Manuale di utilizzo degli script relativi alla anonimizzazione

Tutti gli script sono disponibili nella repository github:

https://github.com/marco-caputo/anonimizzatore-digit-care

Indice

split.py

Lo script `split.py` serve per leggere uno o più file JSON contenenti diari di equipe e suddividere il relativo contenuto in più file JSON separati, ognuno contenente la porzione di testo relativa ad un singolo paziente.

Formato di Input

Il JSON in input può contenere una sezione `anagrafica` (lista di dizionari) contenente i dati dei pazienti coinvolti e deve contenere una sezione `testi` (lista di dizionari) contenente i testi non strutturati con diversi pazienti.

Deve quindi rispettare il seguente formato:

```
{
  "anagrafica": [
    {
      "nome": "CAMILLO",
      "cognome": "BENSO",
      "ragSoc": "CAMILLO BENSO CONTE DI CAVOUR",
      "nazione_nascita": "ITALIA",
      "luogo_nascita": "TORINO",
      "data_nascita": "10-07-1810",
      "nazione_residenza": "ITALIA",
      "luogo_residenza": "TORINO",
      "prov_residenza": "TO",
      "idAna": 123
    },
    ...
  ],
  "testi": [
    {
      "tipo": "Diario d'equipe",
      "data": "02-01-2026 17:44",
      "testo": "BENSO: Testo relativo a Camillo..."
    },
    ...
  ]
}
```

```
Rosso: parte obbligatoria
Verde: parte facoltativa
```

Nota: non è necessario che tutte le anagrafiche relative ai pazienti che appaiono nei testi siano elencate, né che siano presenti tutti i campi indicati per ogni dizionario anagrafico.

Formato di Output

I vari file JSON in output conterranno una sezione `anagrafica` (singolo dizionario) contenente il cognome rilevato dal testo e gli eventuali altri dati del paziente presenti nel file in input, e una sezione `testi` (lista di dizionari) contenente i frammenti relativi al paziente dei testi non strutturati in input.

Rispetta quindi il seguente formato:

BENSO_856f2ea4b.json

```
{
  "anagrafica": {
      "nome": "CAMILLO",
      "cognome": "BENSO",
      "ragSoc": "CAMILLO BENSO CONTE DI CAVOUR",
      "nazione_nascita": "ITALIA",
      "luogo_nascita": "TORINO",
      "data_nascita": "10-07-1810",
      "nazione_residenza": "ITALIA",
      "luogo_residenza": "TORINO",
      "prov_residenza": "TO",
      "idAna": 123
  },
  "testi": [
    {
      "tipo": "Diario d'equipe",
      "data": "02-01-2026 17:44",
      "testo": "Testo relativo a Camillo..."
    },
    ...
  ]
}
```

Rosso: parte obbligatoria
Verde: parte facoltativa

Utilizzo

Per richiamare la procedura utilizzare il comando in una delle sue versioni:

```
python split.py -o "cartella_output" "input1.json" "input2.json" ...
```

```
python split.py -o "cartella_output" "cartella_input_1" "cartella_input_2" ...
```

Esempio:

```
python split.py -o "C:\Users\Mario\Desktop\output"
"C:\Users\Mario\Desktop\diario1.json"
```

```
python3 split.py -o "~/Desktop/output" "~/Desktop/diario1.json"
```

merge.py

Lo script `merge.py` serve per leggere una cartella contenente più file JSON relativi a singoli pazienti (struttura anagrafica + testi), e unificare automaticamente i file che si riferiscono allo stesso paziente, producendo un unico nuovo file JSON per ciascun paziente identificato tra i file forniti in input.

Formato di Input

La cartella in input deve contenere dei file JSON con una sezione `anagrafica` (singolo dizionario) contenente almeno un dato tra cognome e idAna e deve contenere una sezione `testi` (lista di dizionari) contenente i testi non strutturati relativi al singolo paziente.

Devono quindi rispettare il seguente formato:

```
{
  "anagrafica": {
      "nome": "CAMILLO",
      "cognome": "BENSO",
      "ragSoc": "CAMILLO BENSO CONTE DI CAVOUR",
      "nazione_nascita": "ITALIA",
      "luogo_nascita": "TORINO",
      "data_nascita": "10-07-1810",
      "nazione_residenza": "ITALIA",
      "luogo_residenza": "TORINO",
      "prov_residenza": "TO",
      "idAna": 123
  },
  "testi": [
    {
      "tipo": "Diario d'equipe",
      "data": "02-01-2026 17:44",
      "testo": "Testo relativo a Camillo..."
    },
    ...
  ]
}
```

```
Rosso: parte obbligatoria
Giallo: almeno una
Verde: parte facoltativa
```

Nota: più file possono riferirsi allo stesso paziente anche se una anagrafica contiene meno informazioni dell'altra. Tuttavia, se nella cartella in input esistono due file attribuiti a pazienti aventi lo stesso cognome *A* ma idAna diversi e un file contenente unicamente l'informazione del cognome valorizzato ad *A*, quest'ultimo potrà essere unito indistintamente a uno qualsiasi dei due possibili pazienti.

