' codec can't decode byte 0xce in position 0: ordinal
not in range(128)
>>> f.close()
>>> import codecs
>>> f=codecs.open(path,encoding='utf8')
>>> line=f.readline().strip()
>>> print line.encode('utf8')
Αυτό το κείμενο περιέχει ελληνικά.
>>> f.close()
```

- Για τα encodings που υποστηρίζονται δείτε:
  - http://docs.python.org/library/codecs.html#standard-encodings

Λεξική Ανάλυση

- Μετατροπή του κειμένου από ακολουθία χαρακτήρων σε ακολουθία δομικών μονάδων.
- Οι λέξεις που αναγνωρίζονται είναι υποψήφιες για αντιπροσωπευτικοί όροι του κειμένου.
- Απλούστερη μορφή: αναγνώριση των ορίων των δομικών μονάδων στα κενά.

```
>>> text='This is not the way to tokenize.\nWill you split 12.5%, star-crossed o
r U.S.A.?'
>>> print text
This is not the way to tokenize.
Will you split 12.5%, star-crossed or U.S.A.?
>>> text.split(' ')
['This', 'is', 'not', 'the', 'way', 'to', 'tokenize.\nWill', 'you', 'split', '12
.5%,', 'star-crossed', 'or', 'U.S.A.?']
```

- Η λεξική ανάλυση περιλαμβάνει πολλά περισσότερα...

Λεξική Ανάλυση – όχι και τόσο απλή...

- 4 βασικά θέματα:
  - Ψηφία
  - Παύλες
  - Σημεία στίξης
  - Κεφαλαία/πεζά
- Web-specific
  - Διευθύνσεις email
  - URLs
- ...

Λεξική Ανάλυση – Λύσεις

- NTLK tokenizer

```
>>> text='This is not the way to tokenize.\nWill you split 12.6%, star-crossed o
r U.S.A.?'
>>> nltk.word_tokenize(text)
['This', 'is', 'not', 'the', 'way', 'to', 'tokenize', '.', 'Will', 'you', 'split
', '12.6', '%', ',', 'star-crossed', 'or', 'U.S.A.', '?']
```

- Regular Expressions

```
>>> text='This is not the way to tokenize.\nWill you split 12.6%, star-crossed o
r U.S.A.?'
>>> pattern=r'''(?x)
...     ([A-Z]\.)+
...     |[0-9]+(\.[0-9]+)?%?
...     |\w+(-\w+)*
...     |\.\.\.
...     |[][.,;"'?():-_`]
... '''
>>> nltk.regexp_tokenize(text,pattern)
['This', 'is', 'not', 'the', 'way', 'to', 'tokenize', '.', 'Will', 'you', 'split
', '12.6%', ',', 'star-crossed', 'or', 'U.S.A.', '?']
```

Λεξική Ανάλυση - Επιλογές

- Δεν υπάρχει ενιαία λύση που να καλύπτει όλες τις περιπτώσεις. Δεν επιτυγχάνεται 100% απόδοση.
- Αποφασίζουμε τι είναι token και τι όχι ανάλογα με το πεδίο εφαρμογής.
- Αντιμετωπίζουμε τις περιπτώσεις που συναντάμε συχνότερα στα δεδομένα μας.
- Στο NLTK περιλαμβάνεται παράδειγμα του «ιδεατού» tokenization:

```
>>> nltk.corpus.treebank_raw.raw()[:400]
'.START \n\nPierre Vinken, 61 years old, will join the board as a nonexecutive d
irector Nov. 29.\nMr. Vinken is chairman of Elsevier N.V., the Dutch publishing
group. \n\n.START \n\nRudolph Agnew, 55 years old and former chairman of Consoli
dated Gold Fields PLC, was named a nonexecutive director of this British industr
ial conglomerate. \n\n.START \n\nA form of asbestos once used to make Kent cigar
ette filters'
>>> nltk.corpus.treebank.words()[0:50]
['Pierre', 'Vinken', ',', '61', 'years', 'old', ',', 'will', 'join', 'the', 'boa
rd', 'as', 'a', 'nonexecutive', 'director', 'Nov.', '29', '.', 'Mr.', 'Vinken',
'is', 'chairman', 'of', 'Elsevier', 'N.V.', ',', 'the', 'Dutch', 'publishing', '
group', '.', 'Rudolph', 'Agnew', ',', '55', 'years', 'old', 'and', 'former', 'ch
airman', 'of', 'Consolidated', 'Gold', 'Fields', 'PLC', ',', 'was', 'named', '*-
1', 'a']
```

Αφαίρεση Τερματικών Όρων

- Τερματικοί Όροι (stopwords)
  - Όροι οι οποίοι δεν φέρουν πληροφορία για το θέμα του κειμένου
- Αφαιρούμε:
  - Συγκεκριμένα μέρη του λόγου
  - Όρους με υπερβολικά μεγάλη συχνότητα σε όλα τα κείμενα

Αναγνώριση μέρους του λόγου

- Χρειάζεται μορφοσυντακτική ανάλυση (Part-Of-Speech Tagging)
  - Προσοχή: Αν χρησιμοποιηθεί tagger τότε στην είσοδό του δεν πρέπει να έχει αφαιρεθεί τίποτα από το κείμενο! (Γιατί?)

