Politecnico di Torino

POLITECNICO DI TORINO

Laurea Magistrale in Ingegneria del Cinema e dei Mezzi di Comunicazione

Tesi di Laurea Magistrale

Blob: uno studio sperimentale sulla replicazione del montaggio del programma tramite il modello multimodale Gemini

Relatrice:

Prof.ssa Tatiana MAZALI

Candidato:

Fernanda GARCIA

Co-relatore/ Tutor aziendale:

Lorenzo CANALE

ANNO ACCADEMICO 2023/2024

Sommario

Blob è un programma televisivo italiano che rielabora creativamente frammenti di vari contenuti audiovisivi per creare nuovi significati attraverso il montaggio.

L'elaborato esplora l'applicazione dell'intelligenza artificiale, in particolare il modello multimodale Gemini, nel comprendere e replicare il montaggio del programma. Lo studio si concentra su due obiettivi principali: la segmentazione automatica delle clip contenenti i tagli "Blob" e il riconoscimento dei tagli "Blob" rispetto a quelli "non Blob".

Per il primo obiettivo, è stato creato un database di clip utilizzando algoritmi di segmentazione video e tecniche di prompt engineering con il modello Gemini.

Nei risultati delle metriche di performance, si è ottenuta una precision di circa 0,89 nel riconoscimento dei tagli di "Blob". Tuttavia, il secondo obiettivo, che mirava a distinguere i tagli "Blob" da quelli "non Blob", ha evidenziato limiti significativi, con metriche di precision, accuracy e F1 in media inferiori a 0.5.

I risultati dimostrano che, sebbene il modello multimodale Gemini, possa supportare la segmentazione video di un programma come Blob, non sia ancora in grado di comprenderne e replicarne il montaggio.

Il montaggio automatico non è stato raggiunto, ma rappresenta una direzione futura che richiederà ulteriori progressi nei modelli multimodali per comprendere meglio i programmi televisivi complessi come quello di Blob.

Indice

1. Introduzione

Può l'intelligenza artificiale essere in grado di replicare il montaggio di un programma televisivo come *Blob*? Con questa domanda si avvia la ricerca di questa tesi. In particolare, è stato testato il modello multimodale di grandi dimensioni (LMM) Gemini. Questi tipi di modelli rappresentano l'ultima frontiera dell'intelligenza artificiale e stanno dimostrando progressi significativi nella comprensione del linguaggio umano, compreso quello creativo. Ad esempio, l'IA generativa è oggi in grado di generare immagini e di creare cortometraggi sperimentali che replicano determinati stili di riferimento.
Capire come questi modelli possano interpretare il linguaggio del montaggio – in questo di un programma autoriale come *Blob* – rappresenta un passo fondamentale per indagare se e come sia possibile replicare contenuti creativi attraverso l'IA. Lo studio proposto adotta un approccio sperimentale e si concentra su due obiettivi principali:

Segmentazione automatica di Blob per l'estrazione delle clip di partenza.

Blob è un programma che utilizza materiali d'archivio provenienti da una varietà di fonti, come programmi televisivi (telegiornali, talk show), estratti di film, contenuti tratti dai social media e altro ancora.
L'obiettivo della prima fase del progetto è stato estrarre dalle puntate di Blob, reperite online, un database di clip, ciascuna contenente un taglio del programma. Queste clip, poi denominate "subclip" nel database, sono state inizialmente individuate attraverso algoritmi di segmentazione video basati sul rilevamento di scene, in particolare utilizzando la libreria PySceneDetect, in modo da rilevare automaticamente i tagli.
Dopo aver generato un primo ampio database di subclip, la seconda fase si è concentrata sull'identificazione dei tagli "editoriali". Questi tagli uniscono clip provenienti da fonti diverse (ad esempio, un film e un programma televisivo) e

rappresentano i tagli effettivi del programma, denominati "tagli di Blob". Poiché all'interno del programma sono presenti anche tagli "formali", ovvero transizioni tra contenuti dello stesso tipo, come diverse inquadrature di uno stesso programma televisivo.
Per affrontare questo compito, è stata testata l'abilità di Gemini nel riconoscere tali tagli tramite un'attenta attività di *prompt engineering*. Sono stati testati diversi tipi di prompt testuali e configurazioni di input, utilizzando non solo le subclip come video ma anche i loro caroselli di immagini (formati da frame chiave delle clip). Questo ha permesso di combinare diverse configurazioni: subclip, subclip + carosello o solo carosello, ciascuna con specifici prompt testuali.
I risultati ottenuti sono stati promettenti, con alcune configurazioni con prompt specifici che hanno raggiunto valori elevati nelle metriche. In particolare, i valori medi di precision, F1 e accuracy per queste configurazioni si sono attestati intorno a 0,80. Questi risultati hanno permesso la costruzione di un database dei tagli di *Blob* rilevati automaticamente, che sarà utilizzato nella seconda fase del progetto per ulteriori analisi e applicazioni.

Riconoscimento e distinzione dei tagli di "Blob" rispetto ai tagli di "non Blob"

Il secondo obiettivo del progetto è stato quello di valutare se il modello Gemini sia in grado di distinguere, in modo binario, i tagli di *Blob* dai tagli *non Blob* attraverso il prompt engineering. Questo obiettivo è stato pensato come propedeutico al montaggio automatico, anche se non si è arrivati direttamente a testare il montaggio come tale. La distinzione tra tagli di *Blob* e tagli *non Blob*, è servita per comprendere quanto il modello sia capace di riconoscere il linguaggio specifico del programma. A tal fine, è stato creato un database di tagli *non Blob* attraverso algoritmi di concatenazione video ed è stato condotto un esperimento con Gemini per testare la capacità del modello di identificare correttamente le differenze usando come nella prima fase del progetto diversi tipi di configurazioni con prompt specifici testuali e diversi tipi di media includendo i trascritti audio dei video

I risultati numerici delle metriche di precision, recall e f1 con valori in media inferiori allo 0,5 si sono rivelati limitati e poco attendibili, dimostrando che, allo stato attuale, i modelli multimodali come Gemini non sono ancora in grado di distinguere e quindi di poter replicare il montaggio di due clip di Blob
ll primo obiettivo – la segmentazione automatica – si è dimostrato un passaggio essenziale per il raggiungimento del secondo obiettivo, poiché ha permesso di creare una base dati solida e strutturata necessaria per ulteriori analisi. Tuttavia, il riconoscimento e la distinzione dei tagli evidenziano i limiti attuali dei modelli multimodali nella comprensione di linguaggi audiovisivi complessi come quello di *Blob*.
Il montaggio automatico non è stato incluso tra gli obiettivi raggiunti in questa tesi, ma rappresenta una possibile direzione per lavori futuri. La capacità di replicare il montaggio autoriale di *Blob* richiederà ulteriori sviluppi nei modelli di IA, soprattutto per quanto riguarda la comprensione delle relazioni semantiche e stilistiche tra i segmenti audiovisivi.

2. Blob: studio del linguaggio

Poiché l'obiettivo principale di questa tesi è esplorare come l'intelligenza artificiale possa comprendere e replicare lo stile del montaggio del programma *Blob*, è necessario un approfondimento del programma attraverso un'analisi semiotica.
In questo capitolo illustrerò le principali caratteristiche di Blob e i suoi riferimenti culturali, approfondendo poi in particolare il montaggio, elemento chiave di questoa ricerca. Tratterò inoltre il programma nella sua dimensione di archivio e repertorio. Concluderò con alcune considerazioni e premesse essenziali per il progetto di ricerca di tesi.

2.1 Definizione di Blob

Figura 1: Immagine tratta dalla sigla di Blob

Blob è un programma televisivo italiano in onda su Rai 3 alle 20:00, diventato negli anni uno dei più longevi e iconici della Rai e della televisione italiana.
"É la cosa più orribile che abbia mai visto" recitano i fotogrammi d'apertura della prima puntata, trasmessa nel 1989, tratti dal film *The Blob (Fluido Mortale)* del 1958 da cui deriva il titolo del programma [1] .
Ideato da Enrico Ghezzi e Marco Giusti, *Blob* propone ogni sera un montaggio di materiali audiovisivi che spazia da programmi televisivi e spot pubblicitari a

contenuti attuali tratti da internet e dai social media, componendo puntate della durata variabile tra i 10 e i 25 minuti.
Per quanto riguarda il genere possiamo affermare che *Blob* è un programma satirico. Qui la satira non si limita alla politica e alla società, bensì si estende a tutto ciò che la televisione, con la sua pervasività, trasmette.
Blob, come definito da Grasso [1], è un programma "meta-televisivo" poiché parla di televisione attraverso una rielaborazione critica del mezzo stesso.
Questo punto di vista non viene mai esplicitato ma sollecitato da un taglio ironico e dal montaggio di immagini che, decontestualizzate, assumono nuovi significati.
Ogni episodio è un collage di frammenti strappati dal loro contesto originale e riutilizzati in una nuova composizione che offre una lettura del tutto diversa, che lo spettatore è invitato a ricomporre.
Il punto di vista critico di *Blob*, come affermato da Rajewsky [2], non è mai esplicitato con intenti didattici, ma è affidato alla libera interpretazione dello spettatore, rendendolo un'opera "aperta" nel senso definito da Umberto Eco, senza una conclusione univoca.
Al contrario, *Blob* offre una "apertura semantica" che invita il pubblico a interpretare attivamente ciò che osserva.
D'altronde come spiega lo stesso autore del programma Enrico Ghezzi [3]:
"Blob non si rivede, non si rimonta. Il completamento, la forma che quella sera, stasera, domani sera ha assunto o assumerà sono affidati da sempre a voi, ai vostri codici sparsi, alla vostra disattenzione, al vagare dei vostri occhi, al vostro desiderio di lettori, alla vostra immaginazione di terroristi o di pacifisti".

Il modo specifico di Blob, inoltre, di montare e intrecciare frammenti selezionati riprende, il concetto di détournement, una pratica artistica introdotta dai situazionisti negli anni Cinquanta e Sessanta, influenzati dai movimenti d'avanguardia come il dadaismo e il surrealismo.

Il détournement consisteva nel riciclare in modo sovversivo i prodotti della società capitalista, capovolgendone il significato e rimuovendoli dal contesto originale per inserirli in nuove ambientazioni, creando così nuovi significati come critica sociale alla società dello spettacolo.

Secondo Filippo Porcelli , un altro degli autori del programma, Blob adotta e reinterpreta questo principio nel suo montaggio unendo immagini che provengono da diversi contesti per stimolare un processo di riflessione critica da parte dello spettatore: “Il détournement è prima di tutto un modo di vedere, qualcosa come una tecnica di invenzione e di riappropriazione, qualcosa che parte dalla coscienza e dal dominio sul proprio inventario di saperi e di saper fare. Il détournement è il contrario della citazione, strappa il frammento dal suo contesto, dal suo movimento e dalla sua epoca” [4]

2.2 Il contesto televisivo di Blob

Secondo Rajewsky [2], Blob può essere considerato un esempio paradigmatico di “neotelevisione” poiché riflette i cambiamenti che hanno trasformato il panorama televisivo italiano negli anni Ottanta.
Il termine "neo-televisione" [5] è stato coniato da Umberto Eco per descrivere il modello di televisione generalista e commerciale, sviluppatosi in Italia dai primi anni Ottanta, che si differenzia dalla "paleotelevisione" della Rai degli anni Cinquanta e Sessanta, caratterizzata da un approccio pedagogico e istituzionale.
A partire dalla metà degli anni Settanta, il sistema radiotelevisivo italiano era dominato dal duopolio tra il settore pubblico della Rai e quello privato di Mediaset, lanciato da Silvio Berlusconi.
In particolare, come afferma Monteleone [6], negli anni Ottanta, la televisione commerciale privata si impose grazie alla capacità di attrarre il pubblico e organizzare i palinsesti in modo efficace.
Tuttavia, questo modello ridusse la qualità dei contenuti televisivi, introducendo una forte commercializzazione che trasformò l’Italia in un mercato dominato dal consumo e dalla pubblicità.
Dal 1988, la Rai ampliò, inoltre, la programmazione giornaliera, offrendo una vasta gamma di contenuti che favorirono l'emergere del fenomeno dello “zapping”

Come spiega Rajewsky [2], Blob si fonda proprio sul principio dello zapping, trasformando questa attività, originariamente legata alla fruizione passiva, in un elemento costitutivo della sua produzione.
Il ritmo rapido e frammentato della programmazione, segnato da interruzioni pubblicitarie frequenti, contribuiva a creare un'estetica della ricezione basata sul frammento "come parte staccata di un'opera che non si ha il tempo né la voglia di consumare nella sua integrità". [6]
Blob sfrutta questa frammentarietà nel suo linguaggio per evidenziare la superficialità e la vacuità della neotelevisione del periodo, trasformando tali elementi in spunti di riflessione critica per lo spettatore.
Le abitudini di zapping, il ritmo sempre più accelerato della produzione e della ricezione tipici della neotelevisione vengono così letteralmente posti di fronte agli occhi dello spettatore di Blob, spingendolo a riflettere sulla natura del medium televisivo

2.3 Il montaggio

Il montaggio è uno degli elementi principali che caratterizzano lo stile e il linguaggio di Blob ed è per questo motivo il principale oggetto di studio per il progetto. Solo capendo il montaggio di Blob e quindi come si struttura il suo linguaggio è possibile cercare di emularlo nella creazione e valutazione con l'intelligenza artificiale.

Teoricamente il montaggio è quell'operazione tecnica che consente di unire la fine di un'inquadratura con l'inizio dell'inquadratura successiva e permette allo spettatore di assistere a quello che viene definito "l'effetto di montaggio" ovvero alla percezione del passaggio da una immagine all'altra.
Questo "effetto di montaggio" risulta evidente in Blob in quanto i materiali montati appartengono a media e linguaggi diversi, e formano così un pastiche di immagini di repertorio che vengono usate in nuovo contesto che ne altera il significato

originale. Montare questi frammenti significa lavorare per concatenamenti, sovrapposizioni e scomposizioni.
La tipologia di montaggio in Blob è estremamente differenziata e varia a seconda delle diverse puntate e infatti non si può definire mai casuale. [7]

Come afferma Marco Giusti uno degli ideatori del programma nella prefazione del libro dedicato: “Blob non è mai ingenuo ordinamento di materiale già andato in onda. Niente è casuale in Blob, essendo una pratica che varia regolarmente di gusto, di intensità, perché si fa in tanti, con teste e culture diverse.”[3]
Il montaggio viene affidato infatti a diversi montatori e il fatto che non sia casuale si può notare nelle varie immagini selezionate che spesso vengono accostate secondo determinate logiche.
All’interno delle puntate si possono notare analogie e contrapposizioni tra i frammenti che si susseguono per ogni taglio.
Le analogie possono riguardare i temi, lo stile o la forma delle immagini.

2.3.1 Montaggio come analogia

Un esempio di analogia può essere notato nella puntata del 15 maggio 2024. Se analizziamo la puntata notiamo il titolo in sovraimpressione “DuelliAlSole”. La puntata, infatti, è incentrata sul tema dei dibattiti tra politici, tema che accumuna i vari spezzoni della puntata. Uno spezzone è tratto dal confronto tra Romano Prodi e Silvio Berlusconi durante le elezioni del 2006. Subito dopo, vediamo un estratto dal film “Qualunquemente” di Antonio Albanese, in cui il protagonista, candidato premier, partecipa a un dibattito politico in televisione. Tuttavia, la puntata contiene anche altri contenuti legati al tema, presentati con il linguaggio satirico di Blob. La puntata, infatti, si apre con una scena tratta da “I sette samurai” di Kurosawa, in cui due guerrieri samurai si affrontano in un duello finale. Più avanti, vediamo un estratto dal film “Rambo”, dove due pugili si scambiano colpi in un combattimento.

In questo caso, tuttavia, si può notare come i vari frammenti appartengono a media diversi dai film d'autore ai talk show televisivi ma sono tutti accomunati dalla tematica del dibattito politico.

