UNIVERSITÀ DI ROMA TOR VERGATA

Corso di Laurea Magistrale in Informatica

Tesi di Laurea Magistrale

Un sistema di classificazione e indicizzazione di supporto ad attività di raccolta, catalogazione e accesso a notizie di carattere medico-sanitario

Relatore
Prof. Giorgio Gambosi
Prof. Danilo Croce

Candidato
Dott. Paolo Ciasco

Anno Accademico 2021-2022

Indice

Capitolo 1

Introduzione

Questa produzione sperimentale si occuperà di utilizzare elementi delle aree informatiche di **Information Retrieval**, **Machine Learning** e **Natural Language Processing (NLP)** per un caso d'uso realistico, in collaborazione con l'**Istituto Superiore di Sanità** [1].
L'indicizzazione, strettamente legata alla ricerca sul web di informazioni, ricopre un ruolo importantissimo nella vita di tutti i giorni di molte persone; basti pensare, difatti, che ogni giorno vengono fatte nel mondo 5.6 miliardi di ricerche su *google.com*[1]. La stessa, però, viene anche utilizzata in ambiti più locali e ristretti come imprese, banche e attività statali affinché possano essere costruiti motori di ricerca ad hoc atti ad esaudirne le richieste logistiche.
Nella prima parte di questa tesi, infatti, il lavoro sarà concentrato ad indicizzare documenti (opportunamente elaborati) di test, ma con la stessa struttura di quelli finali, relativi alla collezione di notizie riguardanti il noto fenomeno virale che ha stravolto il globo terracqueo (*"COVID-19"*) tramite **Apache Solr** [2], una piattaforma di sviluppo open source di motori di ricerca.
L'elaborazione di linguaggio naturale, in stretto connubio con il Machine Learning, è altresì un argomento veramente rilevante nella pletora di elementi che vengono toccati dall'informatica: avere interazioni continue fra uomo e macchina è diventata una normalità della quotidianità di ogni individuo.
Citando da Wikipedia: "Lo scopo è rendere la tecnologia in grado di "comprendere" il contenuto dei documenti e le loro sfumature contestuali, in modo tale che possa quindi estrarre con precisione informazioni e idee contenute nei documenti, nonché classificare e categorizzare i documenti stessi." [3]
È proprio questa definizione che meglio caratterizza quello che verrà svolto nella seconda parte di questo lavoro di tesi: una volta che i dati saranno raccolti, il lavoro sperimentale cercherà di estrarre informazioni dal testo degli stessi cercando di categorizzare e classificare le notizie nella migliore maniera possibile, tramite approcci classici (supervised) diversi di *"Text Classification"*.

[1] Dati raccolti da SEOTribunal.com nel 2019

Nel corso della trattazione della classificazione saranno utilizzati due dataset differenti con due procedure parzialmente differenti fra loro: il primo che verrà usato, infatti, non sarà quello definitivo ma si proverà a categorizzare in classi molto diverse da quelle conclusive; il modello che risulterà essere il "migliore" sarà utilizzato sul dataset definitivo per provare a vedere una compatibilità di risultati che sia appagante.
In un certo senso si proverà una strada alternativa a partire da delle "pseudocategorie" per provare a classificare nelle categorie finali; in ogni caso i modelli saranno addestrati anche sul dataset definitivo.
Nell'ultimo capitolo di questa tesi, poi, verrà sviluppato un servizio WEB (**NewsWorld**) che metterà insieme tutti i singoli procedimenti descritti in questo elaborato e che permetterà ad un utente, dopo aver caricato un file ben strutturato con delle notizie non categorizzate, di avere la classificazione delle stesse, la loro indicizzazione e la possibilità di eseguire delle query.
Questo completerà il lavoro sperimentale nella sua totalità.
Infine, nella conclusione di questo elaborato, si esploreranno eventuali sviluppi futuri atti a migliorare il sistema in toto.

Capitolo 2

Indicizzazione

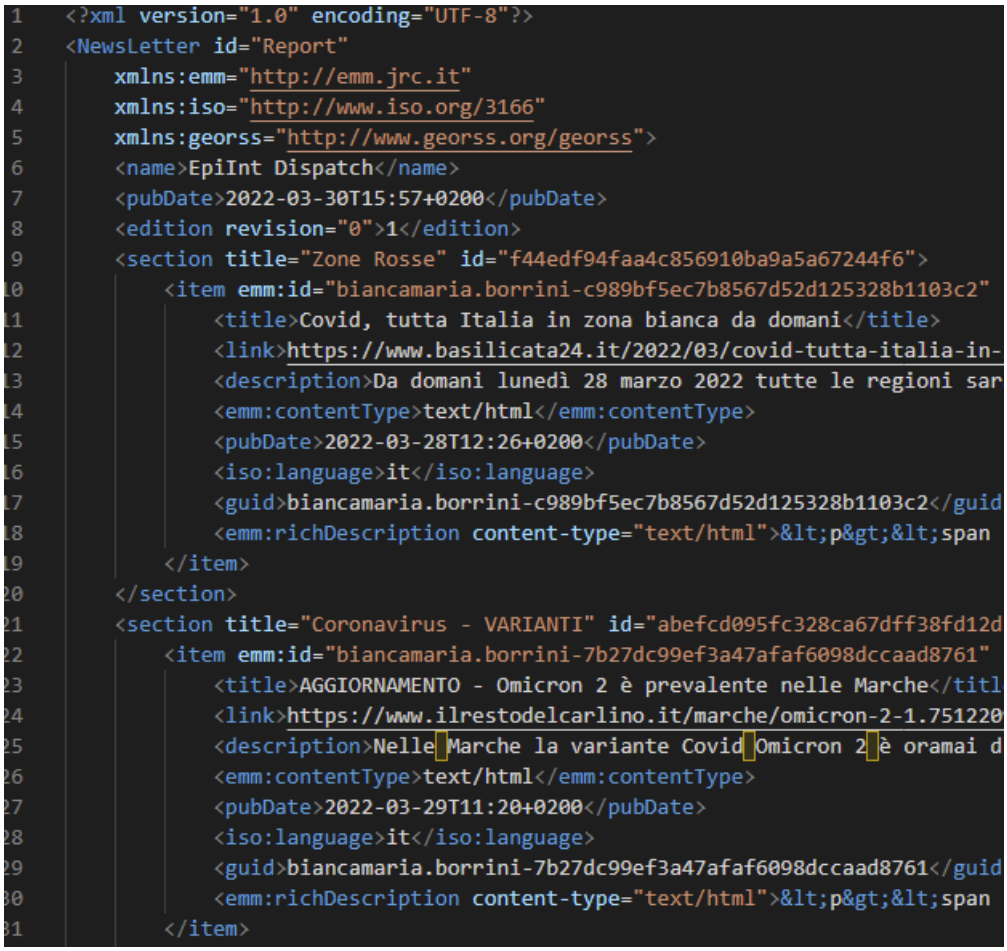
In questa parte verrà spiegato ed illustrato il processo di indicizzazione dei dati tramite **Apache Solr**; verranno descritti i dati a disposizione e la loro struttura intrinseca, verranno preparati affinché siano riconosciuti correttamente dall'handler di Solr per poter essere mandati in indicizzazione ed infine verranno, appunto, indicizzati secondo opportune regole.
I dati forniti, al momento di stesura, rappresentano la base su cui tutto il lavoro di questo capitolo è fondato; è doveroso specificare come quanto segue, specialmente per quanto riguarda il processo di adattamento degli stessi affinché Solr possa immagazzinarli correttamente, è strettamente legato alla struttura fornita e cambiamenti rilevanti della stessa potrebbero invalidare il processo di seguito scritto.

2.1 Analisi preliminare dei dati

I dati vengono consegnati in un file con estensione **"XML"**.
Il formato "XML" [4] è uno standard riconosciuto dall'organizzazione mondiale "W3C" [5] per il *markup* dei documenti. Sfrutta sintassi generica tramite l'utilizzo di semplici *tag* affinché il tutto rimanga il più possibile "human-readable" e la flessibilità di questo metalinguaggio è una delle principali ragioni affinché esso stesso sia ampiamente utilizzato in domini di estrazione totalmente differenti fra loro. Infatti, XML, non fornisce un set ben definito di tags da utilizzare in ogni area di interesse bensì permette, a chi scrive, di creare i propri tags ed elementi necessari allo scopo.
Infine, è doveroso scriverlo, il formato XML va a definire nel documento una vera e propria struttura ad albero che sarà molto utile per lo scopo di questo elaborato, in quanto ci permetterà di eseguire *data manipulation* in modo relativamente semplice e veloce.
Dopo questo breve excursus si pone ora l'attenzione sul topic principale di questa sezione. I dati forniti rappresentano la collezione di notizie riguardanti "COVID-19" raccolte da persone afferenti all'ISS. Dopo aver formattato meglio il codice fornito tramite un "Beautify"[1] del noto editor di testo "Visual Studio Code" [6] per avere più chiara la struttura

[1] XML Formatter by Fabian Lauer - Marketplace VSCode Extension

generale del file, è stata eseguita un'analisi sommaria di quest'ultima.



```
<?xml version="1.0" encoding="UTF-8"?>
<NewsLetter id="Report"
    xmlns:emm="http://emm.jrc.it"
    xmlns:iso="http://www.iso.org/3166"
    xmlns:georss="http://www.georss.org/georss">
    <name>EpiInt Dispatch</name>
    <pubDate>2022-03-30T15:57+0200</pubDate>
    <edition revision="0">1</edition>
    <section title="Zone Rosse" id="f44edf94faa4c856910ba9a5a67244f6">
        <item emm:id="biancamaria.borrini-c989bf5ec7b8567d52d125328b1103c2"
            <title>Covid, tutta Italia in zona bianca da domani</title>
            <link>https://www.basilicata24.it/2022/03/covid-tutta-italia-in-
            <description>Da domani lunedì 28 marzo 2022 tutte le regioni sar
            <emm:contentType>text/html</emm:contentType>
            <pubDate>2022-03-28T12:26+0200</pubDate>
            <iso:language>it</iso:language>
            <guid>biancamaria.borrini-c989bf5ec7b8567d52d125328b1103c2</guid
            <emm:richDescription content-type="text/html">&lt;p&gt;&lt;span
        </item>
    </section>
    <section title="Coronavirus - VARIANTI" id="abefcd095fc328ca67dff38fd12d
        <item emm:id="biancamaria.borrini-7b27dc99ef3a47afaf6098dccaad8761"
            <title>AGGIORNAMENTO - Omicron 2 è prevalente nelle Marche</titl
            <link>https://www.ilrestodelcarlino.it/marche/omicron-2-1.751220
            <description>Nelle Marche la variante Covid Omicron 2 è oramai d
            <emm:contentType>text/html</emm:contentType>
            <pubDate>2022-03-29T11:20+0200</pubDate>
            <iso:language>it</iso:language>
            <guid>biancamaria.borrini-7b27dc99ef3a47afaf6098dccaad8761</guid
            <emm:richDescription content-type="text/html">&lt;p&gt;&lt;span
        </item>
```

Figura 2.1. Screenshot di una porzione di dati visualizzati tramite VSCode dopo la formattazione.

Le informazioni presentano subito due sezioni molto generali: "Zone Rosse" (Riguardante il cambio di colore delle varie regioni imposto dal governo) e "Coronavirus - VARIANTI" (Riguardante notizie principalmente di carattere generale sulla nascita di nuove varianti nel paese o di prevalenza di una specifica in determinate regioni).
All'interno di queste sezioni (e nelle seguenti che analizzeremo poi) è possibile notare informazioni potenzialmente importanti per una successiva indicizzazione come la data di pubblicazione, il testo della notizia, il titolo della stessa e la fonte (generalmente siti di giornali online) mentre è già possibile prevedere un'eventuale troncatura dei dati per quello che riguarda codice HTML, specifica del contenuto HTML (stesso testo con i tag), linguaggio della notizia (tutte le notizie saranno raccolte in italiano) e autore del raccoglimento della notizia (siamo interessati alla notizia in sé).

A seguire sono presenti cinque sezioni che specificano le notizie dei differenti focolai di contagio nei principali contesti urbani: "Coronavirus - Focolai familiari/amici", "Coronavirus - Focolai scolastico", "Coronavirus - Focolai RSA/Case di riposo", "Coronavirus - Focolai ospedalieri" e "Coronavirus - Focolai in altri contesti". A riguardo di queste cinque e delle precedenti due sezioni, si decide di tenere traccia di queste "categorie di appartenenza" delle notizie; potrebbero essere utili in fase di indicizzazione.
Ogni sezione di quelle appena scritte possiede al suo interno una sottosezione che va a specificare le notizie regione per regione; è importante sottolineare che le regioni sono numerate a partire dal numero 010 fino al numero 200 ad intervalli di 10 con l'eccezione dello 040 (che sarebbe dovuto essere il Trentino Alto Adige) sostituito dai numeri 041 e 042 rappresentanti rispettivamente le due province autonome di Trento e Bolzano in quanto province a statuto speciale.
Infine, da una analisi più approfondita, è possibile notare come potrebbe succedere che le notizie siano di carattere nazionale e non specificate su una particolare regione, anche nelle sezioni seguenti alle prime due descritte all'inizio di questo sottocapitolo: bisognerà tenerne conto in fase di trasformazione dei dati.

N.B.: Le categorie qui specificate saranno parzialmente diverse nei dati finali ma il processo automatico creato fornisce molta flessibilità in questo senso.

2.2 Preparazione dei dati

Solr richiede uno schema ben preciso di indicizzazione di documenti in XML (si veda Figura 2.2) e questa parte di questo elaborato è dedicata a ripercorrere i momenti chiavi della manipolazione del documento originario in maniera propedeutica allo scopo.

```
<add>
  <doc>
    <field name="authors">Patrick Eagar</field>
    <field name="subject">Sports</field>
    <field name="dd">796.35</field>
    <field name="numpages">128</field>
    <field name="desc"></field>
    <field name="price">12.40</field>
    <field name="title">Summer of the all-rounder: Test and championship cricket in England
1982</field>
    <field name="isbn">0002166313</field>
    <field name="yearpub">1982</field>
    <field name="publisher">Collins</field>
  </doc>
  <doc>
  ...
  </doc>
</add>
```

Figura 2.2. Esempio di formato necessario all'immissione di documenti in un indice Solr.

In una concezione classica di indicizzazione ci sono molteplici documenti che vengono poi elaborati per costruire un indice ma in questo caso si ha un unico documento monolite che va fatto digerire al motore di Solr.
Il lavoro di preparazione dei dati sarà eseguito tramite la produzione di codice in linguaggio di programmazione **Python** [7].
L'idea di fondo è quella di creare un nuovo file .xml con la struttura voluta copiando mano a mano le informazioni dal file originale.

XML vs JSON [8]
È più che nota la battaglia riguardo l'utilizzo dell'uno o dell'altro formato e non è certo scopo di questo elaborato argomentare su di essa; si è deciso, dopo un'attenta valutazione, di rimanere con lo stesso formato di partenza poiché, sebbene JSON sia più semplice di XML in termini prettamente strutturali, non si è voluto rinunciare ad una eventuale asseverazione di integrità di dati (anche considerando un ente di Stato) e perché l'indicizzazione con Solr possa avvenire senza ulteriori problemi.

Un primo problema affrontato è stato quello di trovare una libreria che potesse facilitare l'accesso e il parsing di file xml; dopo varie ricerche, la scelta è ricaduta su di una dello standard di Python per processare, appunto, dati strutturati: **ElementTree** [9].
Dopo aver eseguito la registrazione dei "Namespace" affinché nel file di output non occorressero tag indesiderati, si è proceduto a caricare il file originale ed a iniziare la struttura del nuovo file xml inserendo come root il tag "add", necessario per "avvertire" l'handler di Solr che da quel punto in poi ci sarebbero stati dei documenti da indicizzare.
Viene illustrato adesso il processo di costruzione della struttura del file di output.

```
et.register_namespace('emm', "http://emm.jrc.it")
et.register_namespace('iso', "http://www.iso.org/3166")
et.register_namespace('georss', "http://www.georss.org/georss")
```

Figura 2.3. Registrazione dei Namespace.

