UNIVERSITÀ DI PISA

DIPARTIMENTO DI
FILOLOGIA, LETTERATURA E LINGUISTICA

CORSO DI LAUREA IN INFORMATICA UMANISTICA

TESI DI LAUREA

Valutazione di strumenti per l'estrazione e l'analisi automatica di sottotitoli impressi nei video

CANDIDATO
Celeste Dipasquale

RELATORE
Prof.ssa Maria Simi
Dott. Davide Italo Serramazza

CONTRORELATORE
Prof. Dell'Orletta Felice

ANNO ACCADEMICO 2020/2021

Sommario

Il presente lavoro di tesi si inserisce nell'ambito del progetto Public History REMIX (PH-REMIX) ossia un progetto di ricerca biennale (2020-2022) nato con l'obiettivo di investigare nuove metodologie per la fruizione, la valorizzazione e il riuso del patrimonio audiovisivo toscano mediante la realizzazione di una piattaforma, PH-Remix, che consente la ricerca e l'editing di porzioni di film documentari per raccontare nuove storie.

Un elemento innovativo del progetto è l'uso di tecniche di Intelligenza Artificiale (IA) per l'analisi del contenuto audiovisivo con lo scopo di frammentare porzioni di video e indicizzarle sulla base delle informazioni estratte.

Il presente elaborato pone l'attenzione sull'estrazione di informazione testuale fornita dai sottotitoli impressi nel flusso video e sulle strategie per poter adattare modelli, ideati per rilevare ed estrarre testo da immagini, a video.

La prima fase del lavoro è incentrata sulla valutazione di varianti dell'output del modello che, a partire da una lista contenente il testo estratto per ogni fotogramma, individua la migliore strategia per raggruppare i fotogrammi che condividono lo stesso sottotitolo e, dunque, costituiscono un'unica clip.

Nell'elaborato emerge, infatti, l'importanza della valutazione sia per individuare il modello che ottiene le prestazioni migliori ma anche, in corso d'opera, per apportare migliorie al modello e, dunque, al risultato.

La fase di valutazione è stata condotta anche sui risultati ottenuti dall'estrazione automatica di informazione linguistica per individuare quale *pipeline* ottiene le prestazioni migliori e, dunque, scegliere quale usare per estrarre automaticamente la categoria grammaticale e identificare e classificare le entità nominali.

Le informazioni estratte costituiscono, dunque, le etichette (label) che indicizzano una clip e consentono la ricerca, da parte dell'utente, delle clip di interesse all'interno della piattaforma.

Indice

Indice figure

Indice tabelle

1. Public History Remix (PH-Remix)

Il presente lavoro si inserisce nell'ambito del progetto PH-REMIX[1] e, in particolare, nella prospettiva di estrazione automatica di informazioni dai sottotitoli.

Il progetto "PH-Remix" (Public History Remix) è nato dalla collaborazione tra il Laboratorio di Cultura Digitale, la Fondazione Sistema Toscana (FST) e il Festival dei Popoli con l'obiettivo di investigare nuove metodologie per la fruizione e la valorizzazione del patrimonio audiovisivo del Festival dei Popoli.

Il Festival dei Popoli[2] è una associazione senza scopo di lucro fondata nel 1959 con l'obiettivo di studiare, conservare e promuovere il cinema di documentazione sociale. L'archivio del Festival dei Popoli contiene oltre 25 000 film-documentari di genere, lingua e stile diversi raccolti nel corso dei sessant'anni di attività del Festival ed è in continua espansione.

Il progetto si pone lo scopo, da un lato, di potenziare la catalogazione tradizionale mediante l'estrazione automatica di informazioni e, dall'altro, di permettere un riuso di segmenti per creare nuovi contenuti. L'obiettivo ultimo del progetto è, dunque, quello di sviluppare una piattaforma web "PH-Remix", basata sull'intelligenza artificiale, per esplorare l'archivio e raccontare nuove storie mediante il remix dei video[3].

1.1. La tecnica del remix nella Digital Public History

Il progetto PH-Remix, come emerge dal titolo stesso, si inserisce nel contesto della *remix culture*. Il termine *remix* indica la pratica della produzione culturale che prevede la modifica di un'opera preesistente tramite l'aggiunta, rimozione o cambiamento con l'obiettivo di creare nuovi contenuti.

Esso nasce tra la fine degli anni Sessanta e l'inizio dei Settanta in ambito musicale per poi ampliarsi ad ogni forma d'arte[4] grazie anche al ruolo centrale svolto dal web e dalle nuove tecnologie digitali: *everything is a remix*[5]. Il remix non è, dunque, un fenomeno

[1] «PH-Remix», Laboratorio di Cultura Digitale, http://www.labcd.unipi.it/ph-remix/.
[2] «Festival dei Popoli», Festival dei Popoli, https://www.festivaldeipopoli.org/.
[3] Giovanni Grasso, Chiara Mannari, Davide Serramazza «"PH-Remix": un progetto interdisciplinare per la valorizzazione del patrimonio audiovisivo del Festival dei Popoli-Festival Internazionale del Film Documentario di Firenze in ambiente digitale»
[4] «Remix Theory » Remix Defined», consultato 4 marzo 2022, https://remixtheory.net/?page_id=3.
[5] Cfr. Ferguson K., Everything is a Remix: https://www.everythingisaremix.info/watch-the-series

legato alla digitalizzazione ma è essa che lo ha reso un fenomeno di massa, *remix culture,* facilitando la fruibilità e la modificabilità dei contenuti.

Il progetto cerca di sperimentare la tecnica del remix e, in particolare, del *remix audiovisivo* nell'ambito della *digital public history*.

Per poter comprendere il perché possa essere utile realizzare un progetto di remix video in ambito di *public history* è importante chiarire cosa sia e quali siano gli obiettivi. La *public history* è una disciplina che ha come obiettivo principale quello di avvicinare storia e pubblico attraverso strumenti al passo con l'evoluzione tecnologia; il *public historian*, infatti, non si limita ai supporti "tradizionali", come la scrittura, ma offre storiografia, crea fonti e costruisce siti per aumentare la consapevolezza della storia e la permanenza delle memorie collettive[6].

In questa prospettiva si inserisce, dunque, l'obiettivo del progetto di creare una piattaforma che consenta, da un lato, l'esplorazione dell'archivio e, dall'altro, l'acquisizione della consapevolezza storica coinvolgendo gli utenti finali nell'elaborazione di contenuti mediante il remix del cinema documentario.

Le fonti utilizzate sono i film documentari, i quali non devono essere visti, meramente, come oggetto di intrattenimento bensì essi acquisiscono un ruolo di documento storico poiché testimoni del mondo contemporaneo e dell'immaginario sociale e, quindi, usati come una risorsa per lo studio e la trasmissione della storia.

Il database della piattaforma contiene ad oggi, aprile 2022, oltre 300 film documentari di genere, stile e lingua diverse provenienti da 67 nazioni per un totale di 316 ore di contenuti e 800 000 clip utilizzabili per il remix.

Nella piattaforma è, infatti, possibile ricercare le clip di interesse mediante un motore di ricerca integrato che consente la ricerca di parole chiave che descrivono gli oggetti presenti nelle clip, i colori dominanti o le parole estratte dai sottotitoli (Figura 1).

[6] S. Noiret, «Public History» e «storia pubblica» nella rete, «Ricerche storiche» XXXIX/2-3 (2009), pp. 276

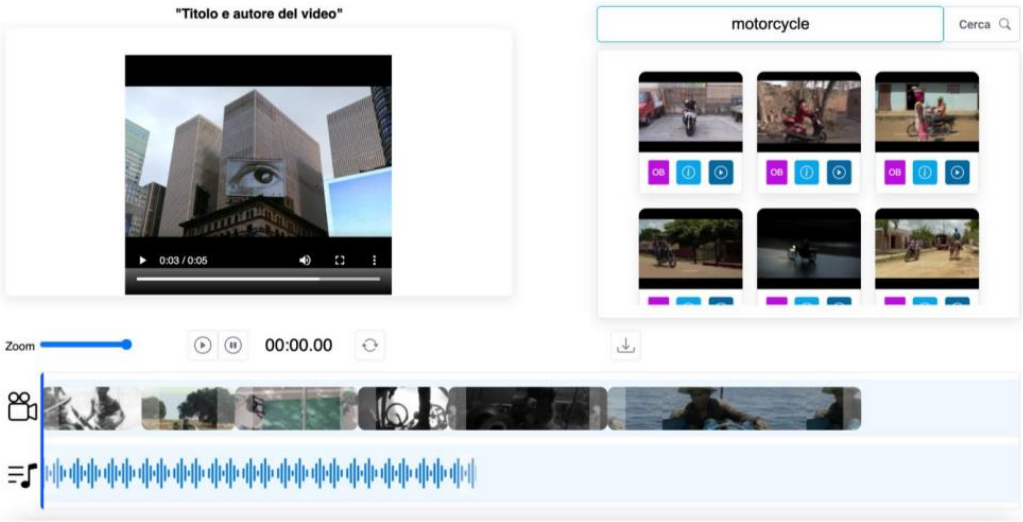


Figura 1: Prototipo dell'interfaccia web della piattaforma per la ricerca e remix di video.

1.2. La piattaforma PH-REMIX

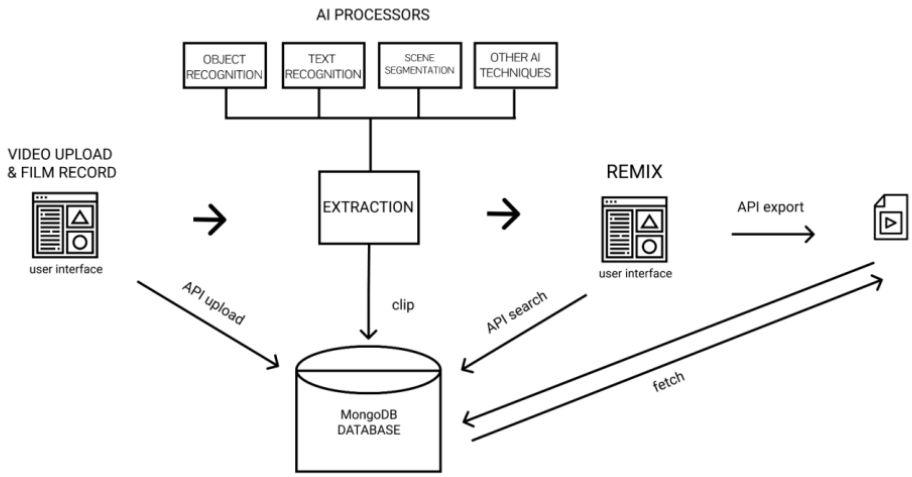


Figura 2: Architettura della piattaforma PH-Remix.

La piattaforma PH-Remix presenta un'architettura software[7] (Figura 2) basata su microservizi che comunicano tra loro attraverso API (*Application Programming Interface*).

È possibile, infatti, individuare tre fasi distinte ma dialoganti: la prima, prevede il caricamento sul server dei film attraverso un'interfaccia web ad accesso riservato; i film vengono, nella seconda fase di estrazione, frammentati in sequenze significative

[7] Giovanni Grasso, Chiara Mannari, Davide Serramazza, «*Intelligenza artificiale e archivi audiovisivi: potenzialità e sfide del progetto "PH-Remix"* » in «*AIUCD 2021 - DH per la società: e-guaglianza, partecipazione, diritti e valori nell'era digitale*» , pp. 141-144

a cui sono associate delle etichette (*label*) necessarie per l'indicizzazione. Le *clip* vengono, così, pubblicate nell'ambiente di remix che consente la ricerca, la visualizzazione e l'editing dei video.
Nella fase di estrazione ciascun film viene analizzato da diversi *processor* basati su algoritmi di *Intelligenza Artificiale* (IA) con lo scopo di frammentare i film in segmenti video sulla base delle informazioni estratte. La clip viene, dunque, rilevata, isolata dal resto del film e indicizzata assegnando un'etichetta che descrive l'informazione estratta e che corrisponderà alla parola chiave usata dall'utente per ricercare la clip.

1.2.1. Il contributo dell'Intelligenza Artificiale

Per la suddivisione dei film in clip e per l'analisi del contenuto audiovisivo sono stati usati diversi *processor* che dividono i film originali in segmenti più brevi sulla base dell'identificazione di alcune caratteristiche.
È stato utilizzato un primo *processor* per l'identificazione delle transizione fra un'inquadratura e la successiva per poter, così, estrarre da un film tutte le clip in cui vi sono dei cambiamenti di inquadratura assegnando come etichetta il colore dominante nella clip.
Il secondo *processor* è un modello usato per estrarre le informazioni relative agli oggetti rilevati nella scena (*object detection*). La *label* sarà, dunque, l'oggetto riconosciuto nella clip.
Il terzo *processor* è utilizzato per estrarre i sottotitoli impressi nei film documentari. I sottotitoli sono impressi (*embedded*) nel flusso video, motivo per cui sono stati usati modelli per il riconoscimento e l'estrazione automatica di testo da un'immagine *(Optical Character Recognition*).
È importante, infatti, evidenziare come i modelli usati sia per *object* che per il *text recognition* siano stati ideati per lavorare su immagini piuttosto che su materiale video, motivo per cui l'output dato dal modello, contenente una lista di informazioni estratte da ogni fotogramma, viene post-processato raggruppando i fotogrammi che condividono lo stesso elemento e che, quindi, formano una clip.
Il presente progetto, nello specifico, si concentra sull'output del modello usato per estrarre i sottotitoli.
La prima fase del lavoro riguarda la valutazione di varianti dell'output del modello che differiscono per i criteri usati nel raggruppare i fotogrammi che costituiscono una clip.

Il testo rilevato dalle clip rappresenta l'input delle fasi successive del lavoro che consiste nell'estrarre informazione linguistica, morfosintattica e semantica, con l'obiettivo di individuare le parole chiave per poter consentire la ricerca all'interno della piattaforma.

Per estrarre, in maniera automatica, le informazioni linguistiche si utilizzano tecniche di *Elaborazione del Linguaggio Naturale.*

Il termine *Elaborazione del Linguaggio Naturale*[8] (o *NLP* dall'inglese *Natural Language Processing*) viene adottato per indicare l'insieme di tecniche computazionali usate per l'analisi e rappresentazione di testi a uno o più livelli di analisi linguistica con lo scopo di ottenere un'elaborazione del linguaggio simile a quella umana.

Il processo di elaborazione viene suddiviso in fasi diverse che rispecchiano i diversi livelli linguistici: analisi lessicale, morfologica, sintattica, semantica e pragmatica.

Nel presente elaborato si pone l'attenzione sui livelli morfo-sintattici e semantici per estrarre le informazioni relative, soprattutto, al tipo di categoria grammaticale (POS dall'inglese *Part of speech*) e individuare, estrarre e classificare le entità nominali (NER dall'inglese *Named Entity Recognition*).

L'oggetto di analisi per ogni fase del lavoro sono quindici dei film presenti nella piattaforma; i film differiscono per il genere, lo stile e la lingua utilizzata ma sono accomunati dall'uso della lingua inglese per i sottotitoli, motivo per cui tutti i modelli utilizzati sono addestrati su *dataset* in lingua inglese.

[8] Ela Kumar, *Natural Language Processing* (I. K. International Pvt Ltd, 2013).

2. Estrazione di sottotitoli impressi nei video

L'obiettivo del progetto è estrarre le informazioni linguistiche dai sottotitoli impressi in video clip.

L'input è, dunque, costituito dai sottotitoli ma è importante evidenziare come non sono forniti da un file esterno bensì sono impressi (*embedded*) nel flusso dei video, motivo per cui è stato necessario estrarli utilizzando tecniche di Intelligenza Artificiale.

L'estrazione automatica dei sottotitoli è stata effettuata, in una fase precedente rispetto al presente lavoro di tesi, mediante la combinazione di due modelli: l'*Efficient and Accurate Scene Text Detector* (EAST) che individua le regioni spaziali in cui sono presenti i sottotitoli; un *Optical Character Recognition* (OCR) che analizza tali porzioni ricavandone il testo contenuto.

È importante evidenziare come questi modelli siano ideati per estrarre testo da immagine piuttosto che da materiale video, motivo per cui l'input della *pipeline*, ovvero l'insieme delle fasi che costituiscono un modello, è rappresentato da ogni fotogramma che viene, successivamente, raggruppato in clip sulla base del testo estratto; una sequenza di fotogrammi che condividono lo stesso sottotitolo formano un'unica clip.

Sono, dunque, state apportate alcune modifiche all'output del modello per consentire di raggruppare nel miglior modo possibile i fotogrammi che hanno lo stesso testo e, quindi, costituiscono lo stesso frammento.

Per poter decretare quale sia il modo migliore è necessario effettuare una valutazione sui risultati del modello; nella prima fase di questo lavoro di tesi è stato, infatti, valutato il risultato di varianti del modello per capire quale ottiene le prestazioni migliori.

In questo capitolo sono analizzati, nella prima parte, i modelli EAST e OCR per comprendere la loro architettura e funzionamento nell'estrarre testo da immagini.

L'obiettivo del progetto, tuttavia, è quello di estrarre il testo da video e, nel secondo paragrafo, si evidenziano le caratteristiche delle varianti dell'output che differiscono per il modo in cui si raggruppano i sottotitoli estratti per ogni fotogramma che formano un'unica clip.

Nel terzo paragrafo viene presentato il lavoro svolto per la realizzazione del *gold standard* e, dunque, per la valutazione delle varianti dell'output.

2.1. Strumenti e Metodologie per l'estrazione di testo da immagini

In questa sezione è esaminata l'architettura dei modelli *Efficient and Accurate Scene Text Detector* e *Optical Character Recognition* per comprendere come avviene, generalmente, l'estrazione automatica di testo da immagini.

2.1.1. Efficient and Accurate Scene Text Detector (EAST)

Efficient and Accurate Scene Text Detector (EAST)[9] è un modello proposto da Zhou et al. con l'obiettivo di rilevare l'area in cui è presente il testo.
L'idea alla base di questo modello è quella di utilizzare una *pipeline* semplice e veloce che individui lo spazio in cui è presente il testo.

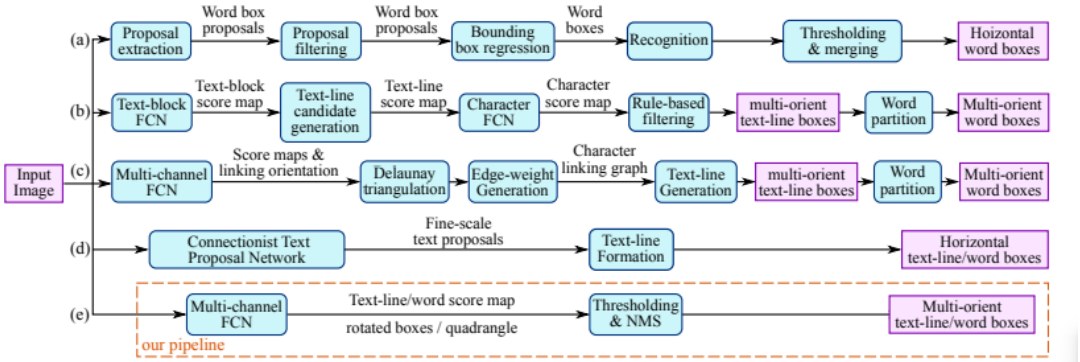


Figura 3: Confronto di diverse pipeline per l'estrazione di testo da immagini: (e) pipeline dell' Efficient and Accurate Scene Text Detector (EAST) che consiste in due sole fasi eliminando gli step intermedi.

