## Introduzione

Il seguente documento illustra il funzionamento di un insieme di moduli Python utilizzati per l’automazione del processo di generazione di tassonomia e regole linguistiche per ogni ramo della stessa a partire da un dominio generico come può essere “Automotive” o “Cloud services”. Il codice sfrutta diverse librerie, tra cui xml.etree.ElementTree, re e pathlib, oltre a interagire con servizi esterni come un modulo GPT (Generative Pre-trained Transformer) per le API di OpenAI e un client per interagiro con il servizio di analisi del linguaggio naturale di Expert.AI.

Il flusso di esecuzione inizia con l'acquisizione di una tassonomia relativa a un dominio di interesse mediante l'interazione con il modulo GPT. Successivamente, vengono generate regole linguistiche per ogni ramo della tassonomia, basate sull'analisi del linguaggio naturale di Expert.AI. L'intero processo è orchestrato dalla funzione principale main, che consente all'utente di specificare il dominio di interesse per la generazione automatica di tassonomie e regole.

Il codice presenta una struttura modulare e un'interessante ibridazione di tecnologie NLP proprietarie di Expert.AI e LLM, facilitando l'automazione della categorizzazione semantica in contesti linguistici specifici.

Di seguito la documentazione dettagliata divisa per moduli.

## **Modulo rules\_generation.py:**

Il file rules\_generation.py contiene funzioni per generare regole linguistiche basate su analisi linguistiche di testi specifici. Esso si avvale dell'API della Platform di Expert.AI per l'analisi semantica di testi e include funzioni per la generazione di regole linguistiche specifiche per un dominio.

### Funzione is\_relevant\_pos

#### Descrizione:

Verifica se un dato token è rilevante in base alla sua parte del discorso (POS). La funzione determina la rilevanza confrontando il tipo di parte del discorso del token con una lista di tipi di POS da ignorare.

#### Parametri:

* token (dict): Un dizionario rappresentante il token linguistico, con informazioni come tipo di POS (type).
* pos\_to\_ignore (list, opzionale): Una lista di tipi di POS da ignorare. Se non specificato, vengono utilizzati i tipi di POS predefiniti da ignorare (default\_pos\_to\_ignore).

#### Output:

* True se il token è rilevante, False altrimenti.

#### Dettagli:

1. La funzione verifica se la lista pos\_to\_ignore è vuota. Se vuota, utilizza la lista predefinita default\_pos\_to\_ignore.
2. Confronta il tipo di POS del token con la lista di tipi di POS da ignorare.
3. Restituisce True se il tipo di POS del token non è nella lista di quelli da ignorare, altrimenti restituisce False.

#### Esempio di Utilizzo:

python

token\_to\_check = {"type": "NOUN", "lemma": "dog"}

ignore\_list = ["CON", "PNT"]

result = is\_relevant\_pos(token\_to\_check, pos\_to\_ignore=ignore\_list)

print(result)

#### Output dell'Esempio:

graphql

True

### Funzione get\_relevant\_parts\_of\_speech

#### Descrizione:

Estrae e restituisce i token rilevanti da una data frase attraverso l'analisi linguistica. La funzione utilizza il servizio di analisi di Essex per ottenere informazioni dettagliate sulla struttura linguistica della frase.

#### Parametri:

* sentence (str): La frase da analizzare per ottenere i token rilevanti.

#### Output:

* Una lista di token rilevanti estratti dalla frase.

#### Dettagli:

1. La funzione inizia creando un dizionario data contenente i parametri necessari per l'analisi linguistica, come il testo della frase, le analisi richieste (analysis), e le features richieste (features).
2. Utilizza il servizio di analisi di Essex (get\_essex\_analysis) per ottenere i risultati dell'analisi linguistica sulla frase.
3. Estrae i token risultanti dall'analisi.
4. Filtra i token rilevanti utilizzando la funzione is\_relevant\_pos.
5. Se non ci sono token rilevanti, crea dei token "fabbricati" basati sulle parole della frase che hanno una lunghezza maggiore di 4 caratteri.
6. Restituisce la lista completa di token rilevanti.