Per ovviare al problema introdotto alla fine del sottocapitolo 2.1 riguardo il fatto che alcune notizie fossero nazionali (e quindi non inserite in una specifica sezione di una regione) ed altre no, si è deciso di agire nel seguente modo: dal momento che ogni documento di interesse è circondato da tag "item", si sono costruiti due cicli differenti, sia per il nazionale che per il regionale, affinché sia garantito, nel file di output, che per ogni tag "item" nel file originale ci sia un tag "doc" (necessario affinché l'handler di Solr riconosca l'inizio e la fine di un documento) nel file da indicizzare. Ovviamente questi cicli sono scritti affinché ogni "item" del documento originale sia raggiunto, ovunque esso sia, nello "standard" imposto.
Affinché i tag "doc" siano inseriti nel file di destinazione, viene sfruttata una lista per avere facilità di accesso nelle operazioni seguenti di copia di informazioni rilevanti.

```
for child in myroot:
    if (child.tag == 'section'):
        for child2 in child:
            if (child2.tag == 'item'):
                doc.append(et.SubElement(newroot, "doc"))


for child in myroot:
    if (child.tag == 'section'):
        for child2 in child:
            if (child2.tag == 'section'):
                for child3 in child2:
                    if (child3.tag == 'item'):
                        doc.append(et.SubElement(newroot, "doc"))
```

Figura 2.4. Iterazione nel file originale per acquisire tutti i tag "item" e costruire le fondamenta della struttura del file di output da indicizzare

Una volta ottenuta la struttura desiderata nel file di output, è il momento di selezionare e copiare le informazioni rilevanti dal file originale a quello nuovo.
Rispecchiando l'analisi fatta in precedenza si è optato per le seguenti:

- **Titolo** - tag = "title"
- **Fonte** - tag = "source"
- **Notizia** - tag = "news"
- **Data** - tag = "date"
- **Categoria di appartenenza** - tag = "category"

Anche in questa parte, come nella precedente, si sono creati due cicli differenti dove ognuno raggiunge l'informazione necessaria e la copia in un nuovo sottoelemento di "doc" denominato "field" (necessario affinché l'handler di Solr riconosca l'elemento in sé da memorizzare).
Rispetto il documento originale vengono acquisiti i campi dei seguenti tag:

- **Title**
- **Link**
- **Description**
- **pubDate**
- **Title (Categoria)**

```
for child in myroot:
    if (child.tag == 'section'):
        for child2 in child:
            if (child2.tag == 'item'):
                for child3 in child2:
                    if (child3.tag == 'title'):
                        et.SubElement(doc[i], "field", name="title").text = child3.text
                    if (child3.tag == 'link'):
                        et.SubElement(doc[i], "field", name="source").text = child3.text
                    if (child3.tag == 'description'):
                        et.SubElement(doc[i], "field", name="news").text = child3.text
                    if (child3.tag == 'pubDate'):
                        et.SubElement(doc[i], "field", name="date").text = child3.text
                        et.SubElement(doc[i], "field", name = "category").text = child.attrib.get("title")
                        i += 1
```

Figura 2.5. Iterazione nel file originale per acquisire tutte le informazioni(in figura quelle a livello nazionale) selezionate e copiarle nel file di output da indicizzare

La presenza degli iteratori *i* e *i2* (per i due cicli) è necessaria affinché tutte le informazioni siano copiate nel documento corrispondente corretto.
Infine, viene generato l'albero finale del file di output con l'indentazione corretta (necessario Python >= 3.9) e codifica "UTF-8" e viene salvato in un file apposito, pronto per essere indicizzato da Solr.

```
<add>
    <doc>
        <field name="title">Covid, tutta Italia in zona bianca da domani</field>
        <field name="source">https://www.basilicata24.it/2022/03/covid-tutta-italia-in-zona-bianca
        <field name="news">Da domani lunedì 28 marzo 2022 tutte le regioni saranno in zona bianca.
        <field name="date">2022-03-28T12:26+0200</field>
        <field name="category">Zone Rosse</field>
    </doc>
    <doc>
        <field name="title">AGGIORNAMENTO - Omicron 2 è prevalente nelle Marche</field>
        <field name="source">https://www.ilrestodelcarlino.it/marche/omicron-2-1.7512209</field>
        <field name="news">Nelle Marche la variante Covid Omicron 2 è oramai diventata dominante.
        <field name="date">2022-03-29T11:20+0200</field>
        <field name="category">Coronavirus - VARIANTI</field>
    </doc>
    <doc>
        <field name="title">AGGIORNAMENTO - Covid, Istituto zooprofilattico: «In Veneto Omicron BA
        <field name="source">https://www.ilgiornaledivicenza.it/rubriche/salute-benessere/news/cov
        <field name="news">La variante BA.2, nota come Omicron 2 di Sars-Cov2 è passata in Veneto
È stato ottenuto il genoma completo di 209 campioni inviati da 12 diversi laboratori distribuiti i
        <field name="date">2022-03-28T14:39+0200</field>
        <field name="category">Coronavirus - VARIANTI</field>
    </doc>
```

Figura 2.6. Screenshot di una parte dei dati dopo il trattamento eseguito.

N.B.: Il codice (*xml_solr_like.py*) è consultabile per esteso visitando la repository https://github.com/PaoloCiasco/TesiMagistrale

2.3 Indicizzazione con Solr

Prima di entrare nel dettaglio tecnico implementativo dell'indicizzazione, è corretto spendere due righe nei confronti di **Apache Solr** [10].
Solr è un motore di ricerca open-source costruito su di una libreria altresì open-source sempre della stessa fondazione no-profit Apache: **Lucene**.
È possibile riassumere il funzionamento di Solr in 3 passaggi chiave:

1. Definire uno *schema*. Lo *schema* è una sorta di lista di regole che guidano il motore a indicizzare nel modo più utile necessario allo sviluppatore.

2. Fornire a Solr i documenti da indicizzare.

3. Utilizzare le funzionalità di ricerca di Solr sulla propria applicazione WEB/locale tramite chiamate, ad esempio, HTTP.

Lucene, su cui ricordiamo è costruito Solr, funziona sulla base di un **indice inverso** per archiviare documenti, ovvero invece di utilizzare una chiave incentrata sul documento utilizza una chiave incentrata sulle parole per un recupero più rapido ed efficiente delle informazioni.
In Solr tutto è memorizzato come un insieme di **documenti** i quali hanno al loro interno un insieme di **fields** che non sono altro che una rappresentazione *"chiave-valore"* dei dati. La raccolta di documenti insieme forma un **indice**. La definizione dello schema prima introdotto e tutte le configurazioni di Solr alloggiano rispettivamente nei file *schema.xml* e *solrconfig.xml*.
Solr introduce al suo interno due concetti fondamentali: **Core** e **Collection**. Differenziare questi due concetti trova senso solo se la ricerca è distribuita, ovvero un indice viene separato in più parti. La Collection è, infatti, un indice logico distribuito su più server mentre il Core è quella parte del server che esegue una Collection.

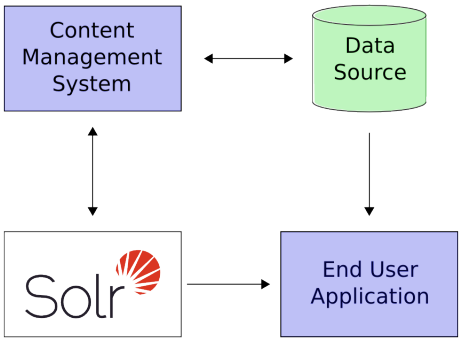


Figura 2.7. Panoramica del funzionamento di Solr in una architettura completa di supporto a una web application.

Sebbene negli ultimi anni la scelta della maggioranza degli sviluppatori stia ricadendo su *ElasticSearch*, un similare di Solr, secondo dati raccolti dal noto portale di ranking "DB-Engines"[2], ancora oggi è usato in moltissime aziende: esempio più che vicino a noi il noto portale online di informazione "www.repubblica.it" [11].
Concluso questo breve preambolo al funzionamento di Solr e alcune sue caratteristiche, si illustrano ora i passaggi svolti per ottenere l'indice finale [12].
Dopo aver avviato un'istanza di Solr in locale e dopo aver creato un core denominato **"Iss_Report"** si è proceduto all'indicizzazione dei documenti a disposizione racchiusi nell'unico file prodotto dopo la trasformazione dei dati. La richiesta al server di Solr è avvenuta tramite una chiamata *"POST"* ed il processo si è concluso positivamente.
Sono stati usati i file di configurazione di default sia per quanto riguarda lo "schema" che il "solrconfig" dal momento che non è stata resa necessaria una customizzazione.
Di seguito l'interrogazione al server (uno screenshot dei primi due risultati), tramite interfaccia WEB grafica, per mostrare i primi 10 documenti indicizzati:


```
{
  "responseHeader":{
    "status":0,
    "QTime":0,
    "params":{
      "q":"*:*",
      "indent":"true",
      "q.op":"OR",
      "_":"1659083010975"}},
  "response":{"numFound":32,"start":0,"numFoundExact":true,"docs":[
      {
        "title":["Covid, tutta Italia in zona bianca da domani"],
        "source":["https://www.basilicata24.it/2022/03/covid-tutta-italia-in-zona-bianca-da-domani-110187/"],
        "news":["Da domani lunedì 28 marzo 2022 tutte le regioni saranno in zona bianca. Anche la Sardegna dunque passa r
        "date":["2022-03-28T12:26+0200"],
        "category":["Zone Rosse"],
        "id":"e9bed68e-b3a4-4b9e-8e6b-e7c3cab4d0c5",
        "_version_":1739674618612940800},
      {
        "title":["AGGIORNAMENTO - Omicron 2 è prevalente nelle Marche"],
        "source":["https://www.ilrestodelcarlino.it/marche/omicron-2-1.7512209"],
        "news":["Nelle Marche la variante Covid Omicron 2 è oramai diventata dominante. Lo dice il virologo Stefano Menzo
        "date":["2022-03-29T11:20+0200"],
        "category":["Coronavirus - VARIANTI"],
        "id":"8bce1105-6609-4f29-bad5-5008e33b92f2",
        "_version_":1739674618648592384},
      {
```


N.B.: I due campi "id" e "_version_" vengono aggiunti autonomamente da Solr (non avendo customizzato i file di configurazione) rispettivamente per rafforzare l'unicità di un documento e per avere uno storico di versioni in caso di aggiornamenti seguenti dell'indice.

N.B.: I dati utilizzati in questo capitolo sono solo dei dati parziali di tutti quelli che poi verranno utilizzati nel processo completo.

[2]https://db-engines.com/en/ranking/search+engine

Capitolo 3

Classificazione

In questo capitolo, il cuore di questo lavoro, verrà eseguito un lavoro di classificazione tramite algoritmi di Machine Learning.
Il capitolo sarà spezzato in due parti: nella prima verrà usato un dataset "dummy" con notizie accumulate dal web (in inglese) ed aventi "pseudocategorie" molto variegate (non è altresì garantito che tutte le notizie siano inerenti al topic del "Covid-19") mentre nella seconda verrà usato il dataset vero e proprio con le notizie sul Covid-19 categorizzate dal personale dell'ISS.
Verranno utilizzati 3 approcci "classici" **supervised** (Logistic Regression, Gradient Boosting e Multinomial Naive Bayes) e una rete neurale (Transformer-Based) per classificare e si proverà, inoltre, a usare il modello "migliore" trovato nella prima parte per, appunto, classificare i dati veri e propri in maniera appagante.

N.B.: Tutti i codici di questo capitolo sono reperibili all'indirizzo https://github.com/PaoloCiasco/TesiMagistrale

3.1 Fase Preliminare

Si fa d'uopo trasformare i dati già precedentemente analizzati nel corso del secondo capitolo in un file *.csv* in modo tale che il lavoro su di essi possa essere relativamente facile da svolgere e funzionale allo scopo.
Affinché questo sia possibile viene modificato il codice Python elaborato precedentemente (*xml_solr_like.py*) per avere un file .xml in uscita più "agevole" per una successiva trasformazione.
Si rende noto che nei due differenti dataset è necessario avere script differenti di trasformazione e quindi verranno prodotti due differenti codici.
Il formato **".csv"** [13] gode di un notevole successo nel contesto di manipolazione dei dati (e quindi di riflesso nel machine learning) a causa della sua facilità di parsing dei dati; rispetto ad un file standardizzato "JSON", infatti, è molto meno flessibile e soprattutto non ammette tipi diversi di dati.
Affinché i valori vengano separati viene generalmente utilizzato il carattere "," e ogni linea

del file viene considerato un "record" dei dati ed avrà lo stesso numero, appunto, di valori. In quello che segue, si procederà ad illustrare i passaggi necessari per arrivare ad un file .csv congruo pronto per l'apprendimento per poi analizzarlo ed eventualmente manipolarlo nei due differenti dataset.
Prima verrà costruito ed analizzato il dataset definito "Dummy" con le pseudocategorie e poi verrà costruito ed analizzato il dataset definitivo.

3.1.1 Costruzione Dataset Dummy

I dati si presentano con il solito file .xml composto, stavolta, di un numero più rilevante di notizie: 1034.
La struttura del file è la stessa per quanto riguarda le sezioni analizzate nel corso del secondo capitolo ma presenta elementi nuovi dei quali, per lo scopo di questa parte, ci interessiamo solo alle categorie.

Figura 3.1. Una porzione di categorie di una news.

Da una prima analisi del numero di categorie delle notizie, si nota che sono di un numero variabile per ognuna di esse raggiungendo un picco di ben 129.
Si rende necessario eseguire una potatura secondo qualche metodica per avere un numero di categorie fisso per ogni notizia e soprattutto più controllabile.
Rispetto a quanto scoperto, le categorie portano in seno un attributo di nome "score" (si veda Fig.3.1) che simboleggia quanto la categoria sia inerente alla notizia secondo una qualche logica: si decide, pertanto, di selezionare le prime cinque categorie con lo score più alto. Il codice prodotto in Python (*"xml_csv_like_with_5top.py"*) estende quanto fatto nel secondo capitolo modificando, però, la struttura di uscita del file rendendolo pronto ad una successiva trasformazione in un file .csv invece che pronto ad essere indicizzato con Solr.
Tralasciando l'acquisizione delle informazioni già trattate precedentemente, per quanto riguarda le categorie viene costruito un array di dizionari: ogni dizionario racchiuderà tutte le categorie con lo score rispettivo di ogni documento. Il codice recupererà le informazioni sulle categorie andandole a inserire nell'array appena citato (nel dizionario

corrispondente al documento correntemente preso in considerazione), verranno ordinate in modo decrescente, verranno acquisite le prime cinque ed infine verranno inserite nel documento .xml di output.

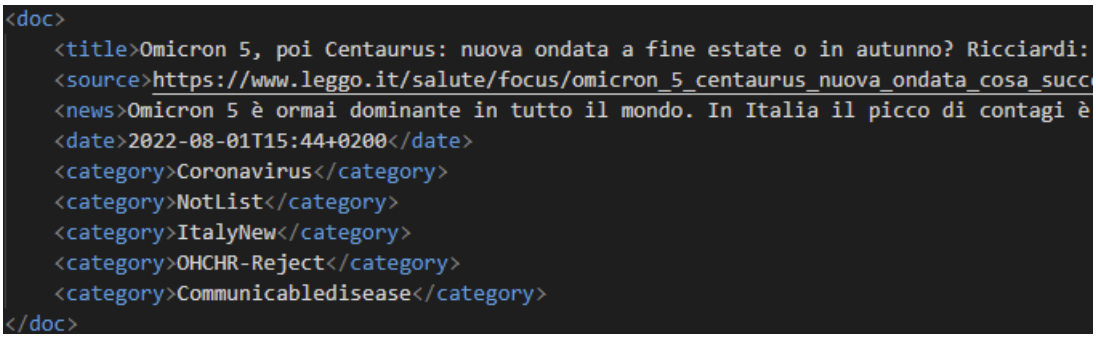


Figura 3.2. Top 5 categorie di una news post elaborazione.

È ora necessario trasformare il file .xml in un file .csv formattato a dovere pronto al task di classificazione.
Il codice (*"xml_to_csv.py"*) espleta questa necessità e viene costruito in modo tale che qualsiasi numero di categorie vengano sottoposte al compilatore esse verranno formattate correttamente (verrà riutilizzato per il dataset vero e proprio). Per far ciò vengono definite alcune funzioni per assicurarsi di recuperare il numero rispettivo alle categorie massime per un documento (in caso si volesse variare il numero delle top 5) e per scrivere il numero esatto delle stesse.
Per eseguire le operazioni di scrittura del file .csv viene sfruttata la libreria built-in di Python: **csv** [14].
Come separatore, invece della virgola usata nei testi delle notizie, si sceglie di usare 3 caratteri underscore per evitare qualsiasi confitto.
Viene quindi creato il **dataset** "*data_dummy.csv*".

