

**Cognitive Computing System applicato allo studio del Glioblastoma**

*Autori: Docenti:*

Francesca Lettieri - M63000956 Prof. Paolo Maresca

Angelo Russo - M63001016 Guido Maria Secondulfo - M63000927

# Indice

1. **Introduzione 3**
   1. Traccia . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
2. **Architettura 3**
   1. Tecnologie utilizzate . . . . . . . . . . . . . . . . . . . . . . . . . 3
      1. E-utilities . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
      2. IBM Cloud Db2 . . . . . . . . . . . . . . . . . . . . . . . 3
      3. GitHub . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
      4. IBM Cloud Foundry . . . . . . . . . . . . . . . . . . . . . 3
      5. IBM Watson Machine Learning . . . . . . . . . . . . . . . 4
      6. IBM Watson Knowledge Studio . . . . . . . . . . . . . . . 4
      7. IBM Watson Discovery . . . . . . . . . . . . . . . . . . . 4
      8. fastai . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
      9. IBM Cloud API . . . . . . . . . . . . . . . . . . . . . . . 4
      10. Jupyter . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
3. **Implementazione 5**
   1. Flusso di esecuzione del sistema globale . . . . . . . . . . . . . . 5
4. **Apprendimento sugli articoli medici tramite NLP 6**
   1. Flusso di esecuzione del sistema di retrieval degli articoli . . . . . 6
      1. E-utilities . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
      2. Query di ricerca . . . . . . . . . . . . . . . . . . . . . . . 7
      3. GitHub e IBM Cloud Foundry . . . . . . . . . . . . . . . 9
      4. IBM Cloud Db2 . . . . . . . . . . . . . . . . . . . . . . . 9
      5. Funzioni implementate . . . . . . . . . . . . . . . . . . . . 14
   2. Flusso di esecuzione del sistema di apprendimento sugli articoli . 16
   3. IBM Watson Knowledge Studio . . . . . . . . . . . . . . . . . . . 16
      1. Import degli Articoli . . . . . . . . . . . . . . . . . . . . . 16
      2. Creazione del Modello Entit`a-Relazione . . . . . . . . . . 16

## 5 Apprendimento sulle Risonanze Magnetiche tramite Visual Re-

**cognition 24**

5.1 Flusso di esecuzione del sistema di apprendimento sulle MRI . . 24

5.2 CNN con la libreria *fastai* . . . . . . . . . . . . . . . . . . . . . . 24

5.2.1 Implementazione . . . . . . . . . . . . . . . . . . . . . . . 24

5.2.2 Risultati ottenuti . . . . . . . . . . . . . . . . . . . . . . . 26

5.3 Visual Recognition di Watson Studio . . . . . . . . . . . . . . . . 29

5.3.1 Implementazione . . . . . . . . . . . . . . . . . . . . . . . 29

5.3.2 Risultati ottenuti - Primo Ciclo . . . . . . . . . . . . . . . 32

5.3.3 Risultati ottenuti - Secondo Ciclo . . . . . . . . . . . . . . 34

5.3.4 Risultati ottenuti - Terzo Ciclo . . . . . . . . . . . . . . . 36

5.3.5 Risultati ottenuti - Quarto Ciclo . . . . . . . . . . . . . . 38

1. **RadiAnt 40**

A.1 Pre-processing effettuato tramite RadiAnt . . . . . . . . . . . . . 41

# 1 Introduzione

## 1.1 Traccia

Il progetto da noi sviluppato si articola in due percorsi distinti:

1. Retrieval da PubMed di articoli inerenti al Glioblastoma Multiforme (GBM), creazione di un database atto a contenere le informazioni rilevanti ed, infine, creazione di un modello di NLP (Natural Language Processing) al fine di realizzare un sistema di apprendimento in grado di realizzare una standard meta-analysis accurata;
2. Acquisizione delle risonanze magnetiche (MRI) di pazienti affetti da GBM e costruzione di un modello di apprendimento su di esse.

# 2 Architettura

## 2.1 Tecnologie utilizzate

### 2.1.1 E-utilities

Le E-utilities (Entrez Programming Utilities) costituiscono un insieme di 8 programmi server-side che forniscono un’interfaccia stabile per il sistema di database e query Entrez. Le E-utilities utilizzano una sintassi a URL fisso che traduce un insieme di parametri di input nei valori necessari ai componenti software del NCBI per ricercare e recuperare i dati richiesti.

### 2.1.2 IBM Cloud Db2

IBM Db2 on Cloud `e un database cloud transazionale progettato per offrire prestazioni elevate. Esso assicura la scalabilit`a orizzontale e verticale per soddisfare le richieste di business.

### 2.1.3 GitHub

Git `e un software di controllo versione distribuito utilizzabile da interfaccia a riga di comando, sviluppato per gestire anche progetti di elevate dimensioni con elevate velocit`a ed efficienza.

### 2.1.4 IBM Cloud Foundry

Cloud Foundry `e una PaaS (platform-as-a-service) open source su IBM Cloud che consente di implementare e scalare le app senza gestire i server. E stato da` noi utilizzato al fine di poter eseguire uno script in linguaggio Python all’interno dell’ambiente IBM Cloud.

### 2.1.5 IBM Watson Machine Learning

IBM Watson Machine Learning aiuta i data scientist e gli sviluppatori ad accelerare l’implementazione di AI e machine-learning. Con i suoi modelli estensibili, aperti, Watson Machine Learning aiuta le aziende a semplificare e sfruttare l’AI in scala su qualsiasi cloud.

### 2.1.6 IBM Watson Knowledge Studio

IBM Watson consente l’apprendimento del linguaggio del dominio con modelli personalizzati che identificano le entit`a e le relazioni specifiche del settore in testo non strutturato. I modelli possono essere creati in un ambiente collaborativo progettato sia per gli sviluppatori che per gli esperti di dominio, senza il bisogno di scrivere codice, e possono essere utilizzati in IBM Watson Discovery, IBM Watson Natural Language Understanding e IBM Watson Explorer.

### 2.1.7 IBM Watson Discovery

Watson Discovery `e una tecnologia pluripremiata per la ricerca aziendale, basata sull’AI, in grado di abbattere i silos di dati e ottenere risposte specifiche alle tue domande, mentre analizza le tendenze e le relazioni nascoste nei dati aziendali. Watson Discovery applica le piu` recenti innovazioni nel machine learning, comprese le funzionalit`a di elaborazione del linguaggio naturale, ed `e semplice da addestrare sul linguaggio del tuo dominio. Watson Discovery pu`o essere implementato su qualsiasi ambiente cloud o on-premise.

### 2.1.8 fastai

fastai `e una libreria di deep learning che fornisce agli operatori componenti di alto livello in grado di fornire rapidamente e facilmente risultati di ultima generazione in domini di deep learning standard, e fornisce ai ricercatori con componenti di basso livello che possono essere mescolati e abbinati per costruire nuovi approcci. fastai mira a fare entrambe le cose senza compromessi sostanziali in facilit`a d’uso, flessibilit`a, o prestazioni. Questo `e possibile grazie ad un’architettura a strati, che esprime schemi comuni di base di molte tecniche di deep learning e di elaborazione dati in termini di astrazioni disaccoppiate. Queste astrazioni possono essere espresse in modo conciso e chiaro sfruttando il dinamismo del linguaggio Python sottostante e la flessibilit`a della libreria Pytorch.

### 2.1.9 IBM Cloud API

Le IBM Cloud API sono delle API che consentono di garantire l’accesso ai servizi IBM Cloud ai propri servizi ed alle proprie applicazioni che lo necessitano.

### 2.1.10 Jupyter

Il Progetto Jupyter `e un’organizzazione creata per sviluppare software opensource, e supportare ambienti di esecuzione in decine di lingue.

JupyterLab offre un’interfaccia per la gestione di documenti di diverso formato, per attivit`a di text editing e attivit`a da terminale.

# 3 Implementazione

## 3.1 Flusso di esecuzione del sistema globale

Il diagramma di flusso rappresentante l’elaborazione effettuata `e il seguente:

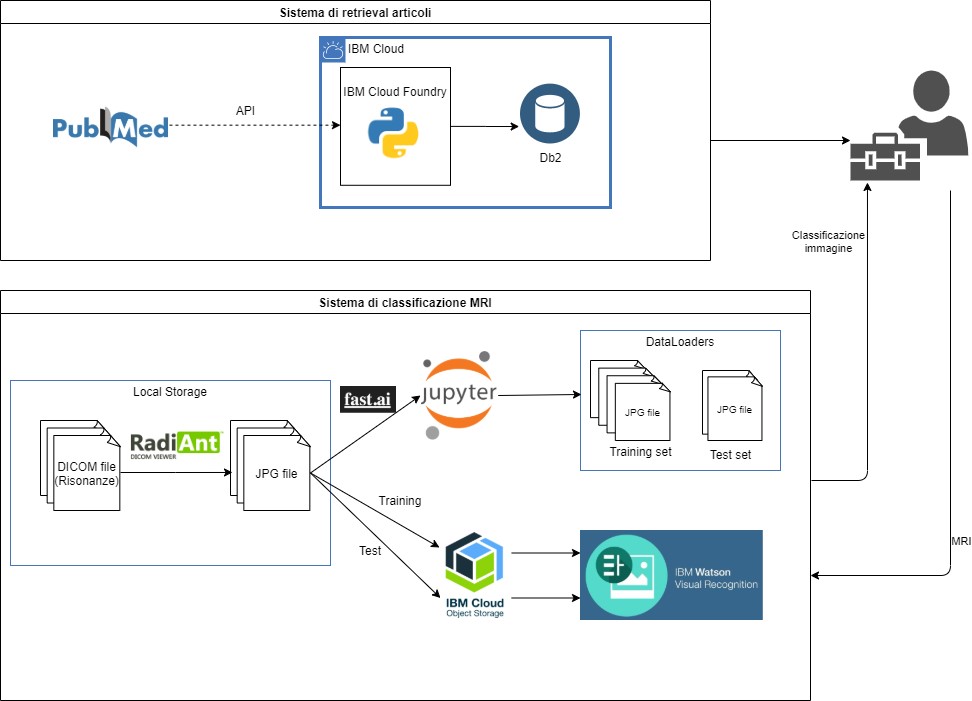


Figura 1: Diagramma di flusso dell’esecuzione Al fine di agevolare la trattazione, affronteremo i problemi separatamente.

# 4 Apprendimento sugli articoli medici tramite NLP

## 4.1 Flusso di esecuzione del sistema di retrieval degli articoli

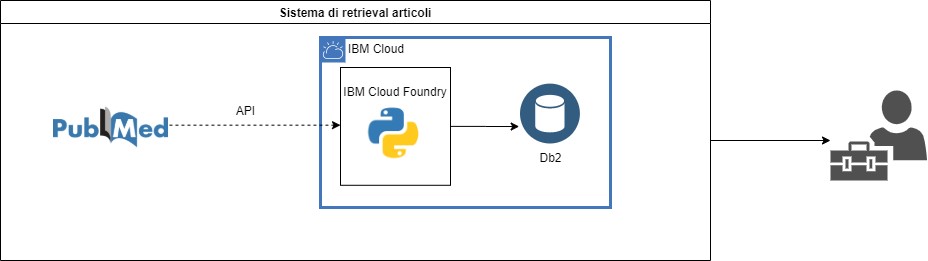


Figura 2: Diagramma di flusso dell’esecuzione

Innanzitutto `e stato effettuato il download degli articoli desiderati da PubMed. Dopo una prima elaborazione di tali documenti, essi sono stati memorizzati in un database relazionale all’interno dell’ambiente IBM Cloud.

### 4.1.1 E-utilities

In seguito ad un confronto con gli esperti di dominio, abbiamo ritenuto opportuno effettuare un retrieval di articoli provenienti unicamente da PubMed, in quanto source affidabile e attendibile.

A tal fine `e stato sviluppato uno script in linguaggio Python (*pubmed.py*) che, sfruttando le E-utilities, `e in grado di effettuare il retrieval degli articoli desiderati da PubMed stesso.

In particolare, la funzione *getPubMedIDArticles* recupera gli ID degli articoli desiderati da PubMed, in base ai termini specificati. L’implementazione di tale funzione `e la seguente:

**def** getPubMedIDArticles ( query ) : test = ”esearch . fcgi ?db=pubmed”

term = ”&term=”+query *#query della ricerca* maxArts =”&retmax=100000” *#Numero massimo di a r t i c o l i recuperati dalla ricerca*

link = ”https :// eutils . ncbi .nlm . nih . gov/entrez/ eutils /%22+test+term+maxArts

s = urllib . request . urlopen ( link ) contents = s . read () f i l e = open(” export idArticles . xml” , ’wb ’) f i l e . write ( contents ) f i l e . close ()

### 4.1.2 Query di ricerca

La query effettuata `e stata elaborata dagli esperti di dominio. Essa espone quali siano i termini da ricercare all’interno di tutti i campi di un articolo, inclusi i MeSH Terms [[1]](#footnote-1) associati all’articolo stesso.