#### Esempio di Utilizzo:

python

sentence\_to\_analyze = "I have a new Galaxy smartphone with seven cameras."

relevant\_tokens = get\_relevant\_parts\_of\_speech(sentence\_to\_analyze)

print(relevant\_tokens)

#### Output dell'Esempio:

bash

[

{"syncon": 123, "score": 0.75, "start": 10, "end": 15, "type": "NOUN", "lemma": "Galaxy"},

{"syncon": 456, "score": 0.85, "start": 43, "end": 49, "type": "NUM", "lemma": "seven"}

# ... altri token rilevanti ...

]

Questa funzione è fondamentale per estrarre informazioni linguistiche rilevanti da una frase, che possono poi essere utilizzate per la generazione di regole linguistiche personalizzate o per altri scopi specifici del progetto.

### Funzione generate\_rule

#### Descrizione:

Genera una regola linguistica basata su una lista di token, un dominio e una descrizione del dominio. La regola viene formattata secondo uno specifico linguaggio di regole di Expert.AI.

#### Parametri:

* domain (dict): Un dizionario contenente informazioni sul dominio, come il nome (NAME) del dominio e la sua descrizione (DESCRIPTION).
* tokens (list): Una lista di token rilevanti ottenuti da un testo attraverso analisi linguistica.
* domain\_description (str): La descrizione del dominio, usata per generare label della regola.

#### Output:

* Una stringa contenente la regola linguistica formattata.

#### Dettagli:

1. La funzione inizia controllando se la lista di token (tokens) non è vuota.
2. Se la lista di token non è vuota, inizia la creazione della regola formattata.
3. Per ogni token nella lista di token:
   * Verifica se il token ha un syncon valido utilizzando la funzione get\_syncon.
   * Se il token ha un syncon, determina il tipo di token e aggiunge le informazioni corrispondenti alla regola.
     1. Se il token è di un tipo variabile (address, email, company ecc), aggiunge alle alternative all’interno del MANDATORY anche TYPE, oltre agli operand lemma e syncon
   * Se il token non ha un syncon, crea un operand di tipo PATTERN basato sul lemma del token.
   * Aggiunge un operator con distanza arbitraria dopo ogni operand (tranne l'ultimo) per separarli.
4. La funzione restituisce la regola completa.

#### Esempio di Utilizzo:

python

domain\_info = {"NAME": "example\_domain", "DESCRIPTION": "Example Domain Description"}

sample\_tokens = [{"lemma": "example", "syncon": 123, "type": "NOUN"}, {"lemma": "test", "syncon": 456, "type": "VERB"}]

description = "Example rule for testing purposes"

rule = generate\_rule(domain\_info, sample\_tokens, description)

print(rule)

#### Output dell'Esempio:

scss

DOMAIN[Example\_rule\_for\_testing\_purposes](example\_domain) {

MANDATORY{

LEMMA("example"),

SYNCON(123) // example

}

<:4>

MANDATORY{

LEMMA("test"),

SYNCON(456) // test,

TYPE(COM)

}

}

### Funzione generate\_rules\_file

#### Descrizione:

Genera un file di regole linguistiche per un dominio specifico. La funzione accetta un dominio e una lista di possibili descrizioni, generando un file di regole basato su tali informazioni.

#### Parametri:

* domain (dict): Un dizionario rappresentante il dominio per il quale generare le regole. Contiene informazioni come il nome del dominio (NAME) e la descrizione (DESCRIPTION).
* possible\_descriptions (list, opzionale): Una lista di descrizioni alternative per il dominio. Se non specificato, viene utilizzata solo la descrizione predefinita del dominio.

#### Output:

* Genera un file di regole nel formato specificato.

#### Dettagli:

1. La funzione prepara la lista possible\_descriptions, che include la descrizione predefinita del dominio e tutte le descrizioni alternative fornite.
2. Per ciascuna descrizione, chiama la funzione generate\_rule per ottenere le regole linguistiche corrispondenti.
3. Combina tutte le regole in un unico testo nel formato richiesto.
4. Scrive il testo delle regole in un file con il nome del dominio nella cartella specificata (rules\_out\_path).

#### Esempio di Utilizzo:

python

example\_domain = {"NAME": "example", "DESCRIPTION": "Sample domain for illustration"}

alternative\_descs = ["Additional description 1", "Additional description 2"]

generate\_rules\_file(example\_domain, possible\_descriptions=alternative\_descs)

### Funzione get\_all\_domains

#### Descrizione:

Ottiene una lista di domini da una stringa di tassonomia XML. La funzione analizza la tassonomia per estrarre informazioni sui domini, inclusi il nome e la descrizione di ciascun dominio.