N.B.: È stata eseguita un'operazione di "stripping" sui testi delle notizie affinché siano rimossi eventuali caratteri di "nuova linea" per procedere senza problemi alla trasformazione in un dataset che sia parsabile efficacemente.

3.1.2 Analisi Dataset Dummy

Sia l'analisi che la classificazione in sé verranno svolte tramite la creazione di un Jupyter Notebook Python "PseudoCategories_Classification.ipynb".
Come prima cosa viene letto il file prodotto nella precedente sezione tramite la libreria **pandas** [15] sfruttando un motore diverso di parsing da quello di default (python invece di C, a causa della presenza di un separatore multicarattere).

	title	**source**	**news**	**category**	**...**	**category.4**
0	Covid, Iss: registrato ...	https://tg24.sky.it/...	La sottovariante Omicron ...	NaN		None
1	Andreoni "Pochi min..	https://www.ilsuss...	Pochi minuti per ...	ItalyNew		Coronavirus
2	Omicron 5, poi Centa...	https://www.leg...	Omicron 5 è ormai ...	Coronavirus		Communica...
3	3.6m indirect ...	https://www.sunnew...	The Federal Government ...	Sahel-Test		GeneralE...
4	Merrifield says he'...	https://www.taiwa...	MINNEAPOLIS (AP) ...	NotList		ET_SPORTFR
...	...	...	...	...	...	...
1029	Puspusan ang ...	https://tribune.net.ph...	Dahil nalalapit na ...	Content-For-...		CAR_Name...
1030	NASAYANG NA ...	https://tribune.net.ph...	Nananawagan ...	NotList		Coronavirus
1031	MIL-OSI USA: ..	https://foreignaffairs...	Source: United States ...	OHCHR-Reject		Attacker_in...
1032	MIL-OSI USA: ...	https://foreignaffairs...	Source: United States ...	Legislation		dogodki_alert

Tabella 3.1. Visione tabulare di una porzione del dataset dummy.

Il dataset consta di 1034 righe (le notizie che avevamo nel file raw) e 8 colonne (titolo, fonte, notizia e le 5 categorie).
In principio si rende necessaria un'analisi di tipo linguistica per capire con che tipo di notizie stiamo lavorando e quindi di conseguenza come comportarci.
Grazie alla libreria **langdetect** [16] si esegue un conteggio delle notizie in 2 differenti lingue, italiano ed inglese, e si verifica l'esistenza di eventuali notizie in altri linguaggi.
Di seguito i risultati:

Total News	1034
Italian News	74
English News	936
Foreign News	24

Si richiede anche un'analisi per suggellare la presenza o meno di eventuali duplicati da rimuovere poi in fase preliminare di apprendimento: non sono presenti duplicati.

3.1.3 Costruzione Dataset Definitivo

I dati vengono acquisiti come tanti file .xml (149) contenenti notizie riguardanti il 2020, il 2021 e il 2022 fino ad inizio Settembre.
La prima operazione che si rende necessario fare è quella di eseguire il merge dei differenti file .xml in uno unico che poi possa essere elaborato propriamente. Per fare ciò viene creato il codice "*merging_xml.py*" che, appunto, fonde tutti i file in uno unico con la stessa radice.
In questo modo si ha la stessa struttura compatibile con il codice già scritto (con qualche modifica) sia per creare dei file pronti ad essere indicizzati con Solr che ad essere trasformati in .csv.
Una volta ottenuto il file con tutte le notizie accumulate viene utilizzato il programma in Python "*xml_csv_like.py*", un codice leggermente modificato dallo script per produrre file pronti per essere indicizzati.

```
<add>
    <doc>
        <title>Covid, tutta Italia in zona bianca da domani</title>
        <source>https://www.basilicata24.it/2022/03/covid-tutta-italia-in-zona-bianca
        <news>Da domani lunedì 28 marzo 2022 tutte le regioni saranno in zona bianca.
        <category>Zone Rosse</category>
    </doc>
    <doc>
        <title>AGGIORNAMENTO - Omicron 2 è prevalente nelle Marche</title>
        <source>https://www.ilrestodelcarlino.it/marche/omicron-2-1.7512209</source>
        <news>Nelle Marche la variante Covid Omicron 2 è oramai diventata dominante.
        <category>Coronavirus - VARIANTI</category>
    </doc>
```

Figura 3.3. Porzione di output del file xml modificato pronto per essere trasformato in csv.

Il file .xml monolite ottenuto viene poi trasformato in un file .csv tramite lo stesso codice usato per il dataset dummy (*"xml_to_csv.py*)". É doveroso specificare che vengono scritte sul file csv solamente le righe che hanno sia titolo, corpo della notizia, fonte e categoria. A differenza della precedente costruzione, non è qui necessario fare operazioni di "gathering" specifico sulle categorie dal momento che abbiamo una sola macrocategoria che specifica la notizia.
Al termine del processo viene formato il **dataset** "*Official_Data_20_22.csv*".
Per commenti più approfonditi sul procedimento della costruzione, si rimanda alla precedente sezione.

3.1.4 Analisi Dataset Definitivo

Anche in questo caso, sia l'analisi che la classificazione in sé verranno svolte tramite la creazione di un Jupyter Notebook Python "*Categories_Classification.ipynb*".
Le specifiche di caricamento del file sono le stesse usate per il precedente dataset.

	title	**source**	**news**	**category**
20	Varianti in crescita (c...	https://www.ecodiber...	Primi due casi di ...	Coronavirus - VAR...
21	Coronavirus, primo c...	https://milano.repubb...	È arrivata in Italia anche ...	Coronavirus - VAR...
22	AGGIORNAMENTO: ...	https://www.ilmessa...	E sono risultati tutti nega...	SARS-CoV-2
23	EVENTO: Coronaviru...	https://www.ansa.it/...	Primi contagiati in Ital...	SARS-CoV-2
24	AGGIORNAMENTO-...	http://www.meteowe...	Sono saliti a 163 i pazi...	Malattie a trasmi...
25	Varianti Covid Italia ...	https://tg24.sky.it/...	Proseguono le segnala...	Coronavirus - VAR...
26	Covid, metà caso area ...	https://www.ansa.it/...	La metà dei casi di coro...	Coronavirus - VAR...
27	Focolaio Covid ad Arce...	https://www.fanpage...	"Cari concittadini, siamo...	Coronavirus - VAR...
28	Variante inglese, centin...	https://milano.corrie...	Si spengono le luci e p...	Coronavirus - VAR...

Tabella 3.2. Visione tabulare di una porzione del dataset definitivo.

Il dataset consta di 3924 righe (le notizie che avevamo nel file raw complete) e 4 colonne (titolo, fonte, notizie e categoria).
Viene resa necessaria, come nel caso del dataset dummy, un'analisi linguistica per vedere eventuali notizie di lingua diversa dall'italiano (ricordiamo che in questo caso avremo solamente notizie italiane) con la stessa libreria usata nella precedente sezione.
Di seguito i risultati:

Total News	3924
Italian News	3923
English News	1
Foreign News	0

Dal momento che nel dataset risiede una unica notizia in lingua inglese, essa verrà eliminata in fase di pre-processing dei dati.
Viene, inoltre, resa necessaria un'analisi di eventuali **duplicati** (vista la modalità di ricezione) che dovranno poi essere rimossi in fase preliminare di apprendimento.
Le notizie duplicate risultano essere 498.

3.2 Apprendimento Dataset Dummy

In questa sezione, dopo aver eseguito delle operazioni preliminari sui dati propedeutiche alla classificazione di testo, si procederà alla definizione dei modelli e al loro apprendimento valutandone le differenti performance.

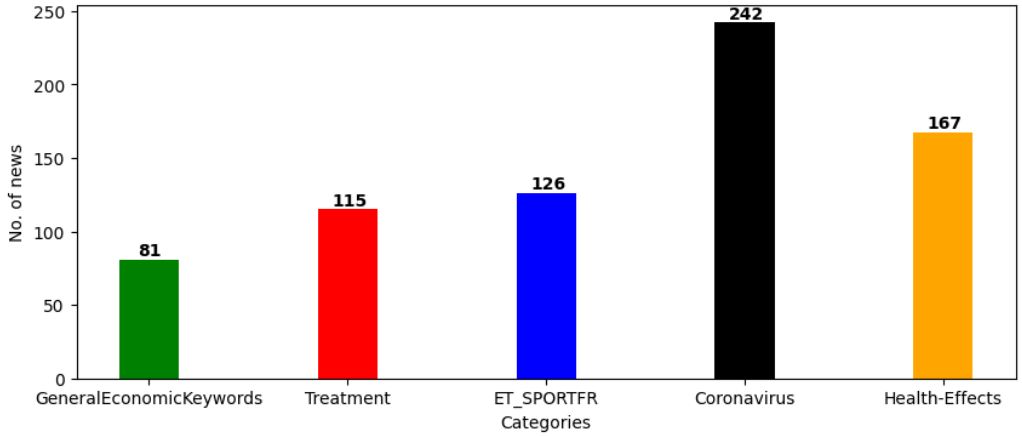
3.2.1 Operazioni Preliminari Sui Dati

Dopo aver importato le librerie necessarie alle operazioni e caricato il dataset creato nelle precedenti sezioni tramite *"pandas"*, si procede a ripulire il dataset dalle notizie che non sono in inglese dal momento che lo stesso è costituito dal 90% da notizie di lingua di stampo anglosassone.
Dopo la ripulitura risultano 936 notizie rimanenti.
Dal momento che ci sono parecchie categorie si decide di scegliere le cinque più popolari e significative rispetto tutte le news. Per fare questo si vanno a contare, per ogni colonna del dataframe contenente le categorie, le occorrenze di quest'ultime; dopo aver scartato quelle risultate insignificanti a livello semantico come 'OHCHR-Reject' e 'NotList', che sebbene fossero fra quelle più presenti non attribuivano, appunto, alcun significato, queste sono le categorie scelte:

- **Coronavirus**
- **Health-Effects**
- **GeneralEconomicKeywords**
- **ET_SPORTFR**
- **Treatment**

Fra le più popolari sono state scelte, in particolare, categorie per lo più attinenti direttamente ad un discorso generale sul Coronavirus, alle terapie riguardo ad esso e agli effetti collaterali sulla salute.
Si è ritenuto utile e necessario inserire anche due macroaree come sport ed economia.
È bene specificare che, sebbene siano state raccolte le categorie con lo score più alto rispetto la ricerca (assumendo che la misura fosse affidabile a priori dal sistema che ha generato il file .xml originale), non è garantita l'accuratezza semantica a livelli massimi di precisione; può capitare, infatti, una assegnazione non pienamente coerente con la notizia.
Si ricorda, inoltre, che i dati non sono annotati manualmente ma sono tag speciali attribuiti da una ricerca dal noto motore di ricerca per notizie "Google News".

In generale è possibile notare una maggioranza di categorie rispetto il coronavirus e alle sfaccettature sportive (ET_SPORTFR) ed economiche (GeneralEconomicKeywords).
Di seguito la distribuzione delle categorie rappresentata tramite un grafico a barre:



Si osserva, in ogni caso, che il numero di categorie distribuite nelle diverse notizie è abbastanza sbilanciato, considerando le poche notizie disponibili.
Dopo aver eliminato le notizie senza categoria attribuita il numero di news rimanenti è 731; questa sarà la grandezza definitiva del nostro dataset.

Inizia ora la fase (conclusiva di questa sezione) di **ripulitura del testo** [17] per passare poi all'apprendimento vero e proprio. Dopo aver rinominato l'etichetta della colonna "cate_def" (contentente le categorie definitive per ogni news) in "category" si è proceduto a trasformare ogni categoria in numeri univoci tramite la funzione "factorize" della libreria "pandas" e a salvarli in una colonna specifica chiamata "category_id".
Le prime due azioni che vengono fatte riguardano l'eliminazione di **caratteri speciali** dal testo e l'eliminazione di **virgolette** tramite la definizione di due funzioni(ved. Fig.3.7).

```
def remove_tags(text):
    remove = re.compile(r'')
    return re.sub(remove, '', str(text))

df['news'] = df['news'].apply(remove_tags)

def special_char(text):
    reviews = ''
    for x in text:
        if x.isalnum():
            reviews = reviews + x
        else:
            reviews = reviews + ' '
    return reviews

df['news'] = df['news'].apply(special_char)
```

Figura 3.4. Funzioni per eliminare caratteri speciali e virgolette.

Dopo aver convertito tutto il testo in caratteri minuscoli (una pratica ben comune in ambito NLP che rende il flow di parsing più consistente) si procede a rimuovere le cosidette **stopwords**.
Le stopwords sono quelle parole (articoli, preposizioni, pronomi, ecc.) che risultano più comuni in ogni linguaggio e pertanto non aggiungono informazione rilevante al testo.
Effettuando un'operazione di rimozione di quest'ultime, il risultato è una riduzione sensibile di informazione di basso livello e quindi non vi sarà particolare impatto sul modello che andremo ad addestrare.
In ogni caso non è sempre indicato rimuovere le stopwords con leggerezza, si consideri il seguente esempio in inglese:

"The meal was not good at all."

Se si volesse eseguire un task di **sentiment analysis**, ovvero di cercare di carpire se la frase sia positiva o negativa (nell'accezione più semplice dello stesso), e si rimuovessero le stopwords il risultato sarebbe:

"meal good."

È lapalissiano dire che un testo che indubbiamente esprimeva una recensione negativa prima del preprocessing, ora ne esprime una positiva.
Per il task in esame in questa tesi, che ricordiamo essere di classificazione testuale, l'attività che stiamo trattando è più che propedeutica ad una buona riuscita complessiva.
Per effettuare questa pratica ci sono due librerie principali che offrono questa possibilità in modo facile e veloce: **NTLK** [18] e **spaCy** [19].
In questa trattazione verrà usata la prima per motivi di esperienza pregressa.

```
def remove_stopwords(text):
  stop_words = set(stopwords.words('english'))
  words = word_tokenize(text)
  return [x for x in words if x not in stop_words]

df['news'] = df['news'].apply(remove_stopwords)
```

Figura 3.5. Funzione per eliminare stopwords ed eseguire la 'tokenizzazione' del testo.

Dopo aver rimosso le stopwords si procede alla tokenizzazione del testo, ovvero separare tutto il testo in entrata in **"token"** che verranno poi processati dal nostro modello.
L'operazione viene effettuata sempre grazie alla libreria appena usata.

L'ultimo processo di questa catena di preprocessing riguarda l'utilizzo di una tecnica fra **"Stemming"** e **"Lemmalization"**.

Stemming vs Lemmalization [20]
Sebbene operino in modo differente, l'obiettivo comune di queste due tecniche è trovare le forme base di termini e verbi. Ad esempio:

$$am,\ are,\ is \rightarrow be$$
$$car,\ cars,\ car's,\ cars' \rightarrow car$$

In generale, una stemmizzazione è un processo un po' rude che tende a elidere la parte finale delle parole nella speranza che il senso non venga alterato mentre la lemmalizzazione si serve principalmente di vocabolari, analisi morfologiche delle parole e di reti neurali (più recentemente) affinché si riesca a trovare, per l'appunto, il *lemma* di una parola.
Un esempio classico per apprezzare al meglio la differenza fra le due tecniche è il seguente:

Stemming: $saw \rightarrow sa$
Lemmaliz.: $saw \rightarrow see\ or\ saw\ (dipende\ dal\ contesto)$

Per sommi capi, a causa del loro funzionamento, per una lingua di stampo anglosassone (a causa di pochi elementi irregolari rispetto alle altre lingue) è preferibile utilizzare uno Stemmer mentre per una lingua di provenienza latina (come l'Italiano) è preferibile seguire la strada della lemmalizzazione.
Lo Stemmer più famoso è lo Stemmer di Porter.
La scelta di utilizzare la stemmizzazione si rende necessaria nel caso di grandi dataset e di poche risorse computazionali a propria disposizione poiché la controparte necessita di più tempo e, appunto, risorse.

Per la realizzazione di questo task su questo dataset, decisamente non di dimensioni grandi, si decide di usare l'approccio della lemmalizzazione grazie alla libreria NLTK.

```
def lemmatize_word(text):
    wordnet = WordNetLemmatizer()
    return " ".join([wordnet.lemmatize(word) for word in text])

df['news'] = df['news'].apply(lemmatize_word)
```

Figura 3.6. Funzione per lemmalizzare i token ottenuti dal testo pre-processato.

Il processo preliminare si conclude qui; si lascia al lettore qualche esempio pre/post catena di preprocessing.