La *pipeline* è costituita, di fatti, da due fasi: la prima prevede l'uso di una *Fully Convolutional Network (*FCN) che produce predizioni riguardanti lo spazio in cui è presente il testo; le predizioni vengono, così, inviate ad un algoritmo di post-elaborazione, *Non Maximum Suppression* (NMS), per produrre i risultati finali.

[9] Xinyu Zhou et al., «EAST: An Efficient and Accurate Scene Text Detector», in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI: IEEE, 2017), 2642–51, https://doi.org/10.1109/CVPR.2017.283.

Il modello è dotato di una rete neurale artificiale, *Fully Convolutional Networks* (FCN), che è allenata per predire, a partire da immagini, le forme geometriche in cui è presente il testo.

La rete FCN è un particolare tipo di rete neurale convoluzionale (*Convolutional Neural Networks*), usata in ambito di segmentazione semantica, che prevede l'assegnazione di un'etichetta a livello dei singoli pixel dell'immagine[10].

Una rete neurale convoluzionale (*CNN o ConvNet*) è un'architettura di rete utilizzata, soprattutto, in ambito di analisi e rilevamento di immagini poiché permette di estrarre le feature apprendendole direttamente dai dati[11]. L'input della rete è un'immagine che ha tre dimensioni *altezza* × *larghezza* × *profondità* dove l'altezza e la larghezza rappresentano le dimensioni spaziali mentre la profondità è data dal numero di colori di base, generalmente 3 (rosso, verde e blu), usati per caratterizzare l'immagine.

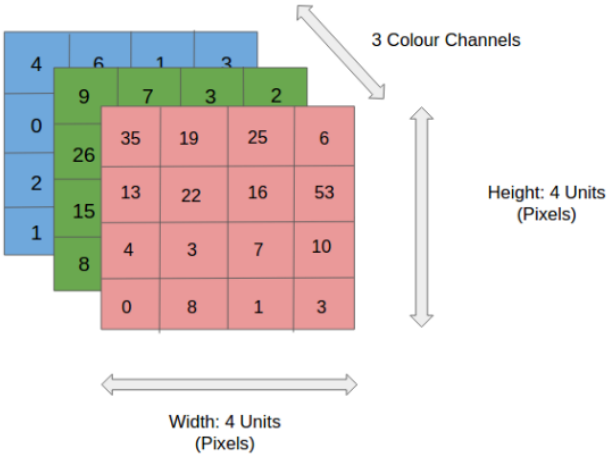


Figura 4: Descrizione numerica di un'immagine a colori.

L'architettura di base di una rete neurale convoluzionale è formata da due strati (*layer):* strato convoluzionale e strato di pooling.

[10] Jonathan Long, Evan Shelhamer, e Trevor Darrell, «Fully Convolutional Networks for Semantic Segmentation», s.d., 10.
[11] Keiron O'Shea e Ryan Nash, «An Introduction to Convolutional Neural Networks», *ArXiv e-prints*, 1 novembre 2015.

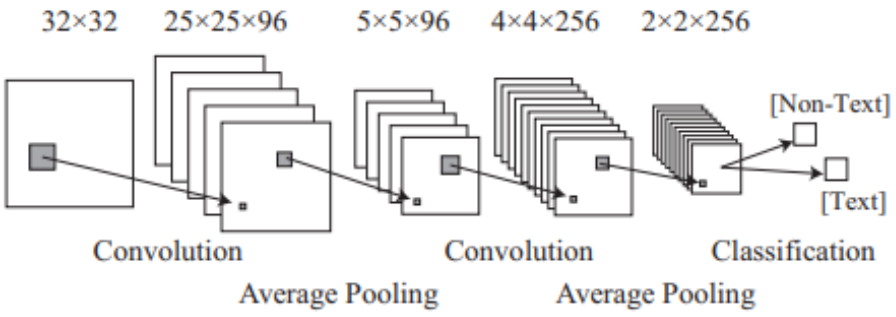


Figura 5: Esempio di architettura di rete neurale convoluzionale usata per il riconoscimento di testo.[12]

Come emerge dalla Figura 5 l'output di ogni *layer* è una matrice a tre dimensioni ($h * w * d$) che dopo ogni strato convoluzionale e di *pooling* si riduce per la dimensione *h* e *w* mentre aumenta la terza dimensione che rappresenta il numero di *feature* estratte dal singolo *layer*; ottenendo, così, un vettore in grado di descrivere quanto ogni singola feature estratta sia rilevante ai fini della classificazione dell'immagine.

Le FCN sono, dunque, reti neurali convoluzionali con la differenza che la classificazione si verifica a livello di singolo pixel; essa, di fatti, fornisce un'immagine segmentata dell'immagine originale in cui sono evidenziati gli elementi richiesti.

Nel caso del modello EAST la rete individua se il pixel possa essere classificato come carattere o no e fornisce le informazioni riguardanti lo spazio in cui esso è presente.

Ad ogni risultato viene applicato un processo di post-processing che prevede una prima sogliatura (*thresholding*) e l'utilizzo, successivo, di un algoritmo *Non Maximum Suppression*.

La *sogliatura*[13] è un metodo di segmentazione di un'immagine che isola gli oggetti (*foreground)* dallo sfondo *(background)*.

I singoli pixel di un'immagine sono, dunque, catalogati come "pixel oggetto" se superano un valore di soglia e come "pixel di sfondo" se il valore è sotto la soglia. L'immagine, solitamente, ha valore pari a 1 dove è presente l'elemento mentre 0 per lo sfondo.

[12] Tao Wang et al., «End-to-End Text Recognition with Convolutional Neural Networks», s.d., 5.
[13] P.K Sahoo, S Soltani, e A.K.C Wong, «A Survey of Thresholding Techniques», *Computer Vision, Graphics, and Image Processing* 41, n. 2 (febbraio 1988): 233–60, https://doi.org/10.1016/0734-189X(88)90022-9.

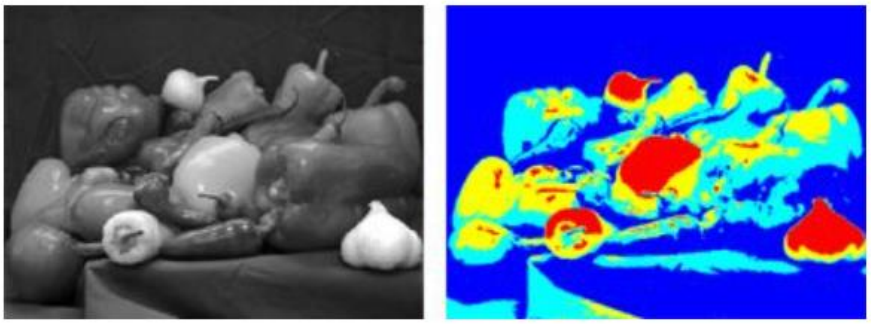


Figura 6: Esempio del metodo di sogliatura multilivello che distingue lo sfondo dagli oggetti.

Gli elementi che hanno punteggi superiori a quelli prefissati sono, dunque, ritenuti caratteri e salvati per la successiva *Non Maximum Suppression (NMS)*[14].

L'*NMS* è un algoritmo, generalmente, usato per selezionare un'entità, come un riquadro di delimitazione, tra molte entità sovrapposte unendo tutti i quadrangoli che appartengono allo stesso elemento.

Figura 7: Esempio di applicazione dell'algoritmo NMS che prevede la selezione di un solo riquadro tra quelli individuati.[15]

Nel caso del modello EAST la selezione si basa sull'assunto secondo cui le forme geometriche di pixel vicini tendono ad essere correlate, motivo per cui è stato proposto di unire le forme geometriche riga per riga e, mentre si fa questo, la forma geometrica incontrata viene fusa con l'ultima unita. In questo caso, dunque, più che parlare di selezione delle forme geometriche dovremmo parlare di "media" poiché le coordinate del quadrangolo unito sono date dalla media ponderata del punteggio dei diversi quadrilateri.

[14] Subrata Goswami, «Reflections on Non Maximum Suppression (NMS)», *Medium* (blog), 13 gennaio 2020, https://whatdhack.medium.com/reflections-on-non-maximum-suppression-nms-d2fce148ef0a.

[15] Sambasivarao K, «Non-Maximum Suppression (NMS)», Medium, 30 aprile 2021, https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c.

I valori ottenuti dopo NMS sono, dunque, considerati i risultati finali della *pipeline* e costituiscono l'input della fase successiva che prevede l'utilizzo di un modello di *Optical Character Recognition* per poter estrarre il contenuto testuale presente nei riquadri selezionati.

2.1.2. Optical Character Recognition (OCR)

I sistemi di riconoscimento ottico dei caratteri (detti anche OCR dall'inglese *Optical Character Recognition*) sono programmi dedicati al riconoscimento automatico di caratteri elaborati otticamente.
Un sistema OCR è costituito, generalmente, da diversi componenti. La prima fase consiste nel digitalizzare il documento analogico usando uno scanner ottico. Quando si trovano le aree in cui è presente il testo, viene estratto ogni simbolo attraverso un processo di segmentazione. I simboli estratti possono essere pre-elaborati, eliminando rumore, per facilitare l'estrazione delle caratteristiche (o *features*) nella fase successiva. I caratteri estratti vengono classificati cioè ad ogni carattere estratto viene assegnata una classe di appartenenza. Le informazioni contestuali, infine, sono usate per ricostruire le parole del testo originale[16].

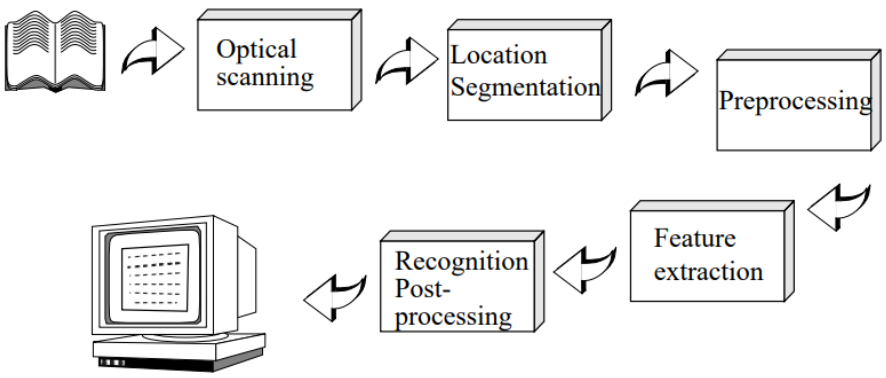


Figura 8: Fasi principali di un sistema OCR

Nel modello utilizzato nel progetto, l'input sono le coordinate spaziali dei riquadri in cui sono presenti i sottotitoli; questo significa che il modello di riconoscimento ottico dei caratteri prevede le fasi, sopracitate, di estrazione di feature, classificazione e post elaborazione.

[16] Line Eikvil, «Optical character recognition», *citeseer. ist. psu. edu/142042. html* 26 (1993).

Il modello OCR è stato, nel progetto, implementato dalla libreria *Tesseract* ossia un software libero (open-source) usato per il riconoscimento ottico dei caratteri[17].

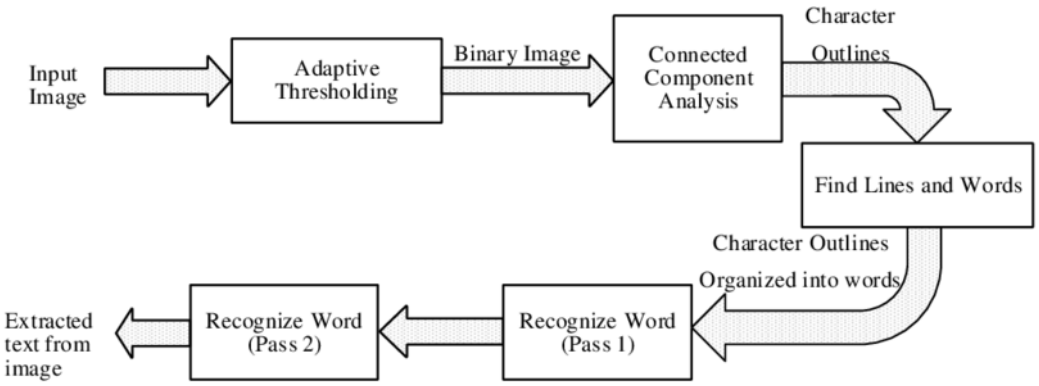


Figura 9: Architettura del software Tesseract OCR.

Le prime versioni di *Tesseract* prevedevano il riconoscimento dei caratteri analizzando le righe di testo per campioni fissi e suddividendole in parole usando la spaziatura dei caratteri, i quali vengono classificati sulla base di un elenco di classi dei caratteri che potrebbero corrispondere a quello ignoto.

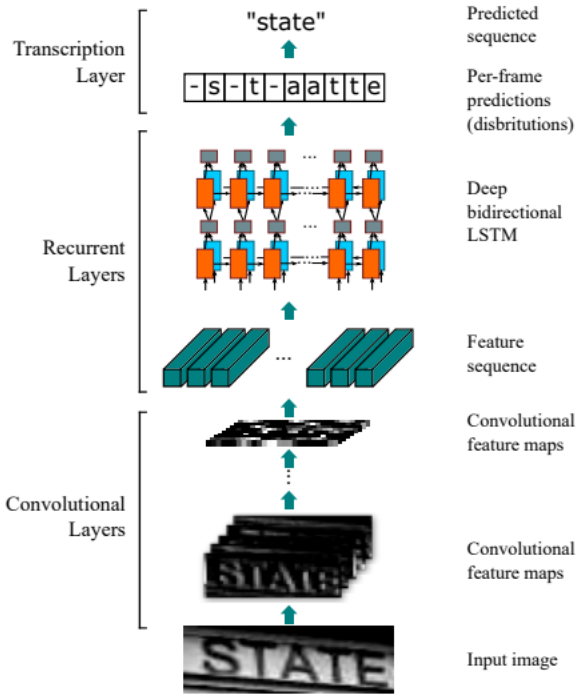
Ogni feature corrisponde ad un vettore di bit di classi a cui potrebbe corrispondere e i vettori di bit sono sommati su tutte le caratteristiche. Le classi con la somma più alta costituiscono l'elenco ristretto per il passaggio successivo.

Viene, così, calcolata la similarità tra il vettore di feature dell'elemento ignoto e il vettore di bit del prototipo della classe a cui potrebbe corrispondere. La decisione finale per una determinata segmentazione è la parola avente la distanza minore.

Questo tipo di classificazione, tuttavia, presuppone che si abbia a propri un elenco delle caratteristiche per ogni carattere.

Negli ultimi anni, con il crescente uso delle reti neurali, anche per il riconoscimento di testo da immagini ci si avvale delle reti neurali.

[17] R. Smith, «An Overview of the Tesseract OCR Engine», in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 2, 2007, 629–33, https://doi.org/10.1109/ICDAR.2007.4376991.



[18]

Figura 10: Architettura di un modello per il riconoscimento di sequenze testuali in un'immagine.

La Figura 10 è un esempio di riconoscimento di caratteri mediante l'uso di reti neurali: una rete *Convolutional Neural Network* (CNN) prevede il riconoscimento della riga mediante una scansione di ogni segmento del testo in ordine di lettura restituendo una mappa di *feature* che viene fornita come sequenza di input alla rete neurale ricorrente (LSTM).

Per ogni feature la rete neurale LSTM produce una sequenza di probabilità condizionata dagli altri caratteri che viene fornita a un meccanismo di *transcription* che si occupa di aggregare i singoli caratteri per formare la parola.

Questo è un esempio, dunque, dell'uso delle reti neurali per il riconoscimento dei caratteri da immagine.

Anche il software *Tesseract*, dalla versione 4.0, prevede l'uso di una rete neurale ricorrente: *Long Short Term Memory* (LSTM) per il riconoscimento delle righe di testo. Non essendo, tuttavia, presente la documentazione relativa alla quarta versione di Tesseract non è possibile analizzare nel dettaglio l'architettura.

Nel progetto il riconoscimento di caratteri viene, infatti, implementato con la libreria Tesseract 4.0 che prende in input le coordinate dell'area in cui è presente il testo, le analizza ed estrae il testo contenuto.

[18] Wang et al., «End-to-End Text Recognition with Convolutional Neural Networks».

Il risultato del modello OCR rappresenta il punto di partenza di tutto il lavoro.

2.2. Estrazione automatica di sottotitoli da video

È importante evidenziare come i modelli usati siano pensati per lavorare su immagini piuttosto che su materiale video, motivo per cui sono state apportate delle modifiche all'output del modello.

L'input della *pipeline* è rappresentato da ogni fotogramma del film e l'output è una lista contente gli elementi testuali estratti da ogni fotogramma.

L'obiettivo è estrarre le clip che contengono sottotitoli diversi, motivo per cui, a partire da questa lista, sono calcolate due metriche: la prima misura la differenza del testo estratto da un fotogramma confrontandolo con il precedente così da isolare l'insieme dei fotogrammi in cui vi è impresso uno specifico sottotitolo. La seconda è usata per fornire una misura della probabilità che la trascrizione sia una frase ben formata per poter selezionare tra tutte le trascrizione quella che è più probabile sia la vera trascrizione.

L'utilizzo di queste euristiche consente di raggruppare i fotogrammi che costituiscono la stessa scena sulla base del testo estratto e di frammentare il film in clip che contengono sottotitoli diversi.

Questo *processor* consente, dunque, di estrarre le clip che contengono specifici sottotitoli e la *label* corrisponde alla trascrizione rilevata.

L'output di ogni film è, dunque, una file .JSON contenente una lista di dizionari ognuno dei quali corrisponde a una clip diversa e ha le informazioni:

- "Processors"*:* tipo di *processor* utilizzato,
- "start"*:* tempo di inizio,
- "VideoId"*:* l'ID identificativo del video,
- "duration"*:* tempo di durata del frammento,
- "extracted_info": sottotitolo estratto,
- "SegId": identificativo del singolo frammento.

```
{
    "adminInfo": {
        "build_time": "2021-09-17 04:09:32.497332",
        "processors": "transcript"
    },
    "start": "00:02:10.631",
    "VideoId": "1293",
    "manually_modified": false,
    "duration": "00:00:30.072",
    "extracted_info": "Way to go, | like you like that! ",
    "SegId": "3530"
},
{
    "adminInfo": {
        "build_time": "2021-09-17 04:09:32.497386",
        "processors": "transcript"
    },
    "start": "00:02:40.744",
    "VideoId": "1293",
    "manually_modified": false,
    "duration": "00:00:02.628",
    "extracted_info": " Listen, everything ok with the soccer shoes? ",
    "SegId": "3531"
},
```


Figura 11: Esempio del file output del modello.

Nel corso del lavoro, grazie alla valutazione dell'output, sono state apportate alcune modifiche che interessano il raggruppamento dei fotogrammi sulla base dei sottotitoli impressi.

Sono state, infatti, realizzate tre versioni diverse di output: la prima prevedeva che il controllo della similarità tra il testo estratto da un fotogramma fosse fatto per un range di cinque fotogrammi così che anche se tra le prime due frasi non vi è corrispondenza è possibile che ci sia per le frasi successive; questo permette di trascurare eventuali rumori.