In altri casi, le puntate non hanno un tema specifico, ma presentano semplicemente estratti dalla televisione del momento, come fatti di cronaca rilevanti o momenti imbarazzanti in talk show o altri programmi televisivi, e talvolta anche contenuti tratti dai social in formato verticale. Tuttavia, in alcuni di questi casi, gli spezzoni dei vari programmi mantengono una connessione logica con il frammento successivo o precedente, perché accomunati da un'analogia di uno stesso tema.
Se analizziamo la puntata del 31 luglio 2024, per esempio, notiamo che non c'è un tema comune tra gli spezzoni della puntata.
Nella puntata sono presenti vari contenuti tratti dalla televisione del momento, come un estratto da un talk show, un reality show e persino dalle Olimpiadi del 2024. Tuttavia, si può notare come alcuni frammenti successivi della puntata siano collegati da una connessione logica. Per esempio, vediamo un'intervista alla premier Giorgia Meloni, seguita subito dopo, in un accostamento ironico, da una pubblicità di meloni. Sempre nella stessa puntata, si parla della caccia a un'orsa in via di estinzione, seguita da un estratto di un film in cui un orso attacca una famiglia.
In questi casi il montaggio segue un accostamento per analogie di temi spesso anche ironici che cambiano di volta in volta e che lo spettatore deve ricostruire

2.3.2 Montaggio come conflitto

Una caratteristica fondamentale del linguaggio e del montaggio di Blob è il contrasto.
In Blob, infatti, si assiste spesso all'unione di immagini contrapposte: vi sono contrapposizioni di stile, temi, mezzi, spazio o tempo. [7]
Negli esempi citati in precedenza, sebbene i frammenti condividano un tema comune, provengono generalmente da contesti molto diversi, come un film

d’autore accostato a un programma televisivo recente o un telegiornale di attualità associato a un reality show demenziale.

In altri casi, il contrasto riguarda anche la durata dei singoli spezzoni: spesso, ad esempio, vediamo un frammento di cinque secondi seguito da uno di tre secondi. Le combinazioni dei vari tagli possono essere infinite, e molte volte sembra che i montatori scelgano volutamente gli accostamenti più stridenti.
Il montaggio in Blob è conflitto, poiché si basa proprio sullo scontro di scene e inquadrature opposte e indipendenti l’una dall’altra.
In questo senso, il montaggio di Blob, come citato anche da Rajewsky [2], riprende il concetto del “montaggio delle attrazioni” teorizzato dal pioniere del cinema russo degli anni Venti, Sergej Ejzenštejn.
Per Ejzenštejn, l’elemento fondamentale del montaggio è il conflitto tra le scene, in contrapposizione alla continuità visiva tipica del cinema narrativo di David Griffith altro regista degli anni 20, pioniere del montaggio.
Nel suo “montaggio delle attrazioni”, noto anche come montaggio alternato, Ejzenštejn contrappone immagini provenienti da contesti differenti per provocare reazioni intellettuali ed emotive nello spettatore.
Un esempio di questa tecnica si trova nel celebre film Sciopero (1925), in cui Ejzenštejn alterna immagini della polizia zarista che massacra il popolo con quelle di un macello di buoi.
Lo stesso Ejzenštejn introduce questo linguaggio affermando: “Attraverso il montaggio avviene uno scontro, e dallo scontro di due fattori nasce un concetto; è il luogo in cui si produce un pensiero che trae origine proprio dallo scontro tra due elementi indipendenti l’uno dall’altro.” [8]

Blob adotta spesso questo principio nel montaggio, specialmente per affrontare tematiche delicate e stimolare una riflessione critica negli spettatori.
Un esempio dell'applicazione di tale principio si nota nell'episodio del 10 ottobre 2024, in cui vengono trattati diversi temi di attualità, tra cui la guerra a Gaza.
In un momento della puntata, vediamo uno spezzone tratto dal film *La vita è bella* (1987), in cui il protagonista si trova con la famiglia in un campo di concentramento nazista. Lo spezzone successivo proviene dal reality show

Temptation Island, in cui una concorrente piange accompagnata da una musica drammatica.
Segue poi un video tratto dai social media che mostra una bambina, presumibilmente in un campo di Gaza, a cui viene dato un lecca-lecca.
Successivamente, vediamo un altro contenuto dai social con il politico Matteo Salvini che canta e balla "E la vita, la vita".
La sequenza si conclude con un video dell'attacco e dell'incendio israeliano in un campo profughi a Gaza.
Questi accostamenti di montaggio riprendono appieno il concetto di 'montaggio intellettuale', poiché il contrasto tra le immagini così opposte suscita emozioni e, in particolare, stimola una riflessione critica sulle tematiche di attualità.

2.4 Blob come programma d'archivio

Se da un lato il montaggio rappresenta uno degli aspetti fondamentali del linguaggio di *Blob*, la raccolta del materiale – e quindi la creazione stessa del database di contenuti mediali d'archivio – è ancor di più la sua componente fondante.
L'autorialità di *Blob*, infatti, non risiede solo nel montaggio, ma anche nella raccolta quotidiana dei materiali da parte degli autori.
È proprio la ricerca di quell'intervista al politico, del momento esilarante ma grottesco nel quiz show o di quel contenuto virale e bizzarro trovato in rete che rende *Blob* così riconoscibile e unico.

Sin dalle origini di *Blob*, accanto ai montatori del programma erano presenti i "visionatori": coloro che si alternavano a guardare la televisione del giorno precedente per diverse ore, seguendo soprattutto programmi in diretta, notiziari, varietà e altri format.
Ma visionare non significava semplicemente guardare la tv. Come afferma Filippo Porcelli: "Si trattava di realizzare una sorta di sguardo congelante, un po' come

quello di una Medusa, per intenderci, che metteva a fuoco l'impercettibile nei frammenti testuali per riconnetterli a una possibile struttura di correlazione".[4]

Blob, tuttavia, non è il primo programma televisivo a utilizzare in modo creativo il materiale d'archivio. Nel 1988 andò in onda la prima puntata di *Schegge*, ideato dagli stessi Enrico Ghezzi e Marco Giusti, insieme a Filippo Porcelli, che sarebbe poi diventato uno dei principali autori di *Blob*. *Schegge* può essere considerato il precursore di *Blob*, essendo stato il primo programma Rai a costruire televisione utilizzando repertori d'archivio, che all'epoca non erano ancora catalogati nelle Teche Rai. L'obiettivo iniziale era accumulare materiale televisivo per colmare i "buchi" del palinsesto, noti come "allacci". Insieme a *Schegge*, nacque anche *Vent'anni Prima*, con la stessa idea di rimontare materiale d'archivio, in questo caso proveniente dai telegiornali.

Questi primi esperimenti gettarono le basi per la nascita di *Blob*, uno dei programmi più iconici della televisione italiana per il suo uso innovativo dei materiali d'archivio. Come descritto da Ghezzi nella prefazione de *Il libro di Blob*: "il culmine televisivo (fino a oggi) di un processo di repertorizzazione e archeologizzazione del presente e, insieme, di 'presentizzazione' e 'direttizzazione' del passato e del reperto".[3]

Negli ultimi anni, *Blob* utilizza come materiale d'archivio non solo programmi televisivi o film, che hanno rappresentato il principale materiale fin dagli albori, ma con l'avvento di Internet anche contenuti tratti dal web e dai social media. È frequente trovare contenuti in formato verticale, presi da pagine social di testate giornalistiche, così come contenuti amatoriali che trattano tematiche d'attualità.

"Blob non esiste, resiste" afferma lo stesso Enrico Ghezzi e questa frase ha un tono quasi profetico.[3]

Dal 1989, infatti, *Blob* è stata l'unica trasmissione televisiva capace di trasformarsi in una categoria estetica riproducibile, diventando non solo un programma ma un vero e proprio stile, un modo unico di rappresentare la realtà[9]

Ha saputo resistere all'avvento del digitale, al broadcasting, al narrowcasting, alla conseguente diffusione di pubblici sempre più selettivi e oggi addirittura ai social

media. Nonostante i profondi cambiamenti che hanno segnato la fine della TV generalista, *Blob* ha continuato a esistere senza perdere la sua identità e preservando l'estetica e lo stile che lo contraddistinguono.[10]

2.5 Considerazioni per il progetto

Dall'analisi mediale e semiotica del linguaggio di *Blob* emergono considerazioni utili per il progetto di ricerca.

Blob, infatti, oltre a essere un programma autoriale, rappresenta un esempio unico nel panorama televisivo italiano. Non possiede uno stile univoco; in particolare, il montaggio, come già evidenziato, risulta estremamente vario e differenziato. Inoltre, *Blob* utilizza materiale d'archivio, ovvero contenuti audiovisivi precedentemente montati da altri autori.

Per quanto riguarda il secondo obiettivo del progetto, ossia comprendere quanto i modelli di intelligenza artificiale possano interpretare e, eventualmente, replicare il montaggio di *Blob*, le premesse sembrano piuttosto complesse. *Blob* presenta uno stile autoriale, difficile da replicare. I modelli di intelligenza artificiale potrebbero incontrare notevoli difficoltà nel distinguere le clip di *Blob* da quelle di "non *Blob*" nella seconda fase del progetto. Tuttavia, alcune caratteristiche specifiche del montaggio, come l'uso dell'analogia e del contrasto, possono essere utilizzate come parametri per la ricerca.

In relazione al primo obiettivo, ovvero la segmentazione automatica dei tagli di *Blob*, essendo un task più tecnico, i modelli di intelligenza artificiale – in questo caso *Gemini* – potrebbero fornire risultati più soddisfacenti. La rilevazione dei tagli non richiede infatti di comprendere l'aspetto autoriale e stilistico del programma, ma si limita a un'analisi più oggettiva delle transizioni tra le clip.

3. Stato dell'arte: segmentazione video

In questo capitolo esploreremo lo stato dell'arte riguardante tre principali tipi di segmentazione video, che possono essere identificati tramite algoritmi o altri metodi. La segmentazione per il rilevamento di scene si è rivelata particolarmente utile per la prima fase del progetto di ricerca, finalizzata alla creazione del database di clip con i tagli di 'Blob'. È quindi fondamentale analizzare i vari approcci di segmentazione video disponibili. In particolare, è stato condotto uno studio sulla letteratura relativa a tre tipologie di segmentazione e ai tool attualmente in uso. Vengono ora definite le tre principali tipologie individuate:

- **Shot Boundary Detection:** Il processo di *Shot Boundary Detection* (SBD) si concentra sull'identificazione dei punti esatti in cui termina un'inquadratura e ne inizia un'altra. Un'inquadratura (*shot*) è una sequenza continua di fotogrammi che mostrano una progressione visiva coerente, catturata senza interruzioni da una videocamera. Le transizioni tra le inquadrature possono includere cambiamenti nell'angolazione della videocamera, nel livello di zoom o persino tagli che portano a prospettive diverse all'interno della stessa scena.
  Lo scopo dell'SBD è rilevare tali transizioni, che possono manifestarsi come tagli netti (*cuts*), dissolvenze (*fades*) o transizioni più graduali (*dissolves*). In altre parole, i confini di uno *shot* rappresentano i limiti fisici in cui avvengono cambiamenti nella videocamera.

- **Scene Boundary Detection**: Il rilevamento dei confini delle scene (*Scene Boundary Detection*) è il processo che identifica le transizioni tra scene diverse all'interno di un video. Una scena rappresenta tipicamente un segmento continuo della narrazione, composto spesso da più inquadrature interconnesse che, insieme, descrivono un particolare ambiente, luogo o evento. Questo approccio mira a rilevare i punti in cui la storia cambia in modo significativo, come il passaggio da un luogo a un altro o uno

spostamento temporale, concentrandosi sulla struttura narrativa ad alto livello piuttosto che sui cambiamenti tecnici nelle inquadrature.
Il compito della *scene detection* è segmentare automaticamente un video in parti significative e narrative, senza alcun supporto da parte del produttore. Questo viene realizzato utilizzando indizi percettivi e caratteristiche multimediali estratte dai dati, come l'analisi del contenuto visivo, audio e delle relazioni semantiche tra le inquadrature, per identificare i confini delle scene in modo autonomo.

- **Content Source-Based Segmentation**: Questo metodo segmenta un video in base all'origine o alla fonte del contenuto, piuttosto che ai cambiamenti narrativi o visivi. Questa tipologia di segmentazione, che non è presente in letteratura, è stata introdotta e denominata per descrivere un approccio utile a segmentare automaticamente programmi come *Blob*, composti da contenuti audiovisivi di diversa origine. L'obiettivo è identificare e delimitare blocchi coerenti di contenuto in base allo stile di produzione, alla qualità o al formato della fonte, piuttosto che rilevare transizioni narrative o tecniche.

Il rilevamento della *Content Source-Based Segmentation* sarà l'obiettivo principale della prima fase del progetto per identificare i tagli di 'Blob'. Questo approccio è particolarmente rilevante per il programma *Blob*, che utilizza clip provenienti da diverse fonti.
In questa sezione, quindi, presenteremo lo stato dell'arte riguardo a ciascuno dei diversi tipi di segmentazione.

3.1 Shot boundary detection (SBD)

3.1.1 Stato dell’Arte della Shot Boundary Detection

La Shot Boundary Detection (SBD) è un compito fondamentale nell’analisi dei video, mirato a segmentare un flusso video in shot distinti, ognuno dei quali rappresenta una sequenza continua di fotogrammi senza transizioni evidenti. La

rilevazione dei confini tra questi shot è utile per l'archiviazione, l'indicizzazione e la ricerca nei video. Nel corso degli anni, diversi approcci sono stati sviluppati per affrontare le sfide di questa task, evolvendo dal trattamento delle transizioni semplici a soluzioni più complesse, che utilizzano reti neurali e metodi statistici avanzati.

Nel 1996, V. Patel e K. Sethi [11] hanno posto le basi per l'analisi dei confini tra shot. Il metodo proposto si concentrava sull'analisi dei cambiamenti di pixel tra frame consecutivi. L'idea era che un cambio significativo tra fotogrammi indicava una transizione di tipo abrupt (improvvisa). Tuttavia, questo approccio era molto sensibile agli effetti di illuminazione e al movimento, che potevano generare falsi positivi, come un semplice cambiamento di luce percepito come una transizione.

Nel 2006 Cotsaces, C., et al. [12] proponevano un avanzamento significativo: non solo si cercava di rilevare i confini degli shot, ma si puntava a rappresentazioni più compatte e agili del contenuto video. Questo permetteva di ridurre la quantità di dati da elaborare, facilitando l'analisi del video. Il metodo esaminava una rappresentazione condensata che sintetizzava le informazioni più rilevanti per il rilevamento delle transizioni, ma, come i metodi precedenti, risultava ancora limitato da problematiche come l'illuminazione variabile.

Nel 2015, Shekar, B. H., e K. P. Uma [13] hanno introdotto un approccio più sofisticato utilizzando le derivate direzionali di Kirsch, che analizzano il cambiamento dei gradienti nei fotogrammi. In questo approccio, le transizioni improvvise sono identificate confrontando la differenza di intensità tra i fotogrammi adiacenti. Il vantaggio principale di questa tecnica era che era in grado di ridurre l'effetto di illuminazione variabile e i movimenti all'interno degli shot, che nei metodi precedenti potevano essere interpretati erroneamente come cambiamenti di scena. Sebbene l'accuratezza fosse migliorata, rimaneva comunque una certa vulnerabilità agli effetti di movimento complessi e rapidi.

Il 2017 ha segnato una grande svolta nell'ambito della SBD con l'introduzione delle reti neurali convoluzionali (CNN), che sono in grado di apprendere autonomamente le caratteristiche più rilevanti dai dati.