14esima notizia rispetto il dataset ottenuto:

pre → *GREENBELT, Md. – A West Virginia man was sentenced Thursday to three years in federal prison after he sent emails threatening Dr. Anthony Fauci and another federal health official for talking about the coronavirus and efforts to prevent its spread. Using an anonymous email account based in Switzerland, Thomas Patrick Connally, Jr.*

post → *greenbelt md west virginia man sentenced thursday three year federal prison sent email threatening dr anthony fauci another federal health official talking coronavirus effort prevent spread using anonymous email account based switzerland thomas patrick connally jr*

23esima notizia rispetto il dataset ottenuto:

pre → *FILE - Dr. Anthony Fauci, director of the National Institute of Allergy and Infectious Diseases, testifies to a House Committee on Appropriations subcommittee on Labor, Health and Human Services, Education, and Related Agencies hearing, about the budget request for the National Institutes of Health,....*

post → *file dr anthony fauci director national institute allergy infectious disease testifies house committee appropriation subcommittee labor health human service education related agency hearing budget request national institute health*

233esima notizia rispetto il dataset ottenuto:

pre → *The epidemic of COVID-19 has slowed the demand for car smart displays. Reduced car sales are expected to reduce demand for vehicles as well as passenger safety measures. The testing of semi-autonomous driving systems, as well as enhanced safety features, has been postponed due to nationwide....*

post → *epidemic covid 19 slowed demand car smart display reduced car sale expected reduce demand vehicle well passenger safety measure testing semi autonomous driving system well enhanced safety feature postponed due nationwide*

3.2.2 Tuning, Training e Test Set

Dal momento che il dataset a disposizione consta di 731 notizie, un numero abbastanza limitato, si decide di appplicare una tecnica di **k-Fold Cross-Validation** [21] per effettuare tuning di iperparametri.
In generale, quando abbiamo a che fare con dati non eccessivamente popolati, questa tecnica permette di avere un risultato meno "biased" rispetto ad un canonico split in train e testset. Una scelta importante riguarda il numero di fold che si vogliono usare, ovvero in quanti parti dividere il dataset. In questo senso si hanno in generale tre macro possibilità:

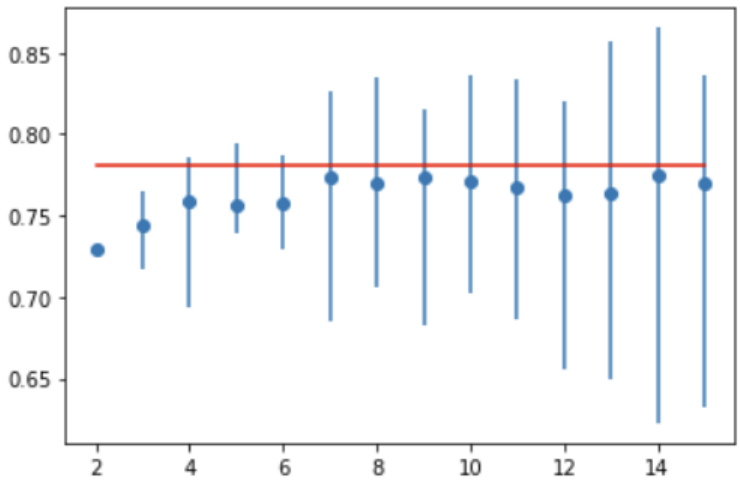
- **Rappresentativa:** Il valore di k è scelto in modo tale che ogni gruppo train/test è grande abbastanza da essere statisticamente rappresentativo riguardo il dataset.
- **k = 5, k = 10:** Tramite euristiche, questi due valori sono stati definiti come "golden values" per l'applicazione della tecnica.
- **k = n:** Caso speciale dove il numero di folds è uguale al numero di elementi nel dataset. Anche noto come "leave-one-out cross-validation".

È propedeutico citare, a questo punto, un passaggio dal libro " An Introduction to Statistical Learning": "*To summarize, there is a bias-variance trade-off associated with the choice of k in k-fold cross-validation. Typically, given these considerations, one performs k-fold cross-validation using k = 5 or k = 10, as these values have been shown empirically to yield test error rate estimates that suffer neither from excessively high bias nor from very high variance.*" (Gareth James et al. , 2017)

In quello che segue, però, verrà cercato il "migliore" valore di k possibile per il dataset in esame; senza passare per delle euristiche già definite a priori.
Un possibile approccio è quello di valutare le performance di uno stesso modello sullo stesso dataset con valori differenti di k e fare una comparazione sui risultati ottenuti. Quello che ci aspettiamo è sicuramente una stima "noisy" per valori di k piccoli che migliora mano a mano che k aumenta; ma "noisy" rispetto a cosa?
Quello che serve è una misura di riferimento il più affidabile possibile: come ottenerla?
La soluzione migliore sarebbe senza dubbio quella di effetturare il training del modello su tutti i possibili dati disponibili e testare la performance su un altro dataset separato, di circa la stessa dimensione, che sia rappresentativo dell'altro.
Questa soluzione, purtroppo, non è quasi mai perseguibile e quindi è possibile simulare questo scenario tramite la **"leave-one-out cross-validation"(LOOCV)** introdotta poche righe fa.
Dal momento che il dataset non è di dimensioni elevate (la computazione è altamente costosa) è una strada perseguibile e rappresenta una buonissima misura di riferimento.
In breve, verrà calcolata la misura "ideale" tramite LOOCV e poi verrà testato un modello, di solito usato per costruire una "baseline", su un numero di folds che va da 2 a 15. Tramite un grafico si sarà aiutati a scegliere il k "migliore".

Il modello di apprendimento usato è la **Logistic Regression**.
Di seguito risultati testuali e grafici:

```
Ideal: 0.781
> folds=2, accuracy=0.729 (0.727,0.732) > folds=10, accuracy=0.772 (0.703,0.836)
> folds=3, accuracy=0.744 (0.717,0.765) > folds=11, accuracy=0.768 (0.687,0.833)
> folds=4, accuracy=0.759 (0.694,0.786) > folds=12, accuracy=0.762 (0.656,0.820)
> folds=5, accuracy=0.757 (0.740,0.795) > folds=13, accuracy=0.764 (0.649,0.857)
> folds=6, accuracy=0.758 (0.730,0.787) > folds=14, accuracy=0.774 (0.623,0.865)
> folds=7, accuracy=0.773 (0.686,0.827) > folds=15, accuracy=0.770 (0.633,0.837)
> folds=8, accuracy=0.770 (0.707,0.835)
> folds=9, accuracy=0.773 (0.683,0.815)
```



I **"k"** più vicini al valore "ideale" di riferimento risultano essere (tutti allo stesso modo) : 7, 9 e 14.
Si decide di utilizzare $\mathbf{k = 7}$.

Per quanto riguarda lo split fra train e test set si è usata la proporzione 72/28 e seme 9.

3.2.3 Metriche di Valutazione

Prima di entrare nel vivo dell'apprendimento si trattano in questa parte operazioni preliminari necessarie ed utili allo stesso.
La tecnica per valutare modelli su di un task multiclasse come quello in esame che verrà usata è quella detta "**One vs Rest**" [22]: ogni classe viene comparata rispetto a tutte le altre nello stesso tempo. In questo scenario, si prende una classe e viene considerata "positiva" mentre tutte le altre sono considerate "negative" e questo processo si ripete per tutte (si veda Fig.3.9).
In questo modo ci riduciamo ad una classificazione di tipo binario dove è possibile utilizzare tutte le metriche classiche di valutazione.

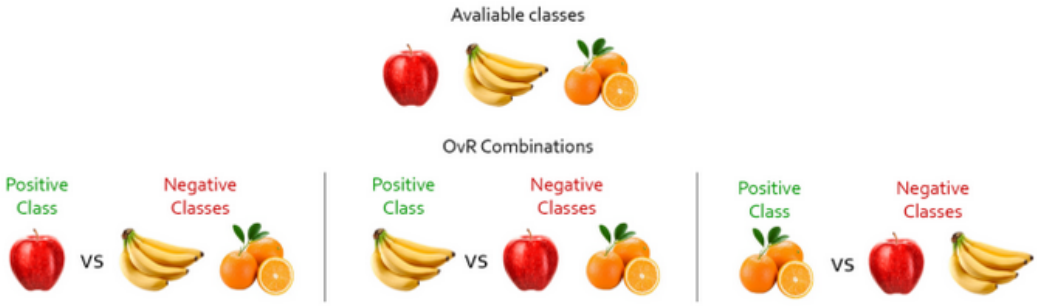


Figura 3.7. Rappresentazione di una classificazione OvR. [23]

Proprio rispetto le metriche di valutazione, si è scelto di utilizzare la **matrice di confusione** e valori di **precision**, **recall**, **f1-score** per ogni classe e globali pesati secondo la micro averaging.
Queste, infatti, sono metriche standard utilizzate in task di questa tipologia.

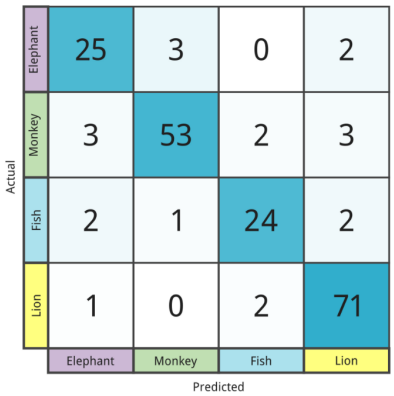


Figura 3.8. Esempio di una matrice di confusione. [24]

3.2.4 Baseline (Logistic Regression)

Come Baseline si sceglie di partire da una classica **Regressione Logistica**.
Di seguito i risultati:

Confusion Matrix

GeneralEconomicKeywords	17	0	0	0	1
Treatment	8	20	0	0	0
ET_SPORTFR	14	0	23	1	0
Coronavirus	22	2	0	54	2
Health-Effects	15	0	0	4	22

Classification Report

Categories	**Precision**	**Recall**	**F1-Score**	**Support**
GeneralEconomicKeywords	1.00	0.44	0.62	18
Treatment	0.91	0.71	0.80	28
ET_SPORTFR	1.00	0.61	0.75	38
Coronavirus	0.89	0.68	0.77	80
Health-Effects	0.88	0.54	0.67	41
Global Micro Avg.	0.91	0.62	0.74	205

Per comodità di lettura si riportano solo i valori globali del modello con l'accuracy:

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Support**
Logistic Regression	0.61	0.91	0.62	0.74	205

3.2.5 Gradient Boosting Text Classification

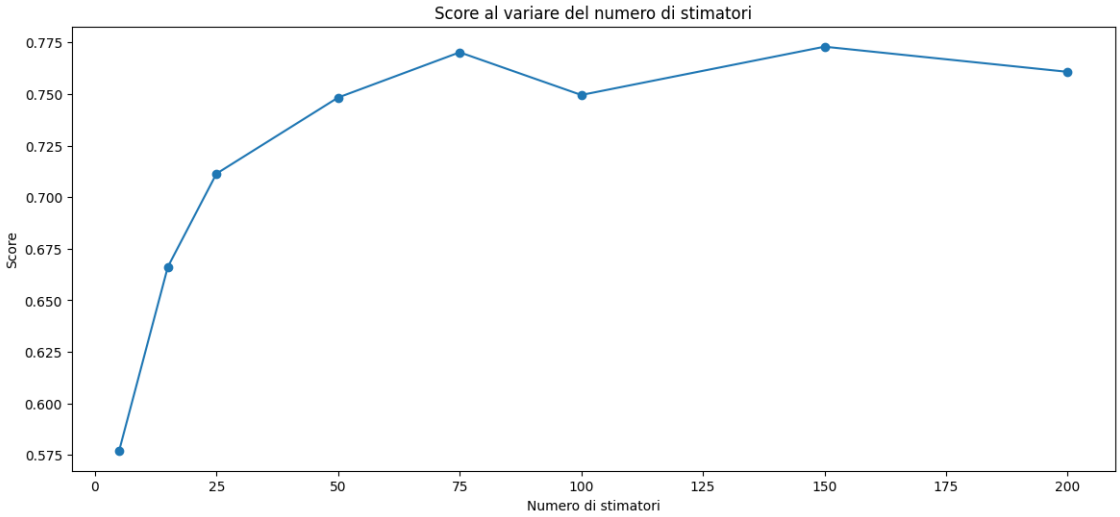
In questa sezione verrà utilizzato un metodo di boosting facente parte della famiglia dei metodi chiamati ensemble: **Gradient Boosting** [21]. I metodi **ensemble** sono chiamati così perché l'addestramento viene eseguito su un insieme di modelli e la predizione viene fatta unendo in qualche modo le predizioni di tutti i modelli utilizzati; è doveroso sottolineare come in questo modo si abbia un'elevanta varianza risultante.
Il Gradient Boosting, molto simile ad Adaboost, è definito come un metodo additivo; partendo dal dataset originale si addestra un primo modello $y^1(x)$, secondo una qualche loss function, e si costruisce un "nuovo" dataset con gli stessi elementi dell'originale ma con i target individuati dal residuo, ovvero dalla differenza fra t_i (target originale) e y_i^1. Viene preso, poi, un nuovo modello $y^2(x)$ e viene eseguito il training sul dataset ottenuto appena spiegato; la predizione sarà la somma delle due precedenti predizioni e se non sono contento con il risultato ottenuto vado avanti nella creazione di modelli. In un certo senso, il ruolo di $y^2(x)$ è di compensatore all'errore di $y^1(x)$.

La prima operazione che andiamo ad eseguire sul modello è il "**tuning**" di 3 iperparametri: *n_estimator*, *learning_rate* e *max_depth*.
Il **learning rate** denota semplicemente quanto il modello apprende velocemente; ogni albero(stimatore) aggiunto modifica il risultato del modello generale, il magnitudo di questa modifica è controllato dal learning rate. Il vantaggio di avere un rate più basso è che il modello sarà più robusto ed efficiente.
Il **numero di stimatori** rappresenta quanti weak learner (modelli) vengono usati nel processo; è buona norma prestare parecchia attenzione a non usarne troppi poiché si può apprezzare un rischio sensibile di andare in overfitting.
Infine la **max depth** rappresenta quanto possono essere profondi gli alberi; più sono profondi più raccolgono informazioni direttamenti sui dati.

Il primo parametro di cui faremo un tuning sarà il numero di stimatori. A differenza degli altri due parametri, dove ci affideremo ad una libreria, mostreremo un grafico per esprimere la variazione dell'accuracy in base a 8 interi che esprimono, appunto, il numero di stimatori.
Di seguito il procedimento:

```
scores_boost = []

r = [5, 15, 25, 50, 75,  100, 150, 200]
for k in r:
    score_boost = cross_val_score(estimator=GradientBoostingClassifier(n_estimators=k),
    X = x, y= y, cv=folds, scoring='accuracy')
    scores_boost.append(score_boost.mean())
```

Questo il grafico:



Viene deciso di utilizzare 75 stimatori.

N.B.: A causa della richiesta computazionale decisamente esosa non si è voluto salire oltre i 200 stimatori.

Come anticipato per i due altri iperparametri si è deciso di utilizzare la libreria **"GridSearchCV"** [25], che esegue una ricerca esaustiva degli score migliori in base alla prova di tutti i setting impostati. Di seguito il procedimento:

```
gbc = GradientBoostingClassifier(n_estimators=75)
parameters = {
    "max_depth":[3,5,7],
    "learning_rate":[0.01,0.1,1]
}
```

0.3s

```
from sklearn.model_selection import GridSearchCV
cv = GridSearchCV(gbc,parameters,cv=folds)
cv.fit(x,y)
```

Al termine del procedimento i valori migliori risultano:

- **learning_rate =** 0.1
- **max_depth =** 3

Finalmente viene fatto addestrare il modello sui dati.
Questi i risultati :

Confusion Matrix

GeneralEconomicKeywords	16	0	0	2	0
Treatment	5	22	0	1	0
ET_SPORTFR	18	0	20	0	0
Coronavirus	23	2	1	52	2
Health-Effects	22	1	0	1	17

Classification Report

Categories	**Precision**	**Recall**	**F1-Score**	**Support**
GeneralEconomicKeywords	0.78	0.39	0.52	18
Treatment	0.88	0.79	0.83	28
ET_SPORTFR	0.95	0.53	0.68	38
Coronavirus	0.89	0.68	0.77	80
Health-Effects	0.89	0.41	0.57	41
Global Micro Avg.	0.88	0.56	0.67	205

Per comodità di lettura si riportano solo i valori globali del modello con l'accuracy:

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Support**
Gradient Boosting	0.57	0.88	0.56	0.67	205

3.2.6 Multinomial Bayesian Text Classification

In questa sezione verrà utilizzato un classificatore Bayesiano come classificatore.
Un classificatore Bayesiano multinomiale [21] è ampiamente utilizzato per assegnare documenti a delle classi basandosi su analisi di tipo statistico; in generale è considerato una discreta alternativa alle ben più complesse reti neurali perché riesce a fornire "buoni" risultati in pochissimo tempo.
Sommariamente la classificazione avviene determinando la probabilità che un documento appartenga ad una classe popolata da altri documenti avendo lo stesso "soggetto" tramite il famoso **Teorema di Bayes**.