```
>>> text= 'This is a cold night.'
>>> tokens=nltk.word_tokenize(text)
>>> tokens
['This', 'is', 'a', 'cold', 'night', '.']
>>> nltk.pos_tag(tokens)
[('This', 'DT'), ('is', 'VBZ'), ('a', 'DT'), ('cold', 'JJ'), ('night', 'NN'), ('
.', '.')]
>>> tokens.remove('a')
>>> tokens
['This', 'is', 'cold', 'night', '.']
>>> nltk.pos_tag(tokens)
[('This', 'DT'), ('is', 'VBZ'), ('cold', 'VBN'), ('night', 'NN'), ('.', '.')]
```

Κανονικοποίηση

- Κανονικοποίηση λέξεων: μετατροπή σε τύπους που μπορούν να ομαδοποιηθούν.
- Επιλογές:
  - Αναγωγή στο θέμα
    - πχ:
      runs, running -> run
      colder, colds -> cold
      beginning,begins->begin
  - Αναγωγή στον πρώτο κλιτικό τύπο
    - πχ:
      am,was -> be
      has,had -> have

Κανονικοποίηση - Αποκατάληξη

- Stemming (αποκατάληξη)
  - Αναγωγή στο θέμα της λέξης
  - Χρησιμοποιεί σύνολο κανόνων αποκατάληξης

```
>>> porter=nltk.PorterStemmer()
>>> tokens=['baby','babies','child','children','am','was']
>>> stemms=[porter.stem(t) for t in tokens]
>>> stemms
['babi', 'babi', 'child', 'children', 'am', 'wa']
```

Κανονικοποίηση - Λημματοποίηση

- Lemmatization (λημματοποίηση)
  - Αναγωγή στον πρώτο κλιτικό τύπο
  - Χρησιμοποιεί μορφολογικό λεξικό
    - Το πόσο ενημερωμένο είναι επηρεάζει την απόδοση!

```
>>> wn=nltk.WordNetLemmatizer()
>>> nouns=['baby','babies','child','children']
>>> [wn.lemmatize(t) for t in nouns]
['baby', 'baby', 'child', 'child']
>>> verbs=['am','was','goes','went']
>>> [wn.lemmatize(t) for t in verbs]
['am', 'wa', 'go', 'went']
```

Επιλογή αντιπροσωπευτικών όρων

- Ποσοτικοποίηση του πόσο σημαντικός είναι ο κάθε όρος
  - Υπολογισμός βαρών
- Επιλογή των πιο σημαντικών (=όρων με τα μεγαλύτερα βάρη)
- Προσοχή: Η μέτρηση συχνοτήτων εμφάνισης και ο υπολογισμός της idf είναι bottleneck!

Τελικά

- Με τους επιλεγμένους όρους κατασκευάζω τον πίνακα:

$$\begin{array}{l} \\ \text{term 1} \\ \text{term 2} \\ \text{term 3} \end{array} \begin{array}{c} \begin{array}{cc} \text{d1} & \text{d2} \end{array} \\ \begin{bmatrix} f_{1,1} & f_{1,2} \\ f_{2,1} & f_{2,2} \\ f_{3,1} & f_{3,2} \end{bmatrix} \end{array}$$

  - Όπου η f είναι:
    - Boolean μοντέλο: 0 ή 1
    - Vector μοντέλο:
      - Συχνότητα εμφάνισης: απλοϊκή επιλογή
      - Tfldf: Συχνότητα όρων & αντίστροφη συχνότητα εγγράφου
  - Πως τον αποθηκεύω? Πως τον φορτώνω στη μνήμη?

Διάνυσμα Κειμένου

- Κάθε κείμενο αντιπροσωπεύεται τελικά από ένα διάνυσμα:

$$\text{vector}(d_n) = \begin{matrix} \\ \text{term 1} \\ \text{term 2} \\ \text{term 3} \end{matrix} \begin{matrix} d_n \\ \begin{bmatrix} f_{1,n} \\ f_{2,n} \\ f_{3,n} \end{bmatrix} \end{matrix}$$

- Το διάνυσμα αναπαριστά φορμαλιστικά το κείμενο στο μοντέλο που εφαρμόζουμε
- Διευκολύνει ποσοτικοποίηση εννοιών όπως πχ ομοιότητα μεταξύ κειμένων
  - Ομοιότητα: πως μπορεί να οριστεί όταν αναφερόμαστε σε διανύσματα?

Υπολογισμός βαρών με TfIdf

$$weight_i = \frac{tf_i \times idf_i}{\sqrt{\sum_{k \in vector} (tf_k \times idf_k)^2}}$$

- Όπου:
  - **tf** (term frequency) είναι η συχνότητα εμφάνισης ενός όρου σε ένα κείμενο
  - **idf** (inverse document frequency) η αντίστροφη συχνότητα κειμένου στη συλλογή
  - παρονομαστής (παράγοντας κανονικοποίησης) **το ευκλείδιο μήκος** του διανύσματος για κάθε κείμενο

$$idf_i = \log \frac{N}{n_i}$$

- Όπου:
  - **N** είναι ο συνολικός αριθμός κειμένων της συλλογής
  - **ni** ο αριθμός των κειμένων της συλλογής στα οποία εμφανίζεται ο όρος i

Recommended Reading

- "Natural Language Processing with Python"
  - Chapter 3: Preprocessing Raw Text