- Hassanien, Ahmed, et al. [14] ha proposto un metodo innovativo che utilizza CNN spazio-temporali. Queste reti non solo prendono in considerazione la variazione spaziale dei fotogrammi (cioè la composizione di ogni singolo fotogramma), ma anche la dimensione temporale, considerando la sequenza dei fotogrammi per identificare le transizioni tra gli shot. Questo approccio ha permesso di affrontare più efficacemente i cambiamenti complessi di illuminazione e movimento, oltre a gestire grandi volumi di dati video, migliorando notevolmente sia la velocità che l'accuratezza del rilevamento. I risultati ottenuti hanno mostrato che le CNN spazio-temporali possono rilevare le transizioni in tempo reale, con prestazioni superiori rispetto ai metodi tradizionali.
- Nello stesso anno Gygli [15] proponeva l'uso di reti convoluzionali completamente convoluzionali (FCN), un'altra tipologia di rete neurale convoluzionale, che offre il vantaggio di un'elaborazione estremamente rapida, mantenendo alta l'accuratezza. Questo approccio ha dimostrato che è possibile ridurre significativamente il tempo di elaborazione, rendendo la SBD applicabile a grandi volumi di dati video in tempo reale senza compromettere la qualità dei risultati.

Nel 2019 Souček, Moravec e Lokoč [16] hanno presentato TransNet, una rete neurale profonda progettata per la rilevazione rapida dei confini di scena (SBD). Questo modello si distingue per l'uso di convoluzioni 3D dilatate, che ampliano il campo visivo della rete senza aumentare significativamente il numero di parametri da addestrare, rendendo il processo più efficiente. La rete è stata addestrata utilizzando due tipologie di transizioni comuni: hard cuts e dissolves. Il risultato è una rilevazione altamente precisa e veloce, con prestazioni all'avanguardia, soprattutto usando il dataset della RAI.
Un aspetto innovativo di TransNet è la sua capacità di funzionare con una velocità di inferenza superiore al tempo reale, anche su una singola GPU di livello medio. Questo rende il modello adatto per applicazioni che richiedono l'elaborazione

rapida dei video, come i sistemi di gestione e recupero dei video. Sebbene il modello sia stato principalmente testato su transizioni di tipo hard cut e dissolve, i risultati sono stati estremamente promettenti, con una bassa percentuale di falsi positivi rispetto ai falsi negativi. Questo approccio offre un'alternativa più efficiente e scalabile rispetto ai modelli tradizionali, che solitamente richiedono post-elaborazione aggiuntiva per affinare i risultati.

Nel 2021 l'articolo [17] ha proposto un nuovo metodo che combina logiche fuzzy, gradienti di Sobel, e istogrammi cumulativi per migliorare la rilevazione delle transizioni sia improvvise che graduali. Il metodo si basa su un'analisi statistica tramite la misura di deviazione standard relativa (RSD) sugli istogrammi MCSH (Mean Cumulative Sum Histogram), permettendo di identificare le transizioni in modo più accurato.
L'analisi delle differenze tra i valori di RSD tra i fotogrammi adiacenti permette di rilevare transizioni improvvise, mentre i modelli di transizioni graduali vengono estratti confrontando le differenze relative di RSD all'interno dei gruppi di fotogrammi. Sebbene il metodo abbia mostrato buone prestazioni sui dataset di riferimento, la ricerca futura si concentrerà sulla riduzione della complessità computazionale e sull'esplorazione di altri descrittori delle caratteristiche visive.

3.1.2 Tool esistenti per la Shot Boundary Detection

Vengono elencati I diversi tipi di tool esistenti per la Shot Boundary Detection:

a. Basati su analisi video

- **PySceneDetect:** utilizza variazioni nella luminanza e nei pixel tra fotogrammi consecutivi per identificare tagli e transizioni visive, come dissolvenze e dissolvenze incrociate. Questo approccio, pur basandosi su regole semplici, risulta molto efficace per segmentare inquadrature in modo rapido [18]

- **FFmpeg:** con il filtro scene, analizza il video fotogramma per fotogramma per individuare cambiamenti netti nella luminosità, offrendo una soluzione versatile e personalizzabile per analisi rapide.[19]

b. Basati su Machine Learning e Deep Learning

- **TransNet V2:** Sviluppato da Google, è una rete neurale progettata specificamente per il rilevamento delle transizioni tra inquadrature. Rispetto agli approcci tradizionali, è in grado di gestire meglio scenari complessi, come transizioni graduali o video con variazioni caotiche. [16]
- **OpenCV:** Combinando OpenCV con algoritmi personalizzati, è possibile costruire soluzioni su misura che sfruttano analisi visiva avanzata e metodi di soglia per rilevare i confini. OpenCV è una libreria open-source che offre una vasta gamma di strumenti per l'elaborazione delle immagini e dei video, rendendola ideale per sviluppare soluzioni personalizzate per la shot boundary detection[20]

c. Software di Editing Video

- **DaVinci Resolve e Adobe Premiere :** Entrambi i software includono una funzione che rileva automaticamente i tagli tra inquadrature, consentendo agli editor di individuare rapidamente le transizioni. Tuttavia, il funzionamento non dà sempre risultati corretti. DaVinci Resolve utilizza l'analisi degli istogrammi per rilevare i tagli, mentre Adobe Premiere Pro si avvale della tecnologia AI di Adobe Sensei[21]

3.2 Scene boundary detection (SBD)

3.2.1 Stato dell'Arte della Scene Boundary Detection

La Scene Boundary Detection (SBD) è una tecnica fondamentale per segmentare i video in unità significative, come scene o capitoli, facilitando l'analisi del contenuto video. La ricerca in questo campo si è evoluta notevolmente nel corso degli anni, passando da metodi basati su caratteristiche visive a approcci più complessi che

combinano segnali multimodali e tecniche di deep learning. Di seguito è presentata una panoramica delle principali tappe di sviluppo della SBD, esaminando i metodi proposti negli articoli più influenti.

Nel 2003, Zeeshan Rasheed e Mubarak Shah [22] hanno introdotto uno dei primi approcci sistematici per il rilevamento dei confini di scena, focalizzandosi su film e programmi televisivi. Il loro metodo si basa sull'analisi delle caratteristiche visive dei frame, come il colore, il movimento e l'intensità, per identificare le transizioni tra scene. Utilizzando un'analisi statistica, hanno rilevato i cambiamenti improvvisi nelle caratteristiche visive, che indicano i confini tra scene. Sebbene semplice rispetto agli approcci moderni, questo metodo è stato fondamentale per lo sviluppo successivo della SBD, creando una base per identificare i confini di scena attraverso metriche visive.

Nel 2005, Chong-Wah Ngo et al. [23] hanno proposto un approccio innovativo che sfrutta i grafi per la segmentazione delle scene. Ogni shot video viene trattato come un nodo, mentre le relazioni tra gli shot (come la somiglianza visiva) vengono rappresentate come archi. Utilizzando tecniche di clustering e ottimizzazione, il loro metodo raggruppa gli shot in scene significative, identificando automaticamente i confini di scena attraverso la modellizzazione grafica. Questo approccio ha mostrato il potenziale di trattare i video come strutture complesse e ha aperto la strada all'uso di rappresentazioni più sofisticate dei contenuti video.

Chasanis et al. [24] hanno introdotto un metodo che combina il clustering degli shot con l'allineamento delle sequenze. In questo approccio, gli shot video vengono raggruppati in cluster in base alla loro similarità, mentre le sequenze di shot vengono allineate temporalmente per individuare le transizioni tra scene. La principale innovazione di questo lavoro è stata l'uso dell'allineamento delle sequenze, che consente di identificare in modo più preciso la continuità narrativa tra gli shot. L'allineamento aiuta a ridurre gli errori nei confini di scena, affrontando le difficoltà legate alla variabilità dei contenuti visivi tra shot adiacenti.

Nel 2013, Del Fabro e Böszörmenyi [25] hanno scritto una rassegna completa sull'evoluzione della SBD, descrivendo le tecniche utilizzate fino a quel momento e identificando le sfide future. Hanno sottolineato che i metodi tradizionali, che si basano principalmente su caratteristiche visive, non erano sufficienti per affrontare la complessità dei video moderni. Hanno evidenziato la necessità di approcci multimodali, che integrano segnali visivi, audio e semantici, per migliorare l'accuratezza del rilevamento. Questa rassegna ha spinto la comunità di ricerca a esplorare nuovi metodi che combinano più fonti di informazione per migliorare la segmentazione delle scene.

Nel 2015, Baraldi et al. [26] hanno introdotto l'uso di reti neurali profonde per il rilevamento dei confini di scena, proponendo una rete siamese per confrontare gli shot video. Le reti siamese sono composte da due reti parallele che condividono gli stessi pesi, progettate per imparare rappresentazioni simili per gli shot video simili. Questo approccio ha migliorato la capacità di rilevare i confini delle scene, poiché le reti erano in grado di apprendere le relazioni tra gli shot in modo più robusto rispetto ai metodi basati su caratteristiche manuali. Inoltre, le reti neurali profonde hanno consentito di gestire meglio la variabilità dei contenuti video, adattandosi automaticamente ai cambiamenti nei dati.

Nel 2022, Islam et al. [27] hanno proposto un approccio basato su transformer, un tipo di architettura di deep learning che ha rivoluzionato il trattamento dei dati sequenziali. Il loro metodo utilizza un modello di stato-spazio per catturare le dipendenze temporali tra gli shot, migliorando l'identificazione delle transizioni di scena. I transformer sono particolarmente efficaci nell'apprendere relazioni a lungo raggio nei dati temporali, il che li rende adatti per l'analisi dei video. Questo approccio ha superato le limitazioni dei metodi precedenti, che si basavano su RNN o CNN, mostrando un miglioramento significativo in termini di accuratezza e efficienza.

Shixing Chen et al. [28] hanno esplorato un approccio di apprendimento auto-supervisionato contrastivo per il rilevamento delle scene. In questo metodo, un modello apprende a riconoscere i confini delle scene confrontando shot simili e

dissimili in modo da sviluppare rappresentazioni più significative senza la necessità di dati etichettati. Questo approccio contrastivo riduce significativamente la dipendenza da grandi dataset annotati, rendendo il processo di training più economico e scalabile. L'uso dell'apprendimento auto-supervisionato ha anche reso il modello più robusto, adattandosi meglio a una varietà di contenuti video.
2023 – Multimodal High-Order Relation Transformer for Scene Boundary Detection
L'ultimo sviluppo significativo nella SBD è stato presentato da Wei et al. nel 2023 [29] con il Multimodal High-Order Relation Transformer. Questo approccio combina segnali multimodali (visivi, audio e semantici) per modellare relazioni complesse tra gli shot. Il modello integra due componenti principali: un encoder ad alto ordine e un decoder adattivo. L'encoder è responsabile di catturare le relazioni complesse tra gli shot, migliorando l'accuratezza delle correlazioni e riducendo i rumori dovuti a caratteristiche imperfette. Il decoder, invece, è progettato per raggruppare dinamicamente gli shot all'interno delle stesse scene, focalizzandosi su pattern di transizione tra scene piuttosto che sull'apparenza visiva. Questo approccio multimodale e adattivo ha mostrato risultati eccezionali nei benchmark, superando i metodi precedenti in termini di precisione e capacità di generalizzazione.

3.2.2 Tool esistenti per la Scene Boundary Detection

a. Basati su Heuristics e Regole

- **PySceneDetect:** può essere ampliato con tecniche di clustering, permettendo di raggruppare inquadrature consecutive con caratteristiche visive simili e trasformarle in segmenti di scene. Questo approccio è utile per una segmentazione di base, ma può risultare limitato in contesti più complessi.[18]

b. Basati su Analisi Multimediale Avanzata

- **Google Cloud Video Intelligence API:** Utilizza sia il contenuto visivo che l'audio per segmentare un video in scene, integrando metadati semantici

per individuare cambiamenti di contesto. Questo approccio permette una segmentazione più precisa e contestuale[30]

c. Basati su Deep Learning

- **CLIP:** sviluppato da OpenAI, collega immagini e testo per identificare cambiamenti concettuali tra inquadrature. Analizzando descrizioni testuali generate per inquadrature consecutive, può rilevare confini di scena basati su variazioni significative, ad esempio da una strada trafficata a un parco tranquillo. Inoltre, può essere integrato con modelli di riconoscimento delle azioni per migliorare la precisione della segmentazione combinando cambiamenti visivi e concettuali.[31]

d. Software di Editing Video

- **Adobe Premiere Pro e DaVinci Resolve:** Offrono soluzioni semi-automatiche per segmentare manualmente scene complesse, supportando il clustering delle inquadrature. Premiere Pro utilizza la tecnologia AI di Adobe Sensei per rilevare automaticamente i tagli delle scene, mentre DaVinci Resolve si basa su analisi istogrammiche e offre molte opzioni di personalizzazione [21]

3.3 Content Source-Based Segmentation

La *content source-based segmentation* è una tipologia di segmentazione che abbiamo introdotto e denominato, poiché non è presente in letteratura, per descrivere un metodo utile a segmentare in modo automatico programmi come *Blob*, costituiti da contenuti audiovisivi di diversa origine. Tuttavia, questa segmentazione può essere applicata anche ad altri tipi di contenuti audiovisivi composti da clip provenienti da fonti diverse, come:

- Telegiornali, che utilizzano clip da varie fonti, come corrispondenti esterni, agenzie di stampa e immagini d'archivio.
- Documentari, che combinano materiali d’archivio, interviste e ricostruzioni.
- Film e serie TV, in cui possono essere integrate clip esterne, come video amatoriali o materiali d’archivio.

- Video remix su piattaforme social come YouTube o TikTok, dove i creators combinano contenuti da diverse fonti per creare nuove narrazioni.

Non esistono studi specifici nella letteratura scientifica che trattino esclusivamente questo tipo di segmentazione. Tuttavia, la *scene boundary detection* può essere impiegata per segmentare contenuti audiovisivi con clip di diversa origine, poiché ogni frammento proveniente da una fonte diversa può essere interpretato come una scena. Va notato, però, che in questi casi un singolo frammento può contenere più scene.

Grazie agli ultimi progressi nella *scene boundary detection*, questa metodologia può fornire risultati efficaci anche per segmentare contenuti in base alla fonte e al tipo di contenuto (*content source-based segmentation*). Questo perché i frammenti di contenuti provenienti da fonti diverse presentano generalmente differenze visive e sonore significative. Solo in rari casi, clip accostate possono mostrare somiglianze tali da complicare la segmentazione automatica.

Per la nostra ricerca abbiamo deciso di utilizzare come strumenti i linguaggi multimodali in particolare Gemini come spiegato nella sezione del progetto di ricerca.

4. Progetto di ricerca

Il progetto di ricerca si basa su una domanda principale:

- **RQ**: Può un algoritmo di intelligenza artificiale essere sviluppato per distinguere lo stile di montaggio di Blob da uno randomico?

Per rispondere a questa domanda, la prima parte del progetto si è concentrata sulla creazione di un dataset (D) per allenare l'intelligenza artificiale. Questo dataset include sia clip con tagli che abbiamo definito di "Blob" sia clip con tagli definiti di "non Blob".
Le clip di "Blob" contengono i tagli originali del programma, mentre le clip di "non Blob" presentano tagli simulati, creati per differire dallo stile di montaggio di Blob. Entrambi i tipi di clip sono stati generati automaticamente utilizzando algoritmi di segmentazione e concatenazione video.
Successivamente, verrà utilizzato il modello di intelligenza artificiale multimodale Gemini per valutare la sua capacità di distinguere tra le clip di "Blob" da quelle di "non Blob".
In questo capitolo sarà descritto il processo del progetto, dalla creazione dei database e dei test tramite prompt engineering, fino al calcolo dei risultati e alla valutazione finale. Nello specifico, la prima parte tratterà la fase di creazione del database dei tagli di 'Blob', mentre la seconda parte affronterà la costruzione del database per i tagli di 'non Blob' e i risultati dell'esperimento sulla capacità di Gemini di distinguere tra i due tipi di tagli, 'Blob' e 'non Blob'.