A differenza del precedente modello, non vi sono molti iperparametri su cui fare tuning se non la variabile di smoothing che va ad elidere un problema che ora verrà descritto.
Il modello che viene trattato utilizza un'assunzione probabilistica in cui ogni termine di un documento (feature) è indipendente dagli altri. Questa è un'assunzione molto stringente che non riscontra la verità assoluta perché, come ben si sa, i termini non sono indipendenti fra loro in uno scritto o in un discorso; in ogni caso, nel calcolo della probabilità, è così possibile eseguire la moltiplicazione delle, appunto, probabilità di tutti i termini del dizionario. In questo modo, però, può succedere che uno specifico documento non abbia un termine della collezione e quindi il documento in toto raccoglierebbe probabilità uguale a 0, ed è proprio in questo che interviene lo smoothing andando ad evitare questo problema.
Un buon parametro di smoothing non altera le probabilità e fornisce una buona risoluzione al problema sopra descritto.

Come prima, la ricerca dell'iperparametro migliore viene effettuata tramite la libreria **"GridSearchCV"**. Di seguito il procedimento:

```
np.logspace(0,-9, num=10)
from sklearn.model_selection import RepeatedStratifiedKFold

cv_method = RepeatedStratifiedKFold(n_splits=7,
                                    n_repeats=3)

params_NB = {'alpha': np.logspace(0,-9, num=100)}

gs_NB = GridSearchCV(estimator=MultinomialNB(),
                     param_grid=params_NB,
                     cv=cv_method,
                     verbose=1,
                     scoring='accuracy')


gs_NB.fit(x,y)
```

Al termine del procedimento il valore migliore risulta:

- **alpha =** 0.004328761281083057

Ora viene fatto addestrare il modello sui dati.
Questi i risultati :

Confusion Matrix

GeneralEconomicKeywords	17	0	0	0	1
Treatment	2	24	0	2	0
ET_SPORTFR	9	1	26	0	2
Coronavirus	16	6	2	53	3
Health-Effects	9	0	1	6	25

Classification Report

Categories	**Precision**	**Recall**	**F1-Score**	**Support**
GeneralEconomicKeywords	0.44	0.78	0.56	18
Treatment	0.75	0.86	0.80	28
ET_SPORTFR	0.81	0.68	0.74	38
Coronavirus	0.81	0.69	0.74	80
Health-Effects	0.68	0.73	0.71	41
Global Micro Avg.	0.70	0.75	0.71	205

Per comodità di lettura si riportano solo i valori globali del modello con l'accuracy:

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Support**
Multinomial Naive Bayes	0.67	0.70	0.75	0.71	205

3.2.7 Neural Network Text Classification

In questa sezione verrà utilizzato un approccio diametralmente opposto rispetto a quelli usati fino ad ora ma ampiamente usato nel processo di linguaggio naturale oggigiorno.
Le reti neurali [21] ricoprono un importante ruolo in tanti contesti diversi ed hanno ottenuto dei risultati sensibilmente migliori (non in ogni istanza) rispetto ai modelli classici di apprendimento di Machine Learning.
Non è scopo di questo trattato fornire una esaustiva spiegazione dell'argomento in questione ma si ritiene utile fornire qualche dettame teorico.
Il nome "Rete Neurale" deriva da un'abitudine storica in quanto ogni strato di una rete può essere considerata come un insieme di neuroni (percettroni).
Come anticipato, una rete neurale dispone di più strati rispetto ai modelli classici considerati fino ad ora: in ogni layer (strato) vi sono delle **funzioni di attivazione** che intervengono fra l'uscita e l'entrata degli stessi.
In generale, in una rete neurale, possiamo definire 3 tipi di layer diversi: input, inner, output.
Dato un vettore x di d feature, $x = (x_1, ..., x_d)$, il primo layer di una rete neurale deriva un certo numero m_1 di attivazioni, $a_1^{(1)}, ..., a_{m_1}^{(1)}$, attraverso combinazioni lineari di $x_1, ..., x_d$.
Gli strati intermedi, anche detti hidden, prenderanno il vettore prodotto dallo strato precedente e vi applicheranno altre combinazioni lineari per produrne di altri da poi proporre allo strato successivo.
Infine lo strato finale è definito in base al task che si sta affrontando: se è regressione verrà effettuata una sola combinazione lineare verso un'unica attivazione su cui verrà usata la funzione identità (ho già il valore target), se è classificazione binaria verrà effettuata una sola combinazione lineare verso un'unica attivazione su cui verrà applicata una sigmoide ed in fine se è una multi-classificazione verranno effettuate k combinazioni lineari verso k attivazioni su cui uso una funzione softmax su ognuna ottenendo un vettore con la codifica "1-of-k".
Per apprendere tutti i coefficienti dei pesi di tutti gli strati viene usata la **backpropagation**.

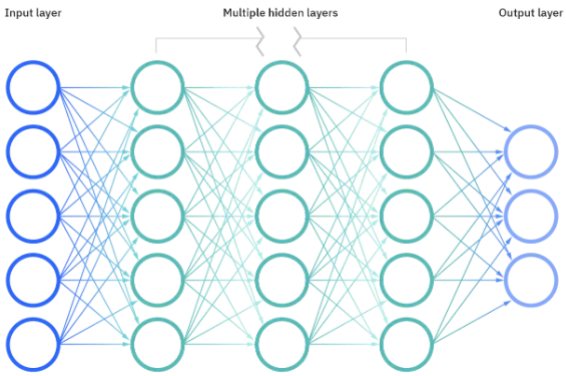


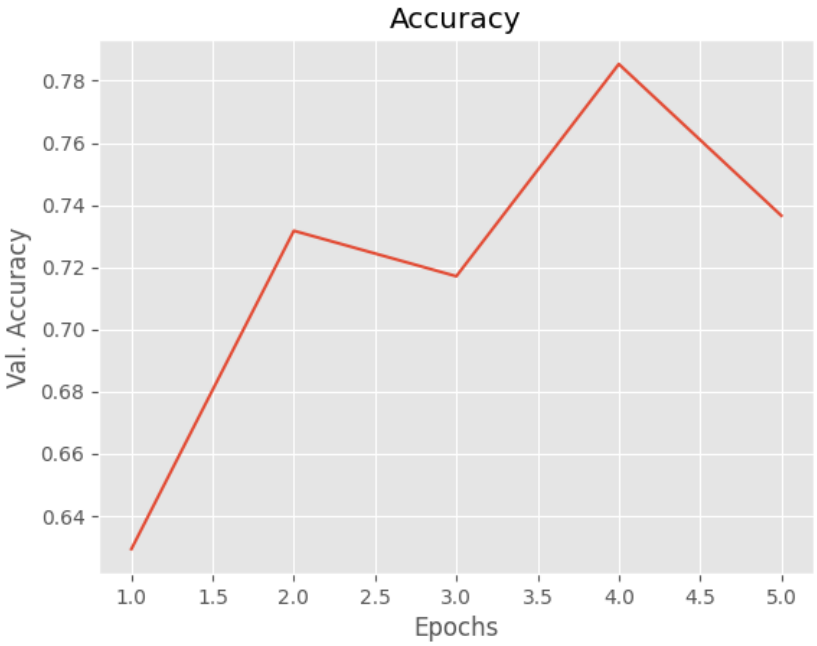
Figura 3.9. Astrazione di una rete neurale. [26]

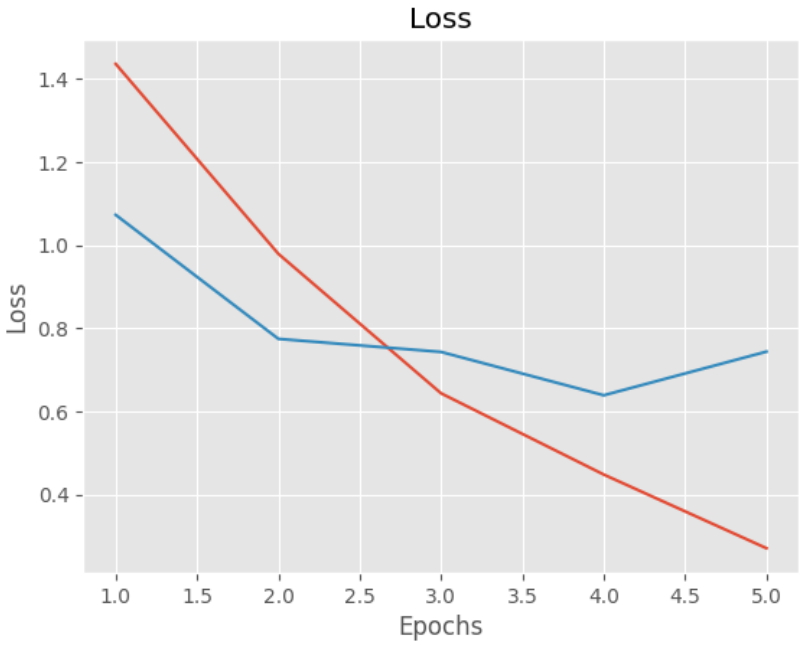
Per il nostro task, il modello che verrà usato è il **BERT Multilingual Base Model** [27]. Viene utilizzato un modello multilingua affinché possa essere riutilizzato, come già spiegato, su dati differenti in lingua italiana ed apprezzarne i risultati. Il procedimento seguito (con ovvie modifiche) è interamente tratto da una lezione di "Computational Linguistics" del 2021 [28] del Prof.Danilo Croce.
Il file di riferimento per questa parte è *"PseudoCategories_NeuralNetwork.ipynb"* ed il dataset su cui si è operato è stato creato tramite il pybook *"Exporting_Definitive.ipynb"*
Dopo aver splittato il dataset con la stessa proporzione e stesso seme usati per i modelli classici si è proceduto ad eseguire il training della rete neurale; si riportano i risultati dell'epoca che ha ottenuto i risultati migliori (cinque in totale):

```
======== Epoch 4 / 5 ========
Training...
  Batch    10  of     15.    Elapsed: 0:01:28.

  Average training loss: 0.448
  Training epoch took: 0:02:10

Running Test...
  Accuracy: 0.785
  Test Loss: 0.638
  Test took: 0:00:19

  Saving the model during epoch 3
  Actual Best Validation Accuracy: 0.785
```

Il risultato ottenuto potrebbe essere considerato deludente sotto alcuni punti di vista ma si sciorinano ora delle motivazioni a favore di esso.
Le reti neurali, per ottenere dei buoni risultati, necessitano di ampie quantità di dati mentre nel task attuale sono state fornite solamente 731 news.
Le categorie risultanti come "top5" sono state ottenute da un processo automatico che si è basato su un certo score su cui non sono state poste in atto (non era possibile) verifiche sensibili, può essere sicuramente quindi successo che l'etichettamento a priori non sia stato dei migliori.
Si sta classificando su 5 categorie, quindi un modello che operi in maniera casuale avrebbe solamente il 20% di scegliere in maniera corretta.

Di seguito l'andamento di **Accuracy** e **Loss** nelle varie epoche:





Di seguito i risultati totali:

Confusion Matrix

Coronavirus	69	5	2	2	2
ET_SPORTFR	4	32	0	2	0
GeneralEconomicKeywords	2	1	14	0	1
Health-Effects	10	4	2	23	2
Treatment	4	0	0	1	23

Classification Report

Categories	**Precision**	**Recall**	**F1-Score**	**Support**
Coronavirus	0.78	0.86	0.82	80
ET_SPORTFR	0.76	0.84	0.80	38
GeneralEconomicKeywords	0.79	0.79	0.79	18
Health-Effects	0.82	0.56	0.67	41
Treatment	0.82	0.82	0.82	28
Global Micro Avg.	0.79	0.77	0.78	205

Per comodità di lettura si riportano solo i valori globali del modello con l'accuracy:

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Support**
NN (BERT - Mutilingual)	0.79	0.79	0.77	0.78	205

3.2.8 Comparazione dei modelli

Ora che tutti i modelli hanno prodotto risultati fa d'uopo eseguire una comparazione degli stessi per selezionarne il migliore.
Di seguito una tabella generale di comparazione delle principali metriche globali:

Models	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Logistic Regression	0.61	0.91	0.62	0.74
Gradient Boosting	0.57	0.88	0.56	0.67
Multinomial Naive Bayes	0.67	0.70	0.75	0.71
NN (BERT - Multilingual)	0.79	0.79	0.77	0.78

È lapalissiano constatare che il modello migliore risulta essere la rete neurale; una valida alternativa, in ogni caso, risulta essere il Multinomial Naive Bayes.
La rete neurale si comporta egregiamente (si ricordino le considerazioni fatte sui dati) sotto ogni punto di vista fornendo una performance decisamente accettabile e spendibile.

I risultati sarebbero potuti essere migliori utilizzando tecniche di oversampling ma, ricordando l'obiettivo finale di questo elaborato, non rappresenta essere uno sforzo di risorse e tempo spendibile.

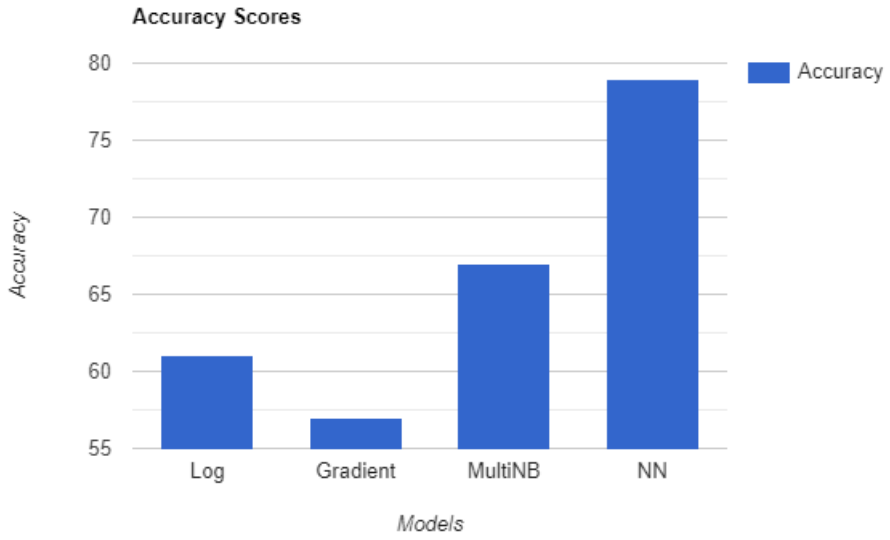


Figura 3.10. Grafico a barre rispetto l'accuracy dei modelli.

3.2.9 Test del modello migliore

Nella parte conclusiva di questa sezione si vuole dare un assaggio delle potenzialità del modello multilingua e verranno utilizzate alcune news, in lingua italiana, della sezione di "Indicizzazione" di questo elaborato.
Di seguito i risultati di qualche esempio:

"Il Covid torna a mandare in apprensione il calcio italiano, l'annuncio circa la positività al Coronavirus gela i tifosi. È un ex calciatore a rivelare gli ultimi rumors, sulle indiscrezioni legate al problema Covid in Nazionale. Il gruppo squadra sarebbe colpito da un vero e proprio focolaio."

Predizione: ET_SPORTFR

"Covid, focolaio nel gruppo sicurezza pubblica della Polizia Locale: più del 10% del personale è positivo (Di lunedì 28 marzo 2022) Continua a crescere il numero di persone positive al Covid all'interno del gruppo sicurezza pubblica Emergenziale del Corpo di Polizia Locale. Alle nove persone già in malttia a partire dalla giornata di ieri, se ne aggiungono oggi altre tre."

Predizione: Coronavirus

"Le società Atlante Eurobasket Roma e Givova Scafati annunciano il rinvio del match di campionato fra i due team, valido per il recupero del 15esimo turno della Serie A2 2021/22, previsto originariamente per questo mercoledi. Il rinvio è stato disposto dal Settore Agonistico della FIP a seguito dei numerosi casi di positività al Covid-19 all'interno del Gruppo Squadra della squadra campana."

Predizione: ET_SPORTFR

Si notano dei risultati, seppur su un numero ridottissimo di esempi, veramente interessanti.