I criteri di inclusione per la selezione degli articoli sono i seguenti:

* Studi randomizzati di confronto fra trattamenti;
* Fase II o III (compresi studi Fase I/II purch`e contengano risultati adeguati della Fase II;
* OS sopravvivenza globale (overall survival) e PFS sopravvivenza libera da progressione (progression free survival) e ORR (Overall Response Rate) Tasso di risposta complessiva;
* Solo Glioblastoma multiforme (GBM) o High Grade Glioma in ripresa dopo pregressa chirurgia e terapia adiuvante o in progressione.

I criteri di esclusione per la selezione degli articoli sono, invece, i seguenti:

* Tumori pediatrici;
* Studi non randomizzati;
* Studi a singolo braccio;
* Studi di fase I;
* GBM in trattamento adiuvante;
* Review/metanalisi;
* Studi preclinici;
* Studi di farmacoeconomia;
* Studi di confronto senza analisi statistiche.

La formulazione risultante della query `e, dunque, la seguente:

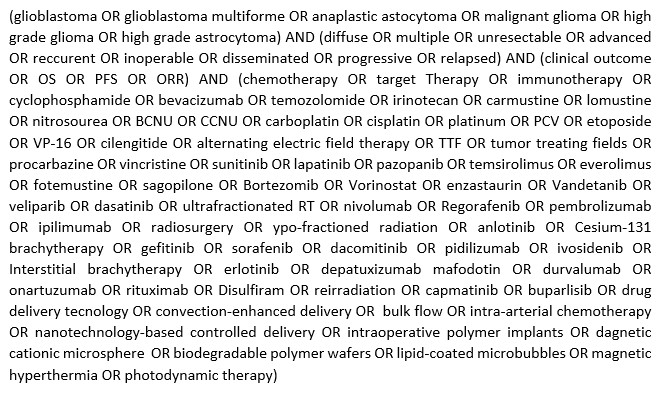


Figura 3: Query originale

Tale query `e stata, quindi, da noi raffinata al fine di poter effettuare il retrieval degli articoli che soddisfacessero i criteri richiesti.

L’operazione di raffinamento `e stata dettata dalla risposta delle API di PubMed a determinati termini (es. ”ypo-fractioned”, ”dagnetic”) che facevano parte della query iniziale; tali termini, omessi in fase di raffinamento, non risultavano presenti all’interno del tesauro (i.e. il MeSH).



Figura 4: Query Raffinata

In entrambi i casi, il numero di articoli recuperati `e pari a 1’318, per questo l’eliminazione dei termini `e risultata ininfluente per quanto riguarda il risultato finale ottenuto. Un ulteriore filtraggio degli articoli recuperati `e stato fatto in base alla data di pubblicazione, secondo quanto richiesto e consigliato dagli esperti di dominio: escludendo gli articoli pubblicati prima del 1990, il numero finale di articoli recuperati `e pari a *1’313*.

### 4.1.3 GitHub e IBM Cloud Foundry

Per poter importare gli articoli in ambiente IBM Cloud, innanzitutto `e stato creato un repository su GitHub all’interno del quale `e stato memorizzato il file *pubmed.py*.

In particolare, sono state effettuate le operazioni di **add**, **commit** e **push** del file all’interno del repository attraverso il prompt dei comandi.

### 4.1.4 IBM Cloud Db2

Innanzitutto `e stata istanziata una versione lite del Database Db2. Utilizzando le credenziali di accesso fornite dal sistema e le API IBM DB, `e stato possibile accedere al Database attraverso una funzione all’intero del file *pubmed.py*. Per la modellazione del database, volendo tener traccia di un’unica entit`a, gli articoli, `e stato realizzato il seguente diagramma ER[[2]](#footnote-2):

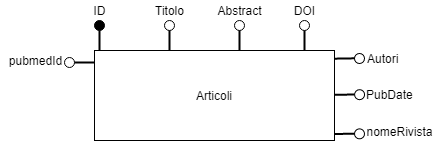


Figura 5: Diagramma ER

Abbiamo ipotizzato di memorizzare, per ogni articolo:

* DOI (Digital Object Identifier) - quando disponibile;
* ID dell’articolo nel database PubMed;
* Titolo;
* Lista di autori - quando disponibile;
* Data di pubblicazione - quando disponibile; • Rivista di pubblicazione - quando disponibile;
* Abstract - quando disponibile.

Osserviamo che, durante l’attivit`a di recupero dalle API Pubmed, il DOI era assente in alcuni articoli, il che lo rendeva inadeguato al ruolo di chiave primaria dell’entit`a. Per questo motivo, abbiamo utilizzato un ulteriore identificativo numerico con auto-incremento come chiave primaria.

Quindi, `e stato possibile creare una tabella ARTICOLI attraverso la seguente query:

### CREATE TABLE articoli ( id INTEGER NOT NULL PRIMARYKEY GENERATED ALWAYS AS

**IDENTITY** (START WITH 1 INCREMENT **BY** 1) , pubmedId **INTEGER UNIQUE NOT NULL**,

doi **VARCHAR**(80) ,

titolo **VARCHAR**(1000) , autori **VARCHAR**(5000) , dataPubblicazione **DATE**, abstract **VARCHAR**(10000) , nomeRivista **VARCHAR** (1000)”

)

La connessione al database `e realizzata attraverso la funzione *connectionDB2*, mentre tale connessione viene chiusa attraverso la funzione *endConnectionDB2*. L’implementazione di entrambe le funzioni `e riportata di seguito:

**def** connectionDB2 () : conn info = ””DATABASE=BLUDB;HOSTNAME=dashdb−txn−sbox −yp−lon02 −13.

services . eu−gb . bluemix . net ;PORT=50000;PROTOCOL=TCPIP; UID=xxd26106 ;PWD=s5bzk6pˆ3s94hght ; ”” conn = ibm db . connect ( conn info , ”” , ””) **return** conn

**def** endConnectionDB2(conn) :

ibm db . close (conn)

Si noti che le informazioni relative al Database, quali nome dello stesso, l’Hostname, il porto, il protocollo, l’username e la password, sono strettamente legate all’account IBM Cloud che ha istanziato il database. In caso di migrazione del database, o di creazione di uno nuovo, `e necessario modificare l’intera variabile *conn info* con le rispettive Credenziali di Servizio generate sul portale IBM Cloud.

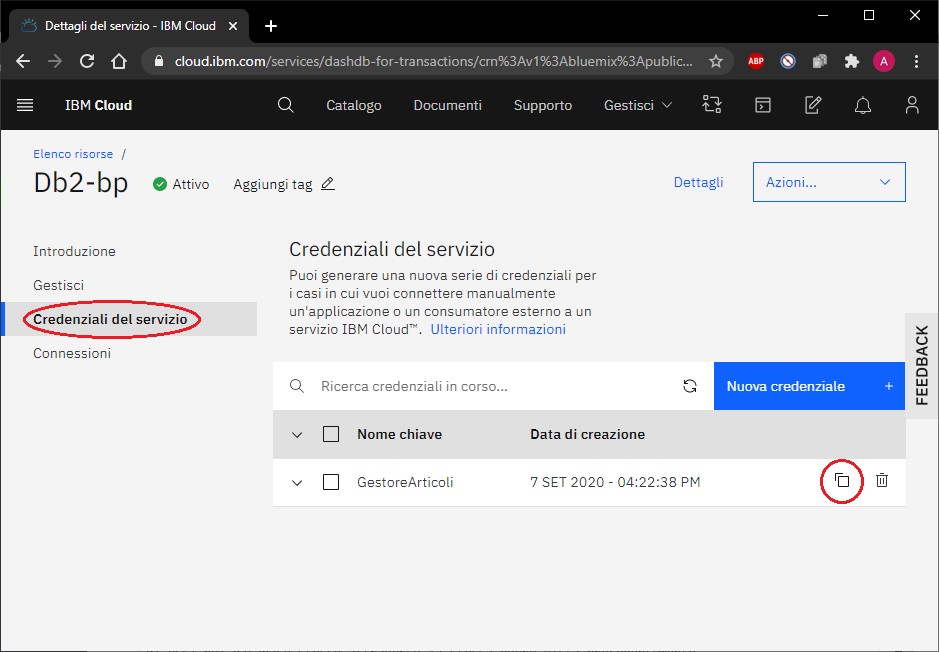


Figura 6: Schermata DB2 per ottenere le credenziali del servizio

Infine, il popolamento del database pu`o essere effettuato attraverso due diverse funzioni. La prima, *insertArticleDB2*, inserisce un singolo articolo all’interno del database attraverso la seguente query sql:

**INSERT INTO** articoli (pubmedId , doi , titolo , autori , dataPubblicazione , abstract , nomeRivista )

**VALUES** ( valore pubmedId , valore doi , valore titolo , valore autori ,

valore dataPubblicazione , valore abstract , valore nomeRivista ) ;

La seconda, *insertArrayDB2*, `e quella che `e stata utilizzata per realizzare il primo popolamento del database. Essa riceve in ingresso un array di articoli, andando ad inserire ognuno di essi nel database stesso attraverso la stessa query vista precedentemente.

In entrambi i casi, si `e resa, quindi, necessaria la definizione di una classe Articolo, la cui implementazione `e riportata di seguito:

**class** Articolo () :

**def** i n i t ( self , titolo , autori , pubDate , pubMedId , doi , abstract , journalTitle ) :

s e l f . titolo = titolo s e l f . autori = autori s e l f . abstract = abstract s e l f . doi = doi s e l f . pubMedId = pubMedId s e l f . pubDate = pubDate s e l f . journalTitle = journalTitle **def** s t r ( s e l f ) :

**return** ”PUBMED ID: %s\nDOI: %s\nTitolo : %s\ nAutori : %s\nData di Pubblicazione : %s\ nAbstract : %s\nJournal Title : %s\n\n” % ( s e l f . pubMedId , s e l f . doi , s e l f . titolo , ’ , ’ . join (

s e l f . autori ) , s e l f . pubDate , s e l f . abstract , s e l f . journalTitle )

L’implementazione delle funzioni sopra menzionate `e la seguente:

**def** insertArticleDB2 ( articolo ) :

abstract temp = ” ” titolo temp = ” ” jtitle temp = ” ” autori insert = ’ , ’ . join ( articolo . autori ) **if** ( autori insert != None) : autori insert = autori insert . replace (” ’” , ” ’ ’”)

**if** ( articolo . titolo != None) : titolo temp = articolo . titolo . replace (” ’” , ” ’ ’”)

**if** ( articolo . abstract != None) :

abstract temp = articolo . abstract . replace (” ’” , ”

’ ’”)

**if** ( articolo . journalTitle != None) :

jtitle temp = articolo . journalTitle . replace (” ’” ,

” ’ ’”) insert = ibm db . exec immediate (conn , ”INSERT INTO articoli (pubmedId , doi , titolo , autori ,

dataPubblicazione , abstract , nomeRivista ) VALUES ( ’”+articolo . doi+” ’ , ’”+titolo temp+” ’ , ’”+ autori insert+” ’ , ’”+ articolo . pubDate+” ’ , ’”+ abstract temp+” ’ , ’”+jtitle temp+” ’ ’)”)

**print**(” [ ”+**str** ( i )+” ] − Inserito \n”) *#DEBUG d e l l ’ articolo inserito*

**def** insertArrayDB2 ( articoli ) :

**for** articolo **in** articoli : abstract temp = ” ” titolo temp = ” ”

jtitle temp = ” ”

autori insert = ’ , ’ . join ( articolo . autori ) **if** ( autori insert != None) :

autori insert = autori insert . replace (” ’” , ”

’ ’”)

**if** ( articolo . titolo != None) :

titolo temp = articolo . titolo . replace (” ’” , ”

’ ’”)

**if** ( articolo . abstract != None) :

abstract temp = articolo . abstract . replace (” ’” , ” ’ ’”)

**if** ( articolo . journalTitle != None) :

jtitle temp = articolo . journalTitle . replace (” ’” , ” ’ ’”)

insert = ibm db . exec immediate (conn , ”INSERT INTO articoli (pubmedId , doi , titolo , autori , dataPubblicazione , abstract , nomeRivista )

VALUES ( ’”+articolo . doi+” ’ , ’”+titolo temp+” ’ ,

’”+ autori insert+” ’ , ’”+ articolo . pubDate+”

’ , ’”+abstract temp+” ’ , ’”+jtitle temp+” ’ ’)”)

**print**(” [ ”+**str** ( i )+” ] − Inserito \n”) *#DEBUG d e l l ’ articolo inserito*

Inizialmente, si era pensato di mantenere in memoria le informazioni degli articoli recuperati, per poi caricarle tutte insieme sul database: questo metodo per`o `e risultato poco efficiente a causa della grande mole di articoli da caricare. Tale risultato `e stato ottenuto stressando il sistema tramite una query generica di prova, che ricercasse gli articoli che contenessero in qualsiasi campo (sia esso Titolo, MeSH o Abstract) il termine ”glioblastoma”. Questa query di prova ha prodotto come risultato ben 41’364 articoli e pubblicazioni, rallentando di molto il processo di recupero, il mantenimento temporaneo in memoria e infine l’inserimento nel Database degli stessi. E risultato molto piu` conveniente, piut-` tosto, gestire grosse moli di articoli recuperandoli e inserendoli immediatamente nel database, in modo da non perdere l’eventuale cluster di articoli in caso di disconnessioni causate dal server di PubMed (problema che, piu` volte, `e stato riscontrato durante il testing dello script). Conviene utilizzare, quindi, la funzione *insertArrayDB2* quando vengono recuperati quantit`a minime di articoli (nell’ordine massimo delle centinaia), come nel caso della query utilizzata ed esposta nel sottoparagrafo 3.1.2, dove la query ha restituito 1’318 articoli come esito della ricerca.

