ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Scuola di Scienze
Corso di Laurea in Ingegneria e Scienze Informatiche

# CLIP-ET-IMPERA: TOPIC-CENTRIC VIDEO SEGMENTATION WITH LARGE LANGUAGE MODELS

*Elaborato in*
Programmazione Di Applicazioni Data Intensive

*Relatore*
Prof. Gianluca Moro

*Co-relatore*
Dott. Giacomo Frisoni

*Presentata da*
Luca Babboni

Terza Sessione di Laurea
Anno Accademico 2022 – 2023

# PAROLE CHIAVE

Natural Language Processing

Machine Learning

Large Language Models

Video Segmentation

Topic Segmentation

*A tutti coloro che hanno contribuito*
*e reso possibile il mio percorso.*

# Abstract

Negli ultimi anni abbiamo assistito a un incremento dell'utilizzo di video per la fruizione di contenuti come conferenze e lezioni. Si presenta quindi la necessità di semplificarne la fruizione, permettendo agli utenti di trovare le informazioni di interesse in tempi rapidi.
Questa tesi presenta due contributi principali: un metodo innovativo di segmentazione automatica dei video in capitoli e la creazione di un nuovo dataset.

Mentre le attuali soluzioni si affidano a tecniche di analisi testuale primitive, l'impiego delle avanzate tecnologie di Large Language Models rappresenta un territorio inesplorato. Di conseguenza proponiamo una nuova metodologia che sfrutta questa opportunità, utilizzando tali modelli generativi per dividere le trascrizioni in paragrafi in modo che questi risultino contigui rispetto agli argomenti contenuti, previo addestramento atto a permettergli di prevedere gli indici di frase che ne fanno da delimitatore.

Nonostante l'evidente rilevanza applicativa di questo argomento, la letteratura attuale non dispone di dataset adeguatamente etichettati. In risposta a questa lacuna introduciamo un nuovo set di dati che incorpora trascrizioni e relative suddivisioni in capitoli con annotazioni supervisionate. Questi dati provengono da video che affrontano temi scientifici e presentano diversità nelle caratteristiche e variazioni di durata.

I risultati, valutati con il dataset proposto utilizzando le metriche Pk Score e WindowDiff, indicano che il Large Language Model addestrato presenta una notevole precisione nella suddivisione del testo, fornendo un contributo significativo nel contesto della divisione di video in capitoli.

# Introduzione

Negli ultimi anni abbiamo assistito a un incremento dell'utilizzo del formato video per la fruizione di contenuti, ampiamente utilizzati come metodo d'insegnamento e condivisione di informazioni.

Una sfida centrale per l'utilizzo di questa tipologia di contenuti in contesto informativo risulta quella di permettere ai fruitori un accesso veloce alle informazioni in essi contenuti. Pensiamo ad esempio alla registrazione di una lunga conferenza oppure di una lezione al cui interno sono contenuti diversi argomenti differenti. Nel caso in cui non venga fornito un indice l'utente potrebbe essere costretto a guardare l'intero video per poter trovare le informazioni di interesse, generando quindi un un notevole spreco di tempo.
Un modo per facilitare la fruizione del video è quello di dividerlo in capitoli, delimitandone i confini tra gli argomenti. Questo potrebbe essere possibile nel caso in cui venga effettuata la procedura manualmente, ma richiedendo notevole tempo e sforzo spesso non viene fatto.

Si pone quindi il problema di creare un sistema per la frammentazione automatica dei video, il cui scopo è quello di trovare i limiti tra gli argomenti trattati.

I metodi più comuni per la segmentazione di un video presenti in letteratura coinvolgono l'impiego di sistemi che analizzano sia le immagini che il parlato del video, spesso in modo combinato.

I metodi che prevedono l'utilizzo delle immagini sono caratterizzati da una bassa generalizzazione, e falliscono quando gli vengono proposti video con immagini di scarsa qualità o che non rappresentano il soggetto atteso. Altri, che si basano su rigide euristiche, risultano datati dopo i recenti sviluppi legati all'intelligenza artificiale.

Per permettere la più vasta generalizzazione il metodo proposto all'interno di questo lavoro si basa sulla sola analisi del parlato, analizzato da un Large Language Model sotto forma di trascrizione del video.
Di seguito andremo quindi ad applicare strumenti all'avanguardia per l'estrazione e la segmentazione del testo.

