Analisi linguistica: NLTK

Rachele Sprugnoli

rachele.sprugnoli@unipr.it

UNIVERSITÀ
DI PARMA

NLTK

- Natural Language ToolKit: `import nltk`
  - ultima versione basata su Python 3
  - documentazione: https://www.nltk.org/
  - libro online: http://www.nltk.org/book/
  - dà accesso a più di 50 risorse linguistiche (corpora e lessici)
  - permette di eseguire task NLP
  - addestrabile
  - versione per lingue classiche: CLTK, http://cltk.org/
  - lo useremo per analisi lessicali (corpus linguistics)

NLTK: esempio di moduli

TASK	**MODULI**
Accesso a risorse linguistiche	`corpus`
Analisi lessicali	`text`
Elaborazione di stringhe	`tokenize`
Assegnazione parti del discorso	`tag`
Classificazione testi in base al topic	`classify`
Estrazione gruppi di token (entità nominate)	`chunk`
Metriche di valutazione	`metrics`

NLTK: esempio di moduli

TASK	MODULI
Accesso a risorse linguistiche	`corpus`
Analisi lessicali	`text`
Elaborazione di stringhe	`tokenize`
Assegnazione parti del discorso	`tag`
Classificazione testi in base al topic	`classify`
Estrazione gruppi di token (entità nominate)	`chunk`
Metriche di valutazione	`metrics`

- Passare il testo al modulo Text: `nome_variabile = Text(testo)`

Analisi lessicale: unità di analisi

Esempio: '*La tragedia di Bruxelles è importante perché è una tragedia europea, dell'intera Europa, del calcio europeo, dei tifosi europei, dell'organizzazione sportiva europea, dell'efficienza europea.*'

- TOKEN: unità minime del testo che includono parole, punteggiatura, numeri, sigle
  - *europea+europeo+europei+europea+europea*=5 token
- TIPI: parole graficamente distinte (EN: word types o types)
  - *europea, europeo, europei*=3 tipi
- LEMMI: forme citazionali dei token
  - *europeo*=1 lemma → per questo useremo spaCy

NLTK: contare

- `set(testo)`→ crea una lista ordinata alfabeticamente di parole distinte = tipi lessicali
  - `sorted(set(testo))` → per ordinare alfabeticamente
  - `len(set(testo))`→ per contare i tipi
  - `len(set(testo))/len(testo)`→ type/token ratio, indice di varietà e ricchezza lessicale
- `testo.count("parola")`→ conta la frequenza di una parola
  - `100 * testo.count("parola") / len(testo)` → calcola frequenza relativa

NLTK: distribuzione di frequenze

- `FreqDist(testo)` → funzione per ottenere la frequenza delle parole
  - `FreqDist(testo).most_common(50)` → lista delle 50 parole più frequenti nel testo
  - `FreqDist(testo).hapaxes()` → lista degli hapax legomena
  - `FreqDist(testo).plot(50)` → crea un grafico a linea con le frequenze delle 50 parole più frequenti

NLTK: concordanze

- `testo.concordance('parola')` → mostra le concordanze della parola cercata: elenco delle occorrenze di ciascuna parola con il suo contesto immediato
  - `testo.concordance(["parola1","parola2"])` → mostra le concordanze della sequenza formata dalle parole cercate
  - `testo.concordance('parola',40,10)` → mostra 10 concordanze in testi di 40 caratteri

NLTK: cercare con espressioni regolari

- `TokenSearcher(testo).findall('<pattern>')` → trova nel testo gli elementi che corrispondono al pattern
  - ogni token cercato deve essere posto tra parentesi angolari
  - `TokenSearcher(testo).findall("<token1><token2>")`

NLTK: collocazioni

- `testo.collocation_list(numero_collocazioni)` → estrae le collocazioni dal testo (di default sono 20)
- Collocazione = "una combinazione di parole soggetta a una restrizione lessicale, per cui la scelta di una specifica parola (il collocato) per esprimere un determinato significato, è condizionata da una seconda parola (la base) alla quale questo significato è riferito" - E. Jezek. *Lessico. Classi di parole, strutture, combinazioni*. Il Mulino, 2011.
    - es. *pioggia battente*

NLTK: similarità

- `testo.similar('parola')` → trova altre parole dello stesso testo che appaiono in contesti simili (20 di default)
  - `testo.similar('parola',5)` → trova 5 altre parole dello stesso testo che appaiono in contesti simili
- `testo.common_contexts(['parola1', 'parola2'])` → mostra i contesti simili in cui appaiono le due parole
  - es. al_mio

Un po’ di pratica

- Lezione9.ipynb

  https://colab.research.google.com/drive/1fGu-EALK2l_Y6zawp4LobKski_RziE2Q?usp=sharing

Esercizio 1

- Riprendendo il file usato per la parte pratica (vedi slide precedente), scrivere uno script che stampi in ordine alfabetico i tipi più lunghi di 10 caratteri
    - N.B. Usare una lista per memorizzare i tipi e poi stamparli

Esercizio 2

- Sempre usando il file precedente, trovare tutte le parole che finiscono con il suffisso “-mente”

Soluzioni esercizi

- https://colab.research.google.com/drive/1BgdJqydBjf229NWaU4ru5lGx_4C8-PEp?usp=sharing