#### 4.1.5 Funzioni implementate

Dopo aver ottenuto la lista contenente gli ID degli articoli, la funzione *pubMedIDConverter* recupera le informazioni ad essi associate. L’implementazione di tale funzione `e la seguente:

**def** PubMedIdConverter( articoli ) : data file = ’ export . xml ’ tree = ET. parse ( data file ) root = tree . getroot () id converter = ”https :// eutils . ncbi .nlm. nih . gov/ entrez/ eutils / efetch . fcgi ?db=pubmed&id=”

i=0

**for** child **in** root . findall (”./ IdList/Id”) :

i=i+1

**id** = child . text id converter1 = id converter+**id**+”&retmode=xml”

xml = urllib . request . urlopen ( id converter1 ) . read

()

xml tree = ET. fromstring (xml) *# Root*

**for** child2 **in** xml tree . findall (”PubmedArticle”) :

temp title = ”” temp abstract = ”” temp author = ”” temp doi = ”Mancante” temp pubdate = ””

autori vector = [ ]

**for** child2 0 **in** child2 . findall (” MedlineCitation/ Article ”) : **for** child2 JournalTitle **in** child2 0 . findall (”Journal/ Title ”) :

temp jtitle = child2 JournalTitle . text

**for** child2 1 **in** child2 0 . findall (” ArticleTitle ”) :

temp title = child2 1 . text

**for** child2 2 **in** child2 0 . findall (” Abstract/AbstractText”) : temp abstract = child2 2 . text

**for** child2 3 **in** child2 0 . findall (” AuthorList/Author”) :

*#La presente soluzione e ’ concepita nel caso si voglia tenere traccia*

*solo di i n i z i a l i del nome e*

*cognome ( per intero )*

path cognome = child23 . find (”

LastName”) path iniziali = child23 . find (”

I n i t i a l s ”) **if** (path cognome != None) :

**if** ( path iniziali != None) :

temp author = child2 3 . find (” LastName”) . text +” ”+ path iniziali . text+” . ”

**else** :

temp author = child2 3 . find (”

LastName”) . text autori vector . append( temp author )

**for** child2 4 **in** child2 . findall (”PubmedData”) :

**for** child2 5 **in** child2 4 . findall (”History /PubMedPubDate/[ @PubStatus=’entrez ’] ”)

:

temp pubdate = child2 5 [ 0 ] . text+”−”+

child2 5 [ 1 ] . text+”−”+child2 5 [ 2 ] .

text

childFinale = child2 4 . find (”

ArticleIdList / ArticleId /[@IdType=’doi

’] ”)

**if** ( childFinale != None) : temp doi = childFinale . text **print**(” [ ”+**str** ( i )+” ] − ” , temp doi ) *# Debug d e l l ’ articolo recuperato*

articolo = Articolo ( temp title ,

autorivector , temp pubdate , temp pubmedid

, tempdoi , temp abstract , temp jtitle ) articoli . append( articolo )

**return** articoli

## 4.2 Flusso di esecuzione del sistema di apprendimento sugli articoli

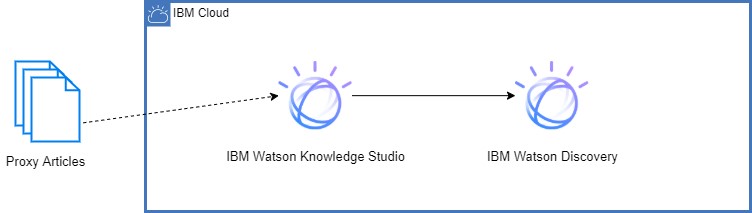


Figura 7: Diagramma di flusso dell’esecuzione

Osserviamo innanzitutto che all’interno di IBM Watson Knowledge Studio vengono memorizzati direttamente gli articoli completi. In particolare, momentaneamente gli articoli sono stati scaricati manualmente sfruttando i Proxy di diverse universit`a campane, quali l’Universit`a degli Studi di Napoli Federico II, l’Universit`a degli Studi della Campania Luigi Vanvitelli e l’Universit`a degli Studi di Salerno.

## 4.3 IBM Watson Knowledge Studio

### 4.3.1 Import degli Articoli

Come anticipato, inizialmente `e stato utilizzato un set costituito da soli 8 articoli scaricati manualmente dalle rispettive riviste, attraverso i DOI corrispondenti memorizzati all’interno del database DB2.

In particolare, nella seguente immagine `e possibile osservare gli articoli memorizzati all’interno di IBM Watson Knowledge Studio:



Figura 8: Primo set di articoli caricati su IBM Watson Knowledge Studio

### 4.3.2 Creazione del Modello Entit`a-Relazione

Per realizzare il modello di NLP, innanzitutto, `e stato necessario definire le Entit`a e le Relazioni che dovranno essere annotate all’interno degli articoli.

A seguito di un primo confronto con gli esperti di dominio, essi hanno fornito un primo insieme di entit`a e relazioni sufficienti per effettuare un’operazione di pre-annotation[[3]](#footnote-3). In particolare, in seguito a tale primo colloquio, sono state definite le seguenti relazioni e le seguenti entit`a (evidenziate in grassetto), con rispettivi sinonimi, abbreviazioni e termini appartenenti a quella data entit`a:

# Entità

**TUMOR** Tumor

Neoplasm

Malignancy

Malignant glioma

Recurrent malignant glioma

Progressive malignant glioma

Oligodendroglioma

Astrocytoma

Anaplastic glioma

Anaplastic astrocytoma

HGG

GBM

Progressive Glioblastoma multiforme

Recurrent Glioblastoma multiforme

High grade glioma

WHO grade IV

**Molecular classification** 1p/19q codeletion

IDH1 or IDH2 mutation

TERT mutation

EGFR amplicons

MGMT

MGMT status

PTEN mutation

ATRT

Peroxisome proliferator-activated receptor gamma

**TRIAL**

Clinical trial registry

Registry

Phase I

Phase II

Phase III

Study design

Open-label

Exclusion criteria

Inclusion criteria

**CHEMOTHERAPY** Chemotherapy

Nimustine

Temozolomide

Carboplatin

Procarbazine

Lomustine

Vincristine

Carmustine Verubulin

Etoposide

Cyclophosphamide

Ifosfamide

Teniposide

Interferon

Convention-enhanced delivery

CED

Gliadel wafers

Carmustine wafers

Chemotherapy delivery

Intravenous delivery

Intra-arterial delivery

**TARGET THERAPY**

Target therapy

Anti-VEGF

Bevacizumab

Anti VEGFR

Sorafenib Anti-EGFR erlotinib HIV protease inhibitors

Ritonavir

Lopinavir

**RADIOTHERAPY** Radiotherapy

Radiation therapy

Irradiation

Re-irradiation

Gamma Knife

Fractioned radiation therapy

Unfractioned radiation therapy

Stereotactic radiosurgery

**SURGERY**

Surgery

Biopsy

Stereotactic neurosurgery

Neuronavigation

Inoperable

Operable

Gross total tumor resection

Complete tumor resection

Partial resection

Residual tumor

**MAGNETIC RESONANCE IMAGING** Magnetic Resonance Imaging

MRI

Enhancement

Contrast-enhancement

Delayed contrast extravasation MRI

Treatment Response Assessment Maps

TRAMs

Perfusion imaging

Dynamic Susceptibility Contrast-enhanced

DSC

Dynamic Contrast-Enhanced

DCE

Susceptibility Weighted Imaging

**RESPONSE TO THERAPY**

Survival

Overall survival

OS

Progression free survival

PFS

Karnofsky status

Life quality

Objective response rate

ORR

**ADVERSE EFFECT**

Collateral effect

Adverse effect

Adverse reaction

Secondary reaction

Dosage

Toxicity

Efficacy

Tolerance

Safety

**RESPONSE ASSESSMENT**

Progression

True progression

Regression

Complete response

Partial response

Stable disease

Pseudoprogression

Pseudoresponse

Radionecrosis

Cerebral radiation necrosis

Response assessment in neuro-oncology

RANO criteria iRANO criteria

# Relazioni

Tumor – molecular classification

Tumor – chemotherapy

Tumor - TARGET THERAPY

Tumor- radiotherapy

Tumor- surgery

Tumor- Magnetic Resonance Imaging

Tumor - trial

Trial – chemotherapy

Trial- radiotherapy

Trial- target therapy

Trial-surgery

chemotherapy - response to therapy chemotherapy – response assessment chemotherapy – adverse effect target therapy – responce to therapy target therapy - response assessment target therapy - adverse effect radiotherapy- response assessment radiotherapy – adverse effect radiotherapy – responce to therapy surgery-response assessment surgery– adverse effect surgery – responce to therapy

# Sinonimi e abbreviazioni

Neoplasm - malignancy - tumor

Glioblastoma - GBM - High grade glioma - HGG

Therapy - treatment

HIV protease inhibitors - HIVPI

Radiotherapy - irradiation - radiation therapy - RT

Gross total tumor resection - Complete tumor resection

Tolerance - tolerability

Overall Survival - OS

Progression free survival - PFS

Karnofsky status – KS

Objective response rate - ORR

Temozolomide - TMZ

Nimustine - INN

Interferon – IFN

Gliadel – carmustine

Reccurent-progressive

CED-convection enhanced delivery

Response assessment in neuro-oncology criteria – RANO

Magnetic Resonance Imaging - MRI

Dynamic Susceptibility Contrast-enhanced – DSC

Dynamic Contrast-Enhanced – DCE

Treatment Response Assessment Maps - TRAMs

Definito il modello, `e stato definito un Dizionario tale che associasse i termini specificati nell’elenco ricevuto dagli esperti di dominio alle rispettive entit`a.

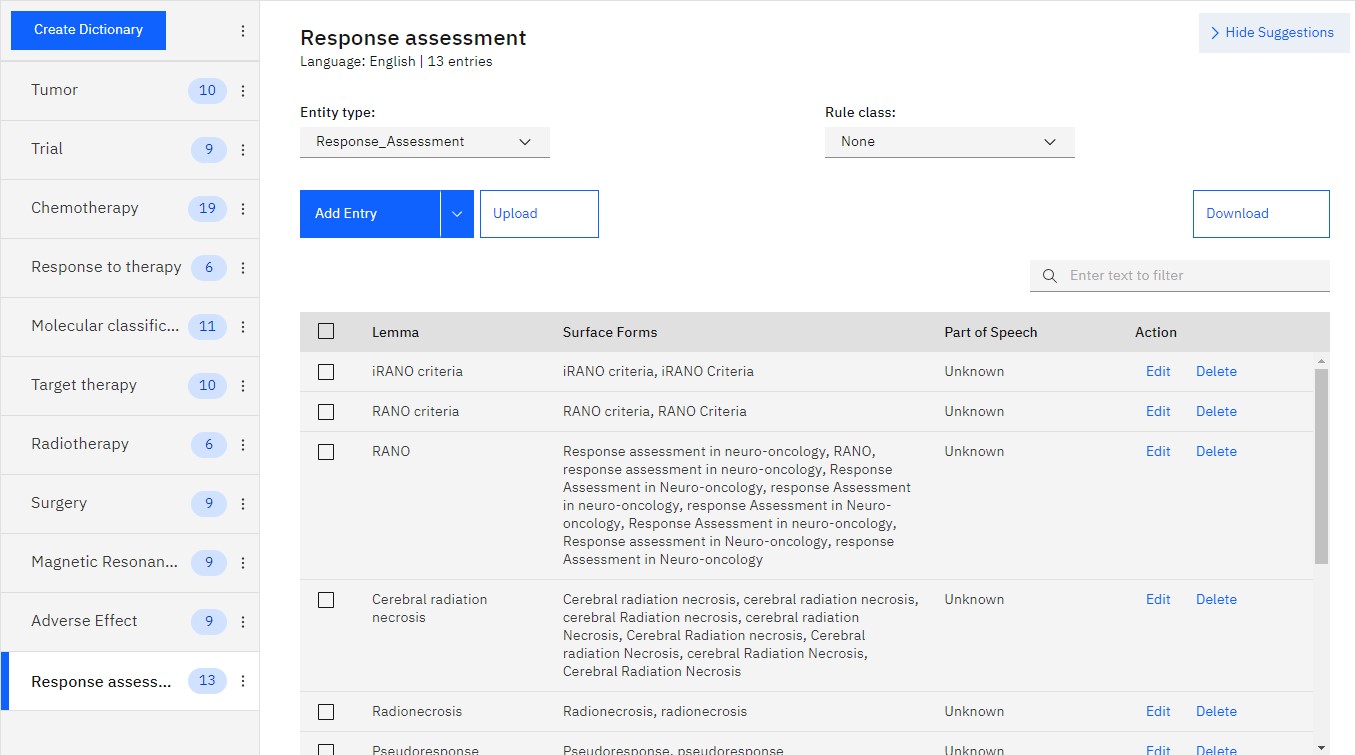


Figura 9: Dizionario definito

Una volta prodotto il Dizionario, `e stata processata la pre-annotazione dei testi precedentemente caricati.