Inoltre, spinti dalla scarsa qualità dei dataset preesistenti, come contributo aggiuntivo questa tesi propone anche lo sviluppo di un nuovo dataset contenente la trascrizione di oltre 4000 video, dotati di una suddivisione in capitoli effettuata manualmente e provenienti da fonti liberamente accessibili online.

Il lavoro di tesi è stato suddiviso nei seguenti capitoli:

- **Capitolo 1** - Background teorico, contenente una introduzione alle nozioni necessarie per comprendere la tesi.
- **Capitolo 2** - Related Works, comprensivo di panoramica sulle varie soluzioni preesistenti in letteratura e descrizione delle tecnologie utilizzate.
- **Capitolo 3** - Metodologia proposta, trattante la spiegazione dettagliata dei contributi e statistiche sul dataset raccolto.
- **Capitolo 4** - Configurazioni utilizzate, esperimenti e risultati, contenente le valutazioni del modello e le configurazioni utilizzate al fine di garantire la riproducibilità

# Indice


**1 Background teorico** **1**
1.1 Large Language Models . . . . . 1
1.1.1 Architettura . . . . . 2
1.1.2 Prompt engineering . . . . . 3
1.1.3 Fine-tuning efficiente di un LLM . . . . . 4
1.1.4 Limiti dei LLM . . . . . 6
1.2 Automatic speech recognition . . . . . 7
1.3 Topic segmentation . . . . . 8
1.4 Metriche di valutazione . . . . . 8

**2 Related works** **11**
2.1 Panoramica sulle soluzioni preesistenti in letteratura . . . . . 11
2.1.1 A Framework for Lecture Video Segmentation from Extracted Speech Content . . . . . 12
2.1.2 Temporal Lecture Video Fragmentation Using Word Embeddings . . . . . 14
2.1.3 Lecture video indexing using boosted margin maximizing neural networks . . . . . 16
2.2 Analisi dei dataset . . . . . 18
2.2.1 Artificially-generated Lecture Video Fragmentation Dataset and Ground Truth . . . . . 18
2.2.2 AVLecture . . . . . 18
2.2.3 VT-SSum . . . . . 19
2.3 Tecnologie impiegate . . . . . 20
2.3.1 Whisper . . . . . 20
2.3.2 Peft . . . . . 21
2.3.3 Llama 2 . . . . . 22
2.3.4 Weight & Bias . . . . . 23

**3 Metodologia proposta** **27**
3.1 Dataset Proposto . . . . . 27
3.1.1 Creazione del dataset . . . . . 27

3.1.2 Analisi del dataset . . . . . 29
3.2 Preparazione del modello Clip-et-impera . . . . . 33
3.2.1 fine-tuning di LLama2 . . . . . 33
3.3 Clip-et-impera, metodologia proposta . . . . . 35
3.3.1 Trascrizione dell'audio . . . . . 35
3.3.2 Segmentazione del testo usando il modello Clip-et-impera 37
3.3.3 Ricostruzione della soluzione . . . . . 39

**4 Configurazioni utilizzate, esperimenti e risultati 41**
4.1 Processo di valutazione del modello . . . . . 41
4.2 Risultati quantitativi e iperparametri . . . . . 45
4.3 Dettagli implementativi . . . . . 46

**Conclusioni e sviluppi futuri 49**

**Ringraziamenti 51**

**Bibliografia 53**

# Elenco delle figure


1.1 Modello dell'architettura transformer. [1] . . . . . 2
1.2 Crescita in numero di parametri da parte dei LLM. . . . . 5
1.3 Grafico rappresentante in confronto di due differenti risultati (A-0 e A-1) con la stringa di riferimento "base truth" (Ref). . . 9

2.1 Panoramica del processo di estrazione delle feature audio proposto da "A Framework for Lecture Video Segmentation from Ex-tracted Speech Content". [2] . . . . . 12
2.2 Panoramica del processo di segmentazione del video in Temporal Lecture Video Fragmentation Using WordEmbeddings. [3] . . . 14
2.3 Illustrazione della similarità tra segmenti adiacenti attraverso il sistema usato dallo studio "Temporal Lecture Video Fragmentation Using Word Embeddings". [3] . . . . . 15
2.4 Architettura del sistema proposto da Lecture video indexing using boosted margin maximizing neural networks. [4] . . . . . 17
2.5 Architettura del modello di whisper. . . . . 20
2.6 Diagramma dimostrante come gli artefatti di W&B possono essere utilizzati all'interno dell'intero processo di sviluppo di un progetto. . . . . 23
2.7 Esempio di grafici relativi a una specifica run forniti dalla schermata dedicata sulla piattaforma W&B. . . . . 24
2.8 Esempio di tabella fornita dallo strumento Traces per lo studio dei prompt. . . . . 24