3.3 Apprendimento Dataset Definitivo

In questa sezione, dopo aver eseguito delle operazioni preliminari sui dati (simili a quanto eseguito per l'altro dataset), si procederà a valutare il modello migliore della precedente sezione (la rete neurale) su questi nuovi dati e a riaddestrare tutti i modelli valutandone le nuove performance.
È doveroso specificare che molte operazioni sono state già fatte nella precedente sezione e non verranno descritte di nuovo.

N.B.: Le notizie sono state raccolte in uno spazio temporale che va dal Febbraio 2020 sino ai primi giorni di Settembre 2022.

3.3.1 Operazioni Preliminari Sui Dati

Come anticipato, la prima operazione che viene eseguita è la rimozione di tutte le notizie duplicate all'interno del dataset.
Dopo la pulizia di 498 duplicati, le notizie risultano essere 3426. Rispetto al precedente dataset, si procede questa volta ad eliminare l'unica notizia inglese presente per qualche errore nei dati in modo tale che risultino solo news di lingua italiane.
Dopo la ripulitura risultano 3425 notizie, che costituiranno le notizie definitive su cui verrà effettato apprendimento.
Viene ora iniziata un'analisi di distribuzione delle notizie nelle varie categorie per capire con quali e con quante di queste ultime si ha a che fare.
Le categorie risultano essere 16 e di seguito si fornisce una tabella riassuntiva:

Category	**No. of News**
Coronavirus - Focolai in altri contesti	952
Coronavirus - Focolai RSA/Case di riposo	687
Coronavirus - Focolai scolastico	578
Coronavirus - Focolai ospedalieri	423
Coronavirus - VARIANTI	311
Coronavirus - Focolai familiari/amici	259
Zone Rosse	57
Malattie a trasmissione interumana diretta	40
2019-nCoV	38
SARS-CoV-2	36
Coronavirus - Focolai familiari	19
Coronavirus - VARIANTE INGLESE	12
Malattie con altro tipo di trasmissione	5
Coronavirus - Focoali RSA/Case di riposo	4
Malattie a trasmissione ambientale	3
Malattie a trasmissione oro-fecale	1

Tabella 3.3. Categorie del dataset ufficiale

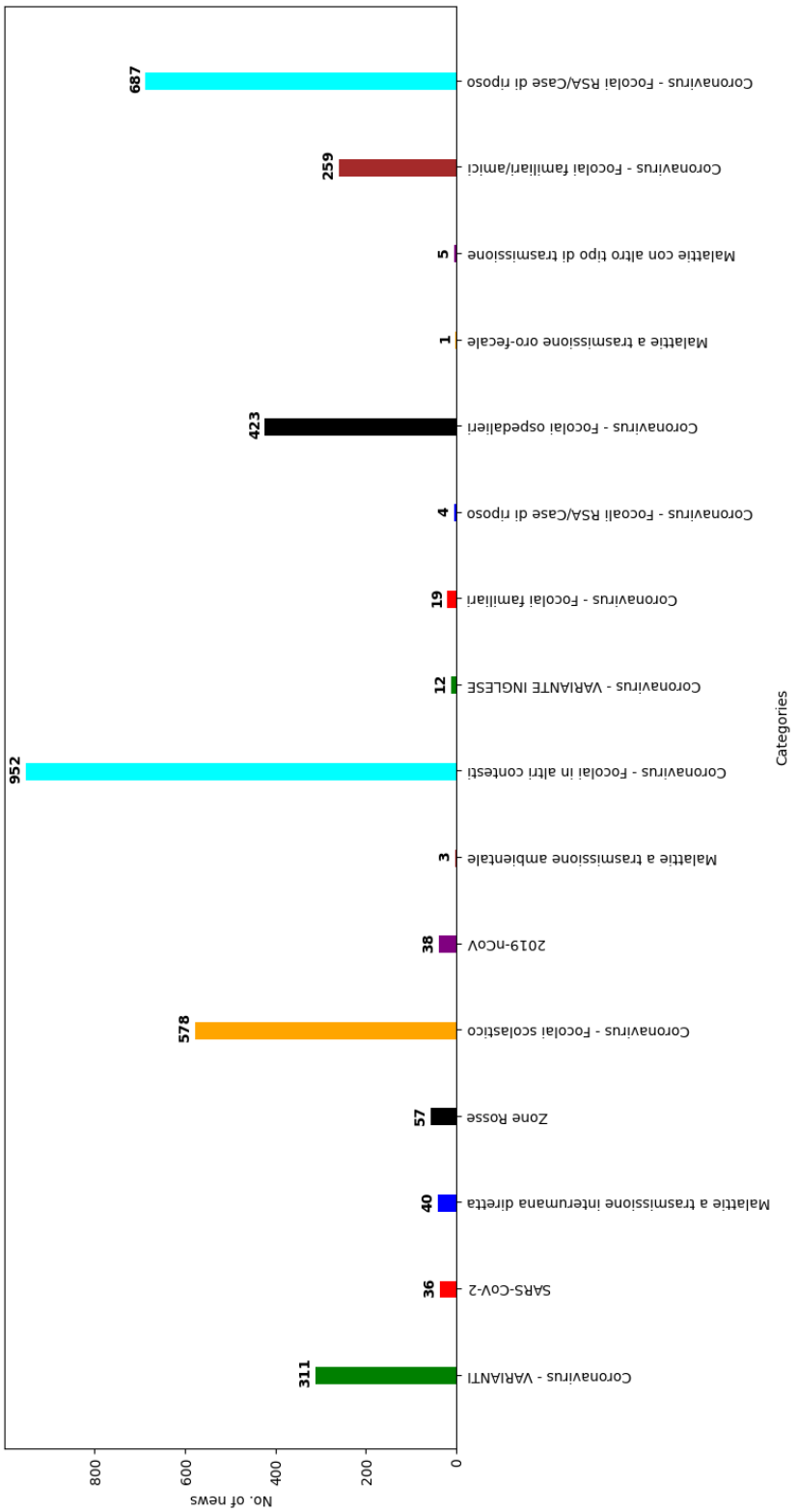


Figura 3.11. Grafico a barre delle categorie del dataset.

Indubbiamente, da una rapida occhiata alle categorie, è necessario operare alcune modifiche. La prima incongruenza che è possibile notare risiede in un errore di battitura: le notizie della categoria **Coronavirus - Focoali RSA/Case di riposo** sono sicuramente da istanziare nella categoria **Coronavirus - Focolai RSA/Case di riposo**.
Successivamente possiamo notare come esista la categoria **Coronavirus - Focolai familiari** e la categoria più generale **Coronavirus - Focolai familiari/amici**: anche qui le notizie della prima vengono indirizzate verso la seconda.
A questo punto è stato utile un incontro con gli esperti dell'ISS in data 13/09/2022 per capire in modo netto quali fossero le categorie che potessero cogliere il loro interesse in maggior modo.
Per quanto riguarda la categoria **Coronavirus - VARIANTE INGLESE** viene deciso che le notizie appartenenti ad essa vengano trasferite nella più generale **Coronavirus - VARIANTI**.
Le seguenti categorie (e quindi anche le relative notizie) vengono eliminate in quanto non più utili ai fini di accumulo di informazioni; si discuterà nel corso dell'ultima sezione di questo elaborato un eventuale utilizzo delle stesse.

- Zone Rosse
- Malattie a trasmissione interumana diretta
- 2019-nCoV
- SARS-CoV-2
- Malattie con altro tipo di trasmissione
- Malattie a trasmissione ambientale
- Malattie a trasmissione oro-fecale

Alla fine di questa elaborazione rimangono 3245 news distribuite su 6 categorie (si veda Fig.3.12).
La fase di ripulitura del testo e di tokenizzazione è la stessa del precedente dataset e si rimanda pertanto il lettore alla sezione 3.2.1.

Alla prossima pagina viene mostrato il grafico a barre rappresentante le notizie finali nelle categorie finali trovate.

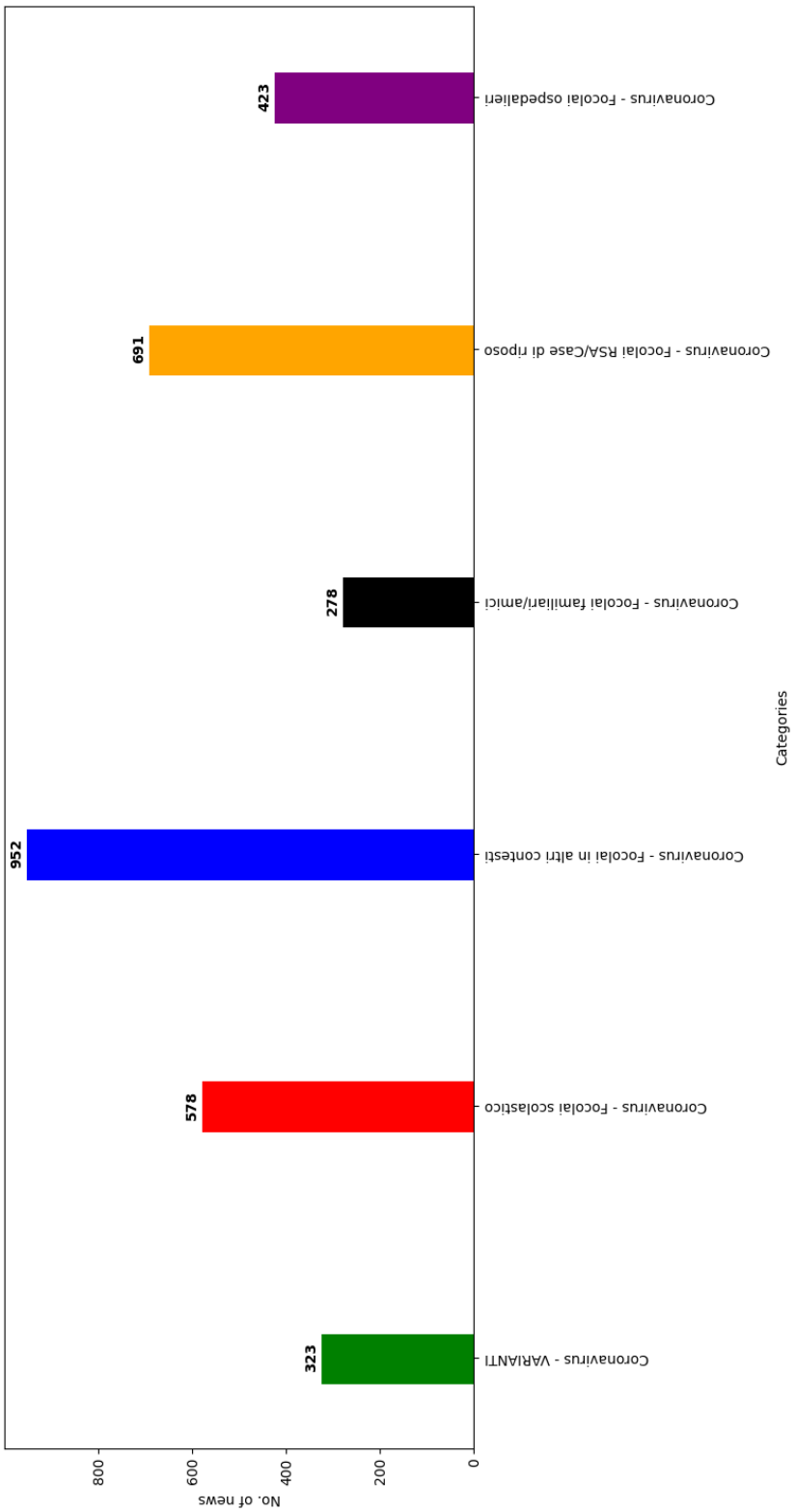


Figura 3.12. Grafico a barre delle categorie finali del dataset e relativo numero di news.

3.3.2 Valutazione approssimativa del modello migliore delle pseudocategorie

Prima di passare all'addestramento dei modelli su questi nuovi dati, come promesso, si vuole provare ad utilizzare il modello migliore trovato dai precedenti dati per categorizzare nelle precedenti categorie, in cerca di possibili risultati interessanti.
Si precisa che non verranno usate metriche di valutazione particolari ma si vuole fornire una visione d'insieme su un numero limitato di notizie per vederne il comportamento; in ogni caso si ricorda che le categorie dei precedenti dati sono abbastanza informative (Economiche, Sportive, ecc.) ma allo stesso tempo parecchio diverse rispetto a quelle dei dati in esame in questa sezione.
Le tre notizie che vengono scelte dal dataset in esame sono selezionate dall'autore di questo elaborato, in maniera che possano appartenere a tre categorie diverse, per poter cercare di apprezzare e valutare una classificazione positiva o negativa.

"La presenza di nuovi focolai covid-19 accertati all'interno dei gruppi squadra hanno costretto la Lega Pallavolo Serie A Femminile al rinvio di ben 6 partite su 7 in programma nella 1a giornata di ritorno del campionato di serie A1."

Categoria Originale: Coronavirus - Focolai in altri contesti
Predizione: ET_SPORTFR

"A Gussago 4 classi in isolamento per coronavirus „ Il virus non allenta la presa. Nuovi casi di contagio da Covid 19 nelle scuole di Gussago. A finire in isolamento fiduciario, questa volta, due classi della scuola primaria di Casaglio, una della secondaria Venturelli e un'altra ancora della primaria di Ronco. In quest'ultima, in particolare, sarebbero risultati positivi due alunni e tre insegnanti."

Categoria Originale: Coronavirus - Focolai scolastico
Predizione: Coronavirus

"Nasce il consorzio per il sequenziamento del Sars-CoV-2 che riunisce il ministero della Salute, Iss e Aifa. Rezza: «Puntiamo a sistema di allerta precoce». Palù:«Le prossime pandemie arriveranno dagli animali». Sileri: «Monitorare serve anche all'economia»"

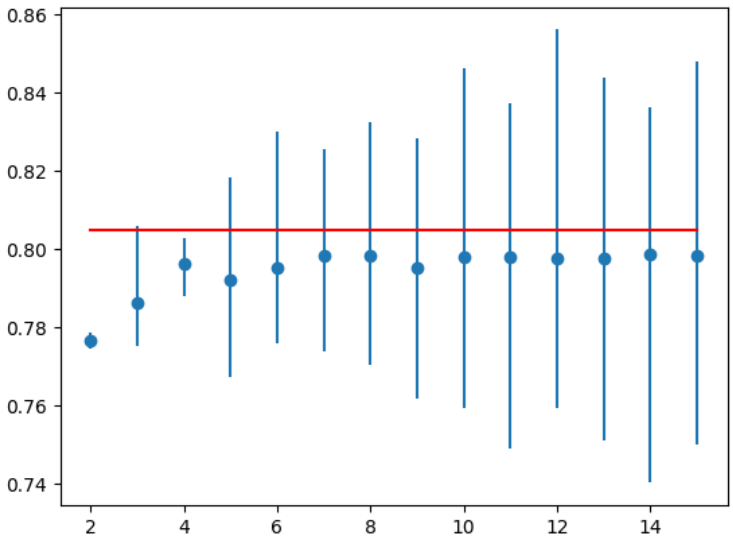
Categoria Originale: Coronavirus - VARIANTI
Predizione: Health-Effects

Sebbene si ribadisce, come fatto anche per lo scorso dataset, che il numero di esempi sia veramente limitato, si vuole far notare come un modello addestrato su notizie in inglese (anche molto diverse da queste) performi comunque in maniera interessante.
Per quanto riguarda la prima notizia, il modello riconosce che si sta parlando di una notizia afferente alla sfera di tipo sportiva, che nel dataset di cui stiamo parlando è riferita alla sezione di "Altri contesti".
Nella seconda, invece, riesce a riconoscere la categoria più generale di Coronavirus mentre nella terza ci si poteva aspettare una classificazione più tendente ad un topic economico.

3.3.3 Tuning, Training e Test Set

Viene utilizzato lo stesso approccio per definire il numero migliore di **folds** per la **Cross Validation** per effettuare tuning di iperparametri.
Di seguito il risultato prima testuale e poi grafico:

```
Ideal: 0.805
> folds=2, accuracy=0.777 (0.774,0.779)
> folds=3, accuracy=0.786 (0.775,0.806)
> folds=4, accuracy=0.796 (0.788,0.803)
> folds=5, accuracy=0.792 (0.767,0.818)
> folds=6, accuracy=0.795 (0.776,0.830)
> folds=7, accuracy=0.798 (0.774,0.825)
> folds=8, accuracy=0.798 (0.770,0.833)
> folds=9, accuracy=0.795 (0.762,0.828)
> folds=10, accuracy=0.798 (0.759,0.846)
> folds=11, accuracy=0.798 (0.749,0.837)
> folds=12, accuracy=0.798 (0.759,0.856)
> folds=13, accuracy=0.798 (0.751,0.844)
> folds=14, accuracy=0.799 (0.740,0.836)
> folds=15, accuracy=0.798 (0.750,0.848)
```



Dopo un'analisi del grafico si decide di utilizzare $\mathbf{k = 4}$.