Di seguito `e possibile osservare alcuni esempi di risultati ottenuti sui 10 testi:

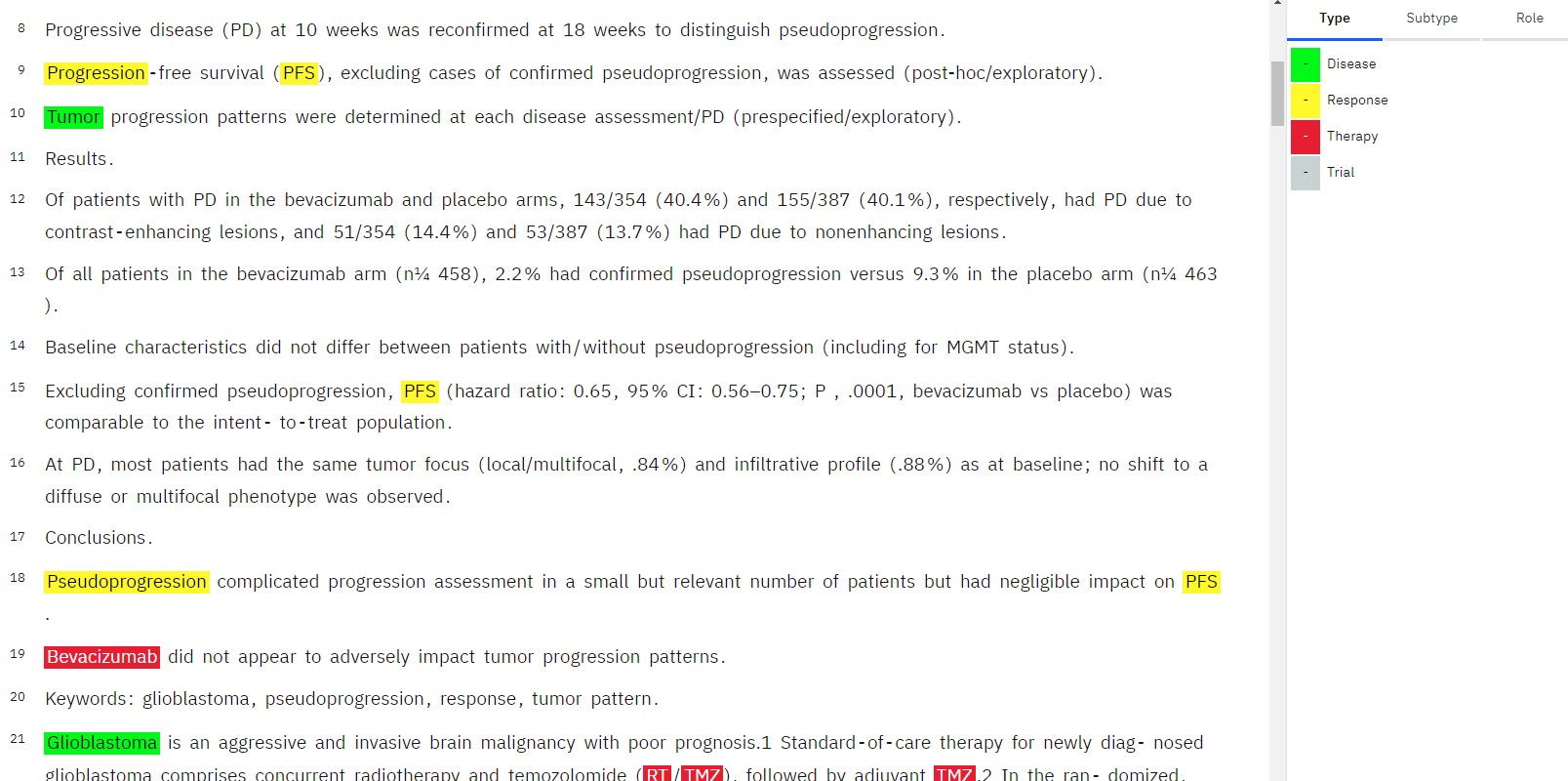


Figura 10: Esempio di annotazione

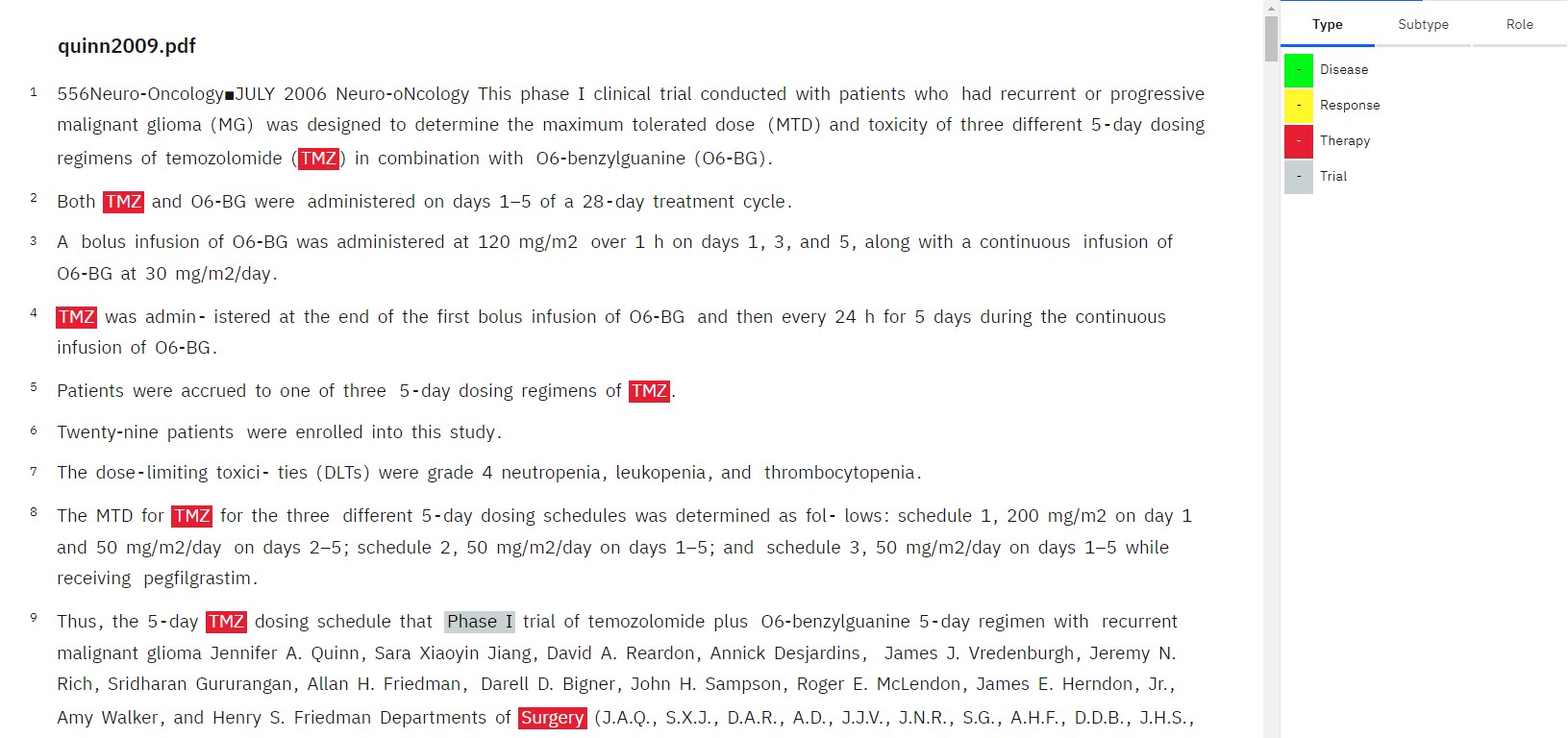


Figura 11: Esempio di annotazione

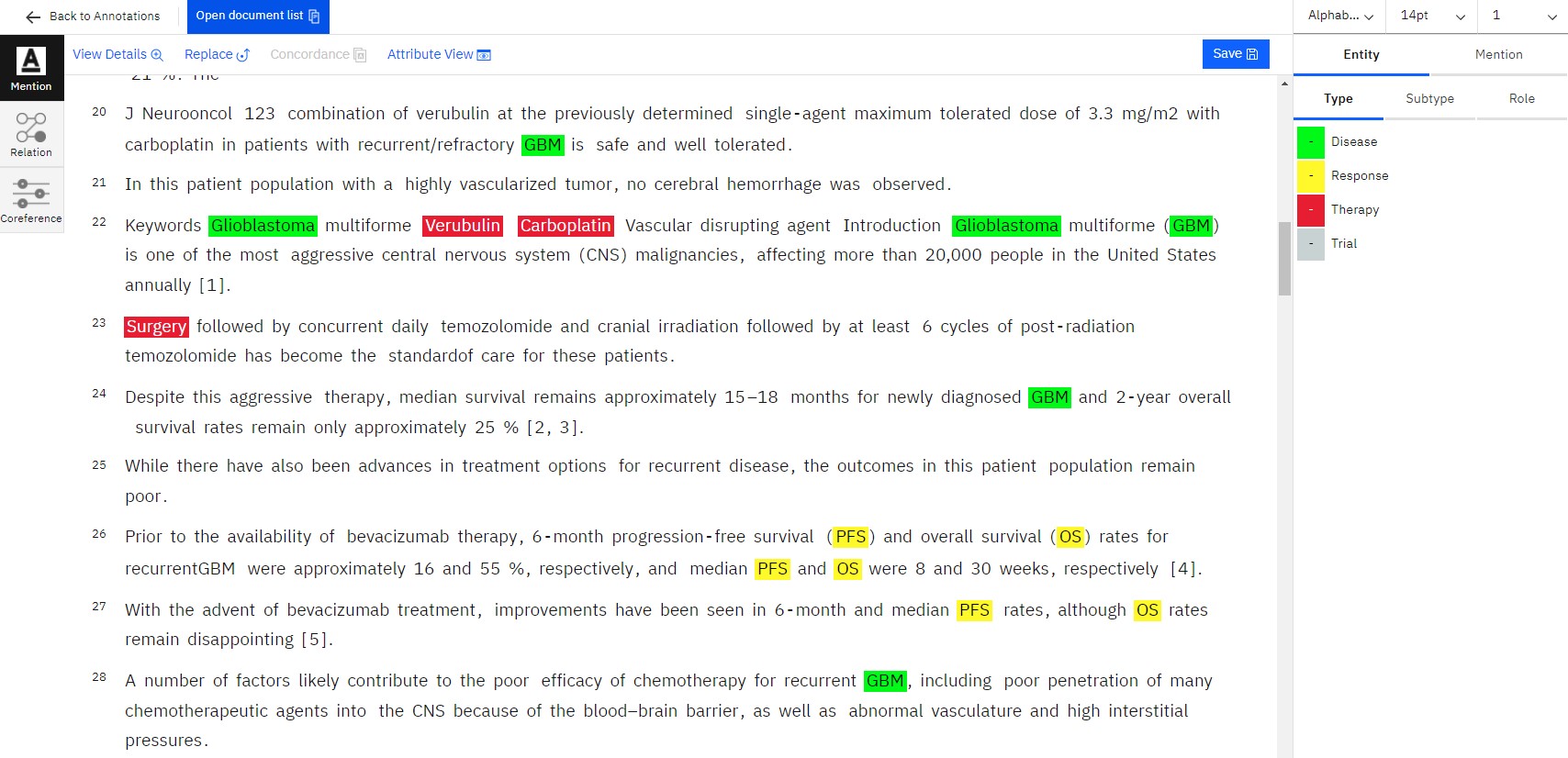


Figura 12: Esempio di annotazione

## 5 Apprendimento sulle Risonanze Magnetiche tramite Visual Recognition

### 5.1 Flusso di esecuzione del sistema di apprendimento sulle MRI

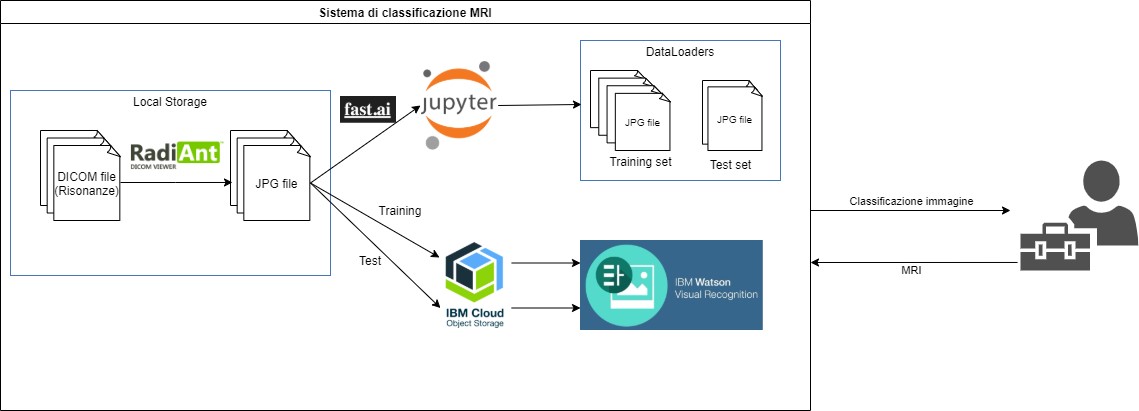


Figura 13: Diagramma di flusso dell’esecuzione

In un primo momento, i file DICOM sono stati sottoposti ad una fase di preprocessing: essi sono stati aperti tramite il tool *RadiAnt DICOM Viewer* e convertiti in formato JPG: laddove fosse necessario, successivamente alla conversione in JPG i file sono stati anche ridimensionati ad una dimensione pari a 256x256 pixel.