Formato di Output

I vari file JSON in output conterranno una sezione `anagrafica` (singolo dizionario) contenente l'unione dei vari dati anagrafici raccolti nei file in input, e una sezione `testi` (lista di dizionari) contenente tutti i testi raccolti relativi al paziente dai file in input.

Ciascuno dei file prodotti avrà il seguente formato:

```
{
  "anagrafica": {
      "nome": "CAMILLO",
      "cognome": "BENSO",
      "ragSoc": "CAMILLO BENSO CONTE DI CAVOUR",
      "nazione_nascita": "ITALIA",
      "luogo_nascita": "TORINO",
      "data_nascita": "10-07-1810",
      "nazione_residenza": "ITALIA",
      "luogo_residenza": "TORINO",
      "prov_residenza": "TO",
      "idAna": 123
  },
  "testi": [
    {
      "tipo": "Diario d'equipe",
      "data": "02-01-2026 17:44",
      "testo": "Testo relativo a Camillo..."
    },
    ...
  ]
}
```

```
Rosso: parte obbligatoria
Giallo: almeno una
Verde: parte facoltativa
```

Utilizzo

Per richiamare la procedura utilizzare il comando:

```
python merge.py “cartella_input_e_output”
```

Esempio:

```
python split.py "C:\Users\Mario\Desktop\output_pazienti"
```

```
python3 split.py "/Users/mario/Desktop/output_pazienti"
```

Logica di Fusione

Due file vengono considerati riferiti allo stesso paziente quando:

- Le anagrafiche sono esattamente uguali, oppure
- Una anagrafica contiene un sottoinsieme degli stessi campi con gli stessi valori.

Al termine dell'esecuzione verrà prodotto un nuovo file JSON per ciascun paziente e i file originali verranno eliminati.

Esempio valido di fusione:

File in input		Output

<pre>{
  "anagrafica": {
  "cognome": "BENSO"
  },
  "testi": [
  {
  "testo": "Testo 1…"
  }
  ]
}</pre>
	<pre>{
	  "anagrafica": {
	  "nome": "CAMILLO",
	  "cognome": "BENSO",
	  "idAna": 123
	  },
	  "testi": [
	  {
	  "testo": "Testo 2…"
	  },
	  {
	  "testo": "Testo 3…"
	  },
	  ]
	}</pre>
		<pre>{
		  "anagrafica": {
		  "nome": "CAMILLO",
		  "cognome": "BENSO",
		  "idAna": 123
		  },
		  "testi": [
		  {
		  "testo": "Testo 1…"
		  },
		  {
		  "testo": "Testo 2…"
		  },
		  {
		  "testo": "Testo 3…"
		  },
		  ]
		}</pre>

anonymize.py

Vero e proprio comando per l'anonimizzazione. Comprende l'anonimizzazione completa dei testi utilizzando modello NER e regole aggiuntive di matching.

Formato di Input

Ogni file in input può avere estensione **.txt**, **.docx**, **.pdf** o **.json**.
Nel caso in cui l'input sia un file JSON, esso può contenere una sezione `anagrafica` facoltativa e dovrebbe contenere una lista `testi` di stringhe da anonimizzare.
In tutti gli altri casi, l'intero contenuto del file verrà elaborato per l'anonimizzazione.

Il formato atteso dei singoli file JSON è il seguente:

```
{
  "anagrafica": {
      "nome": "CAMILLO",
      "cognome": "BENSO",
      "ragSoc": "CAMILLO BENSO CONTE DI CAVOUR",
      "nazione_nascita": "ITALIA",
      "luogo_nascita": "TORINO",
      "data_nascita": "10-07-1810",
      "nazione_residenza": "ITALIA",
      "luogo_residenza": "TORINO",
      "prov_residenza": "TO",
      "idAna": 123
  },
  "testi": [
    {
      "tipo": "Tipo di testo",
      "data": "02-01-2026 17:44",
      "testo": "Testo contenente dati personali..."
    },
    ...
  ]
}
```

```
Rosso: parte obbligatoria
Verde: parte facoltativa
```

Note:

- Tutte le informazioni aggiuntive relative ai testi (`tipo`, `data`) verranno mantenute nei file di output.
- Le informazioni contenute in `anagrafica` saranno utilizzate per anonimizzazione aggiuntiva basata su match di stringhe.
- Includendo l'`idAna` in `anagrafica` , questo sarà utilizzato per nominare i file di output anonimizzati e verrà incluso come dato all'interno degli stessi (trattasi dell'unico dato anagrafico che rimane negli output).