#### Parametri:

* taxonomy\_string (str): Una stringa contenente la rappresentazione XML della tassonomia.

#### Output:

* Una lista di dizionari, ciascuno contenente le informazioni di un dominio. Ogni dizionario include il nome del dominio (NAME) e la descrizione (DESCRIPTION).

#### Dettagli:

1. La funzione utilizza il modulo xml.etree.ElementTree per analizzare la stringa XML e ottenere un elemento radice (root) della tassonomia.
2. Itera su tutti gli elementi DOMAIN all'interno della tassonomia, estraendo il nome e la descrizione di ciascun dominio.
3. Aggiunge le informazioni del dominio a una lista di dizionari (domains\_list).
4. Restituisce la lista dei domini ottenuti dalla tassonomia.

#### Esempio di Utilizzo:

python

taxonomy\_example = """<DOMAINTREE>

<DOMAIN NAME="example1" DESCRIPTION="Description of Example 1">

<DOMAIN NAME="sub\_example1" DESCRIPTION="Subdomain 1 of Example 1" />

</DOMAIN>

<DOMAIN NAME="example2" DESCRIPTION="Description of Example 2" />

</DOMAINTREE>"""

domains\_info = get\_all\_domains(taxonomy\_example)

print(domains\_info)

Questa funzione è utile per estrarre informazioni sui domini da una tassonomia XML, consentendo l'utilizzo successivo di tali informazioni per scopi come la generazione automatica di regole linguistiche o l'analisi di domini specifici.

### Funzione generate\_rule\_files\_per\_domain

#### Descrizione:

Genera regole linguistiche per ciascun dominio nella lista fornita. Per ogni dominio, ottiene alternative descrizioni utilizzando la funzione get\_alternative\_descriptions e genera file di regole utilizzando la funzione generate\_rules\_file.

#### Parametri:

* domains (list): Una lista di dizionari, ciascuno contenente informazioni su un dominio. Ogni dizionario deve avere le chiavi 'NAME' e 'DESCRIPTION' per rappresentare il nome e la descrizione del dominio.

#### Dettagli:

1. La funzione itera su ciascun dominio nella lista domains.
2. Per ogni dominio, stampa un messaggio indicante che sta generando le regole per quel dominio.
3. Utilizza la funzione get\_alternative\_descriptions per ottenere alternative descrizioni per il dominio corrente.
4. Chiama la funzione generate\_rules\_file per generare i file di regole per il dominio, utilizzando le alternative descrizioni ottenute.

#### Esempio di Utilizzo:

python

sample\_domains = [

{"NAME": "example1", "DESCRIPTION": "Description of Example 1"},

{"NAME": "example2", "DESCRIPTION": "Description of Example 2"}

]

generate\_rule\_files\_per\_domain(sample\_domains)

### Funzione main

#### Descrizione:

La funzione main rappresenta il punto di ingresso principale per il processo di generazione di tassonomia e regole linguistiche. Richiede all'utente di specificare un dominio generico, ottiene la tassonomia associata a quel dominio e genera regole linguistiche per ciascun ramo della tassonomia.

#### Parametri:

* general\_domain (str): Il nome del dominio generico fornito dall'utente.

#### Dettagli:

1. Richiede all'utente di inserire il nome di un dominio generico tramite l'input.
2. Ottiene la tassonomia associata al dominio specificato utilizzando la funzione get\_taxonomy.
3. Estrae informazioni sui sotto-domini dalla tassonomia utilizzando la funzione get\_all\_domains.
4. Chiama la funzione generate\_rule\_files\_per\_domain per generare regole linguistiche per ciascun sotto-dominio.

#### Esempio di Utilizzo:

python

main("cloud services")

Il file contiene inoltre funzioni per la pulizia del testo, la generazione di regole e la gestione delle entità specifiche, oltre a funzioni per ottenere domini da una tassonomia XML e generare regole per ciascun dominio.