La seconda versione prevede, invece, un confronto non solo con la frase precedente ma anche con le due precedenti. Questo permette di raggruppare ulteriormente i fotogrammi che costituiscono la stessa scena.

Nella terza versione, invece, non è stato effettuato un controllo in un range di cinque frasi ma ogni frase è stata confrontata solo con la frase precedente e, se vi è una corrispondenza, viene presa quella più verosimile; nel caso contrario le due frasi vengono considerate come sottotitoli che appartengono a clip diverse e si passa ai successivi.

Per ogni diversa versione sono stati creati i file .JSON di ogni film che contengono i dizionari aventi le informazioni relative all'estrazione di sottotitoli.

Il lavoro consiste, dunque, nel valutare quale delle varianti del modello ottiene le prestazioni migliori.

2.3. Valutazione del modello per estrazione di sottotitoli

2.3.1. Realizzazione del gold standard

È stato, in primo luogo, necessario valutare le varianti dei risultati del modello e per farlo è stato realizzato un file *gold standard.*

Il corpus di partenza consiste in 15 file .JSON che contengono, dunque, i sottotitoli estratti in maniera automatica per ogni clip diversa da ogni documentario.

I film-documentari sono di lingua, genere e stile diversi ma ciò che li accomuna è l'uso della lingua inglese per i sottotitoli.

Da ogni file .JSON sono estratti i primi cinquanta elementi realizzando un unico file contenente una lista di dizionari che corrispondono ai primi cinquanta frammenti per ogni film documentario; i dizionari contengono le informazioni relative alla frase estratta, all'identificativo del film e al tempo di inizio, di durata e di fine della clip (Figura 12).

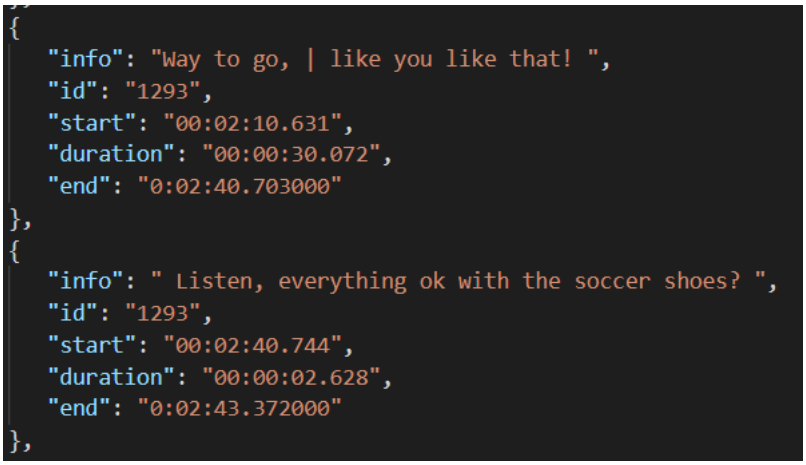


Figura 12: Esempio di elementi presenti nella lista contenenti i primi cinquanta sottotitoli estratti automaticamente da 15 film.

A partire da questo file .JSON sono stati rivisti e annotati manualmente i sottotitoli mediante la visione dei primi minuti di ogni film documentario realizzando un *gold standard* che presenta solo i sottotitoli diversi.

Il *gold* è, dunque, un file formato .txt contenente 512 frasi che corrispondono ai sottotitoli diversi.

L'idea di base per effettuare la valutazione dei risultati del modello è quella di confrontare la frase del *gold* con la corrispondente estratta automaticamente dal modello e vedere quanto esse siano simili.

I problemi emergono nel comprendere quale sia la frase corrispondente poiché avendo solo l'informazione testuale pur confrontando la similarità tra le due stringhe essa non è sufficiente in quanto vi sono casi in cui alcuni sottotitoli non sono riconosciuti dal modello portando ad un disallineamento dei dati e casi in cui il modello divide uno stesso sottotitolo in più clip (Figura 13):


```
{
    "info": "otherwise I'm going to buy Legea for all of you! ",
    "id": "1293",
    "start": "00:02:49.127",
    "duration": "00:00:00.417",
    "end": "0:02:49.544000"
},
{
    "info": "otherwise I'm going to buy Legea for all of you! ",
    "id": "1293",
    "start": "00:02:49.670",
    "duration": "00:00:00.876",
    "end": "0:02:50.546000"
},
{
    "info": "otherwise I'm going to buy Legea for all of you! ",
    "id": "1293",
    "start": "00:02:50.587",
    "duration": "00:00:00.918",
    "end": "0:02:51.505000"
},
```


Figura 13: Esempio tratto dal file .JSON che evidenza la divisione in più clip pur trattandosi dello stesso sottotitolo

L'idea è, dunque, quella di utilizzare gli intervalli temporali per poter avere un ulteriore informazione che consenta il confronto tra la frase annotata manualmente e quella estratta automaticamente dal modello.

Si è deciso, dunque, di realizzare un file .JSON che ha una lista di dizionari contenenti i sottotitoli annotati manualmente e gli intervalli temporali, tempo di inizio e tempo di fine, estratti in maniera automatica; essendo complicato per un umano individuare i secondi si è scelto di considerare le informazioni temporali estratte automaticamente dal modello.

Per poter aggiungere il tempo estratto automaticamente alla frase annotata manualmente è necessario, in primo luogo, individuare la corrispondenza tra le due frasi e per far questo si è calcolata la similarità utilizzando la libreria di Python *fuzzy*[19] che permette di confrontare le stringhe e, avvalendosi dell'*edit distance*, di calcolarne le differenze.

[19] YouGov Plc, *Fuzzy: Fast Python phonetic algorithms*, versione 1.2.2, POSIX, Python, consultato 15 marzo 2022, https://github.com/yougov/Fuzzy.

L'edit distance è una metrica usata per calcolare quanto siano simili due stringhe contando il numero minimo di operazioni di sostituzione, aggiunta o rimozione di caratteri necessarie per trasformare una stringa nell'altra[20].

La libreria *fuzzy* ha il metodo *.ratio* che calcola la similarità normalizzata tra le stringhe vedendo quante operazione siano necessarie per trasformare una stringa nell'altra e attribuendo a tutte le operazioni lo stesso peso. Il valore della similarità tra due stringhe data dal metodo *ratio di fuzzy* è uguale alla somma della lunghezza delle due stringhe meno il numero di operazioni necessarie per trasformare la prima stringa nella seconda il tutto diviso la somma della lunghezza delle due stringe (1):

$$ratio = \frac{lensum - ldist}{lensum} * 100 \quad (1)$$

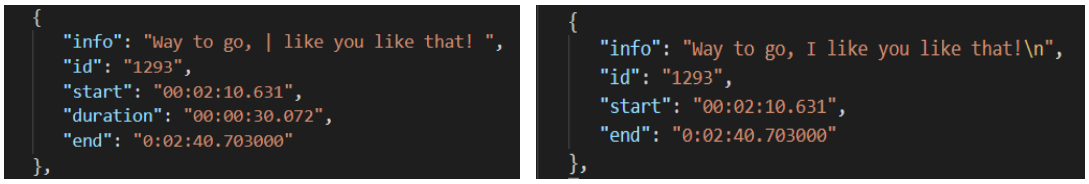
Due frasi uguali avranno valore pari a 100 mentre due frasi molto diverse avranno un valore più basso.

Questa metrica, tuttavia, risulta essere limitante nel caso di stringhe con lunghezza diversa o che presentano solo una parte in comune; in questi casi è più opportuno utilizzare il metodo *partial ratio*[21] che prende in considerazione le sub stringhe permettendo così di considerare anche casi in cui il modello riconosce solo una parte del sottotitolo. Si è scelto di considerare anche questo aspetto poiché si è osservato come, talvolta, il modello riconosceva solo una parte del sottotitolo.

Ogni frase del gold viene, dunque, confrontata con la frase estratta automaticamente e se il valore di similarità tra le due frasi supera un valore soglia allora viene ipotizzato che si tratti dello stesso frammento e, quindi, viene creato un elemento nel nuovo file .JSON che contenga la frase annotata manualmente e gli intervalli temporali estratti automaticamente (Figura 14b).

[20] «Edit distance», consultato 22 marzo 2022, https://nlp.stanford.edu/IR-book/html/htmledition/edit-distance-1.html.

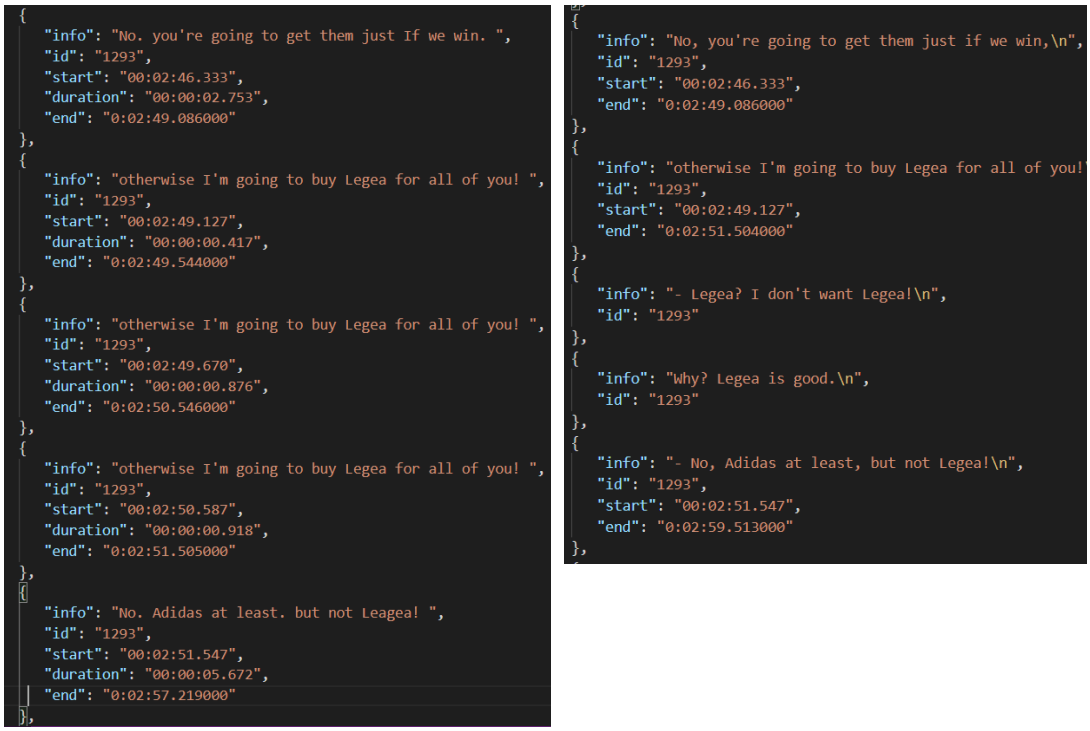
[21] «FuzzyWuzzy: Fuzzy String Matching in Python - ChairNerd», consultato 15 marzo 2022, https://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/.



(a) **(b)**

Figura 14: Esempio di confronto tra elemento estratto automaticamente dal modello (a) e l' elemento del nuovo dizionario .JSON (b) che contiene la frase annotata manualmente e i tempi estratti automaticamente.

Nei casi in cui non si trova una similarità tra la frase *gold* con una frase del modello viene aggiunto nella lista del nuovo file .JSON un elemento contenente solo la frase annotata manualmente senza l'informazione temporale (Figura 15b).



(a) **(b)**

Figura 15: Esempio di confronto tra il file output del modello(a) e il nuovo file .JSON (b) in cui possiamo notare la presenza di due elementi che contengono solo la frase annotata manualmente senza l'informazione temporale poiché non è presente nell'output.

Viene, inoltre, effettuato un controllo se nell'output del modello vi è uno stesso sottotitolo ripetuto in più frammenti calcolando la similarità di ogni elemento con l'elemento successivo sino a quando abbiamo un valore inferiore a un valore soglia e, quindi, si tratta di un nuovo sottotitolo; nel caso in cui vi è un sottotitolo ripetuto più

volte l'elemento da aggiungere al nuovo file .JSON presenterà, dunque, la frase una volta sola e il tempo di inizio corrisponderà al tempo di inizio del primo *frame* mentre il tempo di fine al tempo di fine dell'ultimo *frame* avente la stessa frase (Figura 16):

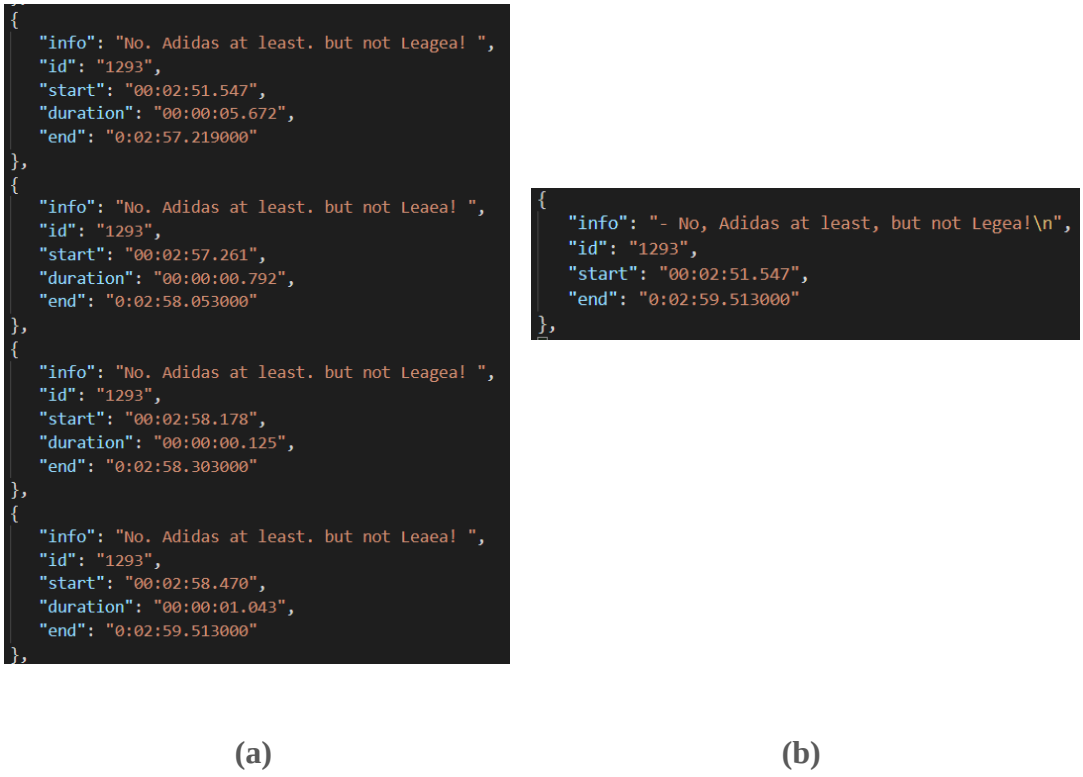


Figura 16: Esempio di uno stesso sottotitolo che viene diviso in più clip dal modello (a) e che è stato incorporato in un unico frammento che ha come tempo di inizio quello del primo elemento e come tempo di fine quello dell'ultimo elemento.

Viene, così, realizzato un unico file .JSON che ha una lista di dizionari contenti la frase annotata manualmente, il numero identificativo del film e i tempi di inizio e di fine della clip.

Il file è stato integrato, successivamente, anche con le informazioni estratte in maniera automatica dalla seconda versione così da ottenere maggiori informazioni temporali. Esso rappresenta, dunque, il file *gold* utilizzato per la valutazione e confrontato con il file .JSON avente la lista di dizionari dati dal modello.

2.3.2. Confronto tra i sottotitoli gold e l'output del modello

La valutazione prevede il confronto tra il file .JSON *gold* e l'output del modello per poter valutare quanto il modello sia accurato nel riconoscere un sottotitolo e quale delle varianti dell'output per il raggruppamento di fotogrammi in clip ottiene le prestazioni migliori.

Se il modello riconosce un frase che è presente nel *gold*, allora, significa che il modello ha individuato correttamente la presenza di un sottotitolo e viene, dunque, calcolata la similarità tra le frasi per valutare l'accuratezza. Nei casi in cui, invece, il modello riconosce qualcosa che non è nel *gold,* allora, si tratta di un *falso positivo* (FP) cioè una previsione positiva sbagliata che non si verifica nella realtà. Quando il modello non riconosce la presenza di un sottotitolo si tratta di un *falso negativo* (FN) cioè una previsione negativa sbagliata poiché il modello non riconosce la frase che, in realtà, è un sottotitolo.

Per poter valutare l'accuratezza del modello nel riconoscere i sottotitoli è necessario, in primo luogo, individuare se vi è una corrispondenza tra le frasi *gold* e l'output del modello utilizzando gli intervalli temporali.

L'idea è, dunque, quella di vedere la somiglianza tra due frasi presenti nello stesso intervallo di tempo per poter valutare quanto sia stato accurato il modello nel riconoscere i caratteri del sottotitolo.

È importante sottolineare, tuttavia, che le informazioni temporali sono estratte in maniera automatica questo comporta, da un lato, che non tutti gli elementi del dizionario *gold* abbiano le informazioni temporali; dall'altro, questo fa sì che non possiamo limitarci ad effettuare il confronto utilizzando solo il fattore tempo.

Ho, dunque, scelto di distinguere i casi in cui vi è anche l'informazione temporale da quelli in cui vi è solo l'informazione testuale.

Nel caso in cui non abbiamo informazioni temporali ci si limita a considerare la similarità tra le due frasi ma i confini dati dai segmenti precedenti e successivi permettono, comunque, l'allineamento.

Nell'esempio (Figura 17) notiamo nel file gold (a) la presenza di due elementi che non contengono le informazioni temporali; in questi casi viene calcolata la similarità tra la frase annotata manualmente e la corrispondente estratta automaticamente; se il valore è minore di un valore soglia, allora, si ipotizza che il modello non abbia riconosciuto quella frase e quindi si incrementa il calcolo dei falsi negativi. Si passa, così, al confronto tra l'elemento successivo nel *gold* e lo stesso elemento estratto dal modello sino a che non si ha o una similarità tra le frasi o un elemento che ha un'informazione temporale.

Figura 17: Esempio di elementi del gold (a) che non presentano l'informazione temporale.

Per gli elementi in cui sono presenti le informazioni del tempo di inizio e di fine della clip sono state considerate tre diverse situazioni.

La prima prevede una corrispondenza degli intervalli temporali cioè i tempi di inizio e di fine della frammento estratto automaticamente rientrano negli intervalli temporali di quello *gold* e viene, dunque, calcolata la similarità tra le due frasi. Nella Figura 17 (b) notiamo come i due elementi estratti dal modello presentano un tempo di inizio e di fine inclusi negli intervalli temporali degli elementi corrispondenti del *gold.*

Nel caso in cui non si individua una similarità tra le due frasi viene effettuato un ulteriore controllo per comprendere se il modello non ha riconosciuto la presenza di un sottotitolo o se ha riconosciuto qualcosa che non è sottotitolo.

Per far questo si considera il tempo di fine della frase *gold* poiché se non è stata trovata una similarità tra le frasi e il tempo di fine della frase *gold* è maggiore di quello della frase riconosciuta automaticamente, allora, significa che la frase estratta dal modello, in realtà, non è presente nel gold e, dunque, viene incrementato il calcolo dei falsi positivi (Figura 18).