Formato di Output

Tutti i testi presenti nei file in input riferiti allo stesso paziente vengono salvati in un unico file JSON di nome:

- `<idAna>.json`
  se `idAna` è fornito nelle informazioni anagrafiche

- `UKNOWN_xxxxxxxx.json`
  se `idAna` non è fornito nelle informazioni anagrafiche ma sono comunque disponibili altre informazioni anagrafiche utili ad eseguire un raggruppamento, dove `xxxxxxxx` è una sequenza esadecimale casuale.

- `text_x.json`
  se nè `idAna` nè altre informazioni anagrafiche sono state fornite per un testo, dove `x` è un incrementale relativo al singolo testo presente nel file.

Il formato effettivo dei file JSON in output è il seguente:

```
{
  "idAna": <id o null se non fornito>
  "testi": [
    {
      "tipo": "Tipo di testo",
      "data": "02-01-2026 17:44",
      "testo": "Testo anonimizzato..."
    },
    ...
  ]
}
```

```
Rosso: parte obbligatoria
Verde: parte facoltativa
```

Nota: i testi possono essere salvati in tanti file diversi attraverso l'impostazione `DEFAULT_OUTPUTS_IN_SINGLE_FILE`.

Utilizzo

La forma generale del comando prevede:

```
python anonymize.py [input] [opzioni]
```

- `input` : una o più stringhe che rappresentano percorsi ai file o cartelle che contengono i file da anonimizzare.
- `opzioni` : lista di opzioni per anonimizzazione e salvataggio (vedere sotto).

Esempio:

```
python anonymize.py "C:\Users\Mario\Desktop\documenti_pazienti"
"C:\Users\Mario\Desktop\documento1.json" --output-dir
"C:\Users\Mario\Desktop\documenti_anonimizzati"
```

```
python3 anonymize.py "/home/mario/Desktop/documenti_pazienti"
"/home/mario/Desktop/documento1.json" --output-dir
"/home/mario/Desktop/documenti_anonimizzati"
```

Opzioni:

- `--output-dir`
  Usato per indicare il percorso della directory di output. Se non specificato e l'input fornito è unico, l'output viene salvato nella stessa cartella dell'input.

- `--text`
  Usato in alternativa agli input per fornire direttamente il testo da anonimizzare.
  In questo caso, se non viene specificata una directory di output, il testo anonimizzato viene stampato a video.

  Esempio:

  

  ```
  python anonimize.py --text "Mario Rossi nato a Milano il
  01-01-1980"
  ```

- `--entities`
  Usato per specificare le entità da anonimizzare. Se non specificato, vengono usate le entità di default (vedi sezione configurazione).

  Esempio:

  

  ```
  python anonimize.py "file.txt" --entities "PER" "LOC" "DATE"
  ```

- `--per-matching`
  Indica il livello di matching aggiuntivo nel range `[0, 2]` per le entità di tipo PER tramite dizionari di nomi e cognomi. Se non specificato, vengono usate le entità di

default. Vedi sezione configurazione per ulteriori dettagli sui livelli.

Esempio:

```
python anonimize.py "file.txt" --per-matching 2
```

- `--personal-data`
  Permette di fornire il percorso ad un file JSON contenente dati personali specifici da anonimizzare tramite matching di stringhe, in sostituzione al dizionario anagrafica che può essere indicato nei file JSON.

  Esempio:

```
python anonimize.py "file.txt" --personal-data
"C:\Users\Mario\Desktop\dizionario.json”
```

- `--gui`
  Avvia l’interfaccia grafica dell’anonimizzatore. Nessun altro input o opzione viene considerata.

  Esempio:

```
python anonimize.py --gui
```

Configurazioni

Il comportamento degli script è controllato tramite una serie di parametri definiti nel file di configurazione `config.py`.