## **Modulo gpt.py:**

Il file gpt.py include funzioni per interagire con l'API di OpenAI, in particolare il modello GPT-3.5-turbo. Le funzioni in questo file sono utilizzate per ottenere alternative di descrizioni e generare una tassonomia dettagliata attraverso interazioni con il modello.

### Funzione get\_alternative\_descriptions

#### Descrizione:

La funzione get\_alternative\_descriptions utilizza il modello GPT-3.5-turbo per generare una lista di espressioni linguistiche alternative per un concetto specifico fornito come input. La risposta è formattata in modo che ciascuna espressione sia preceduta da un trattino (-) per facilitarne la parserizzazione.

#### Parametri:

* original\_desc (str): Il concetto originale per il quale generare espressioni alternative.

#### Dettagli:

1. Invia una richiesta di completamento al modello GPT-3.5-turbo utilizzando le API di OpenAI.
2. Analizza la risposta del modello per ottenere la lista di espressioni alternative.
3. Restituisce una lista di espressioni.

#### Esempio di Utilizzo:

python

original\_description = "Street address"

alternative\_expressions = get\_alternative\_descriptions(original\_description)

print(alternative\_expressions)

Questa funzione ha l’obiettivo di generare variazioni espressive per il concetto di partenza in modo tale da generare regole più varie per un determinato concetto.

### Funzione get\_taxonomy

#### Descrizione:

La funzione get\_taxonomy utilizza il modello GPT-3.5-turbo per generare una tassonomia dettagliata e ramificata di concetti relativi a un dominio specifico fornito come input. La risposta è formattata in formato XML, seguendo uno specifico formato di domini nidificati.

#### Parametri:

* domain (str): Il dominio per il quale generare la tassonomia.
* minimum\_elements (int, predefinito a 70): Il numero minimo di elementi (domini) richiesti nella tassonomia.

#### Dettagli:

1. Invia una richiesta di completamento al modello GPT-3.5-turbo utilizzando le API di OpenAI.
2. Gestisce iterativamente le risposte finché la tassonomia generata non termina con </DOMAINTREE>.
3. Formatta la risposta in formato XML, aggiungendo l'intestazione XML.
4. Verifica se il numero di domini generati è uguale o superiore al numero minimo richiesto.
5. Se soddisfatto, salva la tassonomia in un file e restituisce la tassonomia generata.
6. Se il numero minimo di elementi non è raggiunto, offre all'utente la scelta di mantenere o ritentare la generazione.

#### Esempio di Utilizzo:

python

taxonomy\_str = get\_taxonomy("cloud services", minimum\_elements=10)

print(taxonomy\_str)

Questa funzione semplifica il processo di produzione di una tassonomia dettagliata per un particolare dominio, con la possibilità di specificare il numero minimo di elementi richiesti.

## **Modulo runner\_client.py:**

Il file runner\_client.py fornisce una funzione get\_essex\_analysis che invia dati a un servizio esterno tramite richieste HTTP POST e restituisce i risultati dell'analisi.

Questo file è necessario per ottenere le analisi linguistiche usate per generare regole linguistiche nel file rules\_generation.py.

### Funzione get\_essex\_analysis

#### Descrizione:

La funzione get\_essex\_analysis invia una richiesta HTTP POST al servizio di analisi linguistica con i dati forniti come input. Restituisce l'analisi risultante se la richiesta ha successo (status code 200), altrimenti stampa un messaggio di errore indicando il codice di stato della risposta.

#### Parametri:

* data (dict): Dati da inviare nella richiesta POST.

#### Dettagli:

1. Utilizza la libreria requests per inviare una richiesta POST al servizio esterno specificato dall'URL con i dati forniti.
2. Verifica se la richiesta ha successo controllando il codice di stato HTTP.
3. Se la risposta ha successo (status code 200), converte la risposta JSON in un dizionario e restituisce l'analisi risultante.
4. Se la risposta non ha successo, stampa un messaggio di errore indicando il codice di stato della risposta.

#### Esempio di Utilizzo:

python

data\_to\_analyze = {

"projectId": "example\_project",

"text": "This is a sample text for analysis."

}

result\_analysis = get\_essex\_analysis(data\_to\_analyze)

print(result\_analysis)

Questa funzione fornisce un'interfaccia per interagire con il servizio di analisi del linguaggio naturale di Expert.AI.