3.1 Processo di creazione dati proposto. . . . . 27
3.2 Differenza tra le occorrenze dei video in lingua inglese e quelli in altre lingue all'interno dei dati raccolti . . . . . 29
3.3 Grafico mostrante la distribuzione degli argomenti trattati all'interno dei video contenuti nel dataset proposto. . . . . 30
3.4 Durata dei video all'interno del dataset proposto. . . . . 31
3.5 Numero di parole pronunciate all'interno dei video contenuti del dataset proposto. . . . . 32
3.6 Schema generale della soluzione proposta. . . . . 35

3.7 Schema del processo di trascrizione del video. . . . . 35
3.8 Schema del processo di interrogazione del modello ed ottenere i limiti per la segmentazione del video. . . . . 37

# Elenco delle tabelle

2.1 Performance relative alle varie distribuzioni di Whisper. . . . . 21
2.2 Confronto tra le prestazioni di Llama 2 e altri LLM open source. [5] . . . . . 22

4.1 Risultati dei fine-tuning con diverse configurazioni di iperparametri. 45

# Elenco dei frammenti di codice


3.1 Esempio di input per l'addestramento con formato coerente a Llama2 chat. . . . . 34
3.2 Conversione da video ad audio usando la libreria moviepy. . . . 36
3.3 Estrazione della trascrizione utilizzando Whisper. . . . . 36
3.4 Esempio di contenuto all'interno una cella dell'array. . . . . 37
3.5 Preparazione del modello. . . . . 38
3.6 Esempio di input formattato. . . . . 38
3.7 Processo di interrogazione del modello. . . . . 39

# Capitolo 1

# Background teorico

Il capitolo che segue è dedicato a fornire una base teorica per la comprensione delle tecnologie, dei concetti e dei principi fondamentali che saranno utilizzati nelle fasi successive della realizzazione del progetto di tesi.
Inizieremo con una panoramica delle principali tecnologie utilizzate, per poi andare a definire le diverse tipologie di "compiti" (Task) affidate all'intelligenza artificiale.

## 1.1 Large Language Models

I Large Language Model (LLM), traducibile come "Modello di Linguaggio di Grandi Dimensioni", sono una classe di sistemi di intelligenza artificiale che rappresenta lo stato dell'arte nell'ambito dell'elaborazione del linguaggio naturale (NLP).
La loro particolarità principale è quella di possedere una grande capacità di comprensione del linguaggio umano. Grazie infatti alla loro struttura sofisticata i LLM sono in grado di comprendere il contesto, le sfumature semantiche e persino l'ironia all'interno dei testi. Questa caratteristica permette loro di svolgere compiti prima ritenuti impossibili come quelli di generazione di testi o risposte coerenti a svariate categorie di domande, la creazione di riassunti e il rilevamento delle entità citate all'interno del testo.

Questi modelli sono addestrati su enormi quantità di testo e dati linguistici, principalmente provenienti da diverse fonti online, contenenti una grande varietà di contenuti e lingue.

I LLM hanno rivoluzionato l'ambito del NLP e trovano applicazione in svariati settori, inclusi l'assistenza virtuale e l'analisi dei dati testuali.