Dopo la fase di pre-processing, le risonanze sono state utilizzate per la realizzazione di due reti differenti:

1. una CNN implementata nella libreria *fastai*;
2. il tool di Visual Recognition di Watson Studio.

### 5.2 CNN con la libreria *fastai*

#### 5.2.1 Implementazione

I file DICOM forniteci dagli esperti di dominio sono stati etichettati dagli stessi esperti, che hanno distinto due diverse classi di immagini. In particolare, ci `e stato fornito un insieme di 691 immagini, divise in due diverse classi. La prima `e la classe GBM che contiene 489 immagini di glioblastomi multiformi, mentre la seconda `e la classe MET che contiene 202 immagini di metastasi.

Innanzitutto, i file DICOM etichettati dagli esperti di dominio sono stati suddivisi nei due insiemi di Training e Validation. In particolare, per realizzare la suddivisione `e stata sfruttata la tecnica dell’Holdout[[4]](#footnote-4), per cui il Training Set contiene l’80% dei file, mentre il Validation Set contiene il 20% rimanente.

Ai file DICOM deve essere associato un file **.csv** in cui si specifica l’etichetta associata ad ognuno dei file. Specificando il path in cui `e presente il file di etichettamento, il dataset viene caricato utilizzando la classe built-in *DataBlock*:

gbm = DataBlock( blocks=(ImageBlock , CategoryBlock ) , get x=**lambda** x : gbm source/ f ”{x [0]} ” , get y=**lambda** x : x [1] , batch tfms=aug transforms ( size =300))

* *gbm*: variabile che conterr`a l’intero DataBlock, ovvero l’intero dataset delle risonanze;
* *ImageBlock*: funzione che specifica cosa dovranno contenere i blocchi;
* *gbm source*: variabile precedentemente dichiarata contenente il Path del file di labeling delle risonanze.

Caricato il dataset, si generano il Training Set ed il Validation Set che verranno utilizzati per l’addestramento del sistema e la valutazione della sua accuracy. Ci`o viene effettuato tramite la funzione *dataloaders* presente all’interno della classe DataBlock:

dls = gbm. dataloaders ( df . values , num workers=0, bs = **int**

(**len**( df . values ) ∗0.80) , drop last=True)

* *df.values*: carica i valori delle etichette;
* *num workers*: indica il numero di sottoprocessi per il caricamento dei dati. Il valore di default 0 indica che viene effettuato dal main process;
* *bs*: indica il dimensionamento dei due Set tramite una suddivisione in **batch**;
* *drop last*: elimina eventuali batch generati in eccesso non contenenti elementi del dataset.

E possibile visualizzare il caricamento effettuato sui due set:`

dls . train . show batch (max n=100)

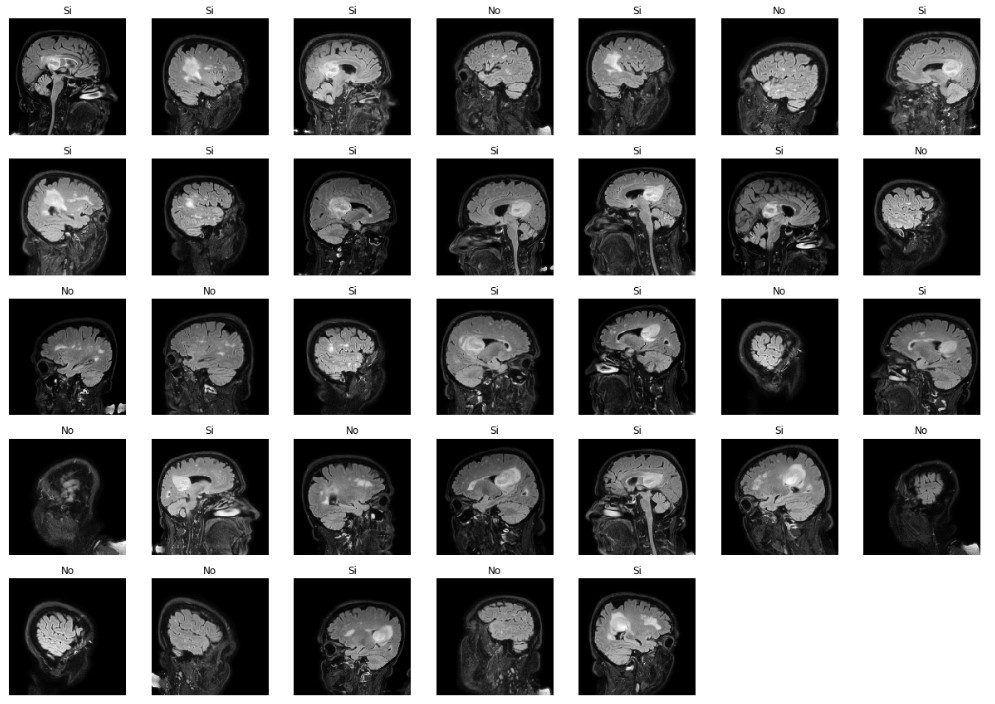


Figura 14: Visualizzazione del Training Set

dls . valid . show batch (max n=100)

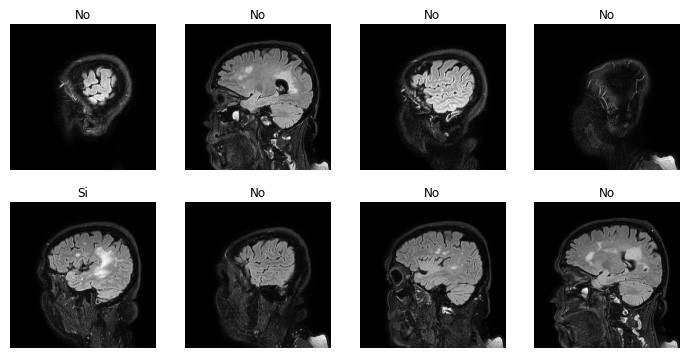


Figura 15: Visualizzazione del Validation Set

#### 5.2.2 Risultati ottenuti

Generati Training e Validation Sets, `e stato effettuato l’addestramento del classificatore, che ha impiegato un tempo pari a 6’03”.

Dopodich´e, il classificatore ottenuto `e stato valutato tramite le metriche di **Accuracy**, **Precision** e **Recall**, ripetendo i cicli di addestramento della CNN fino a quando non raggiungesse un valore di accuracy pari all’83%. Tale valore di soglia `e stato scelto in modo tale da evitare il verificarsi del fenomeno dell’overfitting per valori maggiori di accuracy.

Un esempio di addestramento della CNN `e osservabile nella figura sottostante:

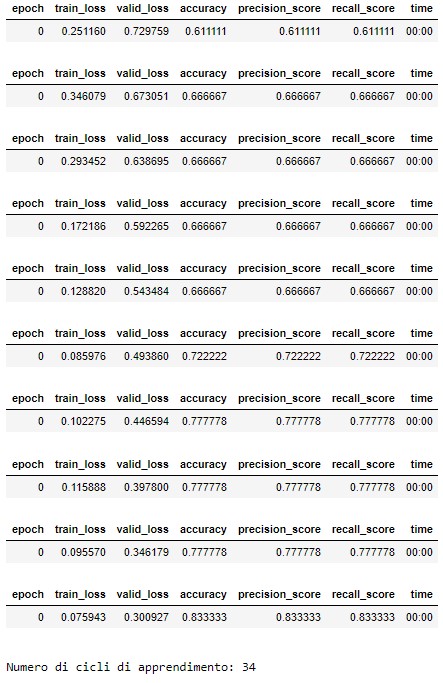


Figura 16: Esempio di addestramento della CNN

Il risultato della classificazione ottenuto `e, quindi, il seguente:

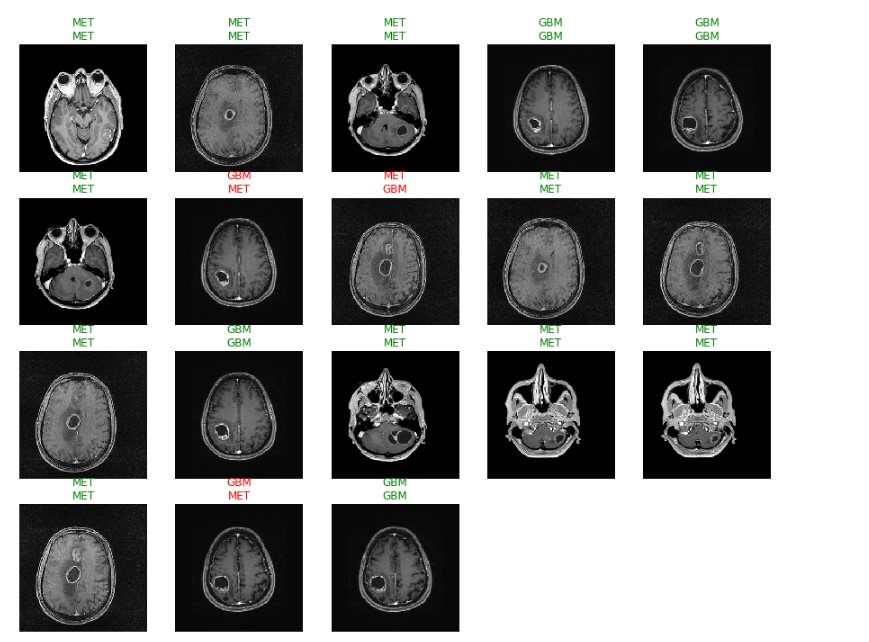


Figura 17: Classificazione effettuata dalla CNN

### 5.3 Visual Recognition di Watson Studio

L’addestramento del classificatore attraverso Watson Studio `e stato addestrato attraverso diversi cicli di raffinamento che analizzeremo nel dettaglio di seguito:

1. Primo Ciclo: il classificatore `e stato addestrato fornendogli in ingresso un Training set costituito da un totale di 691 immagini, divise in due diverse classi. La prima `e la classe GBM che contiene 489 immagini di glioblastomi multiformi, mentre la seconda `e la classe MET che contiene 202 immagini di metastasi. Osserviamo che, in pratica il Training Set adoperato per realizzare il primo ciclo di addestramento `e, in pratica, lo stesso Training Set utilizzato per addestrare il classificatore con **fastai**;
2. Secondo Ciclo: il classificatore `e stato addestrato fornendogli in ingresso un Training set costituito da un totale di 1’838 immagini, divise in due diverse classi ed un set di immagini definite come Negative. In particolare, le classi GBM e MET sono rimaste invariate rispetto al primo ciclo di addestramento, mentre le 1’147 immagini Negative sono immagini che non appartengono a nessuna di esse;
3. Terzo Ciclo: il classificatore `e stato addestrato fornendogli in ingresso un Training set costituito da un totale di 2’103 immagini, divise in due diverse classi ed un set di immagini definite come Negative. In particolare, le immagini definite come Negative sono rimaste invariate, mentre sono state aggiunte 221 immagini classificate come GBM e 44 immagini clssificate, invece, come MET;
4. Quarto Ciclo: il classificatore `e stato addestrato fornendogli in ingresso un Training set costituito da un totale di 956 immagini, divise in due diverse classi. In particolare, le immagini appartenenti alle classi GBM e MET sono le stesse utilizzate nel terzo ciclo di addestramento. Sono state, invece, rimosse le immagini definite come Negative.

Al termine di ogni ciclo di addestramento sono state valutate le prestazioni del classificatore in termini di metriche di **Accuracy**, **Precision**, **Recall** ed **F1score**, attraverso l’utilizzo di un Test Set costituito da un totale di 327 elementi suddivisi in due classi, quali GBM e MET. In particolare, 223 elementi sono classificati come GBM ed i restanti 104 elementi sono, invece, classificati come MET.

