Tokenization Suddividere il corpus in token e generare un file di testo con un token per linea.	
NLTK	OPEN-NLP
Metodo **nltk.word_tokenize**	Modello **en-token.bin**
Lista di frequenza dei token Generare la lista di frequenza dei tokens del corpus (ordinata dal più frequente al meno frequente) in un file di testo con due colonne (separate da tabulazione): token e frequenza.	
NLTK	OPEN-NLP
Classe **nltk.probability.FreqDist** Metodo **most_common**	Implementare una lista di frequenza utilizzando una qualsiasi struttura dati di Java. Un esempio si trova al link https://stackoverflow.com/questions/29122394/word-frequency-count-java-8
Concordanze Le concordanze si visualizzano nel formato KWIC: una linea per ogni occorrenza, un numero di caratteri predefinito a sinistra e a destra della parola cercata. Ricercare le occorrenze della parola "life" nel corpus e generare un file di testo con 3 colonne (separate da tabulazione): contesto sinistro, occorrenza, contesto destro. I contesti sinistro e destro dovranno avere 100 caratteri (metodo NLTK) oppure 20 parole (metodo Open NLP) ciascuno.	
NLTK	OPEN-NLP
Classe **nltk.text.Text** Metodo **nltk.text.concordance_list**	Implementare la ricerca di concordanze utilizzando l'array di token derivato dalla Tokenization.

Tokenization Suddividere il corpus in token e generare un file di testo con un token per linea.	
NLTK	OPEN-NLP
Metodo **nltk.word_tokenize**	Modello **en-token.bin**
Lista di frequenza dei token Generare la lista di frequenza dei tokens del corpus (ordinata dal più frequente al meno frequente) in un file di testo con due colonne (separate da tabulazione): token e frequenza.	
NLTK	OPEN-NLP
Classe **nltk.probability.FreqDist** Metodo **most_common**	Implementare una lista di frequenza utilizzando una qualsiasi struttura dati di Java. Un esempio si trova al link https://stackoverflow.com/questions/29122394/word-frequency-count-java-8
Named Entity Recognition Annotare le Named Entities nel corpus. Generare un file di testo con una riga per ogni token e 2 Colonne (separate da tabulazione): token, named_entity. Quest'ultima colonna, se generata con open-NLP, dovrà specificare il tipo di named entity (almeno Organization, Person, Location).	
NLTK	OPEN-NLP (limitatamente ai primi 10.000 tokens)
Classe **nltk.tree.Tree** Metodo **nltk.ne_chunk** Metodo **Tree.label()**	Classe **NameFinderME** con i 7 modelli addestrati per NER: **en-ner-location.bin**, **en-ner-organization.bin**, ecc. Concatenare i valori sulla seconda colonna nel caso che più finder annotino lo stesso token.

Tokenization Suddividere il corpus in token e generare un file di testo con un token per linea.	
NLTK	OPEN-NLP
Metodo **nltk.word_tokenize**	Modello **en-token.bin**
Lista di frequenza dei token Generare la lista di frequenza dei tokens del corpus (ordinata dal più frequente al meno frequente) in un file di testo con due colonne (separate da tabulazione): token e frequenza.	
NLTK	OPEN-NLP
Classe **nltk.probability.FreqDist** Metodo **most_common**	Implementare una lista di frequenza utilizzando una qualsiasi struttura dati di Java. Un esempio si trova al link https://stackoverflow.com/questions/29122394/word-frequency-count-java-8
PoS-Tagging Eseguire il PoS-Tagging del corpus. Generare un file di testo con una riga per ogni token e 2 colonne (separate da tabulazione): forma del token, PoS.	
NLTK	OPEN-NLP (limitatamente ai primi 10.000 tokens)
Metodo **nltk.pos_tag**	Classe **PoSTaggerME** con il modello addestrato **en-pos-maxent.bin**