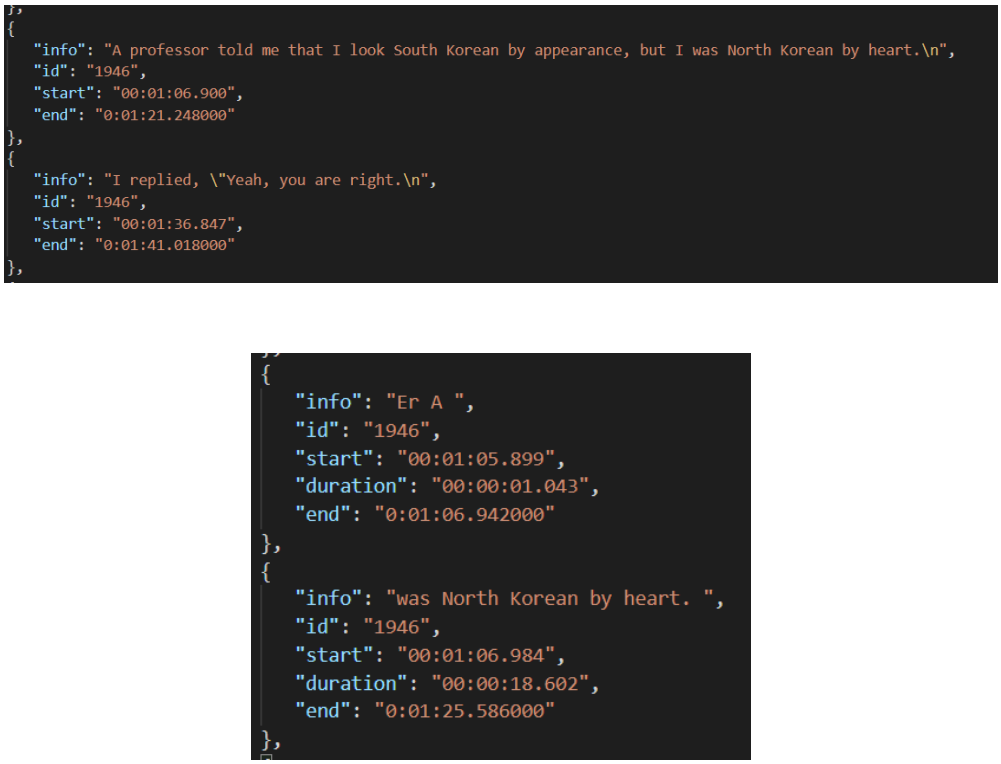


Figura 18: Esempio di falso positivo (b) poiché pur essendo presente nell'intervallo temporale del corrispondente elemento gold non presenta una similarità con la frase gold e ha un tempo di fine minore della frase gold.

Nel caso contrario, ossia che il tempo di fine del frammento gold sia minore di quello estratto in maniera automatica, allora, significa che il modello non ha riconosciuto la frase e, dunque, viene incrementato il calcolo dei falsi negativi.

La seconda casistica comprende, invece, i segmenti del file gold che hanno un tempo di inizio maggiore del tempo di fine del frammento estratto in maniera automatica; questo significa che la frase estratta dal modello non è presente nel *gold*, motivo per cui si incrementa il numero dei falsi positivi. Questo è il caso, per esempio, della Figura 19 in cui notiamo come il sottotitolo *"You know what?"* ha un tempo di inizio 3:12.359 che è maggiore del tempo di fine dell'elemento corrispondente estratto automaticamente 3:12:31 e visto che sino a quel momento non è stato trovato una similarità tra la frase estratta in maniera automatica e quella annotata manualmente allora si ipotizza che la frase estratta dal modello non sia presente nel file *gold* ma si tratti di un falso positivo e si passa al confronto tra la stessa frase *gold e* la successiva frase estratta dal modello.

Figura 19: Esempio di falso positivo (b) poiché il modello riconosce la frase "Den oR aT tells ae syd" che non è presente nel gold.

Il riconoscimento da parte del modello di qualcosa che in realtà non è presente nel *gold* si verifica, generalmente, quando il modello non riesce a individuare i caratteri presenti producendo rumore (Figura 19) o quando rileva testo che non è sottotitolo.

Il terzo caso, invece, considera i segmenti del *gold* che hanno un tempo di fine minore di quelli estratti in maniera automatica questo significa che il modello è andato avanti dal punto di vista temporale senza considera alcuni frammenti; in questo caso si incrementa il numero dei falsi negativi.

Quanto detto può essere esemplificato considerando la Figura 20: in cui emerge come nel *gold* il sottotitolo *sharper and more cynical* ha un tempo di fine 0:11 mentre il corrispondente estratto dal modello inizia al minuto 13 questo, dunque, significa che il modello non ha riconosciuto la presenza di un sottotitolo che è nel *gold*, motivo per cui si incrementa il calcolo dei falsi negativi e si continua confrontando la frase successiva del *gold* con la stessa estratta dal modello.

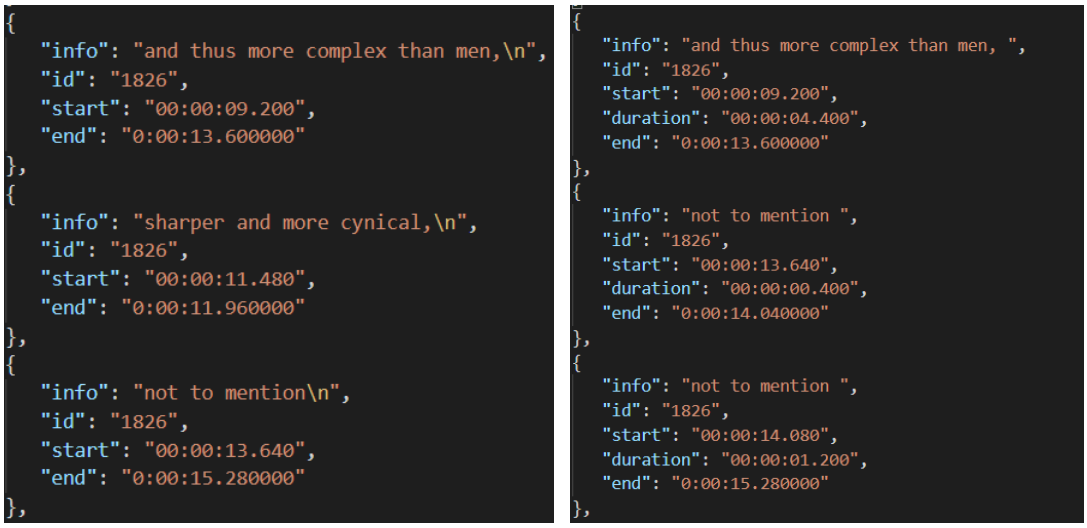


Figura 20: Esempio di falso negativo poiché il modello non riconosce la frase "sharper and more cynical" che è presente nel gold.

In tutti i casi viene considerato non solo il fattore tempo ma viene calcolata la similarità tra le frasi. Per calcolare la similarità tra due stringhe si è scelto di considerare come metriche l'*edit distance* e il W*ord Error Rate* (WER).

In questo caso l'e*dit distance* viene calcolata utilizzando l'algoritmo di *distanza di Levenshtein* (*Levenshtein distance*) che consente di confrontare stringhe con lunghezza differente considerando le operazioni quali rimozione, inserimento o sostituzione di un carattere tra due stringhe.

Nella definizione originale di Levenshtein[22] ogni operazione ha un costo unitario così la distanza di Levenshtein è uguale al minimo numero di operazioni richieste per trasformare una stringa. In realtà, è stata usata la libreria di Python L*evenshtein*[23] e, in particolare, il metodo *.ratio* che calcola la *distanza di Levenshtein* normalizzata e attribuisce un peso diverso alle operazioni in quanto all'operazione di inserimento e rimozione viene attribuito un costo di 1 mentre la sostituzione ha costo 2 sulla base dell'idea secondo cui per sostituire un carattere prima è necessario rimuovere quello presente nella stringa e poi aggiungerne un altro.

Il metodo *.ratio* restituisce un valore compreso in un range [0,1]; stringhe con un valore vicino allo 0 significa che sono molto differenti mentre se il valore è più vicino a 1 è più probabile che ci troviamo dinanzi a frasi molto simili. Si è scelto di calcolare l'edit

[22] Levenshtein, V. I. (1966, February). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady* (Vol. 10, No. 8, pp. 707-710).
[23] «Welcome to Levenshtein's documentation! — Levenshtein 0.18.1 documentation», consultato 22 marzo 2022, https://maxbachmann.github.io/Levenshtein/.

distance normalizzata poiché non sempre abbiamo stringhe della stessa lunghezza; essa consente di calcolare adeguatamente il peso degli errori rispetto alla dimensione delle stringhe confrontate[24].

La *Levenshtein distance* normalizzata è, infatti, uguale a:

$$ratio = \frac{lensum - ldist}{lensum} \tag{2}$$

in questo caso, a differenza del metodo *.ratio* della libreria *fuzzy*, *ldist* è la distanza di Levenshtein pesata cioè ad ogni cambiamento (*edit)* viene attribuito peso 2 se si tratta di una sostituzione e 1 nel caso di rimozioni e aggiunte.

Si è , dunque, ipotizzato che si tratti dello stesso sottotitolo se il valore della distanza di Levenshtein normalizzata è maggiore di 0.50.

Per decretare se vi è una corrispondenza tra la frase annotata manualmente e quella estratta in maniera automatica si è scelto di considerare anche la metrica W*ord Error Rate* (WER).

WER è una metrica che deriva dalla distanza di Levenshtein con la differenza che essa prende in considerazione le sostituzioni, aggiunte o rimozioni a livello di parola e non di singolo carattere.

Essa è uguale al numero di parole diverse diviso il numero totale di parole:

$$WER = \frac{S + D + I}{N} \tag{3}$$

in cui S corrisponde al numero di sostituzioni, D al numero di rimozioni, I al numero di aggiunte e N al numero totale di parole nella frase.

Da questo valore è possibile calcolare anche l'accuratezza intesa come il numero di parole uguali tra le due frasi diviso il numero totale di parole della frase di riferimento.

$$WAcc = 1 - WER = \frac{N - S - D - I}{N} \tag{4}$$

[24] A. Marzal e E. Vidal, «Computation of Normalized Edit Distance and Applications», *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15, n. 9 (settembre 1993): 926–32, https://doi.org/10.1109/34.232078.

2.3.3. Metriche di valutazione

Per valutare l'accuratezza del modello nel riconoscere ed estrarre i sottotitoli si è scelto di calcolare, da un lato, l'accuratezza nell'individuare la presenza di un sottotitolo e, quindi, di segmentare il film in clip sulla base del cambiamento dei sottotitoli; dall'altro, se il modello individua correttamente la presenza del sottotitolo, si valuta quanto sia in grado di riconoscere i caratteri impressi nei video.

Per calcolare quanto il modello sia in grado di riconoscere la presenza di un sottotitolo sono state usate le metriche di precisione (*precision*), richiamo (*recall*) e F1 score.

L'informazione da valutare è, in questo primo caso, se il modello abbia riconosciuto la presenza o meno di un sottotitolo: nel caso in cui il modello ha individuato la presenza di un sottotitolo incremento il calcolo dei veri positivi (*TP*) mentre se il modello riconosce un sottotitolo che, in realtà, non è presente nel *gold* allora incremento il valore dei falsi positivi (*FP*). Il modello potrebbe, inoltre, non riconoscere un sottotitolo e, in questo caso, si incrementa il valore dei falsi negativi (*FN*).

La *precisione* fornisce, dunque, l'informazione su quante predizioni del modello sono, effettivamente, corrette ed è, infatti, uguale al rapporto tra il numero delle predizioni corrette sul numero di volte che il modello lo prevede:

$$precision = \frac{TP}{TP + FP} \tag{5}$$

Il *richiamo*, invece, corrisponde a quanti eventi sono stati correttamente individuati dal modello ed è data dal rapporto tra le previsioni corrette per una classe sul totale dei casi in cui si verifica:

$$recall = \frac{TP}{TP + FN} \tag{6}$$

La precisione può essere, dunque, vista come una misura di esattezza mentre il recupero come una misura di completezza.

L'F1 score è una metrica che tiene in considerazione *precision e recall*, di fatti, viene calcolata come la media armonica di precisione e recupero e viene usata per misurare l'accuratezza di un modello:

$$F_1 = 2 * \frac{p * r}{p + r}$$

(7)

Per valutare quanto, invece, il modello sia accurato nel riconoscere i singoli caratteri si è usata come metrica la distanza di Levenshtein poiché consente di calcolare quanto siano simili due frasi e, quindi, permette di valutare quanto il modello abbia riconosciuto correttamente ogni carattere. Questa metrica è stata usata solo nei casi in cui è stata individuata una corrispondenza tra le due frasi cioè quando il modello riconosce correttamente la presenza di un sottotitolo, di fatti il valore medio dell'accuratezza è dato dalla somma dei valori di accuratezza dei sottotitoli riconosciuti diviso il numero di sottotitoli che il modello ha riconosciuto.

2.3.4. Valutazione e Analisi dei risultati

Nel paragrafo precedente abbiamo evidenziato come, nel corso del lavoro, siano state realizzate, a partire dallo stesso output contenente i testi estratti da ogni fotogramma dei documentari, diverse versioni dell'output di sottotitoli presenti ogni clip.
Ciò che differenza le tre versioni sono le metriche usate per raggruppare i fotogrammi che condividono lo stesso testo estratto automaticamente poiché l'obiettivo è quello di frammentare i video in clip sulla base dei sottotitoli diversi.
Nella prima versione dell'output il controllo di similarità tra i sottotitoli viene effettuato in un range di cinque fotogrammi mentre nella seconda versione c'è un ulteriore controllo non solo con la frase precedente ma anche con le due precedenti. Nella terza versione non viene effettuato il controllo in un range di cinque frasi ma ogni frase viene confrontata solo con la frase precedente e, se vi è una corrispondenza, viene presa quella più verosimile; nel caso contrario vengono considerati come due sottotitoli distinti.
L'obiettivo del presente lavoro di tesi è stato, dunque, quello di valutare queste varianti per individuare quale ottiene le migliori prestazioni; per farlo è stata eseguita la procedura di confronto vista nel paragrafo precedente per l'output di ogni diversa versione.
Dalla valutazione dell'output della prima versione si è osservato come il modello quando individua la presenza di un sottotitolo abbia una accuratezza data *dall'edit distance* pari a 0.89 e questo consente di ipotizzare che il modello riconosce abbastanza bene i singoli caratteri del testo.

Per valutare come il modello lavori nell'individuare la presenza o meno di un sottotitolo si, invece, è calcolata la *precisione* che ha un valore di 0.9; questo permette di sostenere che il modello ha una buona esattezza poiché quando rileva un'informazione testuale essa risulta essere, spesso, un sottotitolo.
Il valore che, invece, non è soddisfacente riguarda il *richiamo* che fornisce un'informazione riguardante la completezza del modello; esso, nella prima versione, è 0.64 e ciò significa che il modello non sempre riesce a rilevare la presenza di un sottotitolo.
Questo lo si può notare dall'osservazione dell'output poiché molto spesso alcuni frammenti non vengono proprio rilevati e questo può essere determinato dal fatto che l'uso dell'euristica di cinque range poteva non essere sufficiente.
La seconda versione prevede, infatti, un confronto delle frasi estratte dal modello sia con la frase precedente ma anche con le due precedenti.
Questo cambiamento ha permesso di avere un maggiore valore di accuratezza 0.93 poiché effettuando un confronto tra due frasi e mantenendo l'euristica di prendere quella più verosimile questo potrebbe aver portato a prendere in considerazione frasi prima trascurate che abbiano un'accuratezza maggiore.
Nel caso della metrica di *precision* notiamo la perdita di qualche punto ottenendo un valore di 0.87 e questo potrebbe significare che il controllo con le due frasi precedenti possa aver portato a rilevare e mantenere rumore.
La metrica di *recall* risulta essere leggermente maggiore 0.67 evidenziando come questo cambiamento possa portare a rilevare alcuni sottotitoli prima trascurati.
La terza versione è stata realizzata per ottenere miglioramenti nel rilevamento di tutti i sottotitoli e, infatti, si è ipotizzato di non effettuare il controllo in un range di cinque frasi ma di confrontare ogni frase solo con la precedente e, se vi è una corrispondenza, prendere quella più verosimile; nel caso contrario di considerali come due sottotitoli distinti e passare ai successivi.
Questo cambiamento ha portato, effettivamente, ad un miglioramento della metrica di *recall* ottenendo un valore di 0.7 ma allo stesso tempo c'è una diminuzione significativa della *precision* poiché il mancato confronto tra più frasi comporta la presenza di molto rumore. Dall'osservazione dell'output notiamo, infatti, come questa versione prevede la presenza di molto rumore ma dall'altro canto notiamo che vengono prese in considerazione frasi che nelle due versioni precedenti erano trascurate e che

nel caso in cui il modello rileva il sottotitolo esso risulta essere molto accurato (accuratezza data dall'*edit distance* normalizzata: 0.91).

Versione	Accuratezza data *Edit Distance normalizzata*	Precision	Recall	F1-score
Prima versione	0.89	0.90	0.64	0.75
Seconda versione	0.93	0.87	0.67	0,76
Terza versione	0.91	0.40	0.70	0,51

Tabella 1: Valori di accuratezza della prima, seconda e terza versione calcolati mediante l'edit distance normalizzata e le metriche di precision, recall e F1-score.

Dai valori ottenuti si è scelto di considerare i risultati dati dalla seconda versione per le fasi successive di estrazione di informazioni poiché essa presenta il valore più alto sia per il riconoscimento dei singoli caratteri nel caso in cui riesce a rilevare sottotitoli sia per la metrica F1 score che descrive l'accuratezza del modello.

Nella prima versione abbiamo, in realtà, un valore leggermente più alto di *precision* questo indica come il modello individui solo i sottotitoli ma allo stesso tempo ne trascura parecchi.

Dal momento che l'obiettivo ultimo del progetto è quello di estrarre le informazioni testuali per consentire la ricerca, mediante parole chiave, di frammenti risulta essere più importante che il modello sia completo e individui il maggior numero di sottotitoli effettivamente presenti.

Da altro canto si è deciso di non considerare la terza versione poiché il leggero miglioramento della *recall*, rispetto alla seconda versione, comporta, tuttavia, un significativo peggioramento della metrica di *precision.*

Queste considerazioni ci hanno spinto a scegliere l'output della seconda versione per le fasi successive di estrazione di informazione testuale.

Per le fasi successive di analisi morfo-sintattica e semantica sono stati utilizzati gli output del modello *Optical Character Recognition* (OCR) dati dalla seconda versione

che fornisce un file .JSON contenenti i dizionari aventi i sottotitoli estratti automaticamente, le informazione temporali, il tipo processore usato e l'identificativo del film e del singolo video clip.

```
[
    {
        "VideoId": "1293",
        "adminInfo": {
            "processors": "transcript",
            "build_time": "2022-02-24 11:45:51.310689"
        },
        "duration": "00:01:52.404",
        "manually_modified": false,
        "start": "00:00:00.000",
        "extracted_info": "No mix audio "
    },
    {
        "VideoId": "1293",
        "adminInfo": {
            "processors": "transcript",
            "build_time": "2022-02-24 11:45:51.310849"
        },
        "duration": "00:00:29.947",
        "manually_modified": false,
        "start": "00:02:10.756",
        "extracted_info": "Way to go, | like you like that! "
    },
    {
        "VideoId": "1293",
        "adminInfo": {
            "processors": "transcript",
            "build_time": "2022-02-24 11:45:51.310907"
        },
        "duration": "00:00:18.268",
        "manually_modified": false,
        "start": "00:01:52.446",
        "extracted_info": "I be with vou in a minute. you have to sign some documents "
    },
```


Figura 21: Esempio dei primi elementi del file .JSON del film "Loro di Napoli" con ID 1293

I sottotitoli presenti nei file .JSON di ognuno dei quindici film presi in esame nel corso del progetto sono stati l'input per l'estrazione di informazioni morfosintattiche e semantiche.