#### 5.3.1 Implementazione

Innanzitutto, una prima parte di dati etichettati forniti dagli esperti di dominio `e stata utilizzata come Training Set con lo scopo di far apprendere il modello. Dopodich´e, un secondo insieme di dati etichettati dagli esperti di dominio `e stato utilizzato come Test Set al fine di poter valutare la metriche di **Accuracy**, **Precision**, **Recall** ed **F1-score**.

In particolare, `e stato realizzato un notebook in linguaggio Python che, sfruttando le IBM Cloud API, ci ha consentito di utilizzare il classificatore precedentemente realizzato per effettuare una classificazione del Test Set.

In primo luogo `e stata generata un’istanza di **VisualRecognitionV3**, che consente di realizzare il collegamento con l’istanza di Visual Recognition addestrata all’interno di Watson Studio. Ci`o `e stato realizzato attraverso la funzione *createVRinstance(parameters)*, che riceve in ingresso un insieme di parametri specificati all’interno di un file JSON. In particolare, i campi specificati all’interno del file sono i seguenti:

* url;
* apikey;
* vr id: id del classificatore da noi istruito;
* test csv file: file csv contenente i nomi delle immagini appartenenti al Test Set e le rispettive classificazioni;
* results csv file: file csv in cui verranno memorizzati i risultati;
* confmatrix csv file: file csv in cui verr`a memorizzata la Confusion Matrix (Matrice di Confusione[[5]](#footnote-5)).

Le informazioni contenute all’interno del file JSON vengono memorizzate all’interno di un oggetto di tipo Parameters:

**def** readParameters (parmsPath) :

*#Lettura dei parametri u t i l i da un f i l e json* vrParmsFile = parmsPath parms = ’ ’ with **open**( vrParmsFile ) as parmFile :

parms = json . load ( parmFile )

parameters = Parameters (parms [ ’ url ’ ] , parms [ ’ apikey ’ ] , parms [ ’ vr id ’ ] , parms [ ’ t e s t c s v f i l e ’ ] , parms [ ’ r e s u l t s c s v f i l e ’ ] , parms [ ’ confmatrix csv file ’ ] ,

parms [ ’ version ’ ])

json .dumps(parms) **return** parameters I parametri appena acquisiti, come anticipato, vengono utilizzati per la creazione di un’istanza di Visual Recognition, ovvero un’istanza della classe **VisualRecognitionV3**, definita all’interno della libreria **ibm watson**. L’implementazione della funzione *createVRinstance(parameters)* `e riportata di seguito:

**def** createVRinstance ( parameters ) :

*#Creazione di una istanza di Visual Recognition* authenticator = IAMAuthenticator( parameters . apiKey) visualRecognition = VisualRecognitionV3 ( version = parameters . version , authenticator = authenticator

)

visualRecognition . set service url ( parameters . url ) **return** visualRecognition

Dopo la generazione dell’istanza della classe **VisualRecognitionV3**, `e possibile effettuare la classificazione delle immagini contenute all’interno del Test Set. A tal fine, innanzitutto `e stata definita la funzione *getVRresponse* che va a classificare una singola immagine restituendo una lista delle classi associate all’immagine stessa, ordinata in base al valore di Confidence (score). La funzione riceve in ingresso 3 parametri, quali l’istanza di visual recognition, l’id del classificatore ed il path dell’immagine da classificare. L’implementazione della funzione `e riportata di seguito:

**def** getVRresponse ( vr instance , classifierID , image path ) :

with **open**( image path , ’rb ’ ) as images file : image results = vr instance . classify ( images file = images file , threshold=’ 0.6 ’ ,

c l a s s i f i e r i d s=classifierID ) . get result ()

*#Ogni singola immagine viene c l a s s i f i c a t a attraverso i l c l a s s i f i c a t o r e da noi i s t r u i t o*

classList = [ ]

**for** c l a s s i f i e r **in** image results [ ’ images ’ ] [ 0 ] [ ’ c l a s s i f i e r s ’ ] :

**if** c l a s s i f i e r [ ’ c l a s s i f i e r i d ’ ] == parameters . vrId

:

classList = c l a s s i f i e r [ ’ classes ’ ]

#### break

*#Le c la ssi vengono ordinate in base allo score* sorted classList = **sorted**( classList , key=**lambda** k : k .

get ( ’ score ’ , 0) , reverse=True)

**return** sorted classList

Sfruttando la funzione appena definita, `e stato possibile definire un’ulteriore funzione *batchVR*, che consente di classificare un batch di immagini restituendo, per ciascuna di esse, i risultati ottenuti. In particolare, i path delle immagini appartenenti al batch da classificare vengono specificati all’interno di un file **csv**, insieme alla classe di appartenenza dell’immagine stessa. La funzione riceve in ingresso 3 parametri, quali l’istanza di visual recognition, l’id del classificatore ed il path del file csv. L’implementazione della funzione `e riportata di seguito:

**def** batchVR( vr instance , classifierID , csvfile ) :

testClasses =[] vrPredictClasses =[] vr predict confidence =[]

images =[] i=0

with **open**( csvfile , ’ r ’ ) as csvfile : csvReader=csv . DictReader ( csvfile ) **for** row **in** csvReader :

testClasses . append(row [ ’ class ’ ]) vr response = getVRresponse ( vr instance , classifierID , row [ ’image ’ ])

vrPredictClasses . append( vr response [ 0 ] [ ’ class

’ ])

vr predict confidence . append( vr response [ 0 ] [ ’ score ’ ])

images . append(row [ ’image ’ ]) i = i+1 **if** ( i%250 == 0) : **print**(””) **print**(”Processed ” , i , ” records”)

**if** ( i%10 == 0) :

sys . stdout . write ( ’ . ’ )

**print**(””) **print**(”Finished processing ” , i , ” records”)

**return** testClasses , vrPredictClasses , vrPredictConfidence , images

##### 5.3.2 Risultati ottenuti - Primo Ciclo

Il tempo impiegato per effettuare il Training del classificatore dato il Training Set descritto precedentemente (che consta di 691 immagini) `e pari a 36’17”. Le metriche di **Accuracy**, **Precision**, **Recall** ed **F1-score** sono state stimate attraverso l’utilizzo delle funzioni *accuracy score* e *classification report*, entrambe definite all’interno della libreria **sklearn**. Di seguito sono riportati i risultati ottenuti:

accuracy = accuracy score ( testClasses , vrPredictClasses ) **print**( ’ Classification Accuracy : ’ , accuracy )



Figura 18: Accuracy della classificazione ottenuta

classification report ( testClasses , vrPredictClasses , labels=labels )



Figura 19: Valori di Precision, Recall e F1-Score

Inoltre, `e stata generata e memorizzata all’interno di un file **csv** la matrice di confusione attraverso la funzione *confusion matrix*, anch’essa definita all’interno della libreria **sklearn**:

labels=**list** ( **set** ( testClasses ) ) vrConfusionMatrix = confusion matrix ( testClasses , vrPredictClasses , labels )

confmatrix2csv ( vrConfusionMatrix , labels , parameters . confMatrixCsvFile )

La matrice di confusione ottenuta `e, quindi, la seguente:

df cm = pd. DataFrame(vrConfMatrix , **range**(2) , **range**(2) ) sn . **set** ( font scale =1.4) . *# Label size*

sn . heatmap(df cm , annot=True , annot kws={” size ” : 16})

*# Font size*

plt . show()

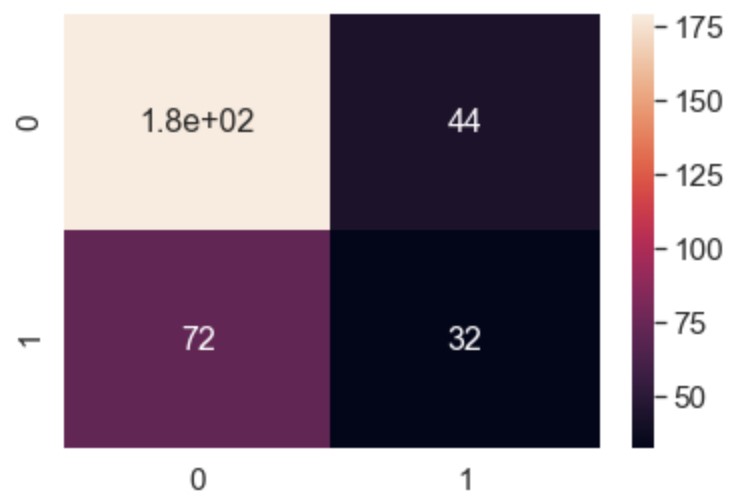


Figura 20: Matrice di Confusione

Infine, tale matrice `e stata memorizzata all’interno di un file **csv** attraverso la funzione *confmatrix2csv*, la cui implementazione `e riportata di seguito:

**def** confmatrix2csv ( conf matrix , labels , csvfile ) :

with **open**( csvfile , ’w’ ) as csvfile : csvWriter = csv . writer ( csvfile )

row=**list** ( labels ) row . insert (0 ,””) csvWriter . writerow (row) **for** i **in range**( conf matrix . shape [0]) :

row=**list** ( conf matrix [ i ]) row . insert (0 , labels [ i ]) csvWriter . writerow (row)

##### 5.3.3 Risultati ottenuti - Secondo Ciclo

Il tempo impiegato per effettuare il Training del classificatore dato il Training Set descritto precedentemente (che consta di 1’838 immagini) `e pari a 1h56’59”. Le metriche di **Accuracy**, **Precision**, **Recall** ed **F1-score** sono state stimate attraverso l’utilizzo delle funzioni *accuracy score* e *classification report*, entrambe definite all’interno della libreria **sklearn**. Di seguito sono riportati i risultati ottenuti:

accuracy = accuracy score ( testClasses , vrPredictClasses ) **print**( ’ Classification Accuracy : ’ , accuracy )



Figura 21: Accuracy della classificazione ottenuta

classification report ( testClasses , vrPredictClasses , labels=labels )

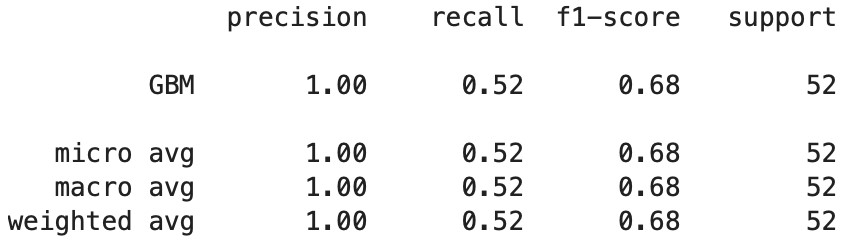


Figura 22: Valori di Precision, Recall e F1-Score

Osserviamo, quindi, che c’`e stato un calo dell’accuracy del classificatore di circa il 20% rispetto al primo ciclo di addestramento. Tale diminuzione `e una conseguenza del fatto che il classificatore non sia piu` stato in grado di riconoscere le metastasi (immagini appartenenti alla classe MET). Infatti, tutte le immagini classificate come MET appartenenti al Test Set sono state erroneamente riconosciute come Negative.

Inoltre, `e stata generata e memorizzata all’interno di un file **csv** la matrice di confusione attraverso la funzione *confusion matrix*, anch’essa definita all’interno della libreria **sklearn**:

labels=**list** ( **set** ( testClasses ) ) vrConfusionMatrix = confusion matrix ( testClasses , vrPredictClasses , labels )

confmatrix2csv ( vrConfusionMatrix , labels , parameters . confMatrixCsvFile )

La matrice di confusione ottenuta `e, quindi, la seguente:

df cm = pd. DataFrame(vrConfMatrix , **range**(2) , **range**(2) ) sn . **set** ( font scale =1.4) . *# Label size*

sn . heatmap(df cm , annot=True , annot kws={” size ” : 16})

*# Font size*

plt . show()

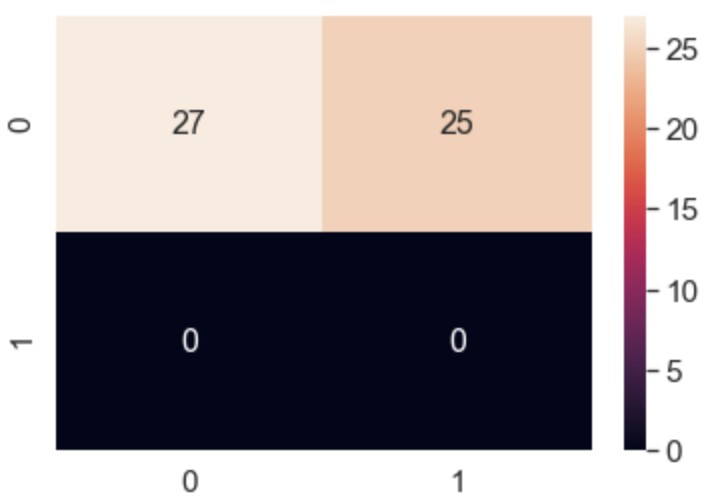


Figura 23: Matrice di Confusione

Dalla matrice `e possibile osservare come, appunto, il classificatore non abbia classificato affatto le immagini appartenenti alla classe MET che, invece, sono state riconosciute tutte come Negative.