4. 1 Rilevazione tagli “blob”

4.1.1 Definizione task

L’obiettivo di questa prima fase del progetto sperimentale è la creazione di un dataset di clip video tratte dalle puntate del programma televisivo “Blob”.
Le puntate sono state reperite su YouTube per costituire la base del database.
Tuttavia, per formare il dataset è necessario segmentare ogni puntata in numerose clip di pochi secondi, seguendo i tagli di montaggio caratteristici del programma.
Considerando, inoltre, lo stile di "Blob" ogni puntata contiene frammenti che vengono montati insieme e che provengono da contenuti di diversa origine come programmi televisivi, film o contenuti tratti da social media.
Serve quindi un algoritmo capace di individuare non solo i tagli tra le varie scene di un singolo programma, ma anche di distinguere tra tagli che definiamo editoriali e tagli formali.
I tagli **formali** sono tagli di montaggio tra lo stesso tipo di contenuto mediale, ad esempio tra diverse inquadrature di uno stesso programma.
I tagli che definiamo **editoriali** sono invece tagli tra contenuti mediali diversi, ad esempio, tra due clip tratte da due programmi televisivi diversi.
Questi ultimi rappresentano i veri "tagli di Blob," scelti appositamente dai montatori del programma.
L'obiettivo primario di questa fase è dunque automatizzare l'individuazione dei tagli editoriali per creare un dataset dei cosiddetti "tagli di Blob," essenziale per la successiva analisi sperimentale del progetto.

Sono riportati alcuni esempi di tagli editoriali e formali.

Figura 2: Esempio di taglio editoriale tra due programmi televisivi diversi

Figura 3: Esempio di taglio editoriale tra due programmi televisivi diversi

Figura 4: Esempio di taglio formale tra due inquadrature tratte da un film

Figura 5: Esempio di taglio formale tra due inquadrature tratte da un programma televisivo.

Considerando le tre tipologie di segmentazione video, definite nel capitolo 3 dello stato dell'arte sugli algoritmi di segmentazione, possiamo dire che l'obiettivo di questa prima fase del progetto è cercare di trovare algoritmi per rilevare la "Content Source-Based Segmentation". Siccome Blob è un tipo di programma che è composta da clip di diversi tipi di riprese perché di diversa origine

4.1.2 Creazione Dataset

Per segmentare, in una prima fase, le puntate di Blob in diverse clip e creare il dataset, è stato utilizzato un algoritmo di "Scene Boundary Detection" su Google Colab, chiamato "Scene detect," basato su Python e OpenCV. Questo algoritmo rileva i cambi di scena e suddivide ogni puntata, della durata di circa 10-20 minuti, in sottoclip di 10 secondi ciascuna. In particolare, per ogni cambio di scena

rilevato, l'algoritmo estrae i 5 secondi precedenti e successivi, ottenendo così diverse clip di 10 secondi dalle puntate originali.

Le clip segmentate sono state denominate "subclip" nel database, mentre le puntate originali sono indicate come "clip." Ogni subclip è stata etichettata con un identificativo alfanumerico che include un prefisso comune per tutte le clip estratte dalla stessa puntata o estratto di *Blob*, e un suffisso numerico che indica il secondo in cui la scena è stata rilevata (ad esempio, 'k7e3-NXqe8_28_37', con 'k7e3-NXqe8' come prefisso che identifica la chiave della stessa puntata)."

Oltre alle clip e subclip, sono stati creati anche immagini di tipo "carosello." Questi media, ottenuti tramite un altro algoritmo Python, presentano una sequenza di frame delle subclip in un'immagine composita. Ogni carosello ha una risoluzione di 3840x288 e contiene 10 frame estratti dalle subclip di 10 secondi. I file carosello sono stati nominati con lo stesso identificativo delle rispettive subclip.

Infine, tutte le clip tratte dalle puntate di Blob, le subclip e i relativi caroselli sono stati organizzati in un database su Dropbox nelle rispettive cartelle, consentendo l'accesso per ulteriori analisi e calcoli utilizzando Google Colab.

4.1.3 Test preliminari per scelta algoritmo

Dopo aver creato il database, è necessario uno strumento in grado di rilevare e distinguere, tra le diverse subclip, i tagli editoriali e formali. Attualmente non esiste un algoritmo che sia in grado di rilevare la "Content Source-Based Segmentation". Per questo, sono stati testati vari modelli multimodali di intelligenza artificiale (MLLM) come ChatGPT, Gemini e Video-LLaVA.

Per ciascun modello, è stato formulato un prompt specifico per identificare i tipi di tagli, utilizzando come input la subclip di 10 secondi estratta da Blob, in formato video o carosello. Nei test preliminari, ChatGPT e Gemini hanno mostrato risultati più accurati nella rilevazione dei tagli, mentre Video-LLaVA ha fornito spesso risultati imprecisi, probabilmente a causa della sua natura non industriale e del numero limitato di parametri rispetto a Gemini e ChatGPT.

Nella nostra ricerca, abbiamo scelto di utilizzare **Gemini 1.5 Flash** come modello per l'analisi video multimodale. Questo modello è stato selezionato poiché, secondo il benchmark Video-MME [32], ha dimostrato di essere il miglior modello commerciale nell'elaborazione video, con un'accuracy media del 75%, superiore al 71,9% di GPT-4. Lo studio evidenzia, inoltre, che l'integrazione di sottotitoli e tracce audio migliora significativamente la comprensione dei contenuti video

4.1.4 Prompt Engineering

Prompt	Immagine	Video	OUTPUT(gemini)
Saresti in grado di descrivermi se tra i vari frame ci sono tagli effettivi tra diverse puntate o contenuti mediali , RISPONDI in formato JSON valido contenente una lista di elementi corrispondente ognuno a uno spezzone cos [{"Descrizione programma": il video sembra estratto da un social , "Frame start:0", "Frame end:10}, {"Descrizione programma": il video sembra estratto da un talk show, "Frame start:15", "Frame end:30}, ..] In tutti i tagli effettivi di montaggio tra i vari frame del video dire poi sempre in formato JSON : "1" solo se è un taglio fra due programmi o contenuti mediali diversi in questo caso TAGLIO EDITORIALE "0" solo se è un taglio appartenente a uno stesso programma o contenuto mediale in questo caso TAGLIO FOR	carouselP2_3_0.jpg		"Frame end": 80, "Taglio": 0 }, { "Descrizione programma": "il video sembra estratto da un talk show", "Frame start": 80, "Frame end": 90, "Taglio": 0 }]
"dato questo video che contiene vari tagli di montaggio tra contenuti mediali diversi o tra diverse inquadrature d dammi come output una lista di elementi JSON che per ciascun taglio mi indica se il taglio è editoriale tra conte [{"Frame start":0,"Frame end":3,"Descrizione programma":"Notiziario","Tipo di taglio":"0"// TAGLIO FORMALE}, {"Frame start":4,"Frame end":6,"Descrizione programma":"Notiziario","Tipo di taglio":"1"// TAGLIO EDITORIALE}	carouselP2_3_1_comp.jpg	BLOBP2_4_comp.mov	[{"Frame start": 0, "Frame end": 3, "Descrizione programma": "Notiziario", "Tipo di taglio": "0"}, {"Frame start": 3, "Frame end": 6, "Descrizione programma": "Notiziario", "Tipo di taglio": "1"}, {"Frame start": 6, "Frame end": 7, "Descrizione programma": "Partita di calcio", "Tipo di taglio": "1"}, {"Frame start": 7, "Frame end": 8, "Descrizione programma": "Partita di calcio", "Tipo di taglio": "0"}, {"Frame start": 8, "Frame end": 9, "Descrizione programma": "Partita di calcio", "Tipo di taglio": "0"}, {"Frame start": 9, "Frame end": 11, "Descrizione programma": "Partita di calcio", "Tipo di taglio": "0"}]
dato questo video che contiene vari tagli di montaggio tra contenuti mediali diversi o tra diverse inquadrature de dammi come output una lista di elementi JSON che per ciascun taglio mi indica se il taglio è editoriale tra conte [{"Frame start":0,"Frame end":3,"Descrizione programma":"Notiziario","Tipo di taglio":"0"// TAGLIO FORMALE}, {"Frame start":4,"Frame end":6,"Descrizione programma":"Notiziario","Tipo di taglio":"1"// TAGLIO EDITORIALE	carouselP2_3_1_comp.jpg	BLOBP2_4_comp.mov	{"Frame start":51,"Frame end":75,"Descrizione programma":"Notiziario","Tipo di taglio":"0"// TAGLIO FORM {"Frame start":76,"Frame end":100,"Descrizione programma":"Notiziario","Tipo di taglio":"1"// TAGLIO EDI {"Frame start":101,"Frame end":125,"Descrizione programma":"Calcio","Tipo di taglio":"0"// TAGLIO FORM {"Frame start":126,"Frame end":150,"Descrizione programma":"Calcio","Tipo di taglio":"0"// TAGLIO FORM {"Frame start":151,"Frame end":175,"Descrizione programma":"Calcio","Tipo di taglio":"0"// TAGLIO FORM {"Frame start":176,"Frame end":200,"Descrizione programma":"Calcio","Tipo di taglio":"0"// TAGLIO FORM {"Frame start":201,"Frame end":225,"Descrizione programma":"Calcio","Tipo di taglio":"0"// TAGLIO FORM {"Frame start":226,"Frame end":250,"Descrizione programma":"Calcio","Tipo di taglio":"0"// TAGLIO FORM

Figura 6: Esempio della tabella Excel con i vari prompt e gli output

Il prompt engineering è la scienza della progettazione e dell'ottimizzazione dei prompt per guidare i modelli di AI, in particolare gli LLM, verso la generazione delle risposte desiderate.

Nel nostro caso, l'obiettivo è identificare quale tipo di prompt risulti più performante nella rilevazione dei tagli editoriali di 'Blob', testando diversi casi possibili.

A tal fine, è stato creato un documento Excel contenente i vari prompt testati, i tipi di media utilizzati (immagini o video) e i risultati ottenuti con Gemini, così da tenere traccia dei primi test dei prompt, come illustrato nella figura 6.

Per "prompt" si intende non solo il testo di chat utilizzato per interagire con l'algoritmo, ma anche il tipo di media allegato: video (subclip) o immagini (carousel) creando quindi diverse combinazioni di configurazioni. Ho testato inizialmente solo il testo come prompt, per poi provare diverse combinazioni con i vari tipi di media.

Riporto in seguito alcuni esempi dei prompt utilizzati e rispettivi output:

Esempi di prompt:

Prompt 1:

- Testo 1: “Dato questo video che contiene vari tagli di montaggio tra contenuti mediali diversi o tra diverse inquadrature dello stesso contenuto mediale.
  dammi come output una lista di elementi JSON che per ciascun taglio mi indica se il taglio è editoriale tra contenuti mediali diversi o formale all'interno dello stesso contenuto mediale. Segui il seguente esempio. [{"Frame start":0,"Frame end":3,"Descrizione programma":"Notiziario","Tipo di taglio":"0"// TAGLIO FORMALE},
  {"Frame start":4,"Frame end":6,"Descrizione programma":"Notiziario","Tipo di taglio":"1"// TAGLIO EDITORIALE},..]”
- Media 1 :subclip

Output 1: [{"Frame start": 0, "Frame end": 3, "Descrizione programma": "Notiziario", "Tipo di taglio": "0"}, {"Frame start": 3, "Frame end": 6, "Descrizione programma": "Notiziario", "Tipo di taglio": "1"}, {"Frame start": 6, "Frame end": 7, "Descrizione programma": "Partita di calcio", "Tipo di taglio": "1"}, {"Frame start": 7, "Frame end": 8, "Descrizione programma": "Partita di calcio", "Tipo di taglio": "0"}, {"Frame start": 8, "Frame end": 9, "Descrizione programma": "Partita di calcio", "Tipo di taglio": "0"}, {"Frame start": 9, "Frame end": 11, "Descrizione programma": "Partita di calcio", "Tipo di taglio": "0"}]

Prompt 2:

- Testo 2: “Dato questa immagine formato carosello che rappresenta frame di un video che contiene vari tagli di montaggio , classificare i tagli in due categorie: * Tagli editoriali (1): Tagli tra contenuti mediali diversi(esempio diverse puntate di un programma)
  * Tagli formali (0): Tagli tra diverse inquadrature dello stesso contenuto mediale.
  Restituisci come output una lista di elementi JSON, dove ciascun elemento indica il tipo di taglio. Usa il seguente formato come esempio: [0, // TAGLIO FORMALE 1, // TAGLIO EDITORIALE ...] Output atteso: una lista di oggetti JSON che specificano se ciascun taglio è un taglio editoriale (1) o un taglio formale (0).
- Media 2: carousel

Output 2: [0,0,1,0,0,0,0,0]

Prompt 3:

- Testo 3: “ Ti sto fornendo un video di circa 20 secondi. Il video è un estratto di Blob, una serie di clip prese da diversi programmi televisivi, estratti di film, video social e altri contenuti multimediali rilevanti. Le clip sono unite attraverso il montaggio per creare nuovi significati.Il mio obiettivo è capire se nell’estratto che ti sto dando sono presenti più clip provenienti da contesti diversi, unite insieme tramite montaggio (ad esempio, una porzione di film seguita da uno spezzone di programma televisivo, due programmi

televisivi consecutivi, un contenuto social seguito da un programma televisivo, ecc.).
Potrebbe anche essere che non ci siano clip diverse e il video sia costituito da un'unica clip.
Se noti clip diverse, rispondi con "1" e indica quali clip diverse noti, fornendo una motivazione. Ad esempio: "1. Nei primi frame si nota un reel social (l'aspect ratio è di 9:16) seguito da uno spezzone di film." Oppure: "1. Nel video sono presenti 3 diverse clip: due programmi televisivi seguiti da un video social."
Se non noti clip diverse, rispondi con "0". Ad esempio: "0. Non sono presenti clip diverse."

- Rispetta strettamente questo formato di risposta con lo 0 o l'1 in testa.
- Media 3: subclip

Output 3: 0

I prompt sono stati testati manualmente con Gemini più volte, utilizzando diversi tipi di media, per identificare il metodo più efficace nel rilevare i tagli.
Dalle prime analisi manuali, il prompt più performante è quello che restituisce una lista JSON composto da 0 e 1 (caso del "Prompt 2") rispetto ad altri prompt che restituiscono un output binario di 0 o 1 (caso del "Prompt 3") o invece una lista di dizionari che indicano con una chiave il tipo di taglio (caso del "Prompt 1")
Questa prima analisi è completamente manuale, poiché ogni output ottenuto con Gemini viene confrontato con i tagli visibili nei media presenti nel database.
Tuttavia, gli output ottenuti con i prompt più performanti sono in formato JSON e non permettono un'estrapolazione diretta che indichi se vi siano tagli editoriali o meno.
I risultati in formato JSON derivano dal tipo di prompt utilizzato, che richiede espressamente in input una lista JSON. Si è notato che questo approccio risulta più efficace nei risultati rispetto a una semplice richiesta binaria di 0 e 1 nei prompt, come ad esempio chiedere "dimmi se c'è un taglio editoriale o no".
Per ottenere risultati binari immediati, sono state utilizzate altre funzioni di parsing su Python, le quali restituiscono "0" nel caso di un taglio formale e "1" nel caso di un taglio editoriale.

4.1.5 Sistemazione dati per calcolo delle configurazioni

Dopo aver testato i vari prompt, è stato effettuato un lavoro di pulizia del database delle subclip.
Poiché le puntate intere del programma sono state reperite da YouTube attraverso una ricerca per parole chiave, alcune clip sono state eliminate dal database perché non provenivano da Blob, ma da altri programmi simili, risultando quindi non utili per l'analisi del progetto.
Successivamente, è stata condotta manualmente un'ulteriore analisi, visionando tutte le subclip e individuando, per ciascuna, il numero di eventuali tagli formali ed editoriali. Questo processo di annotazione dei dati è stato lungo e completamente manuale, ma necessario per ottenere una corretta valutazione "umana" dei tagli.
Dopo questa scrematura, è stato creato un nuovo documento Excel contenente tutte le coppie di subclip, identificate dal proprio ID alfanumerico, che presentano almeno un taglio formale (segnalato con 0) e un taglio editoriale (segnalato con 1). La tabella riporta ogni coppia di subclip con lo stesso prefisso, poiché provengono dalla stessa puntata di Blob. Per ogni coppia è indicato un output di 0 (taglio formale) e un output di 1 (taglio editoriale), determinato tramite un'analisi manuale come mostrato nella figura 7.