Impostazioni Generali	
`DEFAULT_NER_MODEL`	Specifica il percorso del modello spaCy utilizzato per il riconoscimento delle entità nominate (NER).
`DEFAULT_ENTITIES`	Lista delle entità che vengono anonimizzate di default. Questa lista può essere sovrascritta tramite il parametro `--entities` da riga di comando. Le entità disponibili sono: `PATIENT`, `PER`, `LOC`, `ORG`, `FAC`, `GPE`, `PROV`, `DATE`, `EVENT`, `NORP`, `AGE`, `PRODUCT`, `WORKS_OF_ART`, `CODE`, `MAIL`, `PHONE`, `URL`.
`DEFAULT_EXTRA_PER_MATCHING_LEVEL`	Controlla il livello di pattern matching aggiuntivo per le entità di tipo PER (oltre al riconoscimento NER): `0` → nessun matching aggiuntivo `1` → riconosce nomi e cognomi non ambigui (che non appaiono nel dizionario italiano) quando compaiono come nomi propri (maiuscoli o con iniziale maiuscola e non preceduti da una preposizione). `2` → riconosce nomi non ambigui in qualsiasi forma e nomi/cognomi ambigui quando compaiono come nomi propri.
`DEFAULT_OUTPUTS_IN_SINGLE_FILE`	Determina il formato dei file di output: `True` → se nel JSON sono presenti più testi relativi allo stesso paziente, vengono salvati in un unico file JSON. `False` → ogni singolo testo viene sempre salvato in un file TXT separato.
Impostazioni per Multiprocessing	
`MULTI_PROCESSING`	Abilita o disabilita l'elaborazione parallela del modello NER e delle regole. Se impostato a `True` più testi vengono anonimizzati contemporaneamente in base al parametro successivo.
`P_CORES`	Numero di core utilizzati in modalità multiprocessing. E' consigliabile impostare il valore al numero di Performance Core della macchina in utilizzo.
Impostazioni sul Formato dei JSON in Input	
`PATIENT_DATA_FIELDS`	Definisce il nome dei campi principali del JSON. Di default: `anagrafica` → dizionario con i dati personali del paziente. `testi` → lista dei testi associati al paziente.

SINGLE_TEXT_FIELDS	Definisce la struttura di ciascun testo, di default: `tipo` → tipologia del documento `data` → data del documento `testo` → testo originale `testo_anonimizzato` → testo anonimizzato (usato per testing) `lista_entita` → lista delle entità riconosciute (usato per testing)
PERSONAL_DATA_FIELDS	Definisce i campi dell'anagrafica del paziente.
PERSONAL_DATA_FORMAT	Mappa ogni campo anagrafico all'etichetta NER corrispondente.
SINGLE_ENTITY_FIELDS	Definisce la struttura di una singola entità riconosciuta nel testo (usato per testing).

Categorie di dato

Le categorie di dati considerate nell'anonimizzazione sono le seguenti. Le entità contrassegnate dal ✔ indicano le etichette applicate secondo le configurazioni di default.

	Etichetta	Descrizione	Esempi
✔	*[PATIENT]*	Nomi e cognomi propri del paziente.	"Marco Rossi", "G. Verdi", "Luca", "Bianchi"
✔	*[PER]*	Nomi e cognomi propri di persone o famiglie.	"Marco Rossi", "G. Verdi", "Luca", "Bianchi"
✔	*[LOC]*	Nomi di luoghi o aree che non rappresentano entità geopolitiche.	"Via Roma, 2", "Parco nazionale del Gran Sasso".
✔	*[ORG]*	Nomi propri di organizzazioni, enti, istituzioni pubbliche o private, società, cooperative, associazioni, fondazioni, servizi territoriali o strutture sanitarie.	"Ser.D. di Bologna", "Cooperativa Nuovi Orizzonti", "Ministero della Salute", "Bar dello Sport"
✔	*[FAC]*	Strutture, edifici o plessi.	"Ospedale San Raffaele", "Ponte di Rialto", "Aeroporto di Fiumicino"
✔	*[GPE]*	Nomi di entità geopolitiche, quali località, comuni, regioni e nazioni.	"Camerino", "Marche", Francia"
✔	*[NORP]*	Nazionalità, orientamenti religiosi o politici	"tedesco", "cattolico", "comunista"
✔	*[AGE]*	Età di persone.	"23 anni", "40enne"
✔	*[DATE]*	Date e riferimenti assoluti di tempo.	"20/03/2021", "5 maggio", "2020"
✔	*[EVENT]*	Eventi, feste o festival.	"Sagra del Tartufo", "Natale"
	[WORKS_OF_ART]	Titoli di opere artistiche e/o commerciali di varia natura.	"La Divina Commedia", "La Sirenetta"
	[PRODUCTS]	Prodotti e marchi.	"iPhone", "Fiat Panda", "Pavesini"
✔	*[CODE]*	Codici di varia natura e forma, quali codici fiscali, postali, di reparto, coordinate bancarie, targhe, numeri identificativi di pratiche, cartelle cliniche, matricole o altre sequenze alfanumeriche.	"RSSMRA85M01H501U", "20123", "F19.20"
✔	*[MAIL]*	Indirizzi di posta elettronica.	"mario.rossi85@mail.it"
✔	*[PHONE]*	Numeri di telefono o fax.	"+39 349 8821345", "0373/847007", "051/177.589"
✔	*[PROV]*	Sigle di province italiane.	"AN"
✔	*[URL]*	Nomi di dominio, URL e indirizzi web.	"it.wikipedia.org", ""https://www.google.it/"