3. Estrazione di informazioni linguistiche

3.1. Annotazione linguistica

Per estrarre l'informazione linguistica dal testo è importante, in primo luogo, soffermarsi sulla fase di annotazione linguistica di un testo.

L'annotazione linguistica[25] consiste nella codifica esplicita di informazione linguistica associata al dato testuale; essa consente, dunque, di rendere esplicita la struttura implicita dei dati testuali.

L'annotazione, generalmente, avviene in relazione ai tradizionali livelli di analisi linguistica: morfologica, sintassi, semantica e pragmatica.

Essa è, dunque, un processo incrementale che rispecchia i diversi livelli di analisi linguistica; l'output di ogni livello costituisce l'input per il livello successivo ma allo stesso tempo ogni output dei livelli intermedi può essere usato per l'estrazione di informazione.

Ogni livello è, infatti, caratterizzato da una propria autonomia ma allo stesso tempo aggiunge informazione all'insieme di informazioni introdotte ai livelli precedenti. Questo comporta che ogni categoria, per ciascun livello, deve essere assegnata sulla base dell'informazione che si ha sino a quel livello di analisi.

3.2. I livelli di annotazione linguistica

3.2.1. Sentence Splitting

Il primo passo di annotazione linguistica è il *Sentence Splitting* che prevede la suddivisione del testo in periodi (s*entence*). Questa fase è fondamentale poiché le frasi costituiscono il punto di partenza per i livelli successivi.

I criteri di segmentazioni in frasi possono cambiare a seconda del genere testuale. In fase di annotazione manuale si è scelto di mantenere la suddivisione in sottotitoli cioè ogni sottotitolo viene considerato come una frase diversa. Questa decisione è stata presa poiché, talvolta, i sottotitoli non terminano con i segni di punteggiatura che,

[25] Alessandro Lenci, Simonetta Montemagni, Vito Pirelli, «*Testo e computer. Elementi di linguistica computazionale*>>. Carocci, Roma, 2005, pp. 211-219

inoltre, possono essere riconosciuti quando invece non sono realmente presenti poiché caratteri facilmente sensibili a rumore.

3.2.2. Tokenizzazione

La fase successiva di annotazione linguistica prevede la *tokenizzazione*[26] ossia il processo di segmentazione del testo in token.
La definizione di token non è banale in quanto non possiamo limitarci a considerarli come un sequenza di caratteri delimitata da spazi poiché vi sono casi, per esempio *don't,* per i quali anche se non vi sono spazi all'interno si tratta di due token distinti ma anche casi di sequenza di caratteri separati da spazi che corrispondono ad un unico token.
La nozione di token è, inoltre, distinta da quella di parola poiché la tokenizzazione non si basa su criteri morfologici, sintattici o semantici che, invece, definiscono la parola.
I token rappresentano le unità di base per i successivi livelli di elaborazione, motivo per cui la loro identificazione deve essere guidata da criteri strettamente linguistici.
La corretta tokenizzazione del testo richiede tre passi fondamentali: stabilire con precisione quali siano le unità linguistiche atomiche di interesse per definire i tipi di token in cui il testo deve essere segmentato, individuare la metodologia e i criteri più opportuni per l'identificazione dei token e le necessarie operazioni di elaborazione del testo, esprimere i criteri e le trasformazioni necessarie in un linguaggio formale che sia traducibile in un programma eseguibile dal computer.
La tokenizzazione è, dunque, un processo dipendente dalla lingua, dalla specificità del sotto linguaggio a cui appartiene il testo e dagli scopi dell'elaborazione computazionale.

3.2.3. Annotazione morfo-sintattica

L'*annotazione morfologica* prevede l'assegnazione ad ogni parola (o *token*) del testo dell'informazione relativa alle possibili categorie grammaticali (o POS dall'inglese *part of speech*) integrata da ulteriori specificazioni morfologiche e dal lemma che la parola assume nel contesto specifico.

[26] Ivi. pp. 102-110

L’annotazione morfologica è, infatti, spesso combinata con l’annotazione per lemma (*lemmatizzazione*) che consiste nel ricondurre ogni *token* al lemma corrispondente.
Questi due livelli di annotazione sono differenti ma strettamente correlati poiché l’identificazione del lemma presuppone la classificazione grammaticale della parola nel contesto specifico.
La variazione dipende, dunque, dalla definizione del concetto di token, dalla teoria linguistica di riferimento e dal dettaglio dell’informazione annotata.
L’annotazione morfo-sintattica è indispensabile per gli altri livelli di annotazione quali l’annotazione sintattica e semantica.

3.2.4. Annotazione sintattica

Con il termine *annotazione sintattica* si fa riferimento alla codifica di informazione relativa all’analisi sintattica delle frasi in un testo.
È possibile individuare due principali approcci alla rappresentazione sintattica: *rappresentazione a costituenti* basata sull’identificazione di costituenti sintattici (come sintagmi nominali, sintagmi verbali ecc.) e sulle loro relazioni e *rappresentazioni a dipendenze* che forniscono una definizione della frase in termini di relazioni binarie di dipendenza tra parole che indicano relazioni grammaticali come soggetto, oggetto, modificatore ecc.
La variazione può, quindi, dipendere sia dall’approccio adottato che da altre scelte come la tipologia di costituenti indentificati.

3.3. Gli schemi di annotazione

Per ciascun livello di analisi esistono molteplici *schemi di annotazione* che differiscono per diverse caratteristiche come, per esempio, il numero o tipologia di categorie grammaticali. Questi elementi di variabilità sono determinati da fattori diversi come gli scopi della ricerca, applicazioni per la quale il corpus è progettato, la teoria linguistica di riferimento, la modalità con cui l’annotazione linguistica viene effettuata, la “granularità” della descrizione linguistica che intendiamo codificare nel testo e le caratteristiche stesse della lingua.
Questo comporta l’esistenza di diversi schemi di annotazione per i quali è possibile, tuttavia, individuare alcuni tratti invarianti.

A questo proposito sono state promosse diverse iniziative per formulare una rappresentazione standard dell'informazione linguistica nel testo, tra le quali menzioniamo il progetto delle *Universal Dependencies* che è stato utilizzato anche all'interno del progetto.

3.3.1. Uno schema di annotazione universale: Universal Dependencies

Il progetto *Universal Dependencies* (UD) nasce nel 2014 con l'obiettivo di creare un unico schema di annotazione linguistica che fosse uno standard di riferimento multilingue e che enfatizzasse le similarità tra le lingue mantenendo, se necessario, le caratteristiche specifiche di ogni lingua[27].

La sfida delle *Universal Dependencies* è quella di mantenere un equilibrio tra i seguenti elementi:

1. soddisfare i livelli di analisi linguistica per le singole lingue;
2. fornire una base linguistica che permetta di far emergere i parallelismi fra le lingue e le similarità fra le famiglie linguistiche;
3. consentire una rapida annotazione da parte degli annotatori umani;
4. deve essere facilmente comprensibile e utilizzabile anche dai non addetti ai lavori;
5. deve essere utilizzabile per il *parsing* con alta accuratezza;
6. deve essere di supporto per i successivi task di comprensione del testo.[28]

Universal Dependencies si basa su una visione lessicale della sintassi che comporta la codifica delle caratteristiche morfologiche come proprietà delle parole senza suddividerle in morfemi; le unità di base dell'annotazione sono, dunque, le parole sintattiche (no parole morfologiche o ortografiche).

Vorrei, in particolare, concentrami sull'annotazione morfosintattica poiché è il livello affrontato nel corso del progetto.

L'annotazione morfologica nello schema UD consiste in tre livelli di rappresentazione:

- *lemma* che corrisponde alla forma base della parola ossia la forma che si trova nei dizionari;

[27] Joakim Nivre et al., «Universal Dependencies v2: An Evergrowing Multilingual Treebank Collection», in *Proceedings of the 12th Language Resources and Evaluation Conference* (LREC 2020, Marseille, France: European Language Resources Association, 2020), 4034–43, https://aclanthology.org/2020.lrec-1.497.

[28] Marie-Catherine de Marneffe et al., «Universal Dependencies», *Computational Linguistics*, 20 maggio 2021, 1–54, https://doi.org/10.1162/coli_a_00402.

- *part of speech tag* corrispondente alla categoria grammaticale della parola;
- insieme di tratti linguistici (*features*) che rappresentano le proprietà lessicali e grammaticali.

I lemmi sono dati dai dizionari mentre le etichette delle *part of speech* e delle *features* sono date da un inventario universale. In seguito (Tabella 2), viene riportata la lista delle *part of speech* che contiene diciassette tag nella seconda versione:

Classe aperta di parole	Classe chiusa di parole	Altro
ADJ: aggettivo	ADP: apposizione	PUNCT: punteggiatura
ADV: avverbio	AUX: verbo ausiliare	SYM: simbolo
INTJ: interiezione	CCONJ: congiunzione coordinante	X: altro
NOUN: nome comune	DET: determinante	
PROPN: nome proprio	NUM: numerale	
VERB: verbo	PART: particella	
	PRON: pronome	
	SCONJ: congiunzione subordinante	

Tabella 2: Etichette di part of speech universali.

Il sito delle Universal Dependencies è stato, di fatti, oggetto di consultazione nella fase di annotazione manuale per la realizzazione del file *gold.*

3.3.2. Formato CoNLL-U

Il progetto *Universal Dependencies* mette a disposizione le risorse linguistiche annotate in formato CoNLL-U ossia una versione rivista del formato CoNLL-X[29].
Ogni anno la *Conference on Computational Natural Language Learning* (CoNLL) presenta un'attività condivisa in cui i partecipanti addestrano e testano diversi sistemi sugli stessi set di dati per poterli confrontare. La decima edizione si è occupata dell'analisi sintattica multilingue (*Multilingual Dependency Parsing*) per la quale si è introdotto un formato unico. Esso prevede tre tipologie di righe: una che contiene l'annotazione di ogni *token* separata con una tabulazione, una riga vuota che definisce i confini tra le frasi e una riga che inizia con il cancelletto (#) e costituisce un commento.
Ogni riga relativa al token contiene 10 colonne contenti informazioni quali:

1. ID: indice del token a partire da 1 per ogni nuova frase
2. FORM: forma della parola o punteggiatura
3. LEMMA: lemma corrispondente alla parola
4. UPOS: categorie grammaticali universali
5. XPOS: categorie grammaticali specifiche della lingua di analisi
6. FEATS: lista dei tratti morfologici dati dall'inventario delle *features* universali
7. HEAD: testa del *token* corrente
8. DEPREL: relazione di dipendenza del token corrente rispetto alla testa
9. DEPS: lista di dipendenze secondarie
10. MISC: altre informazioni di annotazione

[29] Sabine Buchholz e Erwin Marsi, «CoNLL-X Shared Task on Multilingual Dependency Parsing», in *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)* (New York City: Association for Computational Linguistics, 2006), 149–64, https://aclanthology.org/W06-2920.

```
# newdoc id = file_sent_gold.txt
# newpar
# sent_id = 1
# text = - Good morning Antonio! - Hi Arianna.
1   -   -   PUNCT   NFP _   3   punct   _   _
2   Good    good    ADJ JJ  Degree=Pos  3   amod    _   _
3   morning morning NOUN    NN  Number=Sing 0   root    _   _
4   Antonio Antonio PROPN   NNP Number=Sing 3   vocative    _   SpaceAfter=No
5   !   !   PUNCT   .   _   3   punct   _   _
6   -   -   PUNCT   NFP _   3   punct   _   _
7   Hi  hi  INTJ    UH  _   8   discourse   _   _
8   Arianna Arianna PROPN   NNP Number=Sing 3   parataxis   _   SpaceAfter=No
9   .   .   PUNCT   .   _   3   punct   _   SpacesAfter=\r\n

# sent_id = 2
# text = I'll be with you in a minute, you have to sign some documents...
1   I   I   PRON    PRP Case=Nom|Number=Sing|Person=1|PronType=Prs  5   nsubj   _   SpaceAfter=No
2   'll will    AUX MD  VerbForm=Fin    5   aux _   _
3   be  be  AUX VB  VerbForm=Inf    5   cop _   _
4   with    with    ADP IN  _   5   case    _   _
5   you you PRON    PRP Case=Acc|Person=2|PronType=Prs  0   root    _   _
6   in  in  ADP IN  _   8   case    _   _
7   a   a   DET DT  Definite=Ind|PronType=Art   8   det _   _
8   minute  minute  NOUN    NN  Number=Sing 5   obl _   SpaceAfter=No
9   ,   ,   PUNCT   ,   _   5   punct   _   _
10  you you PRON    PRP Case=Nom|Person=2|PronType=Prs  11  nsubj   _   _
11  have    have    VERB    VBP Mood=Ind|Tense=Pres|VerbForm=Fin    5   parataxis   _   _
12  to  to  PART    TO  _   13  mark    _   _
13  sign    sign    VERB    VB  VerbForm=Inf    11  xcomp   _   _
14  some    some    DET DT  _   15  det _   _
15  documents   document    NOUN    NNS Number=Plur 13  obj _   SpaceAfter=No
16  ... ... PUNCT   .   _   5   punct   _   SpacesAfter=\r\n
```

Figura 22: Esempio di file CoNLL-U.

Il formato prevede, inoltre, l'uso del trattino basso (_) nel caso in cui manca l'informazione. Nel progetto è stato utilizzato il formato CoNLL-U ma, poiché ci siamo fermati al livello di annotazione morfo-sintattica, le colonne che dovrebbero contenere l'informazione sintattiche presentano il simbolo "_".

3.4. Strumenti e Metodologie

Nel corso del progetto sono state prese in considerazione tre *pipeline* per scegliere quale usare: UDPipe, SpaCy e Stanza.

È stata annotata manualmente una porzione di sottotitoli per valutare quale *pipeline* avesse le performance migliori.

In questa sezione vengono analizzate le tre risorse ponendo l'attenzione, in particolare, sulle caratteristiche delle analisi morfo-sintattiche.

3.4.1. UDPipe

UDPipe[30] è una *pipeline* che esegue la segmentazione delle frasi, la tokenizzazione, la codifica di POS (*part of speech tagging*), la lemmatizzazione e l'analisi delle

[30] Milan Straka, Jan Hajic, e Jana Strakova, «UDPipe: Trainable Pipeline for Processing CoNLL-U Files Performing Tokenization, Morphological Analysis, POS Tagging and Parsing», s.d., 8.

dipendenze (*dependency parsing*) usando le treebank delle *Universal Dependencies* (seconda versione).

Come abbiamo detto nella sezione precedente, l'obiettivo delle UD è sviluppare treebank per l'annotazione morfologica e sintattica multilingue e quello di UDPipe è di fornire uno strumento semplice che possa essere facilmente addestrato usando un file formato CoNLL-U.

Abbiamo visto come in un file CoNLL-U il testo è strutturato su diversi livelli: è costituito da paragrafi composti da frasi che sono sequenze di token. Un token, generalmente, corrisponde a una parola ma un singolo token può essere composta da diverse parole sintattiche. La segmentazione delle frasi e la tokenizzazione vengono eseguite congiuntamente utilizzando una rete bidirezionale che predice per ogni carattere se è l'ultimo in una frase, l'ultimo o no in un token. Gli spazi non sono consentiti in un token e la rete non predice la fine di un token se non vi è uno spazio.

Per quel che riguarda l'analisi morfo-sintattica il formato CoNLL-U contiene sezioni per il lemma, etichette che definiscono le parti del discorso universali (UPOS) e specifiche della lingua (XPOS) e la lista di caratteristiche morfologiche; *UDPipe* è in grado di riempire questi campi.

UDPipe usa due modelli uno che disambigua tutti i campi morfologici disponibili (POS tag) e l'altro che esegue la lemmatizzazione (un *lemmatizer*) perché la combinazione di due *tagger* porta a prestazioni migliori.

Nel presente progetto è stata utilizzata la catena di annotazione linguistica UDPipe 1.2.0 e il modello delle Universal Dependencies 2.5 addestrato sulla treebank *English Web Treebank* (EWT).

3.4.2. SpaCy

La *pipeline* di spaCy[31] prevede diversi componenti come: *tokenizer*, *tagger*, *parser* e un *entity recognizer*. SpaCy per prima cosa divide il testo in token producendo una oggetto Doc il quale è poi processato in diverse fasi:

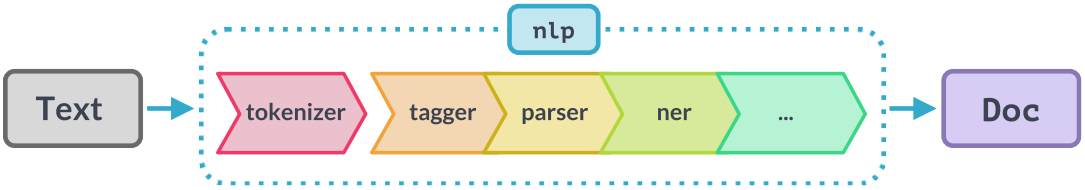


Figura 23: Pipeline di spaCy.

Si tratta di un processo incrementale poiché ogni componente restituisce l'oggetto Doc che poi diviene l'input del componente seguente.

Il *tokenizer* prende in input testo e restituisce l'oggetto Doc che contiene la divisione in token e che rappresenterà l'input per i livelli di analisi successivi.

L'elemento che contraddistingue il tokenizzatore di spaCy è l'uso di regole, difatti l'idea alla base è la divisione del testo grezzo seguendo gli spazi per poi processarlo da sinistra a destra e per ogni sottostringa esegue due controlli: se si tratta di un'eccezione e/o se si tratta di un prefisso, suffisso o infisso.

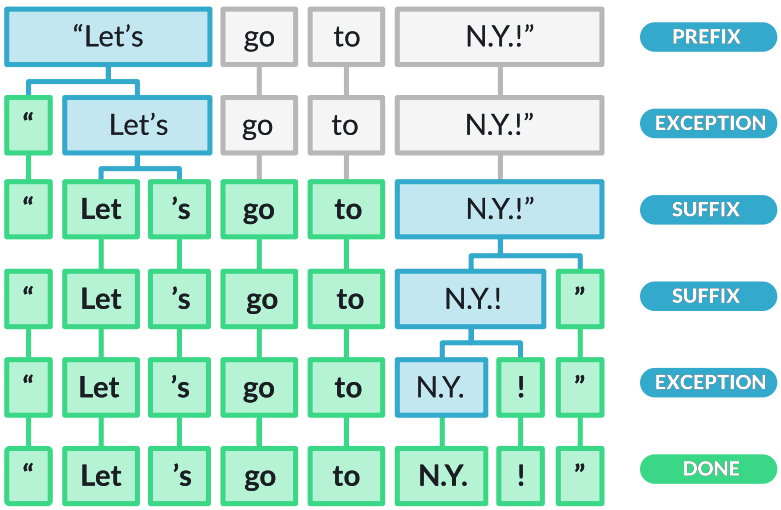


Figura 24: Esempio della tokenizzazione in spaCy basata su regole.