ID	TARGET(1/0)
-k7e3-NXqe8_14_97	0
-k7e3-NXqe8_28_37	1
0AlOI8xoZMA_32_88	0
0AlOI8xoZMA_15_2	1
0MDtvT-H4fk_16_88	0
0MDtvT-H4fk_57_76	1
1fnRYUoPR-Y_28_68	0
1fnRYUoPR-Y_5_6	1
	0
01y7gSAvPoY_5_03	1
2_QWNT7inR4_140_68	0
2_QWNT7inR4_15_36	1

Figura 7: Esempio della tabella Excel con i vari prompt e gli output

L'obiettivo è verificare se questi output, ottenuti dalle annotazioni manuali, corrispondono a quelli generati da Gemini. Per fare ciò, i file contenenti le coppie di clip e i relativi output saranno elaborati in un notebook su Google Colab , che li confronterà con gli output di Gemini tramite algoritmi.

Per eseguire i calcoli sul notebook verrà passata una cartella di file con all'interno la zip dei contenuti carousel e subclip, un file dei prompts, il file index con la tabella dei vari nomi dei file e i target individuati e un file responses con le risposte calcolate con Gemini che viene aggiornato per ogni calcolo.

Successivamente, saranno confrontati i target (output rilevati manualmente) con i predicted (output generati da Gemini) e saranno calcolate, sempre con algoritmi su Google Colab, le metriche di precision, recall, F1 e accuracy. Queste metriche valutano l'efficienza delle varie configurazioni.

Ogni configurazione è caratterizzata da:

- Tipo di media (subclip, carosello o subclip + carosello)
- Tipo di prompt testuale
- Modello di Gemini utilizzato

config	media	prompt
1a	subclip	Ti sto fornendo un video di circa 20 secondi. Il video è un estratto di Blob, una serie di clip prese da
2a	subclip	Ti sto fornendo un video di circa 20 secondi. Il video contiene vari tagli di montaggio tra contenuti
3a	subclip	Ti sto fornendo un video di circa 20 secondi. Il video contiene vari tagli di montaggio, classificare i t
1b	carousel	Ti sto fornendo un'immagine in formato carosello che contiene una sequenza di frame estratti un
2b	carousel	Ti sto fornendo un'immagine in formato carosello che contiene una sequenza di frame estratti un
3b	carousel	Ti sto fornendo un'immagine in formato carosello che contiene una sequenza di frame estratti un
1c	subclip_carousel	Ti sto fornendo 1) un'immagine in formato carosello che contiene una sequenza di frame estratti
2c	subclip_carousel	Ti sto fornendo 1) un'immagine in formato carosello che contiene una sequenza di frame estratti
3c	subclip_carousel	Ti sto fornendo 1) un'immagine in formato carosello che contiene una sequenza di frame estratti

Figura 8: Esempio della tabella Excel con le varie configurazioni

Per il calcolo delle metriche finali sono state considerati 9 configurazioni come illustrato nella *figura 8*. Tuttavia, ogni prompt è stato lanciato 3 volte per avere una maggiore accuratezza nei risultati

Per semplicità nella nostra analisi ogni configurazione sarà rilevata da un identificativo costituito da un numero e da una lettera (ad esempio 1a) e dal tipo di media utilizzato (subclip, carousel o subclip+carousel) come illustrato nella figura 8

4.1.6 Cenni teorici: precision, recall e f1

Precision, recall e la misura F1 sono metriche di performance importanti utilizzate nel riconoscimento di pattern, nel recupero di informazioni, nel rilevamento di oggetti e nella classificazione nel machine learning [33].

Precision (nota anche come valore predittivo positivo) è la frazione di istanze rilevanti tra quelle recuperate da un modello. Si calcola come:

$$\text{Precision} = \frac{\text{Relevant retrieved instances}}{\text{All } \mathbf{retrieved} \text{ instances}}$$

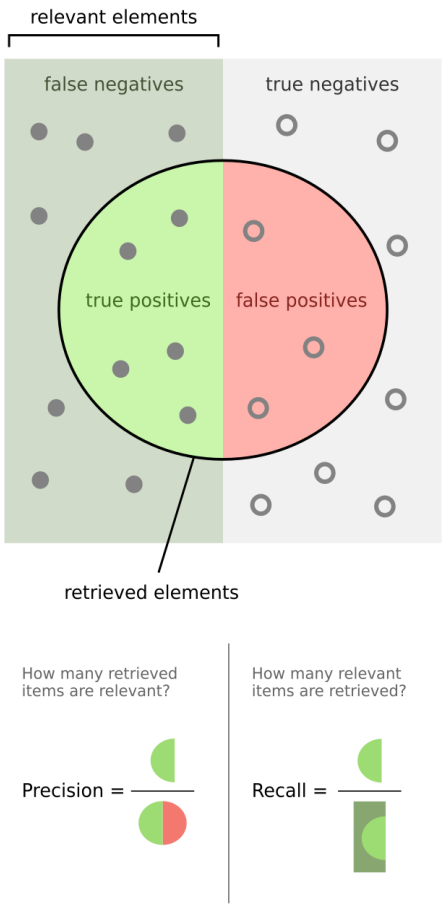

Figura 9: Diagramma che spiega precision e recall

La precision misura essenzialmente la qualità dei risultati recuperati, indicando quante delle voci identificate dal modello sono effettivamente rilevanti. Ad esempio, se un modello identifica otto cani in un'immagine, ma solo cinque di essi sono effettivamente cani (veri positivi) mentre tre non lo sono (falsi positivi), la precisione sarebbe 5/8

Recall (nota anche come sensibilità) è la frazione di istanze rilevanti che sono state correttamente recuperate dal modello. Si calcola come:

$$\text{Recall} = \frac{\text{Relevant retrieved instances}}{\text{All } \mathbf{relevant} \text{ instances}}$$

Il recall misura la quantità di risultati rilevanti recuperati, mostrando quante delle voci effettivamente rilevanti (come i cani nell'immagine) il modello ha identificato con successo. Nell'esempio precedente, se nell'immagine ci fossero 12 cani ma il modello ne avesse identificati solo 5, il recall sarebbe 5/12.

Precision e recall sono entrambe basate sulla rilevanza e sono utili in diversi scenari, specialmente quando si ha a che fare con dati sbilanciati. In alcune situazioni, come la diagnosi medica, potrebbe essere prioritario massimizzare la precision per evitare falsi positivi che potrebbero portare a trattamenti non necessari. In altri casi, come il rilevamento delle frodi, il recall potrebbe essere più importante per garantire che vengano identificate quante più istanze rilevanti possibile (ad esempio, transazioni fraudolente).

La misura F1 combina precisione e recall in una singola metrica calcolando la loro media armonica. È particolarmente utile quando è necessario un equilibrio tra precisione e recall, poiché pesa in modo uniforme entrambe le metriche. La misura F1 si calcola come:

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Questa misura fornisce un unico punteggio che bilancia il compromesso tra precisione e recall, rendendola particolarmente utile nei casi in cui è necessario considerare sia la qualità che la quantità dei risultati restituiti da un modello

4.1.7 Calcolo delle configurazioni

Il calcolo delle configurazioni è stato eseguito su Google Colab utilizzando Python. Il codice implementa un sistema automatizzato per interagire con il modello AI "Gemini" di Google, gestendo file multimediali, generando prompt specifici e inviando richieste al modello, con l'obiettivo di salvare i risultati in modo efficiente. Il sistema segue una serie di fasi coordinate. Prima, definisce i percorsi per i file e le chiavi API, impostando i prompt per vari tipi di media come subclip e carousel. Successivamente, gestisce l'estrazione dei file multimediali dai file ZIP, garantendo che vengano salvati nei percorsi corretti.
Una volta caricati i file, il codice interagisce con l'API di Gemini utilizzando prompt predefiniti e monitorando lo stato delle richieste. I risultati delle elaborazioni vengono salvati in un file *responses* aggiornato per ogni calcolo, contenente le predizioni (predicted) ottenute dal modello
Il processo chiave è un ciclo che esegue operazioni per ciascuna combinazione di modello AI, tipo di media e prompt, verificando se una combinazione è già stata elaborata o fallita. In caso di errore, tenta nuovamente fino a tre volte e registra eventuali fallimenti

A questo punto entra in gioco un'altra parte del codice che introduce funzionalità di pulizia e interpretazione dell'output. Il codice si occupa di pulire le stringhe JSON estratte dall'output del modello, rimuovendo eventuali commenti o caratteri non desiderati.

Una volta pulito il JSON, verifica il tipo di dati contenuti per determinare se si tratta di una lista di numeri o di dizionari, e in base a questo applica una logica differente per interpretare l'output con le specifiche funzioni di parsing precedentemente citate. Questo permette di estrarre informazioni rilevanti dall'output generato e di trasformarle in previsioni numeriche.

Le previsioni così ottenute vengono poi confrontate con i valori target originali per valutare la precisione del modello. Il codice calcola poi le metriche come precision, recall ,F1 e accuracy per valutare le prestazioni del modello su ciascuna configurazione testata.

Queste metriche vengono infine aggregate e salvate in una struttura di dati che permette di analizzare le prestazioni del modello in modo sistematico.

Di seguito sono riportati i commenti alle metriche misurate.

4.1.8 Risposte e analisi risultati

Configurazione ottimale analizzata secondo metriche

Per comprendere quale configurazione risulta ottimale rispetto a ciascuna metrica (precision, recall, F1 e accuracy), è stata analizzata la performance di ogni configurazione in modo dettagliato. Di seguito esaminerò ciascuna metrica separatamente, presentando un'analisi comparativa basata sui risultati ottenuti.

Per ogni metrica, presento un grafico in cui l'asse delle ascisse rappresenta gli identificativi delle configurazioni e il tipo di media utilizzato, mentre l'asse delle ordinate mostra i risultati ottenuti dalle configurazioni per quella specifica metrica.

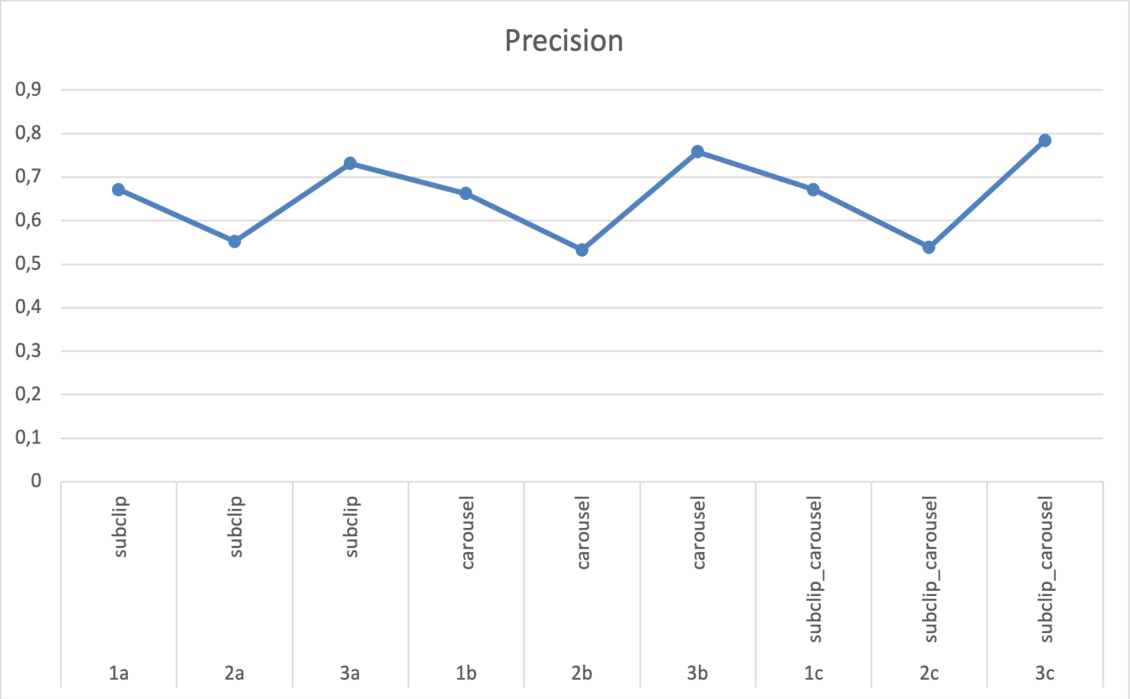

Figura 10: Grafico della precision per le diverse configurazioni

Per quanto riguarda la precision, si può notare dal grafico che la configurazione che sembra funzionare meglio, perché ha il numero più elevato di circa 0,78, è la "3c". Questa configurazione ha le seguenti proprietà

- media : subclip_carousel
- modello gemini: models/gemini-1.5-flash
- prompt:

  Ti sto fornendo 1) un'immagine in formato carosello che contiene una sequenza di frame estratti uno al secondo da un video di circa 20 secondi e concatenati orizzontalmente in ordine sequenziale e 2) il video di circa 20 secondi. Il video contiene vari tagli di montaggio, classificare i tagli in due categorie:

  ** Tagli editoriali (1): Tagli tra contenuti mediali diversi(esempio diverse puntate di un programma)*

  ** Tagli formali (0): Tagli tra diverse inquadrature dello stesso contenuto mediale.*

  Restituisci come output una lista di elementi JSON, dove ciascun elemento indica il tipo di taglio. Usa il seguente formato come esempio:[0, // TAGLIO FORMALE 1, // TAGLIO EDITORIALE ...]

  Output atteso: una lista di oggetti JSON che specificano se ciascun taglio è un taglio editoriale (1) o un taglio formale (0).

Si può notare come anche le altre configurazioni che hanno lo stesso tipo di prompt, identificate infatti dal prefisso “3”, mostrate nella *figura 3i*, hanno i numeri più elevati nella precision e quindi hanno un migliore comportamento.
La configurazione 3a con media subclip ha una precision di 0,73 e la 3b con media carousel di 0,76.