Infine, tale matrice `e stata memorizzata all’interno di un file **csv** attraverso la funzione *confmatrix2csv*.

##### 5.3.4 Risultati ottenuti - Terzo Ciclo

Il tempo impiegato per effettuare il Training del classificatore dato il Training Set descritto precedentemente (che consta di 2’103 immagini) `e pari a 2h0’31”. Le metriche di **Accuracy**, **Precision**, **Recall** ed **F1-score** sono state stimate attraverso l’utilizzo delle funzioni *accuracy score* e *classification report*, entrambe definite all’interno della libreria **sklearn**. Di seguito sono riportati i risultati ottenuti:

accuracy = accuracy score ( testClasses , vrPredictClasses ) **print**( ’ Classification Accuracy : ’ , accuracy )



Figura 24: Accuracy della classificazione ottenuta

classification report ( testClasses , vrPredictClasses , labels=labels )

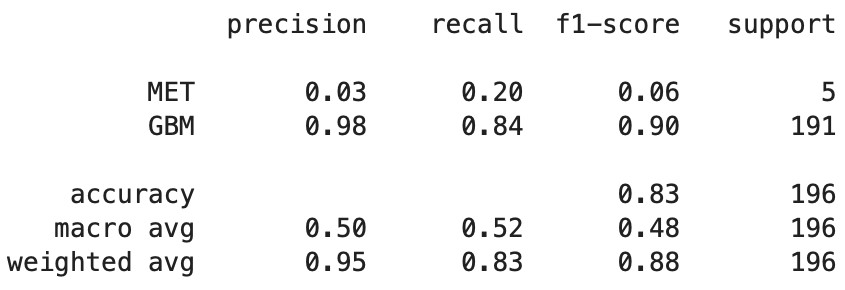


Figura 25: Valori di Precision, Recall e F1-Score

Inoltre, `e stata generata e memorizzata all’interno di un file **csv** la matrice di confusione attraverso la funzione *confusion matrix*, anch’essa definita all’interno della libreria **sklearn**:

labels=**list** ( **set** ( testClasses ) ) vrConfusionMatrix = confusion matrix ( testClasses , vrPredictClasses , labels )

confmatrix2csv ( vrConfusionMatrix , labels , parameters . confMatrixCsvFile )

La matrice di confusione ottenuta `e, quindi, la seguente:

df cm = pd. DataFrame(vrConfMatrix , **range**(2) , **range**(2) ) sn . **set** ( font scale =1.4) . *# Label size*

sn . heatmap(df cm , annot=True , annot kws={” size ” : 16})

*# Font size*

plt . show()

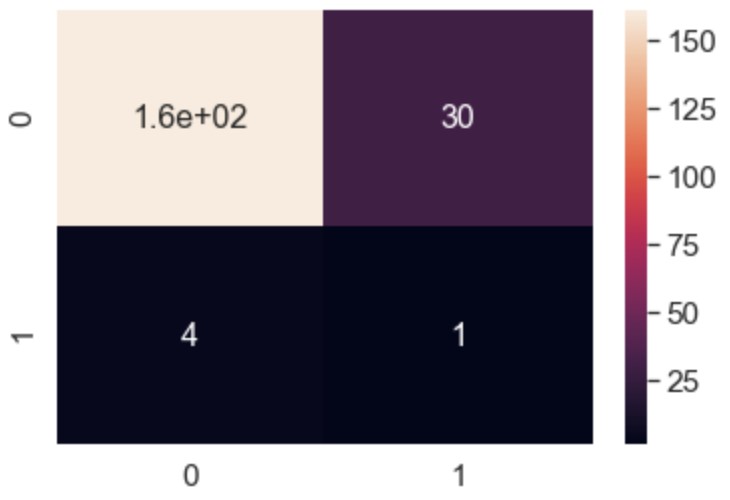


Figura 26: Matrice di Confusione

Infine, tale matrice `e stata memorizzata all’interno di un file **csv** attraverso la funzione *confmatrix2csv*.

##### 5.3.5 Risultati ottenuti - Quarto Ciclo

Il tempo impiegato per effettuare il Training del classificatore dato il Training Set descritto precedentemente (che consta di 956 immagini) `e pari a 1h04’28”. Le metriche di **Accuracy**, **Precision**, **Recall** ed **F1-score** sono state stimate attraverso l’utilizzo delle funzioni *accuracy score* e *classification report*, entrambe definite all’interno della libreria **sklearn**. Di seguito sono riportati i risultati ottenuti:

accuracy = accuracy score ( testClasses , vrPredictClasses ) **print**( ’ Classification Accuracy : ’ , accuracy )



Figura 27: Accuracy della classificazione ottenuta

classification report ( testClasses , vrPredictClasses , labels=labels )

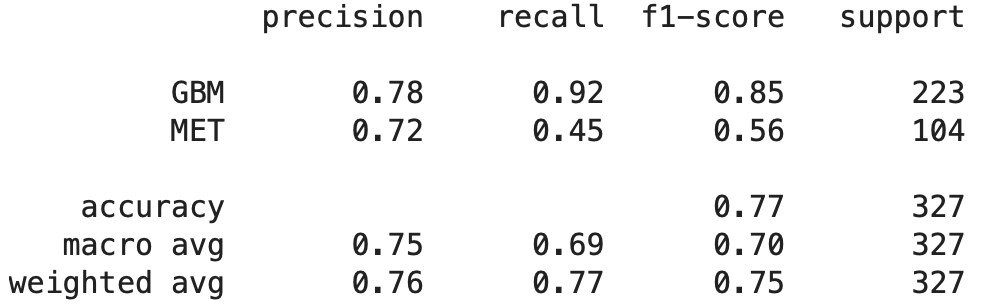


Figura 28: Valori di Precision, Recall e F1-Score

Inoltre, `e stata generata e memorizzata all’interno di un file **csv** la matrice di confusione attraverso la funzione *confusion matrix*, anch’essa definita all’interno della libreria **sklearn**:

labels=**list** ( **set** ( testClasses ) ) vrConfusionMatrix = confusion matrix ( testClasses , vrPredictClasses , labels )

confmatrix2csv ( vrConfusionMatrix , labels , parameters . confMatrixCsvFile )

La matrice di confusione ottenuta `e, quindi, la seguente:

df cm = pd. DataFrame(vrConfMatrix , **range**(2) , **range**(2) ) sn . **set** ( font scale =1.4) . *# Label size*

sn . heatmap(df cm , annot=True , annot kws={” size ” : 16})

*# Font size*

plt . show()

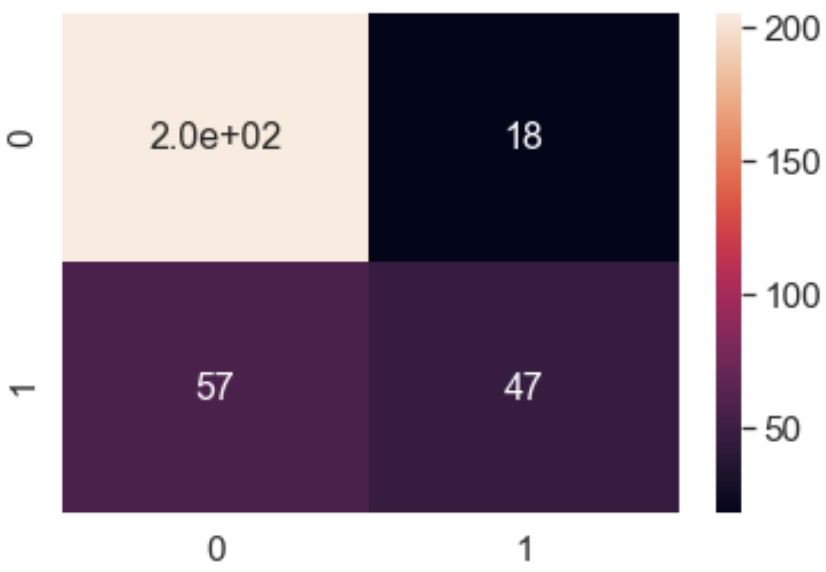


Figura 29: Matrice di Confusione

Infine, tale matrice `e stata memorizzata all’interno di un file **csv** attraverso la funzione *confmatrix2csv*.

**Appendice**

## A RadiAnt

RadiAnt DICOM Viewer `e un visualizzatore di file DICOM sviluppato dalla compagnia Medixant. Il suo primo rilascio, in versione Beta, risale al 2010 con la versione 0.1, ad oggi `e arrivato alla versione 2020.1 (equivalente alla release 6 del software) rilasciato il 6 Aprile 2020. Attraverso un’interfaccia grafica user-friendly, RadiAnt fornisce strumenti per lo studio delle immagini DICOM quali:

* Zoom e spostamento
* Regolazione della luminosit`a e del contrasto
* Rotazione e capovolgimento delle immagini
* Conversione in formati BMP e JPG

Il software supporta l’apertura di file ottenuti da differenti modalit`a di immagini quali:

* Radiografia Digitale (CR, DX)
* Mammografia (MG)
* Tomografia Computerizzata (CT)
* Risonanza Magnetica (MR)
* Tomografia ad Emissione di Positroni PET-CT (PT)
* Ecografia (US)
* Angiografia Digitale (XA)
* Gamma Camera, Medicina Nucleare (NM)
* Immagini secondarie e immagini digitalizzate (SC)
* Structured Reports (SR)

RadiAnt fornisce numerose funzionalit`a, tra cui alcune da noi sfruttate, quali:

### • Confronto di serie o studi differenti

E possibile aprire contemporaneamente piu` serie di un determinato stu-` dio oppure di studi differenti nelle stesse o in diverse finestre al fine di effettuare una comparazione fra essi.

### • Ricostruzioni Multiplanari

Strumento in gradi di ricostruire immagini su piani ortogonali differenti (coronali, sagittali, obliqui, assiali).

### • Esportazione di files DICOM come immagini e filmati

E possibile esportare i file DICOM aperti sia come immagini nei formati` JPG e BMP sia come video in formato WMV (Windows Media Video) e MP4.

### • Rendering del Volume 3D

Strumento che consente di visualizzare grandi volumi di dati generati dagli scanner CT/MR in uno spazio tridimensionale.

### A.1 Pre-processing effettuato tramite RadiAnt

L’utilizzo di RadiAnt `e risultato fondamentale nella fase di pre-processing dei file DICOM prima che venissero dati in pasto al classificatore dell’istanza di IBM Visual Recognition; il software permette la lettura di intere cartelle di file DICOM, unendole in singole sequenze.

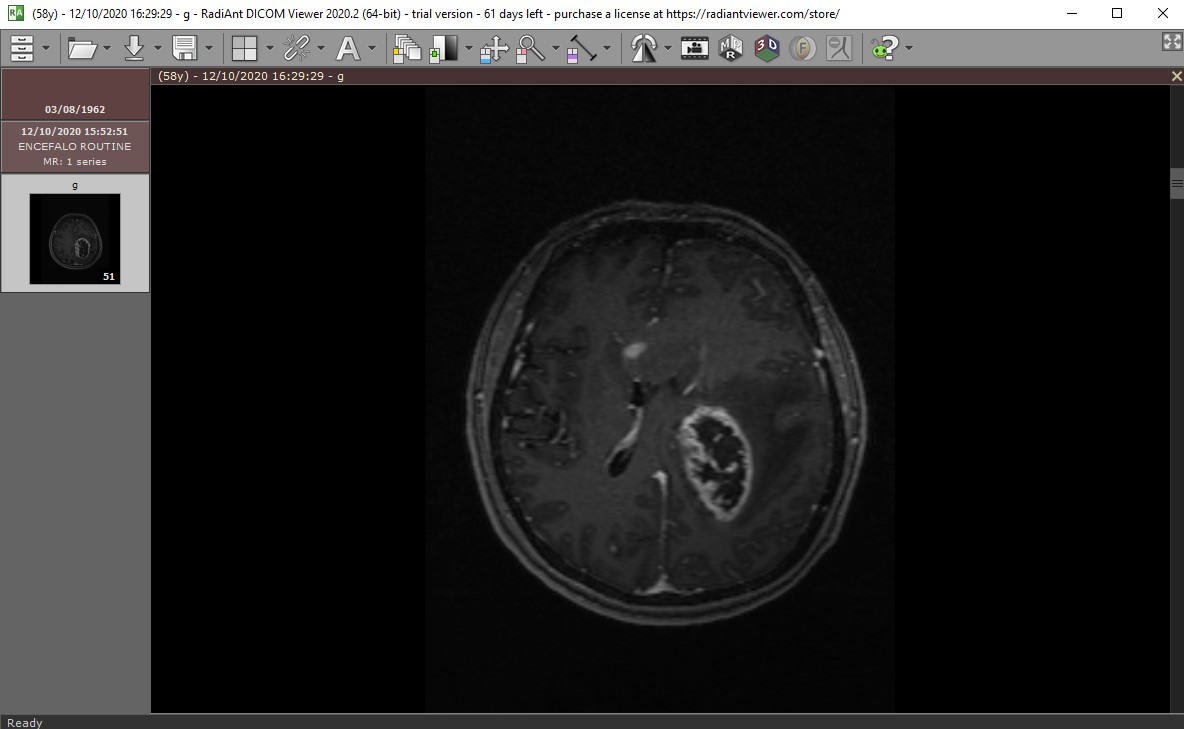


Figura 30: Apertura di una singola cartella contenente una sequenza composta da 52 file DICOM

E anche possibile caricare nel software piu` cartelle contemporaneamente in modo` da poter visualizzare contemporaneamente piu` sequenze.