Per quanto riguarda lo split fra train e test set si è usata la proporzione 70/30 e seme 9.

3.3.4 Baseline (Logistic Regression)

Per questo e soprattutto per i successivi modelli si riportano soltanto le valutazioni e gli eventuali tuning dal momento che sono già stati esaminati nella trattazione del precedente dataset.
Di seguito i risultati:

Confusion Matrix

Coronavirus - VARIANTI	104	3	6	0	0	0
Coronavirus - Focolai scolastico	38	123	11	3	0	0
Coronavirus - Focolai in altri contesti	59	5	200	9	2	1
Coronavirus - Focolai familiari/amici	28	6	23	17	1	0
Coronavirus - RSA/Case di riposo	30	2	8	1	152	3
Coronavirus - Focolai ospedalieri	20	0	18	0	5	96

Classification Report

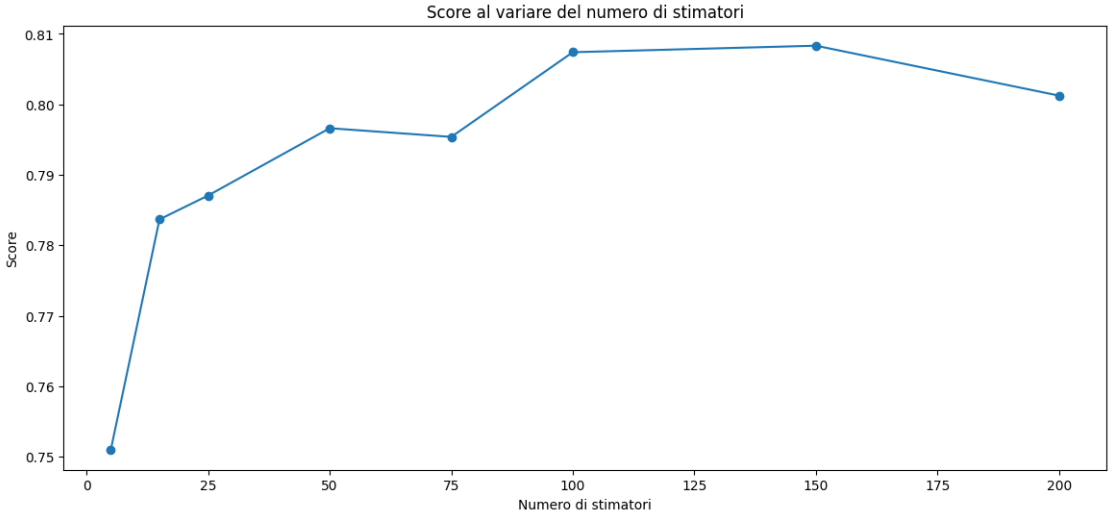
Category	**Precision**	**Recall**	**F1-Score**	**Support**
Coronavirus - VARIANTI	0.90	0.79	0.84	113
Coronavirus - Focolai scolastico	0.89	0.72	0.79	175
Coronavirus - Focolai in altri contesti	0.72	0.72	0.72	276
Coronavirus - Focolai familiari/amici	0.57	0.32	0.41	75
Coronavirus - RSA/Case di riposo	0.93	0.79	0.86	196
Coronavirus - Focolai ospedalieri	0.96	0.73	0.83	139
Global Micro Avg.	0.84	0.71	0.77	974

Per comodità di lettura si riportano solo i valori globali del modello con l'accuracy:

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Support**
Logistic Regression	0.68	0.84	0.71	0.77	974

3.3.5 Gradient Boosting Text Classification

Di seguito i risultati del tuning del modello rispetto gli stessi iperparametri trattati nel precedente dataset:



Viene deciso di utilizzare **125** stimatori.

N.B.: Anche qui a causa della richiesta computazionale decisamente esosa non si è voluto salire oltre i 200 stimatori.
Il procedimento per trovare i migliori parametri di **learning_rate** e **max_depth** è lo stesso del precedente dataset ma, ovviamente, con un numero di stimatori diverso.
Questo il codice:

```
gbc = GradientBoostingClassifier(n_estimators=125)
parameters = {
    "max_depth":[3,5,7],
    "learning_rate":[0.01,0.1,1]
}
```

0.2s

```
from sklearn.model_selection import GridSearchCV
cv = GridSearchCV(gbc,parameters,cv=folds)
cv.fit(x,y)
```

Al termine del procedimento i valori migliori risultano:

- **learning_rate** = 0.1
- **max_depth** = 3

Di seguito i risultati del modello:

Confusion Matrix

Coronavirus - VARIANTI	108	2	1	2	0	0
Coronavirus - Focolai scolastico	53	118	2	1	1	0
Coronavirus - Focolai in altri contesti	108	9	153	4	1	1
Coronavirus - Focolai familiari/amici	45	7	9	13	1	0
Coronavirus - RSA/Case di riposo	25	2	2	2	163	2
Coronavirus - Focolai ospedalieri	32	0	2	0	5	100

Classification Report

Category	**Precision**	**Recall**	**F1-Score**	**Support**
Coronavirus - VARIANTI	0.78	0.81	0.80	113
Coronavirus - Focolai scolastico	0.84	0.73	0.78	175
Coronavirus - Focolai in altri contesti	0.90	0.56	0.69	276
Coronavirus - Focolai familiari/amici	0.44	0.20	0.28	75
Coronavirus - RSA/Case di riposo	0.93	0.86	0.89	196
Coronavirus - Focolai ospedalieri	0.93	0.73	0.82	139
Global Micro Avg.	0.86	0.68	0.76	974

Per comodità di lettura si riportano solo i valori globali del modello con l'accuracy:

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Support**
Gradient Boosting	0.63	0.86	0.68	0.76	974

3.3.6 Multinomial Naive Bayes Text Classification

Il procedimento per la ricerca del miglior parametro di smoothing è esattamente lo stesso e pertanto viene omesso.
Al termine di quest'ultimo il parametro migliore risulta essere **0.2848035868435802**.
Di seguito i risultati del modello:

Confusion Matrix

Coronavirus - VARIANTI	106	4	2	1	0	0
Coronavirus - Focolai scolastico	18	149	5	2	1	0
Coronavirus - Focolai in altri contesti	28	31	195	8	12	2
Coronavirus - Focolai familiari/amici	20	14	27	12	2	0
Coronavirus - RSA/Case di riposo	9	2	10	4	170	1
Coronavirus - Focolai ospedalieri	7	0	5	0	33	94

Classification Report

Category	**Precision**	**Recall**	**F1-Score**	**Support**
Coronavirus - VARIANTI	0.77	0.85	0.81	113
Coronavirus - Focolai scolastico	0.74	0.85	0.79	175
Coronavirus - Focolai in altri contesti	0.76	0.79	0.77	276
Coronavirus - Focolai familiari/amici	0.36	0.51	0.42	75
Coronavirus - RSA/Case di riposo	0.70	0.92	0.80	196
Coronavirus - Focolai ospedalieri	0.74	0.86	0.80	139
Global Micro Avg.	0.71	0.82	0.76	974

Per comodità di lettura si riportano solo i valori globali del modello con l'accuracy:

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Support**
Multinomial Naive Nayes	0.66	0.71	0.82	0.76	976

3.3.7 Neural Network Text Classification

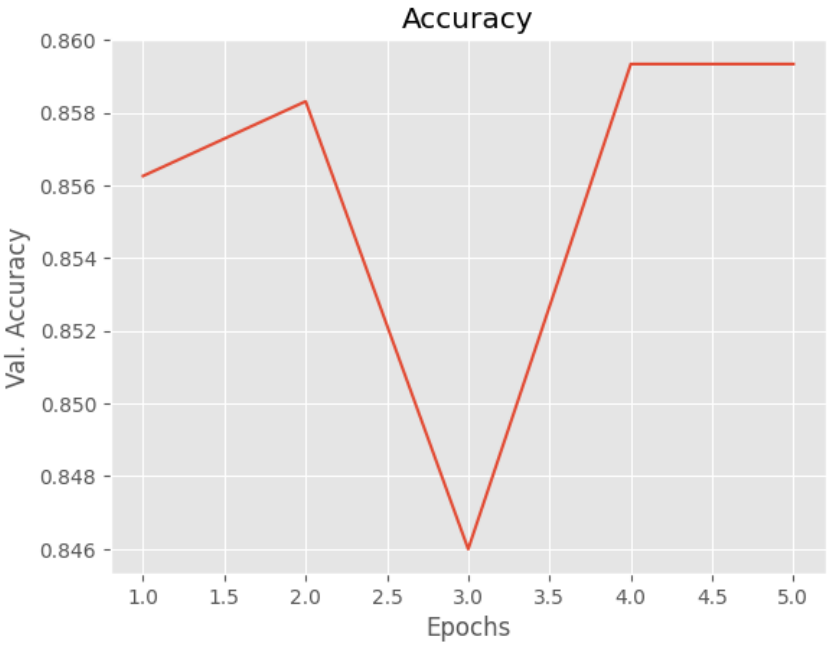
Questa volta, a differenza della precedente, il modello che verrà usato è **UmBERTo** [29], che è stato pre-addestrato su grandi corpora italiani.
Il file di riferimento per questa parte è "*Categories_NeuralNetwork.ipynb*", dopo aver eseguito l'exporting del file .csv pronto per essere elaborato tramite il pybook "Exporting_Definitive.ipynb", simile a quello usato nel precedente dataset.
Questo il risultato della migliore accuracy dell'epoca migliore (l'ultima):

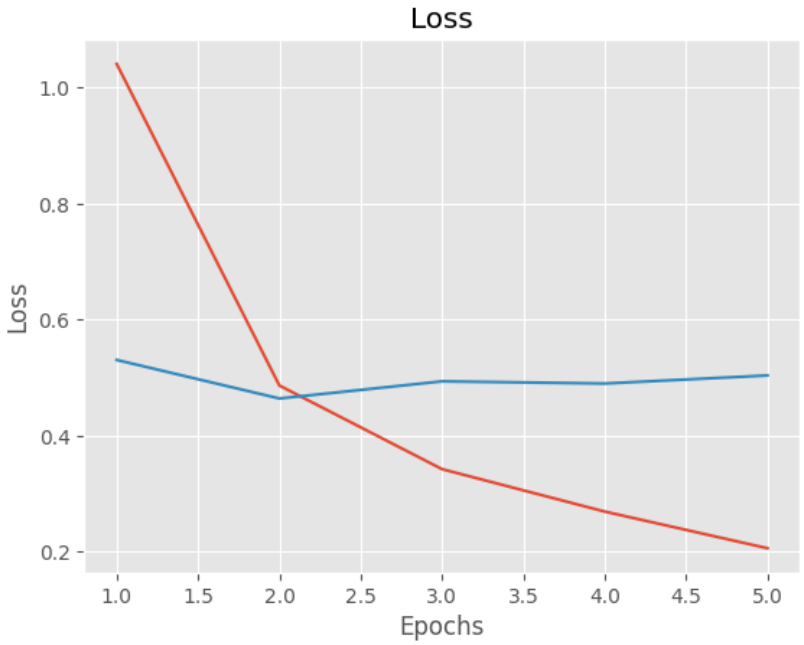
```
  Batch   280  of    341.    Elapsed: 0:23:12.
  Batch   290  of    341.    Elapsed: 0:24:01.
  Batch   300  of    341.    Elapsed: 0:24:51.
  Batch   310  of    341.    Elapsed: 0:25:41.
  Batch   320  of    341.    Elapsed: 0:26:30.
  Batch   330  of    341.    Elapsed: 0:27:20.
  Batch   340  of    341.    Elapsed: 0:28:09.

  Average training loss: 0.206
  Training epoch took: 0:28:12

Running Test...
  Accuracy: 0.859
  Test Loss: 0.504
  Test took: 0:04:02
```

I risultati sono soddisfacenti: si ricorda che stiamo classificando su 6 diverse categorie, che quindi significherebbe avere approssimativamente il 16% di indovinare a caso la categoria giusta.
Si specifica che, rispetto la precedente elaborazione, in questo caso si avevano a disposizione molti più dati e soprattutto l'annotazione degli stessi è stata fatta da esperti che, notizia per notizia, hanno scelto la categoria più adatta.
Probabilmente i risultati potrebbero essere migliorati con altri dati e con la specificazione più dettagliata della classe "Altri Contesti" ma è un argomento che ci si riserva di discutere parzialmente nella sezione conclusiva di questo elaborato.

Nelle prossime pagine, le conclusive di questo capitolo, si riportano i risultati ottenuti dal modello tramite le solite metriche introdotte pagine addietro.

Di seguito l'andamento di **Accuracy** e **Loss** nelle varie epoche:





Di seguito i risultati totali:

Confusion Matrix

Coronavirus - Focolai RSA/Case di riposo	183	0	11	0	2	0
Coronavirus - Focolai familiari/amici	1	51	19	0	4	0
Coronavirus - Focolai in altri contesti	7	25	232	4	6	2
Coronavirus - Focolai ospedalieri	16	3	7	111	0	2
Coronavirus - Focolai scolastico	0	5	9	1	160	0
Coronavirus - VARIANTI	0	3	7	0	3	100

Classification Report

Category	**Precision**	**Recall**	**F1-Score**	**Support**
Coronavirus - RSA/Case di riposo	0.88	0.93	0.91	196
Coronavirus - Focolai familiari/amici	0.59	0.68	0.63	75
Coronavirus - Focolai in altri contesti	0.81	0.84	0.83	276
Coronavirus - Focolai ospedalieri	0.96	0.80	0.87	139
Coronavirus - Focolai scolastico	0.91	0.91	0.91	175
Coronavirus - VARIANTI	0.96	0.89	0.92	113
Global Micro Avg.	0.85	0.84	0.85	974

Per comodità di lettura si riportano solo i valori globali del modello con l'accuracy:

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Support**
UmBERTo	0.86	0.85	0.84	0.85	976

3.3.8 Comparazione dei modelli

Di seguito una tabella generale di comparazione delle principali metriche globali:

Models	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.68	0.84	0.71	0.77
Gradient Boosting	0.63	0.86	0.68	0.76
Multinomial Naive Bayes	0.66	0.71	0.82	0.76
NN (UmBERTo)	0.86	0.85	0.84	0.85

Anche in questo caso, il modello che raggiunge i risultati migliori risulta essere la rete neurale.

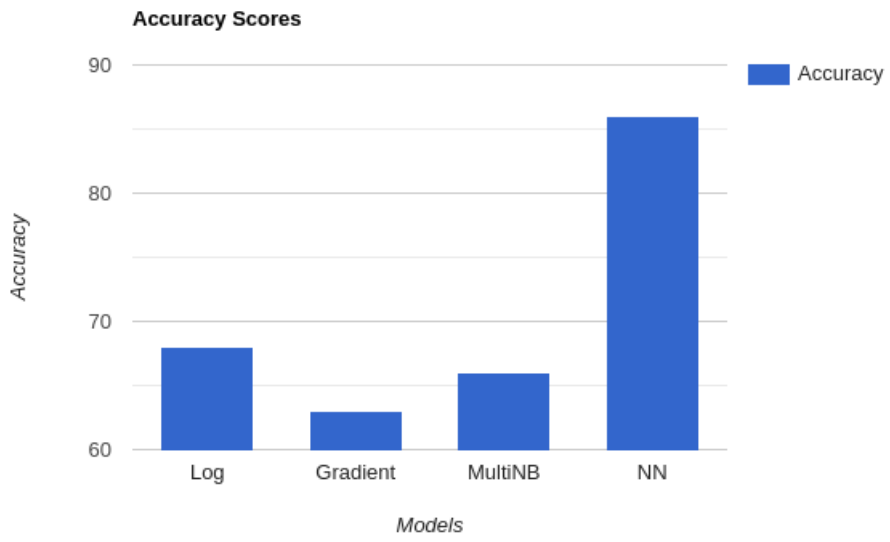


Figura 3.13. Grafico a barre raffigurante l'accuracy dei vari modelli.