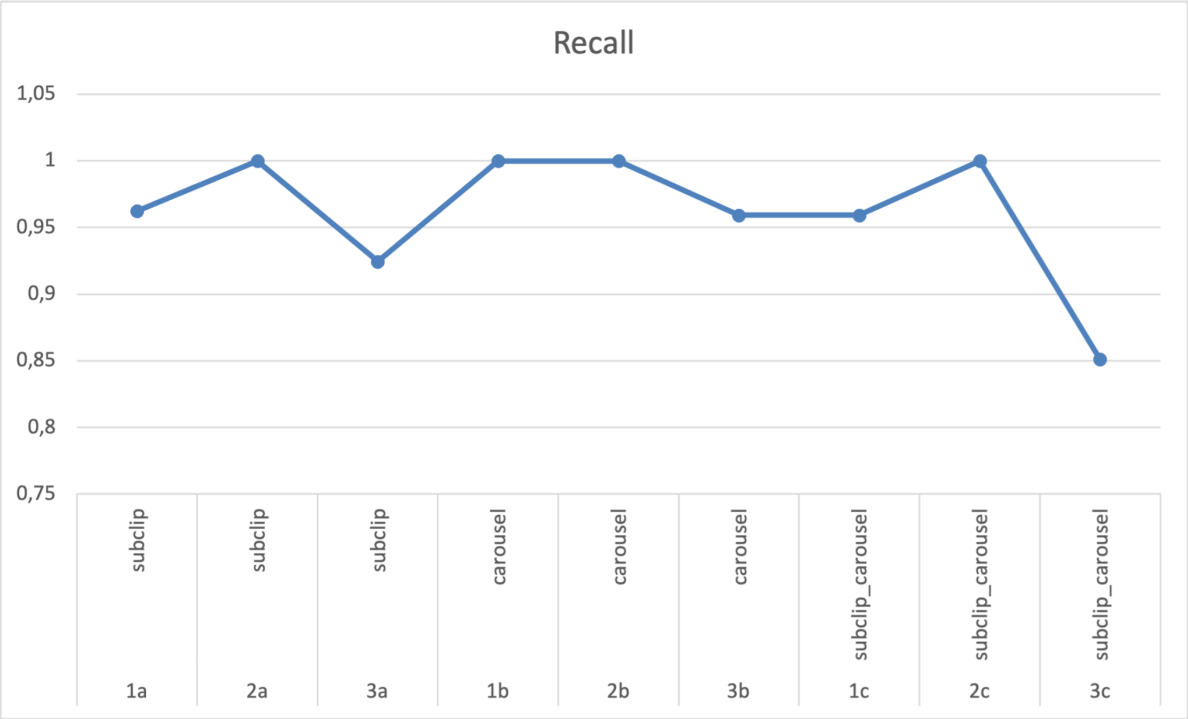

Figura 11: Grafico della Recall per le diverse configurazioni

La Recall ha invece risultati diversi rispetto alla precision e meno conformi. Alcuni risultati infatti hanno valore uno indipendemente dal tipo di media o dal prompt

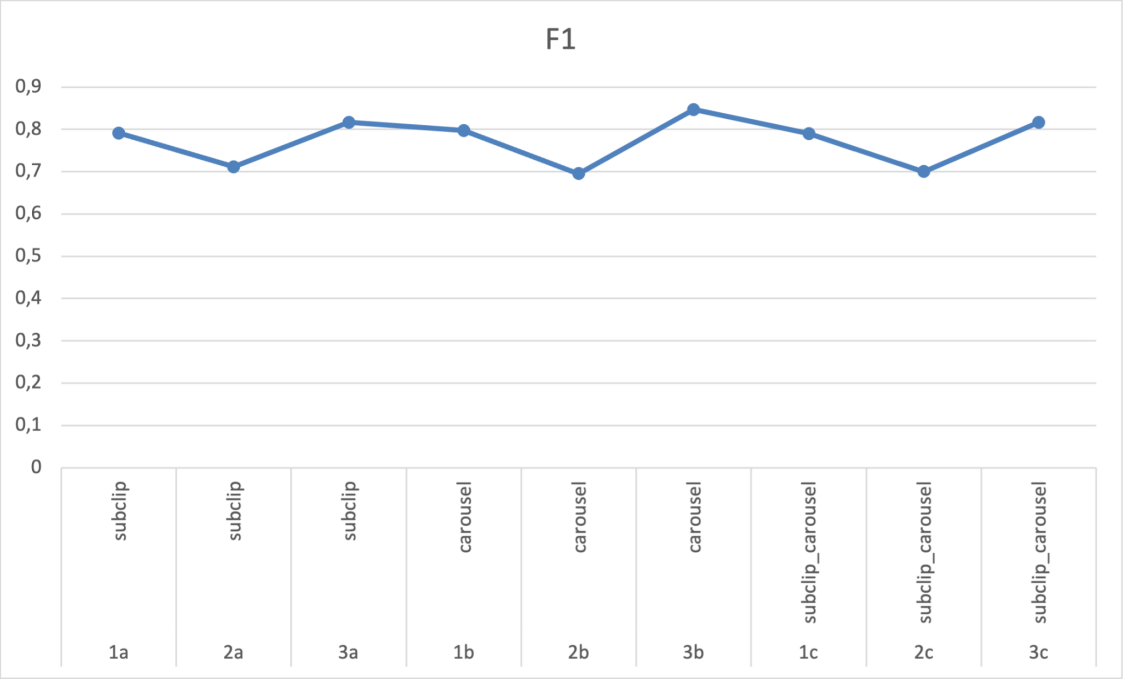

Figura 12: Grafico di F1 per le diverse configurazioni

F1, invece, ha una tendenza simile e conforme alla precision con risultati alti per la configurazione 3a con media subclip e per la configurazione 3c con media subclip-carousel con entrambi un valore di 0,82 . In questo caso, tuttavia, il risultato più alto si ottiene per la configurazione 3b con media di tipo carousel e un valore di circa 0,85 di f1.

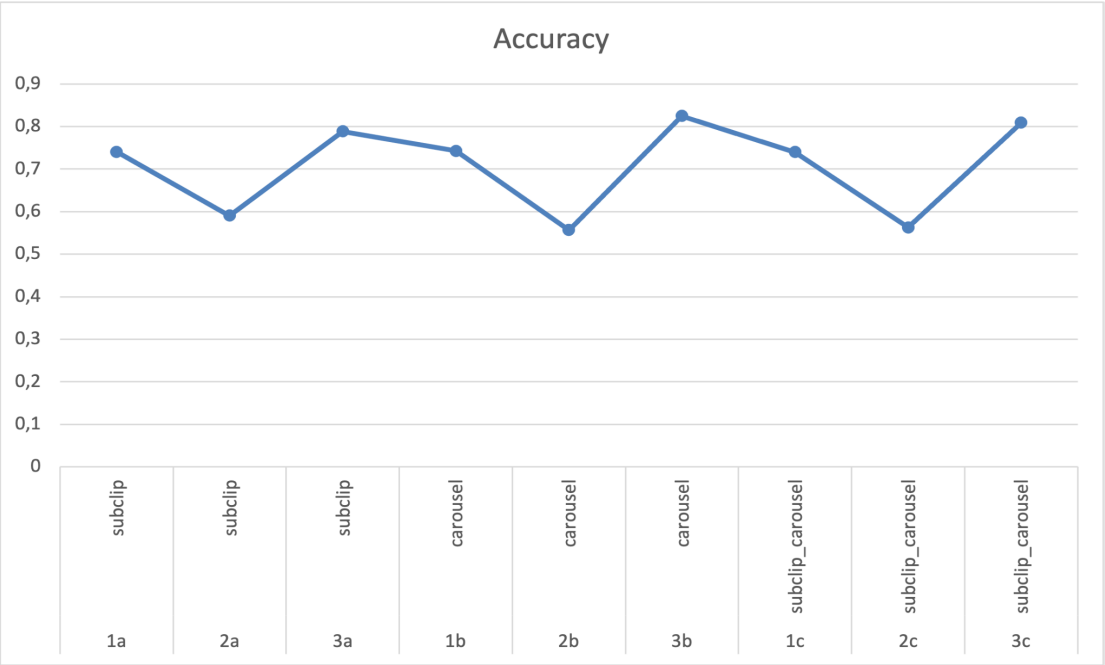

Figura 13: Grafico di Accuracy per le diverse configurazioni

L'accuracy ha una tendenza simile a F1. Le configurazioni più alte sono 3a, 3b, 3c. Il valore più alto si ha per la configurazione 3b con carousel come tipo di media e un valore di 0,82

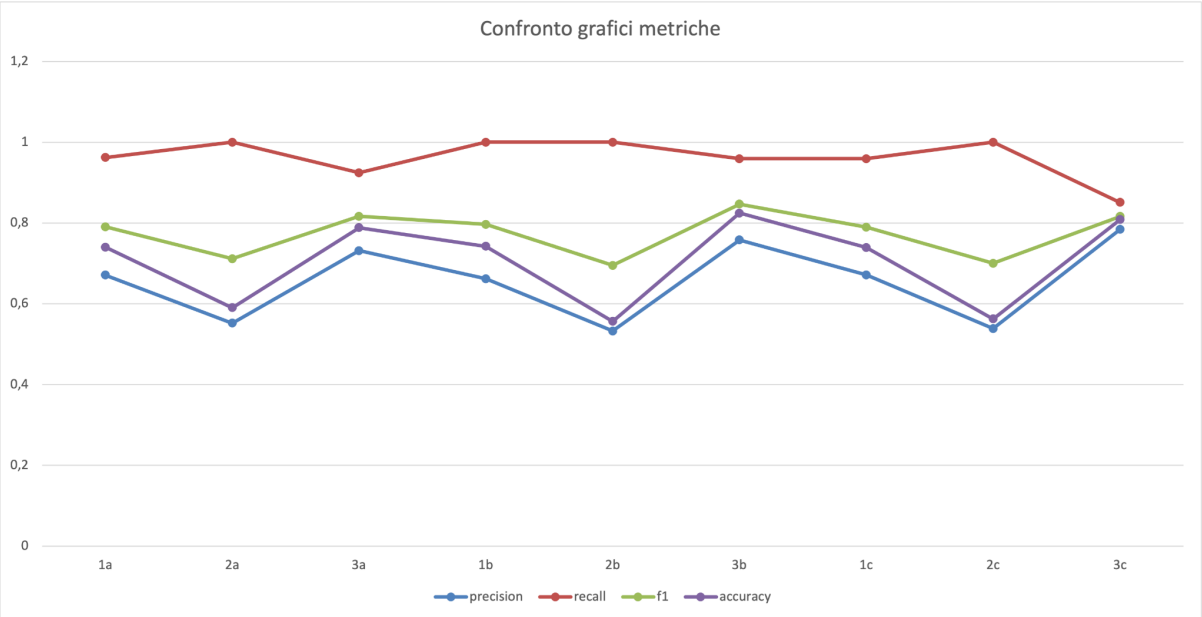

Figura 14: Grafico che mostra andamento delle varie configurazioni

In generale , a parte la recall che ha risultati più altalenanti, i risultati più alti si hanno in media per le configurazioni 3a, 3b e 3c. La 3c, che ha come tipo di media subclip e carousel, sembra essere quella che ha risultati più alti tra le varie

metriche. Probabilmente perchè unendo al prompt l'immagine e il video si dà una maggiore possibilità di correttezza per l'algoritmo.

Influenza del tipo di media

Esaminando l'influenza del tipo di media sui risultati ottenuti, ovvero considerando le configurazioni con solo carosello, carosello+video e solo video, non emergono differenze significative in termini di performance generale attraverso le diverse metriche. Tuttavia, si nota un leggero miglioramento nei risultati medi per le configurazioni che utilizzano i media subclip_carousel e anche carousel. Questo suggerisce che queste configurazioni potrebbero avere un margine di vantaggio, probabilmente per il media carousel ciò è dovuto alla loro natura meno complessa rispetto ai media più pesanti come il subclip mentre per i media subclip_carousel ciò può essere dovuto all'unione dei due media che dà maggiori possibilità di efficienza.

Dimensionality: percentuale risposte ottenute

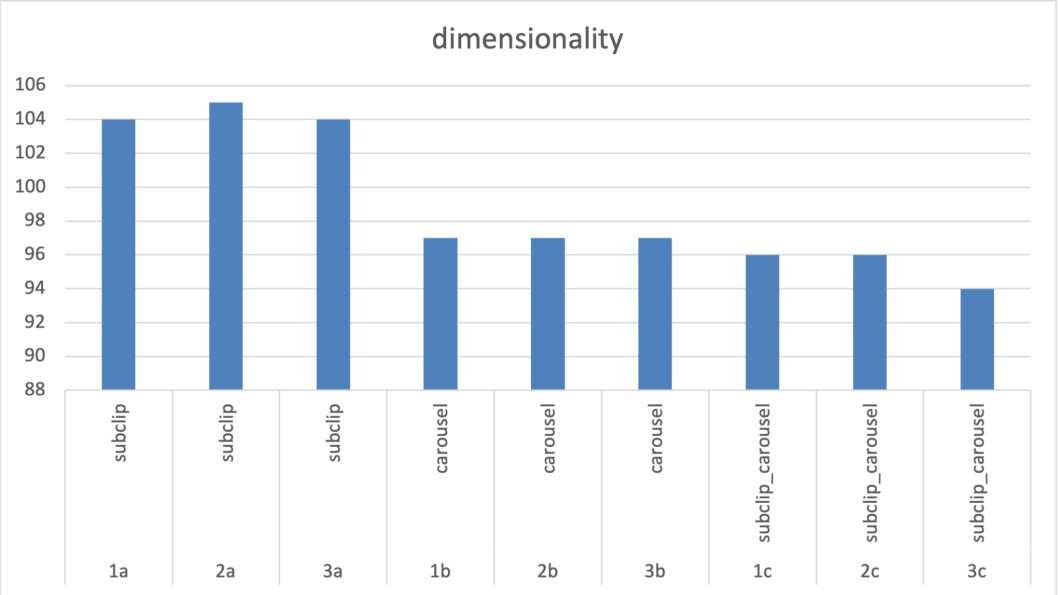

Figura 15: Grafico della dimensionality

Analizzando quale configurazione permette di ottenere il maggior numero di risposte, ci concentriamo sulla metrica della dimensionality, che rappresenta la percentuale di risposte elaborate da Gemini al termine del calcolo di tutte le

configurazioni. Dalla valutazione emerge che la configurazione che genera più risultati è la 2a, con 105 risposte, utilizzando il tipo di media subclip. Anche altre configurazioni che impiegano subclip come media mostrano risultati elevati, suggerendo che questo formato, grazie alla quantità di informazioni che contiene, potrebbe facilitare la generazione di un numero maggiore di risposte.

D'altro canto, i valori più bassi di dimensionality si osservano nelle configurazioni che utilizzano subclip_carousel come tipo di media. In particolare, la configurazione che ottiene il minor numero di risposte è la 3c. Questo dato può indicare che, mentre le subclip da sole forniscono una ricca base informativa, la combinazione di sottoclip e caroselli potrebbe complicare l'elaborazione, portando a un numero inferiore di risultati, probabilmente a causa di un aumento degli errori nell'elaborazione dei dati.

Prompt più efficace per F1 e Accuracy

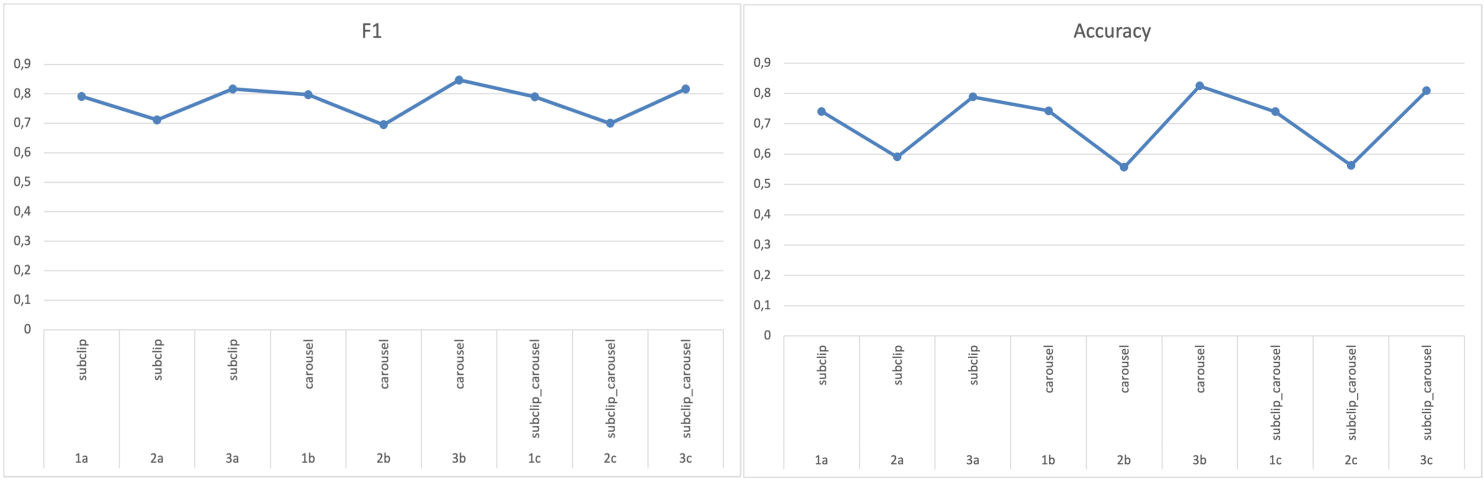

Figura 16: Grafici di F1 e Accuracy

Per identificare il prompt più efficace in termini di F1 e accuracy, si osserva che le configurazioni che ottengono i punteggi più elevati sono la 3a, 3b e 3c. Queste configurazioni condividono un elemento comune decisivo: utilizzano tutte lo stesso prompt testuale, associato ai diversi tipi di media . Questo suggerisce che il prompt in questione, che definiamo di tipo “3” è particolarmente efficace nel massimizzare le performance in termini di F1 e accuracy, consolidando la sua rilevanza rispetto

ad altre varianti analizzate. Viene riportato un esempio del prompt testuale migliore nel caso di input unito a un media di tipo carosello.

