Linguistica Computazionale - Progetto finale
A.A. 2024-2025

Obiettivo

Realizzazione di **due programmi** scritti in Python o su Notebook (Jupyter, Colab, ...) che utilizzino i moduli di NLTK per analizzare linguisticamente due corpora di testo inglese, confrontarli sulla base di alcuni indici statistici, ed estrarre da essi informazioni.

Creazione dei corpora

I due corpora devono essere creati rispondendo alle seguenti caratteristiche:

1. Devono essere in lingua **inglese**
2. Devono contenere almeno **5000 parole**
3. Ciascuno deve essere rappresentativo di uno specifico **genere testuale** (ad es. Testo giornalistico, prosa letteraria, blog, social media, articoli scientifici, ecc.)
4. Devono essere salvati in un file ciascuno di testo semplice con **codifica UTF-8**

Programmi da sviluppare

Ciascuno dei due programmi deve anzitutto **leggere** i file e **analizzarne il contenuto** linguisticamente *almeno* **fino al Part-of-Speech Tagging**.

- Se si sceglie di sviluppare il codice come un **file Python**, il programma deve prendere in input da **riga di comando** i file da analizzare.
- Se si sceglie di sviluppare il codice come **Notebook**, è accettabile che il/i file sia/siano specificati all'interno del codice utilizzando un ***path relativo***.

Programma 1

Il codice sviluppato deve prendere in input i **due corpora**, effettuare le operazioni di annotazione linguistica minime richieste (sentence splitting, tokenizzazione, PoS tagging) e la lemmatizzazione (utilizzando I PoS Tags), e produrre un confronto dei corpora rispetto a:

1. Numero di **frasi** e **token**;
2. **Lunghezza media delle frasi** in token e **lunghezza media dei token**, a eccezione della punteggiatura, in caratteri;
3. Distribuzione delle PoS nei primi 1000 token
    - Confrontare la distribuzione delle categorie grammaticali nei primi 1000 token di ciascun corpus;
4. Dimensione del **vocabolario e ricchezza lessicale** (Type-Token Ratio, TTR), calcolata per **porzioni incrementali di 200 token** fino ad arrivare a tutto il testo
    - i.e., i primi 200, i primi 400, i primi 600, ..., per tutti i token;
5. Numero di **lemmi distinti** (i.e., la dimensione del vocabolario dei lemmi)
6. Numero medio di **lemmi per frase**.
7. Distribuzione delle frasi positive e negative
    - Per classificare le frasi in POS e NEG è possibile utilizzare il classificatore di polarità visto a lezione (Notebook) o un altro classificatore di Sentiment addestrato dallo

studente o pre-addestrato. Se il classificatore NON È addestrato dallo studente, deve essere descritto;

8. Polarità del documento
    - Utilizzare il classificatore del punto 7. per determinare la polarità complessiva del corpus, sommando la polarità di tutte le frasi.

Programma 2

Il codice sviluppato deve prendere in input **un corpus**, effettuare le operazioni di **annotazione** richieste (sentence splitting, tokenizzazione, PoS tagging), ed estrarre le **seguenti informazioni**:

1. I top-50 Sostantivi, Avverbi e Aggettivi più frequenti (con relativa frequenza, ordinati per frequenza decrescente);

2. I top-20 n-grammi più frequenti (con relativa frequenza, e ordinati per frequenza decrescente)
    - Per n = [1, 2, 3]

3. I top 20 n-grammi di PoS più frequenti (con relativa frequenza, e ordinati per frequenza decrescente)
    - Per n = [1, 2, 3, 4, 5]

4. I top-10 bigrammi composti da Verbo e Sostantivo, ordinati per:
    a. frequenza decrescente, con relativa frequenza
    b. probabilità condizionata massima, e relativo valore di probabilità
    c. probabilità congiunta massima, e relativo valore di probabilità
    d. MI (Mutual Information) massima, e relativo valore di MI
    e. LMI (Local Mutual Information) massima, e relativo valore di MI
    f. Calcolare e stampare il numero di elementi comuni ai top-10 per MI e per LMI

5. Considerate le frasi con una **lunghezza compresa tra 10 e 20 token**, in cui almeno la **metà** (considerare la parte intera della divisione per due come valore) dei token occorre **almeno 2 volte nel corpus** (i.e., non è un hapax), si identifichino:
    a. La frase con la **media della distribuzione di frequenza** dei token più **alta**
    b. La frase con la **media della distribuzione di frequenza** dei token più **bassa**
    c. La frase con **probabilità più alta** secondo un **modello di Markov di ordine 2** costruito a partire dal corpus di input

        NB: la media della distribuzione di frequenza dei token è data dalla **somma delle frequenze** (nel corpus) dei token della frase **diviso il numero di token della frase**

6. Percentuale di Stopwords nel corpus rispetto al totale dei token

```
import nltk
from nltk.corpus import stopwords

nltk.download('stopwords')
print(stopwords.words('english'))
```

7. Frequenza di uso dei pronomi personali
    a. Numero di pronomi personali sul totale di token
    b. Numero medio di pronomi personali per frase

8. Estratte le **Entità Nominate** del testo, identificare **per ciascuna classe di NE i 15 elementi più frequenti**, ordinati per frequenza decrescente e con relativa frequenza.

Risultati attesi

Perché il progetto sia considerato idoneo, **devono essere consegnati** all'interno di una cartella compressa:

1. I **due corpora**, come file di testo

2. I **due programmi/notebook BEN COMMENTATI**
    - Nel caso si scelga di sviluppare programmi in **Python** (file .py), il **risultato** dell'esecuzione deve essere scritto in un **file di testo** (ben formattato) e **consegnato**. Quindi, dovranno essere consegnati tre file di output: uno per il primo programma, e due per il secondo.
    - Nel caso si scelga di sviluppare il codice attraverso un **Notebook**, questo deve essere **consegnato eseguito**. Per farlo, una volta eseguito il codice, è sufficiente esportarlo in formato .ipynb dal menù a tendina. Dovranno essere consegnati quindi 3 notebook: 1 per il primo programma, e 2 copie del notebook per il secondo programma, ciascuna eseguita su un corpus diverso.

Il codice/notebook **deve essere eseguibile** (prestare attenzione ai path assoluti) e replicare i risultati consegnati.
Il progetto **DEVE** essere svolto **INDIVIDUALMENTE**

Consegna

La consegna del progetto deve essere effettuata **almeno una settimana prima della data dell'esame ORALE dell'appello a cui volete partecipare**.
Il progetto va consegnato via e-mail con le seguenti modalità:

- **Allegato**: cartella compressa contenente il codice del progetto (cf. *Risultati Attesi*)
- **Oggetto**: [Progetto LC 24-25] <*Cognome Nome*>
- **Destinatari**:
    - alessandro.bondielli@unipi.it
    - alessandro.lenci@unipi.it