### 1.1.1 Architettura

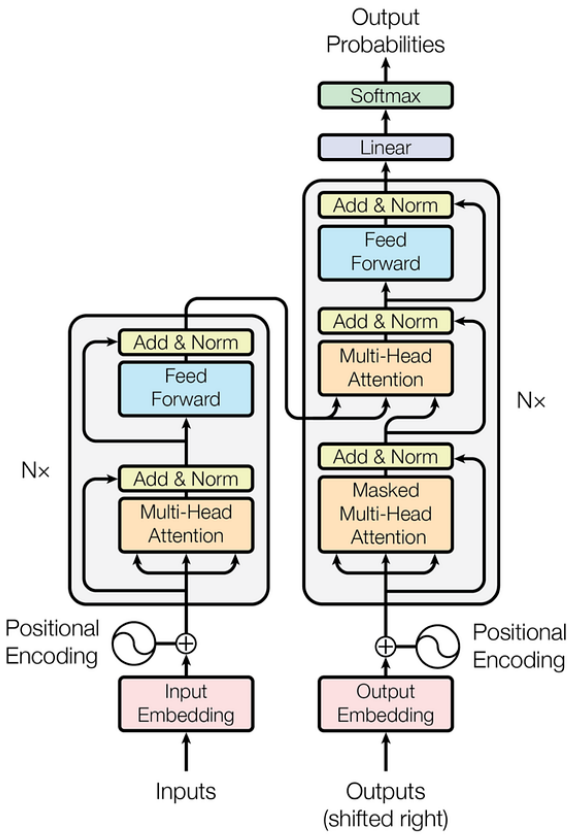



Figure 1: The Transformer - model architecture.

Figura 1.1: Modello dell'architettura transformer. [1]

Tipicamente la struttura alla base dei LLM si basa sull'architettura Transformer (Figura 1.2), presentata per la prima volta all'interno dell'articolo dal titolo "Attention Is All You Need" [1] nel quale viene descritto il meccanismo dell'auto-attenzione (self-attention). Questa particolare tecnica permette di catturare relazioni sintattiche a lungo termine all'interno di una sequenza di parole o token, dando la possibilità al modello di dare un significato alle parole all'interno di un contesto più ampio, superando cosi le limitazioni dei sistemi precedenti.
L'applicazione di questa tecnologia all'interno dei LLM ha comportato un notevole aumento della capacita di gestione di informazioni complesse, migliorando la comprensione del testo e la generazione di testi coerenti.

### 1.1.2 Prompt engineering

Nell'ambito dei LLM, il termine "Prompt" si riferisce all'input testuale fornito al modello, all'interno del quale vengono inserite le istruzioni che lo guidano nella generazione della risposta. Tuttavia, il semplice utilizzo di un generico prompt potrebbe non essere sufficiente al raggiungimento del risultato desiderato, poiché la comprensione e la risposta del modello varia sensibilmente in base alla formulazione delle istruzioni.

Per ottenere risultati maggiormente coerenti con il risultato atteso spesso è necessario ricorrere a un insieme di tecniche noto come "prompt engineering" (ingegneria dei prompt). Questa pratica richiede l'ingegnerizzazione accurata del prompt, attraverso la modifica semantica del contenuto con il fine ultimo di creare istruzioni che siano comprensibili al modello, portandolo a generare la soluzione desiderata.

Il prompt engineering si dimostra una componente molto importante sopratutto con l'utilizzo di LLM preaddestrati, permettendo di massimizzare le capacita di tali modelli e ottenere buoni risultati per compiti sui quali non avevano ricevuto addestramento specifico, evitando quindi una nuova procedura di addestramento.

Una pratica molto efficace nell'ambito del prompt engineering è nota come "In Context Learning" (ICL). Questo metodo consiste nel fornire esempi del risultato atteso all'interno del prompt.
Questa tecnica si rivela molto efficace in numerose situazioni, eliminando la necessità di un ciclo di addestramento specifico e consentendo un notevole risparmio di risorse e tempo. L'approccio ICL è anche noto con i nomi di "apprendimento a zero/ uno/ alcuni esempi" (Zero/ One/ Few shot learning) a seconda del numero di esempi inseriti all'interno del prompt.

### 1.1.3 Fine-tuning efficiente di un LLM

I metodi di prompt engineering presentano però alcune limitazioni che limitano le capacità del modello, non permettendo di ottenere sempre il risultato voluto. Ad esempio il metodo ICL analizzato in precedenza presenta alcuni svantaggi. Seppure il modello sia in grado di imparare ad eseguire un compito semplicemente da alcuni esempi, questi dovranno essere ripetiti ogni volta per far si che le performance rimangano costanti, andando a sottrarre spazio al già limitato massimo numero di parole inseribili all'interno del prompt. Inoltre l'utilizzo di questa pratica non garantisce sempre il raggiungimento del risultato desiderato.

Una possibile soluzione ai problemi elencati è quella di effettuare delle ulteriori operazioni di addestramento (Fine-tuning). Il "fine-tuning" è un processo che segue la fase di pre-addestramento di un modello, e consiste nell'addestrarlo per un compito specifico.
Il set di dati utilizzati per la questa procedura è composto da esempi di prompt e risposte etichettate, caratterizzandosi per una dimensione notevolmente ridotta rispetto a quello utilizzato per l'addestramento iniziale e una particolare specializzazione riguardo al compito specifico oggetto dell'addestramento.
Durante questo processo, i pesi del modello vengono aggiornati in base agli errori avvenuti all'interno delle previsioni rispetto alle risposte attese. L'obiettivo è di far imparare al modello la nuova tipologia di compito, riducendo di conseguenza l'errore nelle risposte.