Ti sto fornendo 1) un'immagine in formato carosello che contiene una sequenza di frame estratti uno al secondo da un video di circa 20 secondi e concatenati orizzontalmente in ordine sequenziale e 2) il video di circa 20 secondi. Il video contiene vari tagli di montaggio, classificare i tagli in due categorie:
** Tagli editoriali (1): Tagli tra contenuti mediali diversi(esempio diverse puntate di un programma)*
** Tagli formali (0): Tagli tra diverse inquadrature dello stesso contenuto mediale.*
Restituisci come output una lista di elementi JSON, dove ciascun elemento indica il tipo di taglio. Usa il seguente formato come esempio:[0, // TAGLIO FORMALE1, // TAGLIO EDITORIALE ...]
Output atteso: una lista di oggetti JSON che specificano se ciascun taglio è un taglio editoriale (1) o un taglio formale (0).

Questo prompt si è dimostrato il più efficace durante i test su Gemini, probabilmente grazie alla semplicità dell'output, che restituisce una lista JSON di 0 e 1. Pur mantenendo la stessa configurazione testuale nelle configurazioni 3a, 3b e 3c, questo prompt varia solo in base al tipo di media a cui è associato (subclip, carousel o subclip+carousel).

4.1.9 Valutazione finale

In conclusione, non è possibile identificare una configurazione specifica come nettamente superiore alle altre. Tuttavia, dall'analisi delle metriche emerge che le configurazioni con il prefisso '3', caratterizzate dallo stesso tipo di prompt e dalla generazione di un output in formato JSON composto da una lista di 0 e 1, offrono i risultati più soddisfacenti in termini di prestazioni nelle metriche. Non si riscontra una preferenza particolare per un media specifico (subclip, carousel o subclip_carousel).

Sulla base di questa analisi, si è deciso di focalizzarsi esclusivamente sui prompt con il prefisso '3' per ottenere le clip con i tagli di 'Blob'. L'obiettivo di questa prima fase, infatti, era utilizzare algoritmi e modelli di IA per costruire un database con i tagli di "Blob".

Dal database iniziale sono stati filtrati i dati corrispondenti a questo specifico prompt, e si è creata una nuova lista di dizionari contenente l'ID, il valore target e

una previsione basata sui dati filtrati. La previsione considera tutti i nove prompt che hanno come risultato 'predicted: 1', dove ciascun prompt con le sue tre combinazioni è stato lanciato tre volte per ottenere una maggiore preciisone nei risultati (quindi in totale 9).

Dopo il calcolo delle metriche per il nuovo dizionario, si osserva una precision molto alta, pari a 0,89, che corrisponde a circa il 90% e risulta sufficiente per considerarla la metrica principale siccome, in questo caso, si preferisce privilegiare la qualità dei risultati ed evitare falsi positivi. Di conseguenza, metriche come F1 e recall non sono state considerate rilevanti in questo contesto.

Questo risultato è sufficiente per la seconda parte della tesi, in quanto utilizzeremo le configurazioni con questo tipo di prompt specifico come base per creare un altro database di tagli di “non Blob”.

4.2 Rilevazione tagli di “non Blob” e distinzione dai tagli di “Blob”

4.2.1 Definizione task e calcolo frammenti video

Dopo aver rilevato in modo automatico le clip di tagli di "Blob", l’obiettivo della seconda fase del progetto è la creazione di un database di tagli “non Blob”.

Per fare ciò, abbiamo scelto di utilizzare algoritmi di concatenazione video che combinano frammenti ricavati dalla rilevazione dei tagli ”Blob”. Questi frammenti costituiscono le unità di base per creare combinazioni che generino tagli artificiali di "non Blob". I frammenti sono stati ottenuti partendo da un file JSON che contiene un dizionario, in cui ogni voce include il nome del file con l’identificativo alfanumerico e la chiave che identifica da quale puntata è stato estratto. Per ogni clip, il JSON inoltre definisce

- cut_seconds: il secondo della clip in cui l’algoritmo di segmentazione ha rilevato il taglio,
- start_seconds: il secondo della clip in cui inizia la sottoclip,
- end_seconds: il secondo della clip in cui termina la sottoclip.

Questo dizionario contiene le informazioni solo per i file subclip che sono stati considerati come 1(i tagli di Blob) ovvero che contengono tagli editoriali quindi con concatenazioni di file di origine diversa.

Esempio di file JSON


```
{
  "-k7e3-NXqe8_28_37": {
    "cut_frame": 269,
    "cut_seconds": 28.365594165594164,
    "start_seconds": 23.365594165594164,
    "end_seconds": 33.36559416559416,
    "file": "-k7e3-NXqe8"
  },
  "-k7e3-NXqe8_51_99": {
    "cut_frame": 493,
    "cut_seconds": 51.98601458601458,
    "start_seconds": 46.98601458601458,
    "end_seconds": 56.98601458601458,
    "file": "-k7e3-NXqe8"
  },
  "-k7e3-NXqe8_78_24": {
    "cut_frame": 742,
    "cut_seconds": 78.24264264264264,
    "start_seconds": 73.24264264264264,
    "end_seconds": 83.24264264264264,
    "file": "-k7e3-NXqe8"
  },
  ...
}
```


Il file JSON è stato generato utilizzando l'algoritmo di rilevamento tagli: PySceneDetect. "Cut seconds" rappresenta il secondo esatto in cui l'algoritmo ha identificato un taglio, mentre 'start seconds' e 'end seconds' indicano rispettivamente 5 secondi prima e 5 secondi dopo il momento del taglio rilevato.

Per generare i frammenti, è stato implementato un algoritmo in Google Colab utilizzando la libreria MoviePy per l'editing video. Partendo dalla cartella delle clip intere di "Blob" e dal dizionario JSON, l'algoritmo elabora ogni clip per generare una cartella di output denominata output_videos. Per ogni clip, viene creata una prima parte (che va da start_seconds a cut_seconds) e una seconda parte (che va da cut_seconds a end_seconds) tagliate creando così sottoclip che saranno la base per creare le clip di "non Blob". Queste clip vengono nominate con il nome della clip originale e con un suffisso "scene_1" o "scene_2" in modo da identificare se si tratta della prima o seconda parte della clip tagliata.

4.2.2 Creazione clip con tagli di “non blob”

Per la creazione delle clip “non Blob” sono stati usati algoritmi di concatenazione video per combinare i frammenti di tagli di "Blob" precedentemente calcolati e presenti nella cartella "output videos", seguendo logiche precise. In particolare, sono state create due cartelle di clip video “non Blob” chiamate "combined_same" e "combined_cross", così nominate per le specifiche modalità di combinazione dei frammenti.

“Combined_same” combina i frammenti con la seguente logica: ogni prima parte tagliata di una puntata viene unita in modo casuale a una seconda parte diversa della stessa puntata, senza ripetizioni. Questo significa che, una volta formata una combinazione, quella specifica seconda parte non può essere riutilizzata con altre prime parti della stessa puntata.

“Combined_cross” combina i frammenti con un'altra logica: ogni prima parte tagliata di una puntata viene unita in modo casuale a una seconda parte tagliata proveniente da una puntata diversa. Anche in questo caso, non devono esserci ripetizioni, e in totale devono essere create 1033 clip combinate.

In entrambi i casi, la creazione delle clip avviene tramite un codice in Python su Google Colab, utilizzando precise funzioni di concatenazione video e la libreria MoviePy disponibile su GitHub. Le funzioni estraggono le chiavi dai nomi dei file, verificano se il file contiene “scene_1” o “scene_2” e lo aggiungono a un dizionario che raggruppa i video per chiave.

Un’altra funzione crea le combinazioni di clip video. Per ogni chiave nel dizionario delle clip, mescola casualmente le clip di “scene_2”. Per ogni clip di “scene_1”, cerca una clip di “scene_2” non ancora utilizzata. Se trova una combinazione valida, genera un nome per il file di output e verifica se questo esiste già.

4.2.3 Calcolo delle trascrizioni audio

Dopo aver ottenuto le due cartelle con le clip di tagli di “non Blob” generate automaticamente, sono state generate anche le trascrizioni degli audio delle clip, fondamentali per fornire input completi ai modelli di intelligenza artificiale.

Utilizzare solo le clip video come input rischiava infatti di produrre risultati meno accurati.
La procedura per estrarre le trascrizioni audio ha seguito i seguenti passaggi:

- Estrazione dell'audio dal video (da .mp4 a .wav) utilizzando la libreria FFmpeg.
- Separazione dell'audio principale dalla musica o dai rumori di fondo con Spleeter, che isola la traccia vocale dalla musica o dal rumore di fondo e restituisce il percorso della traccia vocale.
- Trascrizione dell'audio vocale principale tramite Whisper, un modello sviluppato da OpenAI. WhisperX esegue inoltre la trascrizione e identifica i diversi speaker.

I trascritti sono stati salvati in un file JSON, dove per ogni clip video di tagli "non Blob" è associata la trascrizione audio. Whisper è stato scelto per la sua accuratezza, anche se i timestamp sono a livello di pronuncia e possono risultare imprecisi. Tuttavia, si tratta di uno dei modelli più efficienti per la trascrizione audio basata su intelligenza artificiale.
Per avere una maggiore accuratezza possibile nei risultati, inoltre, sono stati calcolati i trascritti anche dei frammenti delle prime parti e seconde parte tagliate delle clip di "Blob"

4.2.4 Prompt engineering

Dopo aver ottenuto le trascrizioni, la fase successiva è stata la creazione di prompt. Come nel primo esperimento, sono stati testati vari tipi di prompt testuali su Gemini per valutare la capacità del modello di distinguere tra tagli di "Blob" e tagli "non Blob," e comprendere il programma.

Per migliorare la precisione, sono state incluse nei prompt testuali descrizioni sintetiche del programma, facendo riferimento all'analisi semiotica del linguaggio del programma fatta nel capitolo 1.

Si è cercato in particolare di riassumere brevemente la definizione del programma e la descrizione del suo linguaggio. È stato inoltre testato un tipo di prompt senza

alcuna descrizione del programma in modo da testare le conoscenze pregresse del modello sul programma Blob.

Alla fine, sono stati selezionati tre tipi di prompt testuali:

1. Un prompt senza descrizione del programma che chiede all'IA secondo le sue conoscenze se nella clip è presente un taglio di "Blob" o di "non Blob"
2. Due prompt con brevi descrizioni che analizzano il programma tramite parole chiave mirate e chiedono all'IA se è presente un taglio di "Blob" o di "non Blob"

Riporto nel dettaglio gli esempi dei tipi di prompt testuali utilizzati

- Prompt Index 0: con descrizione del programma:

Blob è un programma satirico italiano in onda su Rai 3 dal 1989, caratterizzato da un montaggio che combina spezzoni di programmi TV, film, pubblicità e contenuti social. Il montaggio alterna analogie (per tema, immagine, suono, forma) e contrapposizioni (di stile, spazio, tempo), creando un conflitto visivo che riflette il concetto situazionista di 'détournement,' ossia il riutilizzo di oggetti fuori dal loro contesto originale pern suscitare una riflessione critica.

Ho un database con due tipi di clip

'Tagli veri' (1): clip prese direttamente da una puntata di Blob senza alterazioni.

'Tagli falsi' (0): clip combinate tra puntate diverse o da momenti diversi della stessa puntata.

Data una clip, identifica se il montaggio è un 'taglio vero' (1) o 'taglio falso' (0) in base alle seguenti definizioni:

Restituisci come output esclusivamente il numero 1 per un 'taglio vero' o il numero 0 per un 'taglio falso'

- Prompt Index 1: senza descrizione del programma

Tenendo a mente le caratteristiche del montaggio del programma tv di Blob

Ho un database con due tipi di clip:

'Tagli veri' (1): clip prese direttamente da una puntata di Blob senza alterazioni.

'Tagli falsi' (0): clip combinate tra puntate diverse o da momenti diversi della stessa puntata.

Identifica se una clip ha un 'taglio vero' (1) o 'taglio falso' (0) in base alle seguenti definizioni:. Restituisci come output esclusivamente il numero 1 per un 'taglio vero' o il numero 0 per un 'taglio falso'

- Prompt Index 2 : senza descrizione del programma

Blob è un programma satirico italiano in onda su Rai 3 dal 1989, noto per il suo innovativo montaggio di clip tratte da programmi TV, film, spot pubblicitari e, più recentemente, dai social media. Il programma "fagocita" e riassembla contenuti mediatici per rivelarne l'ironia o la follia nascosta. Ogni puntata utilizza il montaggio per accostare o contrapporre spezzoni, creando nuovi significati. Blob gioca su analogie e contrasti per offrire una critica satirica su politica, media e società, sfruttando il concetto postmoderno di détournement. Ho un database con due tipi di clip: 'Tagli veri' (1): clip prese direttamente da una puntata di Blob senza alterazioni. 'Tagli falsi' (0): clip combinate tra puntate diverse o da momenti diversi della stessa puntata. Identifica se una clip ha un 'taglio vero' (1) o 'taglio falso' (0) in base alle seguenti definizioni:. Restituisci come output esclusivamente il numero 1 per un 'taglio vero' o il numero 0 per un 'taglio falso'

Un'altra possibile strategia nel prompt engineering sarebbe stata quella di testare prompt specifici, focalizzati su singole caratteristiche del programma Blob, come il montaggio per contrasto, il *détournement* o la satira. Tuttavia, si è preferito utilizzare descrizioni riassuntive del programma per limitare il numero di prompt e, di conseguenza, il consumo di risorse computazionali.

4.2.5 Calcolo delle configurazioni

Per valutare la performance del progetto, il calcolo delle diverse configurazioni è stato eseguito su Google Colab utilizzando Python, seguendo una logica simile a quella della prima fase, dove erano stati valutati i tagli di “Blob”. In questa fase, tuttavia, sono state considerate complessivamente 36 configurazioni.

Le configurazioni derivano dalla combinazione dei seguenti parametri:

1. Tipi di prompt testuali: sono stati utilizzati tre diversi tipi di prompt.
2. Tipi di media: sono stati considerati quattro casi di gestione dei video e delle trascrizioni, basati su come le clip e i transcript sono combinati:
    2.1. Transcript_unito_video_singolo: l’intero video combinato con il transcript unico.
    2.2. Transcript_separato_video_singolo: l’intero video combinato con due transcript separati, uno per ciascuna clip concatenata.
    2.3. Transcript_separato_video_separato: le due clip mantenute separate, ognuna con il proprio transcript.

2.4. Transcript_unito_video_separato: le due clip separate combinate con il transcript unico dell'intero video.

3. Tipi di combinazioni video: le metriche sono state calcolate considerando tre combinazioni di clip:
   3.1. MIX CROSS AND SAME: combinazione di video "combined_same" (clip della stessa puntata) e "combined_cross" (clip di puntate diverse).
   3.2. CROSS: solo clip combinate tra puntate diverse ("combined_cross").
   3.3. SAME: solo clip combinate all'interno della stessa puntata ("combined_same").

Esempio di alcune configurazioni nella figura 17:

Configurazione	Prompt index	Prompt type	Tipo
1a	0	transcript_unito_video_singolo	MIX CROSS AND SAME(0) AGAINST 1
2a	0	transcript_unito_video_singolo	CROSS(0) AGAINST 1
3a	0	transcript_unito_video_singolo	SAME(0) AGAINST 1
4a	0	transcript_separato_video_singol	MIX CROSS AND SAME(0) AGAINST 1
5a	0	transcript_separato_video_singol	CROSS(0) AGAINST 1
6a	0	transcript_separato_video_singol	SAME(0) AGAINST 1
7a	0	transcript_separato_video_separa	MIX CROSS AND SAME(0) AGAINST 1
8a	0	transcript_separato_video_separa	CROSS(0) AGAINST 1
9a	0	transcript_separato_video_separa	SAME(0) AGAINST 1
10a	0	transcript_unito_video_separato	MIX CROSS AND SAME(0) AGAINST 1
11a	0	transcript_unito_video_separato	CROSS(0) AGAINST 1
12a	0	transcript_unito_video_separato	SAME(0) AGAINST 1
1b	1	transcript_unito_video_singolo	MIX CROSS AND SAME(0) AGAINST 1
2b	1	transcript_unito_video_singolo	CROSS(0) AGAINST 1
3b	1	transcript_unito_video_singolo	SAME(0) AGAINST 1
4b	1	transcript_separato_video_singol	MIX CROSS AND SAME(0) AGAINST 1

Figura 17: Tabella delle diverse configurazioni

4.2.6 Risposte e analisi dei risultati

Dall'analisi delle performance delle varie configurazioni emerge che la maggior parte delle metriche considerate (Precision, Recall, F1-score e Balanced Accuracy) presenta valori generalmente bassi, con risultati poco significativi e accurati.