[31] «SpaCy · Industrial-Strength Natural Language Processing in Python», consultato 29 marzo 2022, https://spacy.io/.

La tokenizzazione in spaCy si basa, dunque, su regole e non su un apprendimento a partire dai dati generando un oggetto Doc che può essere analizzato a livello di annotazione morfosintattica e sintattica.
La *pipeline* addestrata sui dati e i modelli statistici consentono di fare predizioni su quali etichette possono essere applicate in uno specifico contesto.
Un componente addestrato include dati binari che sono prodotti mostrando un sistema di esempi per fare predizioni che generalizzano tra le lingue; per esempio, una parola che segue "the" in inglese sarà, molto probabilmente, un sostantivo.
L'annotazione morfo-sintattica viene, dunque, eseguita da componenti integrati di spaCy e, dunque, seguendo lo schema universale delle *Universal Dependencies* sono stati usati:

UD	ATTRIBUTI SpaCy
LEMMA	lemma_
UPOS	pos_
XPOS	tag_

Tabella 3: Etichette universali per l'annotazione morfosintattica e corrispondenti attributi di spaCy per ottenerli.

Nel progetto è stato utilizzato il modello en_core_web_lg il quale è addestrato su OntoNotes 5.0, ClearNLP Constituent-to-Dependency Conversion (Emory University), WordNet 3.0 e GloVe Common Crawl.

3.4.3. Stanza

Stanza[32] è un pacchetto (*package*) introdotto da *Stanford NLP Group* per l'elaborazione del linguaggio naturale.
Esso dispone di strumenti che possono essere usati in una *pipeline* per l'analisi del testo e che includono tokenizzazione (TOKENIZE), espansione dei token multi-parole (MWT), lemmatizzazione (LEMMA), etichettatura di *part of speech* (POS) e caratteristiche (*features*) morfologiche, analisi delle dipendenze (DEPPARSE) e riconoscimento delle entità nominali (NER).

[32] Peng Qi et al., «Stanza: A Python Natural Language Processing Toolkit for Many Human Languages», *arXiv:2003.07082 [cs]*, 23 aprile 2020, http://arxiv.org/abs/2003.07082.

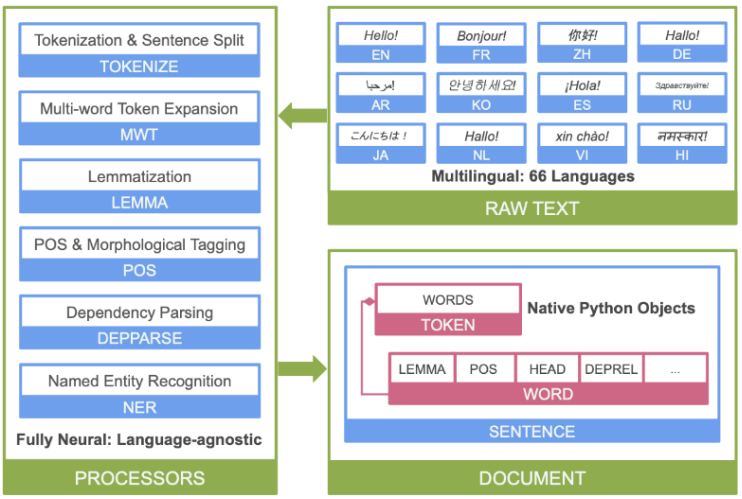


Figura 25: Pipeline di Stanza che prende in input un testo multilingue e restituisce in output le annotazioni come oggetti nativi di Python

Stanza è stato addestrato su un totale 112 dataset che comprendono le treebank delle *Universal Dependencies* e altri corpora multilingue e mostra come la stessa architettura neurale si generalizzi bene e ottiene prestazione competitive su tutte le lingue testate.

Stanza è stato introdotto, in letteratura, per poter superare i limiti degli altri *toolkit*, come *UDPipe* e *spaCy*, usati per l'analisi del linguaggio naturale e lo fa adottando:

- una pipeline *fully neural* che prende in input un testo grezzo e produce annotazioni che includono tokenizzazione, espansione multi-word, lemmatizzazione, etichettatura di part of speech e features morfologiche, analisi delle dipendenza e *named entity recognition*.
- un design indipendente dalla lingua e guidato dai dati
- una *pipeline* che si adatti bene a diversi generi ottenendo prestazioni all'avanguardia e competitivi in ogni fase della *pipeline*.

La *pipeline* neurale di Stanza consiste di modelli che vanno dalla tokenizzazione di testi grezzi all'esecuzione di analisi sintattiche su intere frasi.

Tutti i componenti sono progettati con l'elaborazione di molti linguaggi con scelte progettuali di alto livello che catturano fenomeni comuni a molti linguaggi e modelli basati sui dati che apprendono la differenza tra i linguaggi a partire dai dati.

L'implementazione dei componenti di Stanza è altamente modulare e riutilizza le architetture dei modelli di base, quando possibile, per la compattezza.

La tokenizzazione e la divisione in frasi (*sentence splitting*) da testo grezzo sono combinati in un modulo singolo in quanto è elaborato come un problema di

etichettatura sulle sequenze di caratteri in cui il modello predice se un dato carattere è la fine di un token, di una frase o la fine di un token multi-parola.
Stanza assegna un'etichetta relativa alla categoria grammaticale (POS) ad ogni parola in una frase e analizza le sue features morfologiche universali. Per predirli è stata adottata un rete neurale *long short memory* bidirezionale (Bi-LSTM).
Stanza lemmatizza ogni parola per ricondurla al suo lemma. Il *lemmatizer* di Stanza è implementato come la combinazione di un *lemmatizer* basato su un dizionario e un lemmatizer neurale seq2seq.
È possibile accedere ai risultati delle annotazioni come oggetti nativi Python per consentire per una post-elaborazione flessibile.
Dopo che tutti i processori sono stati eseguiti viene restituita un'istanza di Doc che memorizza tutti i risultati dell'annotazione: *sentences, tokens e word.*

```
# print the text and POS of all words
for sentence in doc.sentences:
    for word in sentence.words:
        print(word.text, word.pos)

# print all entities in the document
print(doc.entities)
```

Figura 26: Esempio di codice per ottenere i risultati dell'annotazione con Stanza.

3.5. Realizzazione del gold standard

Per scegliere quale *pipeline* utilizzare sono state testate le prestazioni di ognuna di esse sui dati annotati manualmente.
Sono stati annotati manualmente, sino a livello morfosintattico, i sottotitoli rilevati manualmente mediante la visione dei primi minuti di ogni film e usati nella fase precedente del lavoro.
Il processo di realizzazione del file *gold* ha previsto una prima annotazione automatica, solo a livello di tokenizzazione, del corpus utilizzando la catena di annotazione UDPipe.
La tokenizzazione è stata rivista manualmente e questo ha rappresentato il punto di partenza per l'annotazione automatica di ogni diversa *pipeline*.

Per poter confrontare i risultati è necessario, infatti, avere la stessa divisione sia in frasi che in *token;* questo è stato possibile poiché ogni *tool* offre la possibilità di utilizzare un testo *pre-tokenizzato.*

L'input in tutti e tre i casi consiste, dunque, nel file formato CoNLL-U che contiene la divisione in token.

Il passo successivo è quello di "riempire" gli altri campi relativi all'annotazione morfosintattica e, in particolare, i campi di ID, FORM, LEMMA, UPOS e XPOS.

È stato realizzato il file *gold* contente queste informazioni, riviste manualmente, consultando il sito delle Universal Dependencies.

Anche in questo si è partiti da una annotazione automatica con la catena di annotazione UDPipe e si è proceduto ad una rivisitazione manuale.

Le principali differenze hanno riguardato i verbi con funzione di ausiliare o di verbo proprio; si è notato, inoltre, che i token che presentano una lettera maiuscola in alcuni casi sono considerati come nomi propri così come i token che terminano con la 's' e seguono sostantivi sono, talvolta, etichettati come verbi.

Nella sezione successiva sono analizzate, più nel dettaglio, le differenze tra il modo in cui si è deciso di annotare manualmente e l'annotazione automatica data dalle tre pipeline considerate.

3.6. Valutazione e Analisi dei risultati

Nel presente paragrafo ci soffermiamo sulle differenze tra il *gold standard* annotato manualmente sino a livello morfo-sintattico e l'output dato dalle tre *pipeline*.

Per quel che riguarda la lemmatizzazione, non vi sono molte differenze tra i lemmi annotati automaticamente e quelli annotati manualmente; questo si riflette anche nella valutazione in cui emerge come tutte e tre le *pipeline* ottengono valori alti nell'attribuzione del lemma corretto.

Maggiori differenze si osservano per l'attribuzione della categoria grammaticale sia universale che specifica per la lingua inglese.

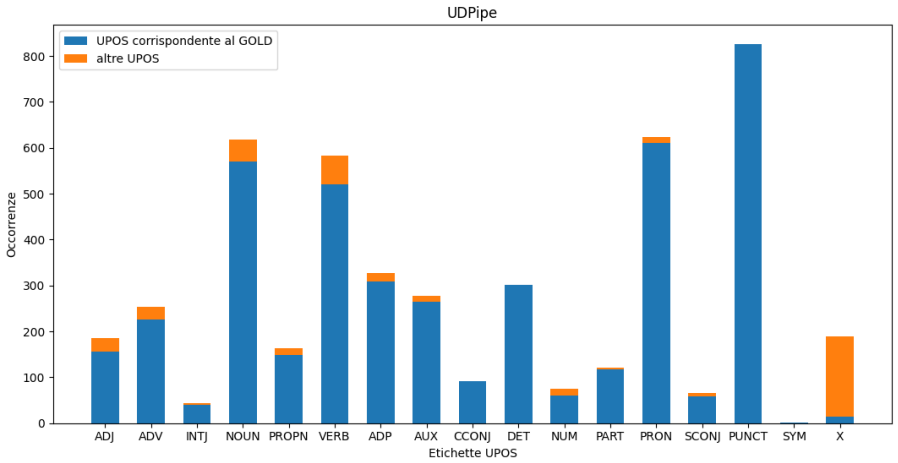


Figura 27: Numero di occorrenze in cui la catena UDPipe individua correttamente l'etichetta UPOS confrontato con il numero di volte che individua altro.

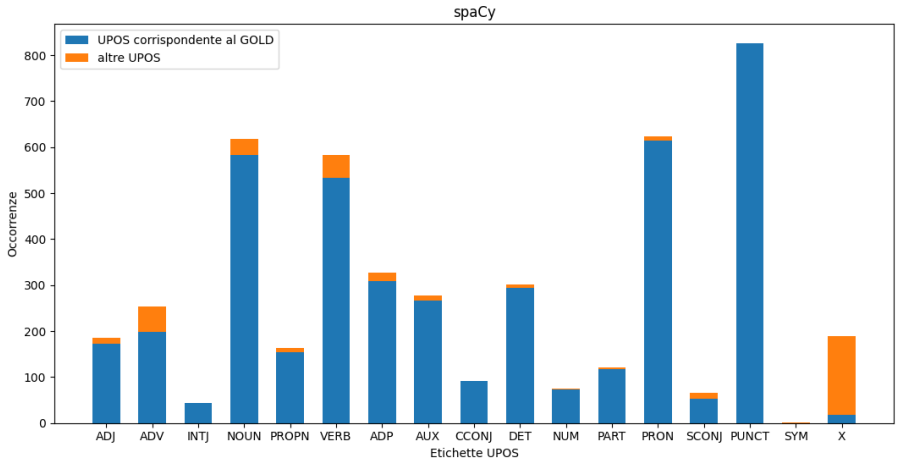


Figura 28: Numero di occorrenze in cui la pipeline di spaCy individua correttamente l'etichetta UPOS confrontato con il numero di volte che individua altro.

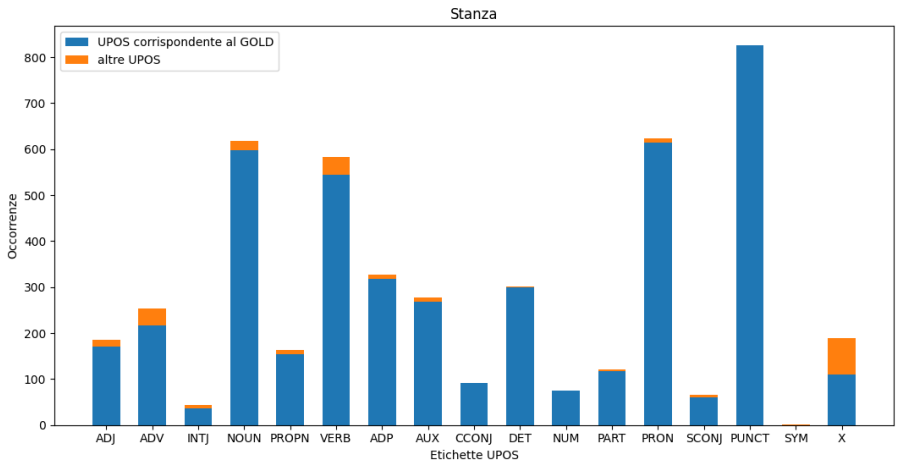


Figura 29: Numero di occorrenze in cui la pipeline di Stanza individua correttamente l'etichetta UPOS confrontato con il numero di volte che individua altro.

Dai grafici emerge, dunque, come alcune difficoltà nel riconoscere la categoria grammaticale corretta si hanno nel caso dei verbi, nomi e avverbi; questo si verifica per tutte e tre le *pipeline*.

Si pone, dunque, l'attenzione su queste categorie grammaticali per comprendere cosa possa determinare confusione nell'attribuire la corretta categoria.

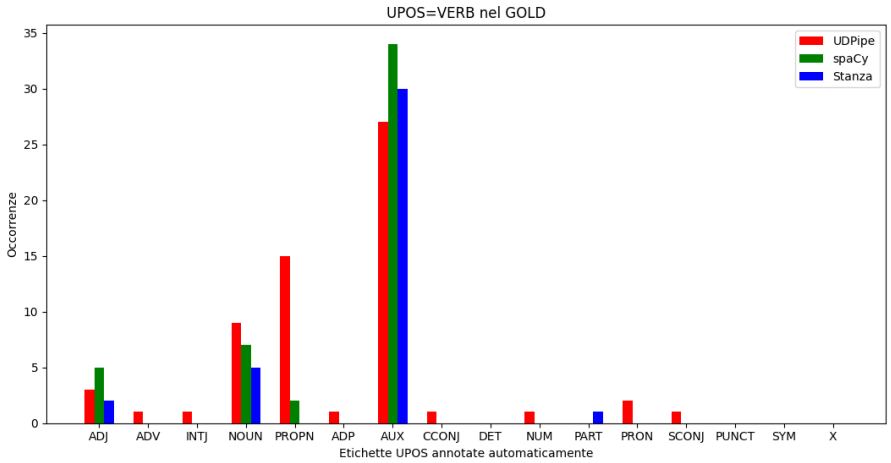


Figura 30: Confronto tra le tre pipeline nel confondere il part of speech VERB *con altre etichette.*

Nel caso del *part of speech* 'VERB' viene spesso identificato come ausiliare e questo è facilmente comprensibile poiché la distinzione tra verbo e ausiliari in verbi come "be" e "have" non sempre è chiara. Notiamo, inoltre, come UDPipe li identifica, talvolta, con l'etichetta 'PROPN' e questo probabilmente si verifica quando il token presenta la lettera maiuscola. L'attribuzione dell'etichetta 'NOUN' si verifica, forse, quando il token termina con la 's'.

Quest'ultimo caso comporta anche il problema contrario ossia l'attribuzione dell'etichetta "VERB" quando , invece, ci troviamo dinanzi alla categoria grammaticale 'NOUN'; come emerge nel grafico seguente.

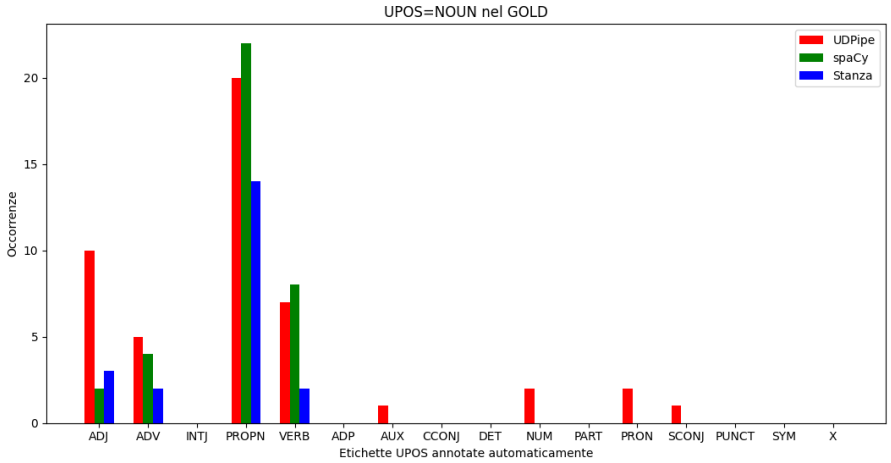


Figura 31: Confronto tra le tre pipeline nel confondere il part of speech NOUN con altre etichette.

Vorrei, inoltre, porre l'attenzione sul etichetta universale 'X' e, in particolare, sul tag specifico (XPOS) 'FW' usato per indicare che si tratti di un token di lingua straniera.

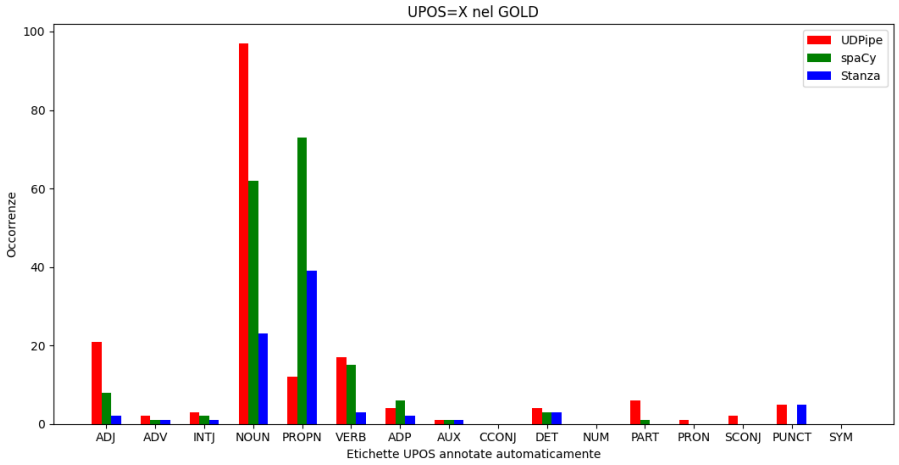


Figura 32: Confronto tra le tre pipeline nel confondere il part of speech VERB con altre etichette.

Dal grafico emerge come la confusione nell'attrbuire l'etichetta 'X' si verifica soprattutto nel caso di UDPipe e spaCy dove il tagger si basa su regole mentre diviene meno problematico nel caso di Stanza che ha il vantaggio di apprendere direttamente dai dati.