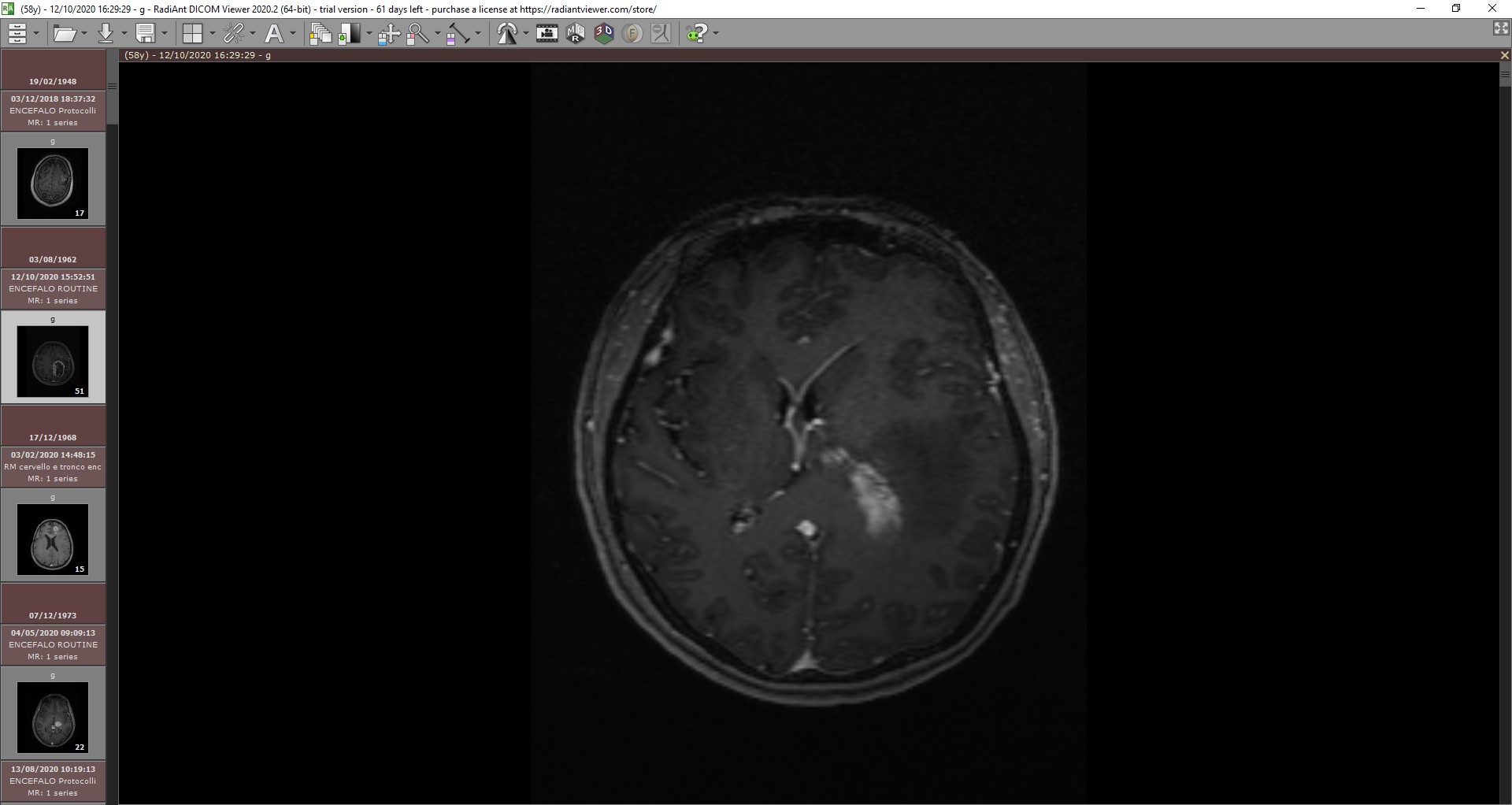


Figura 31: Apertura di piu` sequenze

Una volta aperte le MRI di nostro interesse, `e stata utilizzata la funzionalit`a di conversione in JPG.

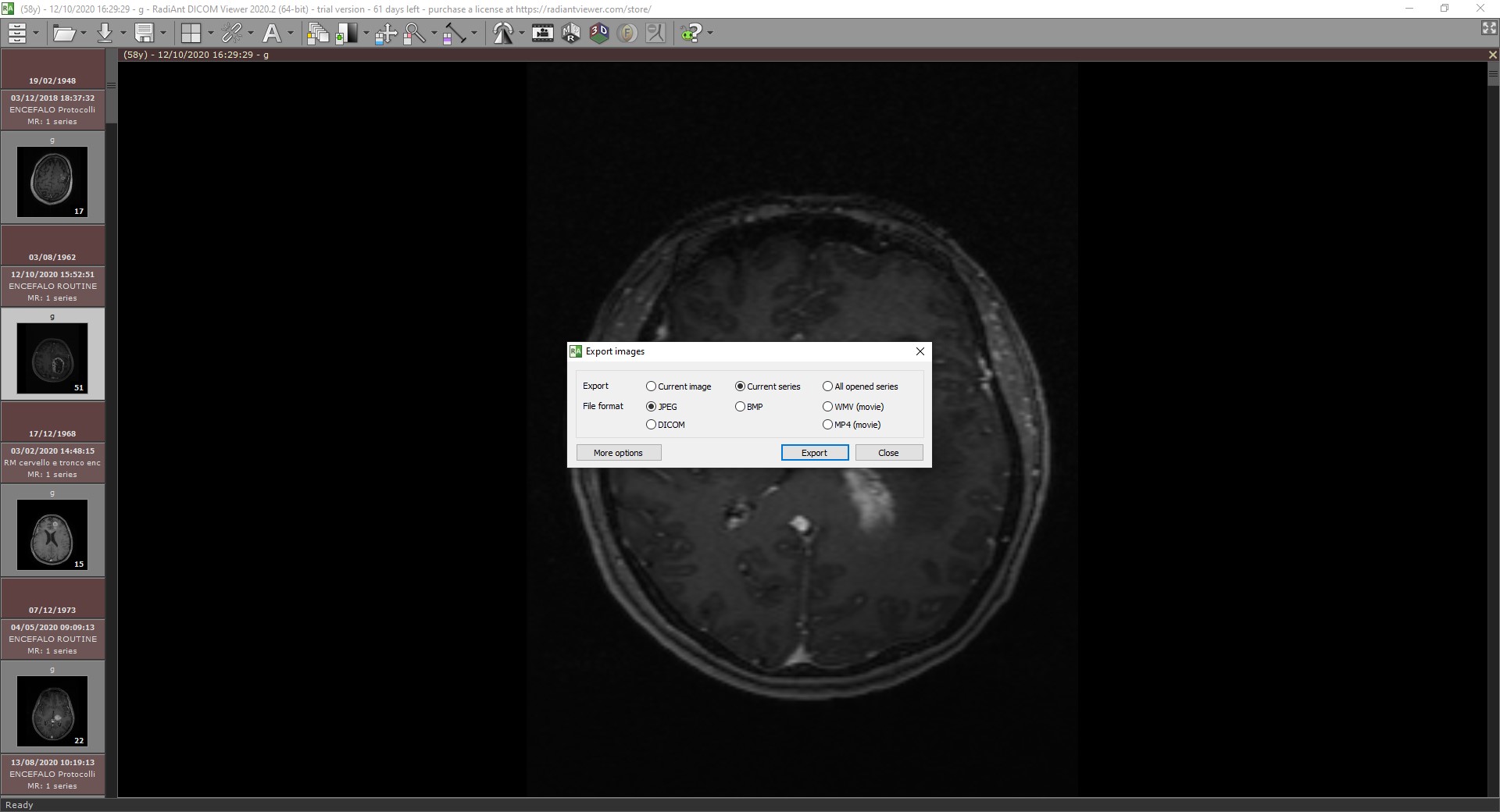


Figura 32: Conversione ed Export in JPG

Le immagini delle risonanze in JPG sono state compresse in un file ZIP in modo da poter caricare l’intero set di sequenze nel Visual Recognition.

## Riferimenti bibliografici

[1] E-utilities: [https://www.ncbi.nlm.nih.gov/books/NBK25501/;](https://www.ncbi.nlm.nih.gov/books/NBK25501/) [2] IBM Cloud Db2: [https://www.ibm.com/it-it/cloud/db2-on-cloud;](https://www.ibm.com/it-it/cloud/db2-on-cloud)

1. IBM Cloud Foundry: [https://www.ibm.com/it-it/cloud/cloud-foundry;](https://www.ibm.com/it-it/cloud/cloud-foundry)
2. GitHub: [https://github.com;](https://github.com/)
3. MeSH: [https://it.wikipedia.org/wiki/Medical\_Subject\_Headings;](https://it.wikipedia.org/wiki/Medical_Subject_Headings)
4. IBM Watson Machine Learning: [https://www.ibm.com/it-it/cloud/machine-learning;](https://www.ibm.com/it-it/cloud/machine-learning)
5. IBM Watson Knowledge Studio: [https://www.ibm.com/it-it/cloud/watson-knowledge-studio;](https://www.ibm.com/it-it/cloud/watson-knowledge-studio)
6. IBM Watson Discovery: [https://www.ibm.com/it-it/cloud/watson-discovery;](https://www.ibm.com/it-it/cloud/watson-discovery)
7. fastai: [https://www.fast.ai;](https://www.fast.ai/)
8. IBM Cloud API:

[https://cloud.ibm.com/apidocs/visual-recognition/ visual-recognition-v3;](https://cloud.ibm.com/apidocs/visual-recognition/visual-recognition-v3)

1. Jupyter: [https://jupyter.org.](https://jupyter.org/)
2. RadiAnt DICOM Viewer: <https://www.radiantviewer.com/it>

1. Medical Subject Headings (acronimo: MeSH) `e un enorme vocabolario controllato (o sistema di metadati) ideato con l’obiettivo di indicizzare la letteratura scientifica in ambito biomedico. Il tesauro `e stato creato dalla National Library of Medicine (NLM) degli Stati Uniti, che `e responsabile anche della sua gestione. Il MeSH viene adoperato per l’indicizzazione degli articoli delle oltre 5000 riviste mediche presenti nel database bibliografico Medline/PubMed e nel catalogo dei libri della NLM. Il vocabolario pu`o essere consultato e scaricato gratuitamente da tutti gli utenti di internet. La stampa dell’edizione cartacea `e stata abbandonata nel 2007. [↑](#footnote-ref-1)
2. In informatica, nell’ambito della progettazione dei database, il modello entity-relationship

   (detto anche modello E-R; in italiano: modello entit`a-associazione o modello entit`a-relazione) `e un modello teorico per la rappresentazione concettuale e grafica dei dati a un alto livello di astrazione, formalizzato dal prof. Peter Chen nel 1976.

   Il modello entit`a-associazione viene spesso utilizzato nella prima fase della progettazione di una base di dati, nella quale `e necessario tradurre le informazioni risultanti dall’analisi di un determinato dominio in uno schema concettuale, chiamato schema E-R (schema entit`aassociazione) o diagramma E-R (diagramma entita`-associazione). [↑](#footnote-ref-2)
3. Un pre-annotator `e un dizionario di Knowledge Studio, un modello Rule-Based oppure un modello di machine learning che puo` essere eseguito per annotare automaticamente un testo. [↑](#footnote-ref-3)
4. La tecnica dell’Holdout prevede di suddividere un dataset in modo tale che 2/3 di esso siano usati come Training Set e il restante 1/3 viene usato come Test Set. [↑](#footnote-ref-4)
5. Nell’ambito dell’Intelligenza artificiale, la matrice di confusione, detta anche tabella di errata classificazione, restituisce una rappresentazione dell’accuratezza di classificazione statistica. Ogni colonna della matrice rappresenta i valori predetti, mentre ogni riga rappresenta i valori reali. L’elemento sulla riga i e sulla colonna j `e il numero di casi in cui il classificatore ha classificato la classe ”vera” i come classe j. Attraverso questa matrice `e osservabile se vi `e ”confusione” nella classificazione di diverse classi. [↑](#footnote-ref-5)