3.3.9 Error Analysis

In questa sezione, l'ultima di questo lungo capitolo, si provvederà ad esaminare qualche errore più significativo per capire il comportamento del migliore classificatore trovato (la rete neurale).
Questa analisi diventa veramente rilevante nel contesto di collaborazione con l'ISS in cui si sta operando, poiché è lapalissiano dire che persone non pienamente afferenti a contesti informatici sono più interessate alle attribuzioni specifiche delle notizie che a numeri di accuratezza, di precisione o altro ed è proprio in questa direzione che vuole andare questa breve discussione.
Per cercare di indagare sugli errori commessi, si fa riferimento alla matrice di confusione per capire meglio su quali notizie non classificate correttamente concentrarsi.
Quanto segue, quindi, sarà un'analisi campionaria di alcune notizie per cercare di estrapolare informazioni rilevanti sulla corretta o meno classificazione o magari addirittura di una non corretta etichettatura iniziale dei dati.

"Cluster in una scuola calcio a Tor Bella Monaca, ora chiusa dalla Asl Roma 2. I bambini positivi, divisi in due squadre, sono allo stato attuale quattro, ma in quarantena ci sono decine di contatti nelle elementari limitrofe. All'incirca dieci classi ..."
La classe predetta dal modello è quella **scolastica** mentre l'assegnazione da parte degli esperti è stata quella di **altri contesti**. Possiamo benissimo notare come questa notizia sia "divisa" in due macroaree: sportiva e scolastica. Evidentemente, l'operatore che ha categorizzato questa notizia, ha preferito inserire la news in "altri contesti" ma non possiamo dire che il categorizzatore abbia sbagliato in modo rilevante.

"Sempre al Santa Maria del capoluogo ci sono quattro ricoverati di nazionalità cinese: nella comunità perugina si è infatti acceso un focolaio importante che è sotto stretto controllo dei sanitari."
La classe predetta dal modello è quella **ospedaliera** mentre l'etichetta assegnata dell'operatore è stata quella di **altri contesti**. Senza fare voli pindarici è ovvio che quest'ultima sia stata sbagliata poiché il Santa Maria è l'ospedale principale di Perugia e si sta parlando di ricoveri in esso; in questo caso il modello ha predetto correttamente la categoria.

"Tornano ad aumentare i casi di coronavirus in Calabria, e vengono scoperti anche diversi focolai, purtroppo ancora in case di cura per anziani e strutture sanitarie. Sono infatti due i nuovi focolai scoperti nel cosentino, che coinvolgono attualmente 55 soggetti risultati positivi seppur asintomatici. Il caso più grande riguarda la struttura di riabilitazione psicosociale Borgo dei Mastri situata nel comune di Paterno, dove una quarantina di pazienti sono risultati positivi al tampone molecolare. Il secondo caso invece riguarda una Rsa di Torano dove sono 15 gli anziani risultati positivi al covid, tutti asintomatici. Secondo una prima ricostruzione, alcuni di questi sarebbero entrati nella struttura come negativi al virus, per poi "positivizzarsi" in un secondo momento."

La categoria predetta dal modello è quella **RSA/Case di Riposo** mentre l'etichetta assegnata in principio è **ospedaliera**. Anche in questo caso ci troviamo davanti ad un errore di etichettatura da parte dell'operatore perché non c'è in alcun modo alcuna informazione riguardante degli ospedali bensì solo su due rsa e quindi si preferisce la predizione del modello.

"Sono una decina i cluster familiari in cui è stata evidenziata la presenza della mutazione inglese del virus Sars-Cov2. Dopo il caso della famiglia loretana emerso alla vigilia di Natale, la settimana scorsa è stata la volta della serie di nuclei residenti in varie zone del territorio provinciale dell'Anconetano." La categoria predetta dal modello è quella **familiare** mentre l'etichettatura fatta dagli operatori è stata rispetto le **varianti**.
Questa notizia è molto interessante da commentare poiché un po' come il primo esempio troviamo due flussi di informazione di differenti categorie; quello rispetto le famiglie e quello rispetto le varianti. L'operatore ha deciso di privilegiare il secondo mentre il modello ha "scelto" l'altro. Anche qui non possiamo dire che si tratti di un errore rilevante.

"Castelsardo, 31 ottobre 2020 – Un focolaio Covid-19 all'interno di un centro per disabili è stato scoperto nel Comune di Castelsardo. La situazione è in continua evoluzione e il sindaco Antonio Capula ha chiesto, tramite l'Unione Sarda, l'aiuto dell'esercito."
La classe predetta dal modello è quella **altri contesti** mentre l'etichetta originaria è **rsa**. Ci troviamo difronte al primo vero errore del modello che non ha saputo riconoscere il "centro per disabili" come RSA e che quindi ha etichettato, sbagliando, come altri contesti.

"Cava de' Tirreni. Sono saliti a 33 i casi di contagio da coronavirus nel reparto femminile di una casa di cura privata di Cava de' Tirreni. La situazione, dunque, si complica e diventa molto più allarmante rispetto alle scorse settimane. Lo riporta La Città di Salerno. I primi casi di positività tra le ospiti del centro di riabilitazione ..."
La classe predetta dal modello è **Rsa e case di riposo** mentre l'etichettatura originale è quella **ospedaliera**. In questo caso l'etichettatura giusta è quella del modello: si parla solamente di una casa di cura.

Come commento finale a questa "lente di ingrandimento" sui lavori svolti sui dati da parte del modello si vuole sottolineare come il valore riportato di accuratezza sia in ogni caso più alto, guardando campioni di malclassificazione qua e là che evidenziano o errori umani o errori poco rilevanti. Questa analisi campionaria, infatti, rafforza il lavoro svolto sui dati e rende ancora più interessante il risultato ottenuto.

Capitolo 4

NewsWorld

Questo breve capitolo, prima di quello conclusivo, riguarderà l'assemblaggio di tutti i singoli procedimenti spiegati ed illustrati nei precedenti capitoli in un'**architettura WEB** che sia fruibile agli scopi. Si precisa che il servizio WEB (chiamato NewsWorld) è un'approssimazione di quello che potrebbe essere una versione finale; non è focus di questo elaborato, infatti, occuparsi in maniera copiosa di stili grafici o altro ma si vuole solo far apprezzare l'operazione di automazione promessa in fase introduttiva.

N.B.: Anche per questa parte, il codice prodotto è interamente reperibile all'indirizzo https://github.com/PaoloCiasco/TesiMagistrale

4.1 Panoramica di Architettura

Su di una stessa macchina è necessario che siano presenti le istanze di un server Solr per l'indicizzazione e di un server vero e proprio che sia accessibile in rete su cui è installato il servizio.
La macchina (dotata di OS Unix-like) che viene sfruttata è ubicata nel laboratorio 25A, aula della macroarea di Scienze dell'Università di Tor Vergata, e sarà adibita a, appunto, server.
Per quanto riguarda la realizzazione back-end dello stesso, si è deciso di utilizzare Flask [30]: un framework molto semplice e veloce per la realizzazione di applicazioni WEB. Il servizio viene fornito sulla porta 5000 mentre Solr viene fornito sulla classica porta 8983.
Un utente potrà caricare notizie non categorizzate in formato ".xml" (rispettando la solita struttura ampiamente discussa nel corso di questa tesi), aspettare la loro, appunto, categorizzazione e ritrovarsele indicizzate su server Solr su cui, in un secondo momento, potrà eseguire delle interrogazioni per visionarli.

4.2 Realizzazione

La realizzazione dell'applicazione WEB è incentrata quasi interamente sul render di template html con Flask.
Come prima cosa viene caricato sul server il migliore modello trovato (la rete neurale) nel precedente capitolo: nel corso dell'addestramento era stato salvato affinché potesse essere riusato per effettuare direttamente predizioni.
Dopo aver fornito all'utente la possibilità di uploadare un file .xml (su cui viene effettuato un controllo di estensione per garantire che il file caricato sia del formato corretto), esso viene prima elaborato tramite lo script scritto in precedenza *xml_solr_like.py* affinché sia pronto per l'indicizzazione in Solr e poi tramite lo script *xml_csv_like.py*, leggermente modificato, verrà effettuato il retrieval delle news delle singole notizie per scriverle, riga per riga, in un file .txt che sarà generato sul server.
Quest'ultimo verrà poi scandagliato (di nuovo riga per riga) e ogni news sarà sottoposta al classificatore e la categoria corrispondente sarà salvata nel file .xml di ingresso.
Una volta che si ha il file .xml con le categorie inserite da parte del classificatore il file viene mandato con una chiamata POST al server Solr, esattamente come illustrato nel primo capitolo di questa tesi, e viene, finalmente, indicizzato.
In questa versione preliminare è possibile effettuare query (secondo il formato Lucene) attraverso l'url **".../query/<query da sottoporre>"** e le news recuperate verranno stampate a schermo.
Si precisa, ancora una volta, che in questa versione preliminare verranno mostrate solo news e categoria corrispondente.

Figura 4.1. Porzione di un risultato di una query.

Capitolo 5

Sviluppi Futuri e Conclusioni

Con la creazione del servizio WEB che racchiude tutti gli elementi trattati nel corso di tutte queste pagine, si chiude il lavoro sperimentale che si voleva proporre.
L'attività di categorizzazione automatica è senza dubbio molto utile e velocizza di molto un lavoro che sarebbe stato tedioso, lungo e, come mostrato nell'ultima sezione del precedente capitolo, "error prone".
L'accuratezza del miglior modello trovato può essere ritenuta soddisfacente e, in ogni caso, un buon punto di partenza per migliorare ancora di più i risultati proposti: si potrebbe pensare di utilizzare un modello pre-addestrato su dati di tipo medico-sanitario per provare a vedere possibili esiti differenti.
Senza dubbio colpisce l'estrema popolazione della classe *"Focolai in altri contesti"*; si potrebbe pensare di eseguire del **clustering** su di essa per provare a estrapolare classi più specifiche che riguardino settori più specifici, come ad esempio notizie di contagi riguardanti la sfera sportiva o quella economica.
Un'estensione interessante di questo lavoro potrebbe riguardare due principali sviluppi molto importanti: flessibilità di riconoscimento di diversi fenomeni virali e prevenzione degli stessi riconoscendo un qualche avvertimento dall'analisi di documenti.
Per quanto riguarda il primo aspetto, potrebbe essere senza dubbio stimolante riuscire, con approcci di tipo **semi-supervised**, passare dalla trattazione di una tematica virale come quella del Coronavirus alla più attuale del vaiolo delle scimmie.
Questo può essere realizzato con tecniche di tipo **few-shot learning** che prevedono la sottomissione al modello di pochissimi esempi per effettuare le nuove predizioni.
Impossibile non citare anche un approccio con la tecnica dell'**active learning** che richiama l'attenzione dell'utente affinché fornisca addirittura nuove categorie per eseguire classificazione.
Tornando agli sviluppi e parlando del secondo, l'attività di prevenzione può risultare veramente rilevante per stabilire quelli che comunemente vengono chiamati "**early warnings**": si potrebbe pensare di cercare di ricavare un qualche tipo di pattern ricorrente da documenti del passato che puntualmente si manifesta prima dell'occorrenza di un virus o più in generale di un fenomeno virale.

Rimanendo, invece, sul lavoro specifico eseguito nel corso di queste pagine, un possibile ovvio potenziamento potrebbe essere quello di ripetere l'apprendimento con l'utilizzo di **cross-validation**, per non parlare di una produzione grafica per il sito ed una miglioria copiosa per quanto riguarda la sezione delle interrogazioni all'indice di Solr (query).
Infine, un'ultima idea molto interessante che potrebbe essere utile sviluppare in futuro può essere la creazione di lessici diversi caratterizzanti le diverse classi, ovvero trovare quelle parole che più specificano l'una o l'altra categoria.

Bibliografia

[1] Website homepage. https://www.iss.it/. Accessed: 2022-07-13.

[2] Website homepage. https://solr.apache.org/. Accessed: 2022-07-13.

[3] Website abstract. https://it.wikipedia.org/wiki/Elaborazione_del_linguaggio_naturale/. Accessed: 2022-07-19.

[4] Elliotte Rusty Harold and W Scott Means. *XML in a nutshell: a desktop quick reference*. " O'Reilly Media, Inc.", 2004.

[5] Website homepage. https://www.w3.org/. Accessed: 2022-07-24.

[6] Website homepage. https://code.visualstudio.com/. Accessed: 2022-07-25.

[7] Website homepage. https://www.python.org/. Accessed: 2022-07-26.

[8] Nurzhan Nurseitov, Michael Paulson, Randall Reynolds, and Clemente Izurieta. Comparison of json and xml data interchange formats: a case study. *Caine*, 9:157–162, 2009.

[9] Python Software Foundation. Short tutorial for using xml.etree.elementtree. https://docs.python.org/3/library/xml.etree.elementtree.html.

[10] Covering Apache Solr. Apache solr quick overview. https://solr.apache.org/guide/8_11/a-quick-overview.html.

[11] Apache Fondation. List of public servers using apache solr. https://cwiki.apache.org/confluence/display/solr/PublicServers.

[12] Covering Apache Solr. Apache solr reference guide. https://solr.apache.org/guide/8_11/.

[13] Website wikipedia page. https://en.wikipedia.org/wiki/Comma-separated_values. Accessed: 2022-08-05.

[14] Website library doc. https://docs.python.org/3/library/csv.html. Accessed: 2022-08-11.

[15] Website library doc. https://pypi.org/project/pandas/. Accessed: 2022-08-11.

[16] Website library doc. https://pypi.org/project/langdetect/. Accessed: 2022-08-11.

[17] Murugan Anandarajan, Chelsey Hill, and Thomas Nolan. Text preprocessing. In *Practical Text Analytics*, pages 45–59. Springer, 2019.

[18] Steven Bird. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 69–72, 2006.

[19] Bhargav Srinivasa-Desikan. *Natural Language Processing and Computational Linguistics: A practical guide to text analysis with Python, Gensim, spaCy, and Keras*. Packt Publishing Ltd, 2018.

[20] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge, 2008.

[21] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[22] Ethem Alpaydin. *Machine learning*. MIT Press, 2021.

[23] Website web page. https://towardsdatascience.com/multiclass-classification-evaluation-with-roc-curves-and-roc-auc-294fd4617e3a. Accessed: 2022-09-08.

[24] Website web page. https://towardsdatascience.com/visual-guide-to-the-confusion-matrix-bb63730c8eba. Accessed: 2022-09-08.

[25] Website library doc. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html. Accessed: 2022-08-20.

[26] Website tutorial page. https://www.ibm.com/it-it/cloud/learn/neural-networks. Accessed: 2022-09-02.

[27] Website tutorial page. https://huggingface.co/bert-base-multilingual-cased. Accessed: 2022-09-02.

[28] Website github page. https://github.com/crux82/AILC-lectures2021-lab/blob/main/AILC_Lectures_2021_Training_BERT_based_models_in_few_lines_of_code.ipynb. Accessed: 2022-09-02.

[29] Website hugging face. https://huggingface.co/Musixmatch/umberto-commoncrawl-cased-v1. Accessed: 2022-09-20.

[30] Miguel Grinberg. *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.", 2018.

Ringraziamenti

Ringrazio il Prof. Giorgio Gambosi per avermi guidato nella realizzazione di questa tesi e di avermi fornito la possibilità di interfacciarmi con una entità importante come l'ISS.

Ringrazio il Prof. Danilo Croce per avermi supportato con preziosi consigli e critiche sempre costruttive per rendere sempre migliori i lavori svolti.

Ringrazio il Prof. Roberto Basili per l'impegno sempre costante nei confronti di questo lavoro e per essere stato, sebbene non ufficialmente, un relatore a tutti gli effetti.

Ringrazio tutto il corpo docenti del corso di laurea magistrale in informatica; sempre pronti ad aiutare ed attenti allo sviluppo accademico dell'individuo.

Ringrazio il mio amico Alessandro Straziota per avermi aiutato con graditissimi e grandissimi suggerimenti che hanno reso più facile e avvincente il processo produttivo di questo elaborato.

Ringrazio l'ambiente del lab.25A: caloroso e sempre disponibile per passare qualche minuto di pausa in allegria.

Ringrazio la mia famiglia, Papà Roberto e Mamma Barbara, per avermi supportato e sopportato in questi (lunghi) anni accademici e per aver rappresentato un faro guida da seguire e replicare.

L'ultimo ringraziamento è dedicato alla mia ragazza Adriana che ho conosciuto proprio nell'ultimo segmento di questi due anni e che ha cambiato la mia vita in meglio. Grazie per avermi spinto a fare sempre meglio e a starci sempre nei momenti no.