Già da questa prima analisi emerge come, talvolta, Stanza funzioni meglio ma per decidere quale *pipeline* usare si è proseguito con la valutazione.

La valutazione è stata eseguita utilizzando lo script fornito durante la *Conference on Computational Natural Language Learning* (CoNLL) 2020[33]. Nella tabella seguente sono riportati i valori di F_1 score relativo a LEMMA, UPOS, XPOS ottenuto con le *pipeline* UDPipe, spaCy, Stanza. Il valore F1 è una misura dell'accuratezza che tiene in considerazione sia della *precision* che del *recall* dove la *precision* è il numero di veri positivi diviso il numero di tutti i risultati positivi mentre il *recall* è il numero di veri positivi diviso il numero di tutti i test che sarebbero dovuti risultare positivi:

$$F1 = 2 * \frac{p * r}{p + r} \tag{8}$$

Pipeline	LEMMA	UPOS	XPOS
UDPipe	96.84	90.89	91.31
spaCy	96.71	92.66	92.54
Stanza	97.95	94.81	95.4

Tabella 4: Valori di F1-score per LEMMA, UPOS E XPOS che indica l'accuratezza dei modelli UDPipe, spaCy e Stanza.

Si è scelto, dunque, di prendere in considerazione la *pipeline* Stanza per estrarre le informazioni morfo-sintattiche dai sottotitoli estratti automaticamente.

Le informazioni morfosintattiche estratte sono state, dunque, aggiunte nei file .JSON di ognuno dei quindici film analizzati e, in particolare, ad ogni dizionario che corrisponde a una diversa clip viene aggiunto un elemento che ha come chiave "*extra_info"* e come valore una lista di dizionari contenenti il token estratto, il lemma e la categoria grammaticale.

[33] «Task Description and Evaluation», consultato 30 marzo 2022, https://universaldependencies.org/iwpt20/task_and_evaluation.html.

Figura 33: Esempio di una lista di dizionari che contiene le informazioni morfosintattiche per ogni token.

4. Estrazione delle entità nominali

4.1. Named Entity Recognition (NER)

Il termine N*amed Entity Recognition* (NER) viene usato per indicare una sottoclasse di task di estrazione di informazioni (*Information Extraction*) che si occupa di identificare le entità all'interno di un testo e associarle alle corrispondenti categorie semantiche delle entità nominali.

In primo luogo, è necessario chiarire che cosa si intenda per *entità nominali*, di fatti non c'è un accordo sulla definizione di *named entity* all'interno della comunità di ricerca[34] poiché da un lato il termine *named* sembra restringere il campo solo ad entità che possono essere definite nomi propri; dall'altro, c'è un accordo sull'includere anche espressioni temporali e alcune espressioni numeriche come quantità monetarie e altri tipi di unità di misura.

Durante la *Conference on Computational Natural Language Learning* del 2003 (CoNLL 2003) le unità nominali sono state definite come: *"Named entities are phrases that contain the names of persons, organizations, locations, times, and quantities."* [35]

Anche questa definizione, tuttavia, risulta essere limitante, motivo per cui si è stabilita l'idea di un'entità nominale come entità che è un *designatore rigido*[36] *(rigid designator*) ossia un termine che designa lo stesso elemento, relativamente, a tutti i mondi possibili e mai un elemento diverso.

Il *task* di riconoscimento dell'entità nominale può essere, dunque, visto come un problema di etichettatura di sequenze che prevede l'identificazione di un intervallo di *token* e la classificazione se si tratti di un'entità nominale o meno e, nel caso in cui lo sia, l'attribuzione di un'etichetta che identifica il tipo di entità nominale.

Per fare questa classificazione i primi sistemi utilizzavano algoritmi basati su regole definendo, quindi, manualmente quali fossero le features che caratterizzavano

[34] «Appendix 5: Named Entities: Current Definitions», in *Named Entities for Computational Linguistics* (John Wiley & Sons, Ltd, 2016), 153–56, https://doi.org/10.1002/9781119268567.app5.

[35] Erik F. Tjong Kim Sang e Fien De Meulder, «Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition», *arXiv:cs/0306050*, 12 giugno 2003, http://arxiv.org/abs/cs/0306050.

[36] Joseph LaPorte, «Rigid Designators», in *The Stanford Encyclopedia of Philosophy*, a c. di Edward N. Zalta, Spring 2022 (Metaphysics Research Lab, Stanford University, 2022), https://plato.stanford.edu/archives/spr2022/entries/rigid-designators/.

un'entità nominale; in tempi recenti, invece, si sono utilizzate tecniche di apprendimento automatico (*Machine Learning*) sia *supervisionato* che prevedono l'individuazione delle regole a partire da una collezione di documenti annotati sia *non supervisionato* che non necessitano dati precedentemente annotati ma sono in grado di eseguire una classificazione sulla base dei dati come, per esempio, la similarità di contesto[37].

Il task di riconoscimento e estrazione di entità nominali suscita molto interesse poiché fondamentale per l'estrazione di informazione semantica e negli anni sono stati sviluppati diversi sistemi NER che possono avere come target domini diversi, usare diverse metodologie, lavorare su lingue diverse, rilevare diversi tipi di entità e supportare diversi formati di input e output.

Nel corso del progetto si è scelto di usare gli strumenti valutati e utilizzati nella fase di annotazione morfo-sintattica: *spaCy e Stanza.*

Anche in questo caso sono state confrontate le prestazioni dei due *processor* con un *gold standard.*

4.2. Realizzazione del gold standard

Per la realizzazione di un file *gold standard* il punto di partenza è il testo tokenizzato realizzato nella fase precedente del lavoro e contente 4743 token; ad ogni token viene assegnata l'etichetta dell'entità nominale individuata. Si è scelto di utilizzare lo schema di annotazione BIOES il quale è un metodo di annotazione sofisticato che distingue un'entità singola da un'entità composta di cui individua l'inizio e la fine. L'acronimo BIOES indica i prefissi che vengono usati per designare la posizione del token all'interno dell'entità:

- il prefisso "B-" viene utilizzato per indicare l'inizio dell'entità (Begin);
- il prefisso "I-" indica che il token è interno all'entità (Inside);
- il prefisso "O-" indica che il token non fa parte dell'entità (Outside);
- il prefisso "E-" indica che il token rappresenta la fine dell'entità (End);
- il prefisso "S-" viene usato quando si ha un'entità con un unico token (Single).

[37] David Nadeau e Satoshi Sekine, «A Survey of Named Entity Recognition and Classification», s.d., 20.

Per scegliere quale tipo di entità nominale assegnare sono state impiegate le etichette usate nel *dataset OntoNotes 5.0* [38] che ne include 18:

ETICHETTA NAMED ENTITY	DESCRIZIONE
PERSON	Nomi di persone inclusi personaggi di finzione
NORP	Nazionalità, gruppi religiosi o gruppi politici
FACILITY	Costruzioni, aeroporti, ponti, ecc.
ORGANIZATION	Compagnie, agenzie, istituzioni, ecc.
GPE	Paesi, città, stati
LOCATION	Luoghi non GPE come corso di fiumi, montagne ecc.
PRODUCT	Prodotti che non indicano servizi come veicoli, cibo ecc.
EVENT	Eventi come battaglie, guerre, eventi sportivi ecc.
WORK_OF_ART	Titoli di libri, canzoni ecc.
LAW	Leggi
LANGUAGE	Linguaggi
DATE	Date o periodi assoluti o relativi
TIME	Tempo minore di un giorno

[38] Weischedel, Ralph et al., «OntoNotes Release 5.0» (Linguistic Data Consortium, 16 ottobre 2013), https://doi.org/10.35111/XMHB-2B84.

PERCENT	Percentuali
MONEY	Valori monetari
QUANTITY	Espressioni che indicano unità di misura
ORDINAL	Numeri ordinali come *"primo", "secondo"*
CARDINAL	Numeri cardinali

Tabella 5: Etichette che indicano il tipo di entità nominale usate nel dataset OntoNotes 5.0 con la relativa descrizione.[39]

Nel file *gold,* dunque, ogni riga corrisponde ad un *token* a cui viene assegnata un'etichetta seguendo lo schema di annotazione BIOES. Il *gold standard,* così realizzato, viene usato per valutare le prestazioni nel riconoscere le entità nominali da parte degli *named entity recognizer di* Stanza e spaCy.

4.3. Strumenti e Metodologie

4.3.1. Stanza

Nella sezione 3.4.3 abbiamo visto la *pipeline* di Stanza e, in particolare, i diversi *processor* implementati.

In questo caso ci soffermiamo sul *named entity recognizer*[40] che è implementato per riconoscere le entità nominali presenti in ogni frase di input.

Per *named entity recognition* è stata adottato un *tagger* (*sequence tagger)* basato sulla rappresentazione di stringhe contestualizzate.

Per prima cosa è stato addestrato un modello linguistico (*language model*) a livello di caratteri precedenti e successivi (*forward and backward character-level)*: *Long Short Term Memory* (LSTM).

Al momento dell'etichettatura sono state concatenate le rappresentazioni alla fine di ogni parola sia dal modello linguistico che da *word embeddings* ossia rappresentazioni

[39] Ivi. p. 12
[40] Qi et al., «Stanza».

distribuite delle parole che memorizzano le informazioni, sia semantiche che sintattiche, delle parole partendo da un corpus non annotato e costruendo uno spazio vettoriale in cui i vettori delle parole sono più vicini se le parole occorrono negli stessi contesti linguistici. Il risultato è inserito in un *tagger sequence* Bi-LSTM con un decodificatore basato su *campi casuali condizionali* (CRF dall'inglese C*onditional Random Field*).

Il modello di Stanza per NER è addestrato su dataset di otto lingue diverse; per l'inglese, di *default*, viene usato il *dataset OntoNotes 5.0.*

La *pipeline* di Stanza, quindi, a partire da un intero documento o da singole frasi restituisce l'etichetta dell'entità nominale individuata per ogni token; l'output è fornito mediante il formato BIOES dove ogni riga rappresenta un token con due campi: il primo è la parola stessa e l'altro il tipo di entità nominale.

Per la valutazione è necessario realizzare un file che ha la stessa divisione in token del *gold*, motivo per cui l'input è il file CoNLL-U che contiene la divisione in token rivista manualmente e da cui viene mantenuta la stessa divisione in token.

L'output deve avere lo stesso schema di annotazione usato per il *gold* standard ossia un file in cui ogni riga corrisponde ad un token a cui viene assegnato l'etichetta dell'entità nominale seguendo lo schema BIOES.

4.3.2. SpaCy

In spaCy il riconoscimento delle entità nominali è implementato dal componente *ner* della *pipeline* che molti modelli hanno di default nella *pipeline* di elaborazione.

Il sistema di riconoscimento delle entità nominali usato nei modelli è un sistema statistico che prevede l'assegnazione delle etichette ad intervalli contigui di token comportando, tuttavia, una forte dipendenza dai dati di addestramento.

Il modello utilizzato è "*en_core_web_lg*" il quale è addestrato su OntoNotes 5.0 e, dunque, presenta lo stesso schema di etichette viste all'inizio del paragrafo consentendo il confronto e la valutazione con il *gold standard* realizzato.

Per consentire la valutazione con il *gold standard* è necessario avere, inoltre, la stessa divisione in token, motivo per cui l'input è il file CoNLL-U avente la tokenizzazione *gold.*

È possibile accedere alle entità nominali mediante l'annotazione delle entità dei token usando gli attributi *token.ent_iob* e token.ent_type.

SpaCy utilizza, dunque, come schema di annotazione dell'entità lo schema IOB che risulta essere più restrittivo rispetto allo schema IOBES poiché prende in considerazione solo se il token è:

- "I-" interno all'entità (*inside*)
- "O-" esterno all'entità (*outside*)
- "B-" l'inizio dell'entità (*beginning*)

Questo, tuttavia, non è un problema per la valutazione poiché è stato convertito l'output del modello da uno schema di annotazione IOB a uno BIOES.

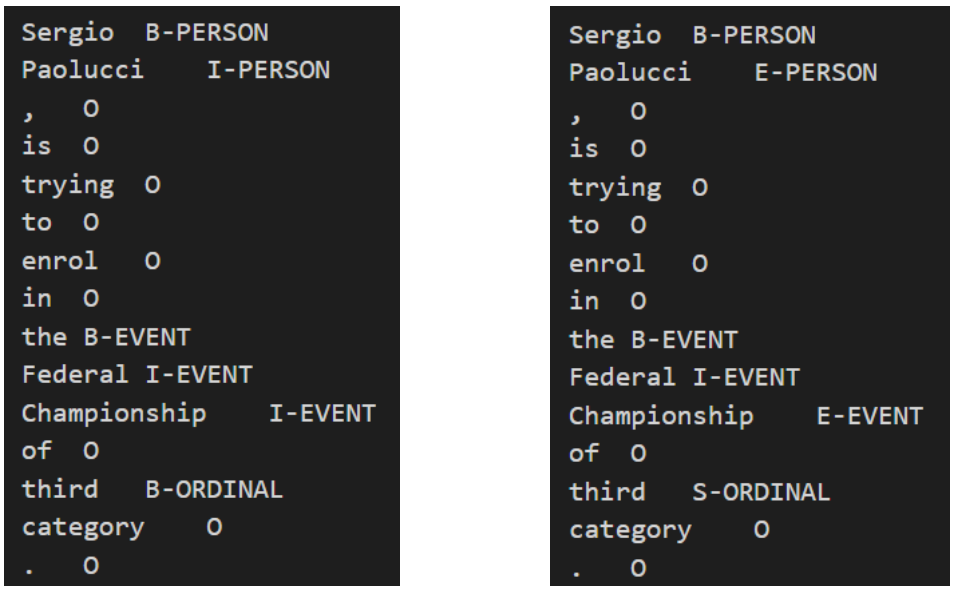


Figura 34: Esempio di annotazione di entità nominale restituita come output da spaCy in formato IOB (a) e convertita in formato IOBES (b).

Viene restituito, anche in questo caso, un file contenente per ogni riga il token e la corrispondente etichetta di entità nominale attribuita da spaCy seguendo lo schema BIOES.

4.4. Valutazione e Analisi dei Risultati

Per consentire la valutazione dei processor di *spaCy e Stanza* per il riconoscimento di entità nominali è necessario confrontare l'output con l'annotazione del *gold standard.* Il file *gold* e l'output dei *processor* devono avere la stessa divisione in token e lo stesso schema di annotazione.

Confrontando l'annotazione del *gold standard* con l'output dei sistemi di *named entity recognition* potrebbero verificarsi tre diversi scenari:

1. Corrispondenza tra l'etichetta *gold* e quella attribuita dal modello (TP);
2. Il sistema ipotizza la presenza di un'entità che in realtà non è presente nel *gold* (FP);
3. Il sistema non riconosce un'entità che è presente nel *gold* (FN).

Considerando solo questi tre scenari abbiamo una valutazione semplice di classificazione che può essere, quindi, misurata in termini di *veri positivi, falsi negativi e falsi positivi* e, successivamente, da cui possiamo calcolare *precisione* (*precision*), *richiamo* (*recall*) e *F1 score* per ogni tipo di entità nominale[41].

In questo modo, tuttavia, non consideriamo altre situazioni come, per esempio, i casi in cui il modello riconosce la presenza di un'entità ma attribuisce una tipologia diversa o casi in cui il modello pur riconoscendo il corretto tipo di entità nominale compie errori nell'individuare i confini dell'entità.

Sono state, di fatti, elaborate diverse metriche per consentire la valutazione dei sistemi di *Named Entity Recognition.*

La valutazione delle performance dei modelli di riconoscimento delle entità nominali in termini di *precision, recall e F1* score è stata introdotta nell'ambito del CoNLL-2003 per il *Language-Independent Named Entity Recognition task*:

"Precision is the percentage of named entities found by the learning system that are correct. Recall is the percentage of named entities present in the corpus that are found by the system. A named entity is correct only if it is an exact match of the corresponding entity in the data file."[42]

Utilizzando queste metriche per confrontare il *gold* standard e l'output di *stanza e spaCy* ottengo i seguenti risultati:

	TP	FP	FN
Stanza	198	54	45
spaCy	177	44	49

Tabella 6: Numero di volte in cui i modelli Stanza e spaCy: individuano l'etichetta corretta delle entità nominali (TP), riconoscono qualcosa che non è un'entità (FP), non riconosce la presenza di un'entità nominale (FN).

[41] Ridong Jiang, Rafael E. Banchs, e Haizhou Li, «Evaluating and Combining Name Entity Recognition Systems», in *Proceedings of the Sixth Named Entity Workshop* (Berlin, Germany: Association for Computational Linguistics, 2016), 21–27, https://doi.org/10.18653/v1/W16-2703.
[42] Sang e De Meulder, «Introduction to the CoNLL-2003 Shared Task».

	PRECISION	RECALL	F1-SCORE
Stanza	0.79	0.81	0.80
SpaCy	0.69	0.76	0.72

Tabella 7: Valori di accuratezza dei modelli di Stanza e spaCy per il riconoscimento delle entità nominali in termini di precision, recall e f1-score.

La valutazione si basa, generalmente, su una completa corrispondenza tra l'etichetta del *gold standard* e quella assegnata dai *processor* ma in alcuni casi questa corrispondenza risulta essere limitante poiché è interessante vedere anche i casi in cui i modelli riconoscono *parzialmente* l'etichetta ossia i casi in cui il modello pur riconoscendo il tipo di entità nominale non individua la posizione corretta del token all'interno dell'entità.

A questo proposito sono state prese in considerazione le metriche introdotte da *Message Understanding Conference* (MUC)[43] che considerano diverse categorie di errori:

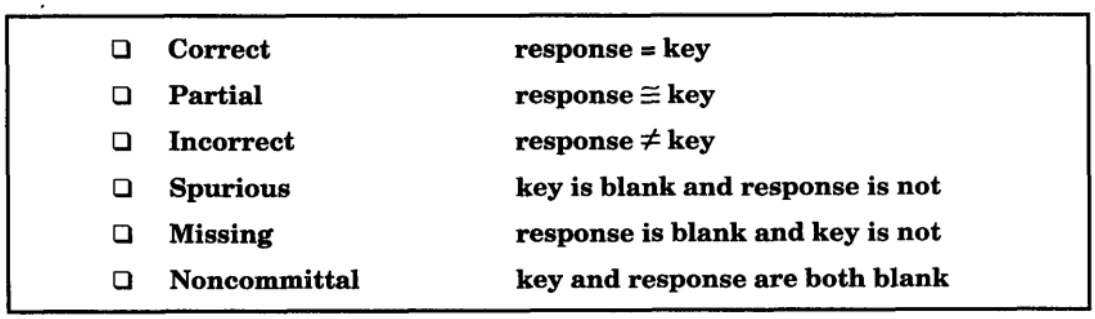


Figura 35: Casiste di errori considerata da MUC per il riconoscimento di entità nominali

Queste metriche, dunque, superano la semplice classificazione considerando, per esempio, anche le corrispondenze parziali.

	COR	PAR	INCOR	SPU	MIS
Stanza	198	11	20	54	45
spaCy	181	6	30	78	57

Tabella 8: Numero di volte in cui i modelli Stanza e spaCy: individuano l'etichetta corretta delle entità nominali (COR), individuano parzialmente le etichette (PAR), la predizione del modello è diversa dall'etichetta corretta (INCOR), riconoscono qualcosa che non è un'entità (SPU), non riconosce la presenza di un'entità nominale (MIS).