Un problema per l'attuazione di queste procedure risiede nelle dimensione dei modelli. Negli ultimi anni il numero di parametri contenuti all'interno dei LLM sono cresciuti sensibilmente (Figura 1.2). Se da una parte questo permette loro di avere migliori prestazioni dall'altra li rende quasi impossibili da addestrare utilizzando schede video non specializzate e dotate di poca memoria grafica. Per mitigare questo problema sono state introdotte diverse tecniche. Di seguito andremo quindi ad esplorare quelle utilizzate all'interno della tesi.

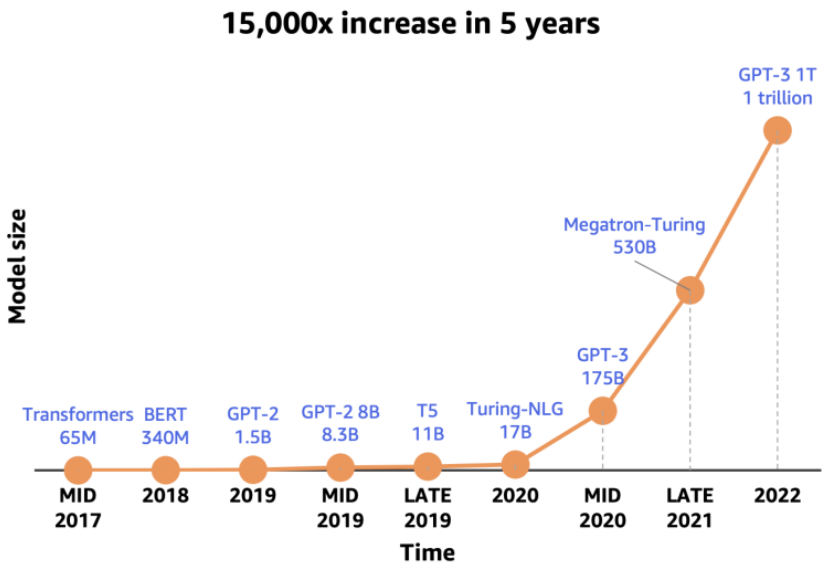



Figura 1.2: Crescita in numero di parametri da parte dei LLM.
da
https://d1.awsstatic.com/events/Summits/reinvent2022/AIM405_Train-and-deploy-large-language-models-on-Amazon-SageMaker.pdf

### Low-Rank Adaption

Introdotto inizialmente da Microsoft Lora [6], acronimo di Low-Rank Adaption, rappresenta una tecnica concepita per ottimizzare il processo di fine-tuning applicato ai LLM, con un'enfasi particolare sull'efficienza computazionale. Questo approccio consente di "congelare" i parametri originali del modello pre-addestrato, mantenendoli invariati, per poi aggiungere un insieme ridotto di parametri modificabili. Durante le fasi di fine-tuning saranno solamente questi ultimi i parametri ad essere aggiornati.

L'adozione di questo procedimento permette di ridurre significativamente il numero di parametri che richiedono addestramento, rendendo fattibile il fine-tuning di LLM di grandi dimensioni su schede video commerciali con limitate risorse di memoria grafica.

#### Quantizzazione

Per capire come funziona questo sistema prima è necessario introdurre il concetto di precisione di un modello. Questo dato si riferisce alla dimensione dei registri utilizzati per salvarne i parametri. Maggiore è la dimensione del registro maggiore è la precisione del dato in formato floating point salvato al suo interno.
Solitamente all'aumentare della precisione sono associate migliori prestazioni da parte del modello.

La procedura che va a svolgere il processo di quantizzazione è quella di effettuare delle approssimazioni sul valore dei parametri in modo da ridurne l'occupazione di memoria. Inoltre utilizzando la tecnica di Activation-aware Weight Quantization (AWQ) [7] è possibile quantizzare i modelli con degradazione nelle performance molto basse.