- Precision e F1 raramente superano lo 0,5, con valori medi vicini a 0,4.
- Recall oscilla tra 0 e 0,5.
- Anche nelle configurazioni migliori, i risultati difficilmente raggiungono lo 0,5, rimanendo insufficienti per ritenere il modello accurato.

- La Balanced Accuracy, particolarmente utile per valutare modelli su dataset sbilanciati, mostra valori medi intorno a 0,5.

Nel caso analizzato, il dataset è sbilanciato, con circa 900 clip etichettate come 'Blob' e 2000 come 'Non Blob'. Per questo motivo è stata utilizzata la Balanced Accuracy come metrica, poiché fornisce una valutazione più equilibrata in presenza di dataset sbilanciati.

In sintesi, avere valori medi delle metriche intorno a 0,5, in particolare con risultati numerici di F1 inferiori a 0,5, indica che il modello non supera le prestazioni di una classificazione casuale, dimostrandosi inadeguato per questo esperimento.

Configurazione Migliore:

La configurazione che si è rilevata più performante è caratterizzata dai seguenti parametri:

- Prompt Index: 1
- Prompt Type: *transcript_unito_video_singolo*

Con i seguenti risultati delle metriche per i tre casi di combinazione video:

Risultati per i tre casi di combinazione video:

1. Caso MIX CROSS AND SAME vs 1
    - Precision: 0,39
    - Recall: 0,53
    - Balanced Accuracy: 0,56
    - F1-score: 0,45
    - Campioni totali: 102
2. Caso SAME vs 1
    - Precision: 0,56
    - Recall: 0,53

- Balanced Accuracy: 0,56
- F1-score: 0,55
- Campioni totali: 68

3. Caso CROSS vs 1
    - Precision: 0,56
    - Recall: 0,53
    - Balanced Accuracy: 0,56
    - F1-score: 0,55
    - Campioni totali: 68

In tutti e tre i casi analizzati, la Balanced Accuracy si attesta a 0,56, un valore leggermente superiore al 50%, ma comunque insufficiente per essere considerato affidabile.

L'F1-score raggiunge 0,55 nei casi Same e Cross, mentre scende a 0,45 nel caso Mix, suggerendo prestazioni peggiori quando le tipologie di video vengono analizzate contemporaneamente

Anche la Recall si mantiene costante a 0,53 in tutte le configurazioni, mentre la Precision si distingue per essere più elevata nei casi Same e Cross (0,56), rispetto al caso Mix, dove cala a 0,39. Questi risultati indicano che l'analisi separata delle tipologie Same e Cross offre prestazioni leggermente migliori al caso Mix, probabilmente perché la loro combinazione introduce maggiore rumore e riduce la chiarezza del modello.

Considerazioni su Prompt e Media:

Per quanto riguarda i prompt utilizzati, il Prompt Index 1 ha dimostrato prestazioni migliori rispetto agli altri. A differenza di Index 0 e Index 2, questo prompt si concentra su input sintetici e diretti, evitando descrizioni dettagliate del programma "Blob" che possono introdurre rumore semantico e confondere il

modello. Tuttavia, con metriche vicine allo 0,5, non si possono formulare valutazioni definitive sull'efficacia di uno specifico tipo di prompt.

Sulla tipologia di media, l'approccio "Transcript Unito" ha prodotto risultati migliori rispetto a quello "Transcript Separato". Il "Transcript Unito" presenta tutte le trascrizioni di un video come un testo coerente, migliorando la continuità narrativa e facilitando il riconoscimento di pattern complessi da parte del modello. Al contrario, il "Transcript Separato" frammenta le informazioni, impedendo al modello di cogliere le relazioni tra le diverse parti. Questa maggiore coerenza narrativa potrebbe spiegare le performance leggermente superiori del "Transcript Unito".

Nonostante alcune configurazioni abbiano mostrato lievi miglioramenti rispetto ad altre, le metriche ottenute rimangono complessivamente insoddisfacenti. Valori vicini allo 0,5 per F1-score e Balanced Accuracy indicano che il modello, indipendentemente dalle configurazioni analizzate, non supera le prestazioni di una classificazione casuale. Pertanto, non è possibile trarre conclusioni definitive sull'efficacia delle variabili testate, e il modello si dimostra inadatto a questo tipo di sperimentazione.

5. Valutazioni e conclusioni

In conclusione, in questa tesi è stata sperimentata l'abilità dei linguaggi multimodali di grandi dimensioni (LLM), in particolare di Gemini, di comprendere lo stile di montaggio del programma televisivo *Blob*.
Nella prima fase, è stata testata la segmentazione delle clip in base all'origine o alla fonte del contenuto, poiché questa è la principale modalità di segmentazione e montaggio video utilizzata nel programma *Blob*, che integra contenuti audiovisivi di diversa origine in un'unica puntata. In questo caso, Gemini ha fornito risultati accurati e significativi, con una precisione di circa il 90%.

Fondamentale è stato anche il lavoro sul prompt engineering, poiché si è osservato che, utilizzando specifici prompt testuali e ottenendo una risposta in formato JSON, i risultati ottenuti erano migliori. Inoltre, fornendo come input non solo la clip video e il prompt di testo, ma anche un formato carosello con i frame del video, si è ottenuto un ulteriore miglioramento.
Se questo primo esperimento, relativo alla segmentazione video in base al contenuto, ha dato risultati soddisfacenti, non si può dire lo stesso per il secondo esperimento, che mirava a distinguere tra clip con tagli di "Blob" e "non Blob".
In questo caso, nonostante l'uso di diversi tipi di media, tra cui i trascritti degli audio delle clip e vari prompt testuali, Gemini ha prodotto risultati deludenti nelle metriche, con percentuali inferiori al 50% nella precision, nella balanced accuracy e nel F1 score.
Questo secondo esperimento ha messo in evidenza la scarsa capacità dei modelli di multimodali, come Gemini, di distinguere lo stile di un programma televisivo da uno generico, dimostrando quindi una comprensione limitata. Questo può essere dovuto al fatto che il montaggio e lo stile di un programma televisivo dipendono da molteplici fattori, che un algoritmo fatica ad analizzare.
Inoltre, come evidenziato nell'analisi semiotica di *Blob*, il programma presenta una forte componente autoriale, frutto del lavoro dei suoi ideatori e montatori, che per anni hanno reso *Blob* un programma iconico, capace di scardinare la televisione italiana e non solo.
Non è quindi ancora qualcosa che un modello multimodale è in grado di comprendere e replicare. Questa constatazione può essere considerata positiva nell'ambito della creazione audiovisiva, poiché implica che l'aspetto autoriale e creativo di Blob non posso ancora essere riprodotto, ma rimanga esclusivo dell'estro creativo umano dei suoi autori e montatori.
In prospettiva futura, però, i progressi nei linguaggi multimodali di grandi dimensioni potrebbero colmare queste lacune.
Nel nostro esperimento, è stato utilizzato un modello pre-addestrato commerciale come Gemini. Sarebbe interessante esplorare l'utilizzo di modelli personalizzati con reti neurali, addestrate su dataset specifici relativi al programma *Blob* o a contenuti audiovisivi simili, per migliorare la comprensione del modello.

Inoltre, l'implementazione di sistemi di feedback continuo potrebbe consentire di affinare ulteriormente le prestazioni, adattandole a situazioni o contenuti diversi. Infine, una collaborazione con esperti del settore creativo — come montatori, registi o autori — potrebbe offrire un contributo decisivo, integrando conoscenze specifiche sul linguaggio narrativo e visivo nel processo di addestramento. Questo approccio consentirebbe di avvicinare i modelli di IA a una comprensione più profonda delle dinamiche umane e delle strutture narrative che caratterizzano la produzione audiovisiva.

Elenco delle figure

Bibliografia

PARTE SEMIOTICA

- [1] Grasso, Aldo. *Storia della televisione*. Milano, Garzanti, 1998.
- [2] Rajewsky, I.O. «'Metatelevision'. The Popularization of Metareferential Strategies in the Context of Italian Television». In Wolf, W. (a cura di), *The Metareferential Turn in Contemporary Arts and Media: Forms, Functions, Attempts at Explanation*, Amsterdam, Rodopi, 2011, pp. 415–444.
- [3] Fava, Vladimir. *Il libro di Blob*. Prefazione di Marco Giusti e Enrico Ghezzi, Torino, Einaudi, 1993.
- [4] Porcelli, Filippo. *Schegge: la TV dopo la TV*. Milano, Il Saggiatore, 2007.
- [5] Menduni, Enrico. «Introduzione a Raymond Williams, *La televisione. Tecnologia e forma culturale e altri scritti sulla TV*». Roma, Editori Riuniti, 2000.
- [6] Monteleone, Franco. *Storia della radio e della televisione*. Venezia, Marsilio, 1992.
- [7] Demarin, Serena. *Il contesto fittizio*. Torino, Università degli Studi di Torino, 2005
- [8] Ejzenstejn, Sergej M. Il montaggio. Venezia: Marsilio, 1986.
- [9] Oliveri, Alice. *«Blob è l'unica parentesi sensata in una TV priva di senso».* The Vision, 10 giugno 2020, https://thevision.com/atlas/blob-tv/
- [10] Valenti, Cecilia. *«Blob resiste: un'analisi di Blob, il laboratorio postmoderno di Enrico Ghezzi»*. Il Tascabile, 11 ottobre 2017, https://www.iltascabile.com/linguaggi/blob-resiste/
-

PARTE SCIENTIFICA

- [11] Patel, Nilesh V., e Ishwar K. Sethi. «Video shot detection and characterization for video databases». *Pattern Recognition*, vol. 30, fasc. 4, aprile 1997, pp. 583–92. *ScienceDirect*, https://doi.org/10.1016/S0031-3203(96)00114-8

- [12] Cotsaces, C., et al. «Video shot detection and condensed representation. a review». *IEEE Signal Processing Magazine*, vol. 23, fasc. 2, marzo 2006, pp. 28–37. *Semantic Scholar*, https://doi.org/10.1109/MSP.2006.1621446.
- [13] Shekar, B. H., e K. P. Uma. «Kirsch Directional Derivatives Based Shot Boundary Detection: An Efficient and Accurate Method». *Procedia Computer Science*, vol. 58, 2015, pp. 565–71. *DOI.org (Crossref)*, https://doi.org/10.1016/j.procs.2015.08.074.
- [14] Hassanien, Ahmed, et al. *Large-scale, Fast and Accurate Shot Boundary Detection through Spatio-temporal Convolutional Neural Networks*. arXiv:1705.03281, arXiv, 27 luglio 2017. *arXiv.org*, https://doi.org/10.48550/arXiv.1705.03281.
- [15] Gygli, Michael *Papers with Code - Ridiculously Fast Shot Boundary Detection with Fully Convolutional Neural Networks*.https://paperswithcode.com/paper/ridiculously-fast-shot-boundary-detection
- [16] Souček, Tomáš, et al. *TransNet: A deep network for fast detection of common shot transitions*. arXiv, 8 giugno 2019. *arXiv.org*, https://doi.org/10.48550/arXiv.1906.03363.
- [17] Rashmi, B. S., e H. S. Nagendraswamy. «Video Shot Boundary Detection Using Block Based Cumulative Approach». *Multimedia Tools and Applications*, vol. 80, fasc. 1, gennaio 2021, pp. 641–64. *Springer Link*, https://doi.org/10.1007/s11042-020-09697-6.
- [18] Castellano, Brandon. *PySceneDetect*. 2014. *GitHub*, https://github.com/Breakthrough/PySceneDetect.
- [19] *Using FFMPEG's Scene Detection To Generate A Visual Shot Summary Of Television News – The GDELT Project*. https://blog.gdeltproject.org/using-ffmpegs-scene-detection-to-generate-a-visual-shot-summary-of-television-news/
- [20] *Using FFMPEG's Scene Detection To Generate A Visual Shot Summary Of Television News – The GDELT Project*. https://blog.gdeltproject.org/using-ffmpegs-scene-detection-to-generate-a-visual-shot-summary-of-television-news/.

- [21] «Exploring Scene Cut Detection of DaVinci Resolve and Adobe Premiere Pro». *ELEMENTS Media Storage*, 26 febbraio 2024, https://elements.tv/blog/exploring-the-new-scene-cut-detection-features-of-davinci-resolve-and-adobe-premiere-pro/.
- [22] Rasheed, Z., e M. Shah. «Scene detection in Hollywood movies and TV shows». *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, vol. 2, IEEE Comput. Soc, 2003, p. II-343–48. *DOI.org (Crossref)*, https://doi.org/10.1109/CVPR.2003.1211489.
- [23] Ngo, Chong-Wah, et al. «Video summarization and scene detection by graph modeling». *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, fasc. 2, febbraio 2005, pp. 296–305. *IEEE Xplore*, https://doi.org/10.1109/TCSVT.2004.841694.
- [24] Chasanis, Vasileios T., et al. «Scene Detection in Videos Using Shot Clustering and Sequence Alignment». *IEEE Transactions on Multimedia*, vol. 11, fasc. 1, gennaio 2009, pp. 89–100. *IEEE Xplore*, https://doi.org/10.1109/TMM.2008.2008924.
- [25] Del Fabro, Manfred, e Laszlo Böszörmenyi. «State-of-the-Art and Future Challenges in Video Scene Detection: A Survey». *Multimedia Systems*, vol. 19, fasc. 5, ottobre 2013, pp. 427–54. *Springer Link*, https://doi.org/10.1007/s00530-013-0306-4.
- [26] Baraldi, Lorenzo, et al. «A Deep Siamese Network for Scene Detection in Broadcast Videos». *Proceedings of the 23rd ACM international conference on Multimedia*, 2015, pp. 1199–202. *arXiv.org*, https://doi.org/10.1145/2733373.2806316.
- [27] Islam, Md Mohaiminul, et al. «Efficient Movie Scene Detection using State-Space Transformers». *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2023, pp. 18749–58. *DOI.org (Crossref)*, https://doi.org/10.1109/CVPR52729.2023.01798.
- [28] Chen, Shixing, et al. «Shot Contrastive Self-Supervised Learning for Scene Boundary Detection». *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2021, pp. 9791–800. *DOI.org (Crossref)*, https://doi.org/10.1109/CVPR46437.2021.00967.

- [29] Wei, Xi, et al. «Multimodal High-order Relation Transformer for Scene Boundary Detection». *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, 2023, pp. 22024–33. *DOI.org (Crossref)*, https://doi.org/10.1109/ICCV51070.2023.02018.
- [30] «Rilevamento dei cambi di inquadratura | Cloud Video Intelligence API Documentation». *Google Cloud*, https://cloud.google.com/video-intelligence/docs/analyze-shots?hl=it.
- [31] Alex. *shonenkov/CLIP-ODS*. 2021. 27 novembre 2024. *GitHub*, https://github.com/shonenkov/CLIP-ODS.
- [32] Fu, Chaoyou, et al. «Video-MME: The First-Ever Comprehensive Evaluation Benchmark of Multi-modal LLMs in Video Analysis». https://video-mme.github.io/home_page.html
- [33]«Precision and Recall». *Wikipedia*, 21 novembre 2024. *Wikipedia*, https://en.wikipedia.org/w/index.php?title=Precision_and_recall&oldid=1258812467.