In questa prospettiva, dunque, possiamo calcolare le metriche di *precision, recall e f1 score:*

[43] Nancy Chinchor e Beth Sundheim, «MUC-5 Evaluation Metrics», in *Fifth Message Understanding Conference (MUC-5): Proceedings of a Conference Held in Baltimore, Maryland, August 25-27, 1993*, 1993, https://aclanthology.org/M93-1007.

$$precision = \frac{CORRECT}{ACTUAL} = \frac{CORRECT}{CORRECT + PARTIAL + INCORRECT + SPURIOUS}$$

(9)

$$recall = \frac{CORRECT}{POSSIBLE} = \frac{CORRECT}{CORRECT + PARTIAL + INCORRECT + MISSING}$$

(10)

	PRECISION	RECALL	F1-SCORE
Stanza	0.72	0.70	0.71
spaCy	0.66	0.61	0.64

Tabella 9: Valori di accuratezza dei modelli di Stanza e spaCy per il riconoscimento delle entità nominali in termini di precision, recall e f1-score considerando anche le etichette parzialmente corrette.

La *precisione* è, dunque, uguale al rapporto tra le occorrenze corrette e la somma dei veri positivi e i falsi positivi ossia i primi corrispondono alle occorrenze corrette, parziali, incorrette mentre i secondi corrispondono alle volte in cui il modello riconosce un'entità che in realtà non è presente nel *gold* (*spurious*).

Il *richiamo*, invece, considera i falsi negativi ossia i casi in cui il modello non riconosce alcune entità (*missing).*

Si è scelto di considerare tutte queste casiste in prospettiva dell'obiettivo ultimo del progetto ossia la ricerca di clip mediante parole chiave.

Dopo la fase di annotazione, sulla base delle prestazioni dei due *recognizer di Stanza e spaCy* e per mantenere anche un continuo con la fase precedente dell'annotazione morfo-sintattica, si è scelto di estrarre le entità nominali utilizzando la *pipeline* di Stanza.

L'estrazione delle entità nominali è stata effettuata su tutti i sottotitoli estratti in maniera automatica dal modello.

Nell'esempio seguente (Tabella 10) possiamo notare quali entità sono state estratte dal film documentario Duelo:

Titolo del film	PERSON	NORP	TIME
'Duelo'	Hail Mary Jesus Amen	Christian	Tonight

Tabella 10: Esempio di entità nominali estratte dai sottotitoli estratti automatica per il film documentario "Duelo".

In alcuni casi, tuttavia, viene attribuita l'etichetta di entità nominale, soprattutto PERSON, LOCATION, GPE E ORGANIZATION, anche ai token che presentano la

lettera maiuscola ma che rappresentano rumore nel riconoscimento automatico di caratteri. Per esemplificare quanto detto sono riportate, nella Tabella 11Tabella 10, le entità estratte dal film documentario *Walking under water:*

Titolo del film	PERSON	GPE	WORK OF ART	DATE	QUANTITY	TIME
'Walking under water	Alexan Uncle Ri Sallang Sellanq Sari Asangan Sema Sama	America NC KS	The Spirit of the Tree Nunuk Every Nunuk	three days Five days three	one gallon	tonight

Tabella 11: Esempio di entità nominali estratte dai sottotitoli rilevati automaticamente per il film documentario "Walking under water".

Si è scelto di aggiungere tutte le entità estratte alla lista di dizionari di partenza poiché l'idea è che un utente non cercherà mai qualcosa che non esiste, quindi, il fatto che vi siano anche parole che non sono parole significative e, quindi, entità nominali non è rilevante ai fini della ricerca mediante parole chiave.

Si è, dunque, partiti da una lista di dizionari contenenti le informazioni relative all'estrazione automatica dei sottotitoli per ogni clip di ogni film documentario e ad ogni dizionario si è aggiunto un elemento che ha come chiave "*extra_info"* e come valori una lista di dizionari contenti le informazioni morfo-sintattiche e l'etichetta del tipo di entità nominale attribuita al singolo token presente nel sottotitolo.

```
{
   "duration": "00:00:05.520",
   "manually_modified": false,
   "VideoId": "1840",
   "start": "00:15:29.920",
   "adminInfo": {
      "processors": "transcript",
      "build_time": "2022-02-24 14:41:37.593110"
   },
   "extracted_info": "Asangan used to say to me Alexan, Alexan you are a great fisherman! ",
   "extra_info": [
      {
         "token": "Asangan",
         "lemma": "Asangan",
         "pos": "PROPN",
         "ner": "S-PERSON"
      },
      {
         "token": "used",
         "lemma": "use",
         "pos": "VERB",
         "ner": "O"
      },
      {
         "token": "to",
         "lemma": "to",
         "pos": "PART",
         "ner": "O"
      },
      {
         "token": "say",
         "lemma": "say",
         "pos": "VERB",
         "ner": "O"
      },
      {
         "token": "to",
         "lemma": "to",
         "pos": "ADP",
         "ner": "O"
      },
      {
         "token": "me",
         "lemma": "I",
         "pos": "PRON",
         "ner": "O"
      },
      {
         "token": "Alexan",
         "lemma": "Alexan",
         "pos": "PROPN",
         "ner": "B-PERSON"
      },
```


Figura 36: Esempio di un elemento del dizionario che costituisce l'output finale del lavoro in cui alle informazioni di estrazione dei sottotitoli sono aggiunte le informazioni morfosintattiche e semantiche per ogni token.

Questi file .JSON contenti le informazioni dei frammenti di video e le etichette che descrivono la clip sono registrati nel database (*MongoDB)* che consente la memorizzazione e interrogazione di grandi quantità di dati.

.

5. Conclusioni e Sviluppi futuri

5.1. Conclusioni

Il presente lavoro si inserisce nell'ambito del progetto PH-REMIX il cui obiettivo è potenziare, da un lato, la catalogazione tradizionale del materiale audiovisivo mediante l'estrazione automatica di informazioni e, dall'altro, permettere un riuso di segmenti di video per raccontare nuove storie.

La piattaforma consente, infatti, all'utente di ricercare, mediante parole chiave, le *clip* di interesse.

L'uso di tecniche di *Intelligenza Artificiale* ha permesso di frammentare i documentari in clip sulla base delle informazioni estratte relative al contenuto del frammento: l'oggetto individuato nella scena, il colore dominante e le parole presenti nei sottotitoli impressi nei video.

Il lavoro di tesi si è, dunque, incentrato sull'estrazione di informazioni linguistiche dai sottotitoli.

Nel corso dell'elaborato è emerso come i modelli utilizzati per estrarre i sottotitoli, in realtà, sono ideati per rilevare testo da immagine piuttosto che da video, motivo per cui è stato necessario adattare l'output del modello, ossia una lista di sottotitoli estratti da ogni fotogramma, raggruppando i sottotitoli uguali in un'unica clip.

Sono state realizzate tre versioni diverse con l'obiettivo di individuare il modo migliore per raggruppare i fotogrammi. Per la realizzazione di queste versioni la valutazione dei risultati del modello ha svolto un ruolo centrale.

Durante il progetto è, infatti, emersa l'importanza della valutazione non solo per cercare il modello che funziona meglio ma anche per individuare quali cambiamenti si possano apportare per migliorarne le prestazioni.

La prima fase del lavoro consiste, infatti, nella valutazione di queste varianti dell'output del modello.

La prima versione prevede un confronto tra i sottotitoli in un range di cinque *frame;* dopo la valutazione è emerso come questa euristica non era sufficiente in quanto il modello non individua alcuni sottotitoli, infatti, il valore della *recall* è 0.67. Dall'osservazione dell'output emerge, inoltre, come nonostante l'euristica di

confronto per un range di cinque *frame* dei sottotitoli vi sono ancora molti elementi che pur trattandosi dello stesso sottotitolo vengono divise in clip diverse.
Successivamente sono state, infatti, apportate delle modifiche che prevedono un ulteriore controllo tra i sottotitoli estratti per i diversi *frame* .
Questo cambiamento ha portato ad un leggero miglioramento per quel che riguarda l'accuratezza del riconoscimento dei caratteri e della *recall.*
Per poter estrare un maggior numero di sottotitoli che corrispondono a clip diverse, nella terza versione, viene eseguito il confronto della similarità solo tra un sottotitolo e quello del fotogramma precedente per verificare se si tratta di una clip diversa.
Questo ha comportato, tuttavia, la divisione in più clip rispetto alle altre due versioni.
La valutazione dell'output di questa versione ottiene valori maggiori per quel che riguarda la *recall* (0.70) in quanto vengono, così, riconosciuti anche i sottotitoli che nelle altre versioni non venivano considerati perché accorpati in altre clip. Si nota, tuttavia, come il valore della *precision* si abbassa notevolmente (0.40) poiché non essendoci un controllo sui fotogrammi successivi vengono considerate clip distinte anche fotogrammi che, in realtà, costituiscono la stessa clip.
È importante, quindi, evidenziare come la valutazione svolge ruolo centrale per scegliere il modello che ha le prestazioni migliori ma non deve essere vista come un'operazione a sé stante rispetto alla creazione del modello bensì come un compito complementare che consente di apportare migliorie al modello anche in corso d'opera.
In seguito alla valutazione si è scelta quale versione restituisce l'output con migliore accuratezza.
I sottotitoli estratti sono, dunque, oggetto di analisi per le fasi successive del lavoro che consistono nell'estrazione di informazione sintattica e semantica.
Anche in questo caso è importare calcolare l'accuratezza dei diversi modelli per poter scegliere quale utilizzare. Sono stati prese in considerazione diverse *pipeline* per l'estrazione delle informazioni sintattiche e delle entità nominali e confrontate con un *gold standard.*
La valutazione, in questo caso, è stata centrale per individuare quale *pipeline* usare per estrarre le informazioni linguistiche. In seguito alla valutazione si è scelto di usare la *pipeline* di Stanza per estrarre le informazioni linguistiche dai sottotitoli estratti automaticamente.

Le parole così estratte rappresentano le *label* per ogni clip consentendo la ricerca, nel portale, della clip mediante le parole, sia parole lessicali che entità nominali, presenti nel sottotitolo.
La ricerca di entità nominali consente, per esempio, all'utente di raccontare una nuova storia su un personaggio o un luogo mediante il *remix* di video clip.

5.2. Sviluppi futuri

Il prototipo della piattaforma è attualmente (aprile 2022) in fase di *test* per poter individuare eventuali problemi sia di sviluppo della piattaforma sia per quel che riguarda la ricerca, all'interno del portale, di clip mediante parole chiave.
La piattaforma è caratterizzata da un'architettura software basata su microservizi in cui le componenti comunicano tra loro ma sono distinte; questo consente di apportare modifiche ad ogni componente aggiungendo nuovi servizi e funzionalità anche in base alle esigenze e richieste degli utenti.
Per quel che riguarda l'estrazione di informazione linguistica da sottotitoli sarebbe possibile, per esempio, aggiungere nuovi criteri di indicizzazione e di ricerca.
L'estrazione automatica di sottotitoli impressi nei video non sempre rileva parole significative, motivo per cui il risultato potrebbe essere migliorato utilizzando un modello di *Machine Learning* per la correzione ortografica automatica (*spelling corrections).*
Per quel che riguarda l'estrazione di chiavi di ricerca si potrebbe associare all'estrazione di entità nominale anche un compito di *Named Entity linking* (NEL) che consente di disambiguare assegnando un'unica identità ad un'entità nominale.
Un ulteriore informazione da estrarre dai sottotitoli potrebbe essere anche la tematica *(topic*) presente nella clip consentendo, così, all'utente di ricercare non solo la clip che contiene quella parola nel sottotitolo ma anche i sottotitoli che affrontano quell'argomento.
Sino a questo momento le chiavi di ricerca sono tutte in lingua inglese e anche l'utente deve effettuare la ricerca utilizzando una o più parole in lingua inglese, uno sviluppo futuro potrebbe essere quello di utilizzare tecniche per *machine translation* (MT) così da tradurre automaticamente la ricerca dell'utente.

Gli sviluppi futuri potranno, dunque, riguardare sia un miglioramento dell'output del modello di estrazione automatica dei sottotitoli ma anche l'ampliamento delle informazioni estratte e, quindi, le ricerche consentite all'interno del portale.

Riferimenti bibliografici

Alessandro Lenci, Simonetta Montemagni, Vito Pirelli, «*Testo e computer. Elementi di linguistica computazionale*>>. Carocci, Roma, 2005,

«Appendix 5: Named Entities: Current Definitions». In *Named Entities for Computational Linguistics*, 153–56. John Wiley & Sons, Ltd, 2016. https://doi.org/10.1002/9781119268567.app5.

Buchholz, Sabine, e Erwin Marsi. «CoNLL-X Shared Task on Multilingual Dependency Parsing». In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, 149–64. New York City: Association for Computational Linguistics, 2006. https://aclanthology.org/W06-2920.

Chinchor, Nancy, e Beth Sundheim. «MUC-5 Evaluation Metrics». In *Fifth Message Understanding Conference (MUC-5): Proceedings of a Conference Held in Baltimore, Maryland, August 25-27, 1993*, 1993. https://aclanthology.org/M93-1007.

«Edit distance». Consultato 22 marzo 2022. https://nlp.stanford.edu/IR-book/html/htmledition/edit-distance-1.html.

Eikvil, Line. «Optical character recognition». *citeseer. ist. psu. edu/142042. html* 26 (1993).

Festival dei Popoli. «Festival dei Popoli». Consultato 24 marzo 2022. https://www.festivaldeipopoli.org/.

«FuzzyWuzzy: Fuzzy String Matching in Python - ChairNerd». Consultato 15 marzo 2022. https://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/.

Giovanni Grasso, Chiara Mannari, Davide Serramazza «"PH-Remix": un progetto interdisciplinare per la valorizzazione del patrimonio audiovisivo del Festival dei Popoli-Festival Internazionale del Film Documentario di Firenze in ambiente digitale»

Giovanni Grasso, Chiara Mannari, Davide Serramazza, «Intelligenza artificiale e archivi audiovisivi: potenzialità e sfide del progetto "PH-Remix" » in «AIUCD 2021 - DH per la società: e-guaglianza, partecipazione, diritti e valori nell'era digitale» ,

Goswami, Subrata. «Reflections on Non Maximum Suppression (NMS)». *Medium* (blog), 13 gennaio 2020. https://whatdhack.medium.com/reflections-on-non-maximum-suppression-nms-d2fce148ef0a.

Jiang, Ridong, Rafael E. Banchs, e Haizhou Li. «Evaluating and Combining Name Entity Recognition Systems». In *Proceedings of the Sixth Named Entity Workshop*, 21–27. Berlin, Germany: Association for Computational Linguistics, 2016. https://doi.org/10.18653/v1/W16-2703.

K, Sambasivarao. «Non-Maximum Suppression (NMS)». Medium, 30 aprile 2021. https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c.

Kumar, Ela. *Natural Language Processing*. I. K. International Pvt Ltd, 2013.

LaPorte, Joseph. «Rigid Designators». In *The Stanford Encyclopedia of Philosophy*, a cura di Edward N. Zalta, Spring 2022. Metaphysics Research Lab, Stanford University, 2022. https://plato.stanford.edu/archives/spr2022/entries/rigid-designators/.

«Levenshtein1966a.pdf». Consultato 22 marzo 2022. https://nymity.ch/sybilhunting/pdf/Levenshtein1966a.pdf.

Long, Jonathan, Evan Shelhamer, e Trevor Darrell. «Fully Convolutional Networks for Semantic Segmentation», s.d., 10.

Marneffe, Marie-Catherine de, Christopher D. Manning, Joakim Nivre, e Daniel Zeman. «Universal Dependencies». *Computational Linguistics*, 20 maggio 2021, 1–54. https://doi.org/10.1162/coli_a_00402.

Marzal, A., e E. Vidal. «Computation of Normalized Edit Distance and Applications». *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15, n. 9 (settembre 1993): 926–32. https://doi.org/10.1109/34.232078.

Nadeau, David, e Satoshi Sekine. «A Survey of Named Entity Recognition and Classification», s.d., 20.

Nivre, Joakim, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, e Daniel Zeman. «Universal Dependencies v2: An Evergrowing Multilingual Treebank Collection». In *Proceedings of the 12th Language Resources and Evaluation Conference*, 4034–43. Marseille, France: European Language Resources Association, 2020. https://aclanthology.org/2020.lrec-1.497.

O'Shea, Keiron, e Ryan Nash. «An Introduction to Convolutional Neural Networks». *ArXiv e-prints*, 1 novembre 2015.

Laboratorio di Cultura Digitale. «PH-Remix». Consultato 24 marzo 2022. http://www.labcd.unipi.it/ph-remix/.

Plc, YouGov. *Fuzzy: Fast Python phonetic algorithms* (versione 1.2.2). POSIX, Python. Consultato 15 marzo 2022. https://github.com/yougov/Fuzzy.

Qi, Peng, Yuhao Zhang, Yuhui Zhang, Jason Bolton, e Christopher D. Manning. «Stanza: A Python Natural Language Processing Toolkit for Many Human Languages». *arXiv:2003.07082 [cs]*, 23 aprile 2020. http://arxiv.org/abs/2003.07082.

«Remix Theory » Remix Defined». Consultato 4 marzo 2022. https://remixtheory.net/?page_id=3.

Sahoo, P.K, S Soltani, e A.K.C Wong. «A Survey of Thresholding Techniques». *Computer Vision, Graphics, and Image Processing* 41, n. 2 (febbraio 1988): 233–60. https://doi.org/10.1016/0734-189X(88)90022-9.

Sang, Erik F. Tjong Kim, e Fien De Meulder. «Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition». *arXiv:cs/0306050*, 12 giugno 2003. http://arxiv.org/abs/cs/0306050.

Smith, R. «An Overview of the Tesseract OCR Engine». In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, 2:629–33, 2007. https://doi.org/10.1109/ICDAR.2007.4376991.

«SpaCy · Industrial-Strength Natural Language Processing in Python». Consultato 29 marzo 2022. https://spacy.io/.

Straka, Milan, Jan Hajic, e Jana Strakova. «UDPipe: Trainable Pipeline for Processing CoNLL-U Files Performing Tokenization, Morphological Analysis, POS Tagging and Parsing», s.d., 8.

«Task Description and Evaluation». Consultato 30 marzo 2022. https://universaldependencies.org/iwpt20/task_and_evaluation.html.

Wang, Tao, David J Wu, Adam Coates, e Andrew Y Ng. «End-to-End Text Recognition with Convolutional Neural Networks», s.d., 5.

Weischedel, Ralph, Palmer, Martha, Marcus, Mitchell, Hovy, Eduard, Pradhan, Sameer, Ramshaw, Lance, Xue, Nianwen, et al. «OntoNotes Release 5.0». Linguistic Data Consortium, 16 ottobre 2013. https://doi.org/10.35111/XMHB-2B84.

«Welcome to Levenshtein's documentation! — Levenshtein 0.18.1 documentation». Consultato 22 marzo 2022. https://maxbachmann.github.io/Levenshtein/.

Zhou, Xinyu, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, e Jiajun Liang. «EAST: An Efficient and Accurate Scene Text Detector». In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2642–51. Honolulu, HI: IEEE, 2017. https://doi.org/10.1109/CVPR.2017.283.