#### QLora

Le tecniche di quantizzazione possono essere anche combinate alla tecnica Lora vista precedentemente. Questa unione prende il nome di QLora [8] ed utilizza una quantizzazione a 4-bit per ridurre lo spazio utilizzato da un modello preaddestrato. Sui parametri cosi modificati vengono poi eseguite le procedure descritte precedentemente nella sezione dedicata a Lora.

### 1.1.4 Limiti dei LLM

I LLM hanno dimostrato di essere molto potenti e versati per una grande varietà di applicazioni. Ciò nonostante ci sono alcune limitazioni significative associate a questi modelli che è importante riconoscere ed affrontare. Ai fini di questa tesi risulta utile conoscere le seguenti limitazioni.

- **Allucinazioni**:
  Il termine Allucinazioni, o come più comunemente noto nella sua forma inglese "Hallucinations", si riferisce a una caratteristica indesiderata del modello. Questa avviene quando questo genera dei testi contenenti informazioni non veritiere, non basate su fatti reali o informazioni fattuali, ma invece originate da associazioni casuali o errori.

- **Context Window Limitata**:
  Il termine "context window" si riferisce alla dimensione della finestra di testo da cui il modello riesce ad attingere le informazioni dal prompt per comprendere l'istruzione e generare una risposta. Avere una context window limitata può creare diversi problemi, soprattutto quando ci si trova ad interfacciarsi al modello con prompt lunghi. Questo limita la capacità del modello di comprendere il contesto di un'informazione all'interno di un testo, generando quindi risposte incomplete o inesatte.

- **Ragionamento**:
  Nel contesto dei LLM il termine ragionamento (reasoning) si riferisce alla capacità di eseguire processi di ragionamento. Questo aspetto è decisivo quando si tratta di generare risposte a problemi complessi su cui il modello non è stato direttamente addestrato. Una delle sfide nell'addestramento di LLM è quella di migliorarne le capacità di reasoning. Questa caratteristica risulta particolarmente importanti durante svolgimenti di compiti che richiedono una profonda comprensione del testo fornito in input.

- **Spiegabilità del risultato**:
  La spiegabilità del risultato (Explainability) è la capacità del modello di comprendere e spiegare il processo decisionale che porta a una determinata risposta a seguito di un prompt. I LLM sono costituiti da un grande numero di parametri interconnessi. Il risultato fornito da questi modelli è il risultato di complessi calcoli matematici tra diversi nodi della rete, rendendo arduo tracciare l'esatto processo che ha portato a una risposta.

## 1.2 Automatic speech recognition

L'Automatic Speech Recognition (ASR), traducibile come "Riconoscimento Automatico del Parlato", è una tecnologia che si occupa di convertire il linguaggio parlato in testo scritto. Questo processo si basa su avanzati algoritmi di elaborazione audio o intelligenze artificiali allenate a riconoscere il parlato a seconda delle caratteristiche audio.

Queste tecnologia trova notevole impiego in una vasta gamma di situazioni, dalla generazione automatica di sottotitoli all'applicazione all'interno di assistenti vocali.

Nonostante i notevoli progressi raggiunti nell'ASR, le sfide persistono, manifestandosi con riconoscimenti meno accurati in contesti ad alta rumorosità e in presenza di variazioni linguistiche regionali. L'adattamento a lingue meno

comuni emerge come un ulteriore ostacolo, poiché i modelli possono dimostrare limitata efficacia in contesti linguistici meno diffusi. Tuttavia, nonostante queste limitazioni l'argomento continua a essere arricchito da nuove proposte, promettendo miglioramenti continui nelle performance.

## 1.3 Topic segmentation

Il compito di "Topic Segmentation", in italiano segmentazione degli argomenti, appartiene al campo del NLP e riguarda la suddivisione di un testo in segmenti, in modo che ciascuno di essi contenga al suo interno un argomento o tema specifico. Attraverso l'uso di tecniche di analisi testuali la finalità di questo processo è quello di individuare i "punti di confine" tra gli argomenti.
La suddivisione del testo in parti più brevi si è dimostra utile in diverse circostanze tra le quali citiamo l'indicizzazione di documenti e la ricerca informativa all'interno di testi molto lungi.

Questa tipologia di task può rivelarsi utile anche applicata ad altre circostanze, come ad esempio all'analisi di video. Anche in questo caso l'obbiettivo è quello di suddividere un video in parti più brevi e a se stanti dal punto di vista degli argomenti trattati.
In questo contesto questa tipologia di task richiede l'utilizzo di tecniche avanzate di analisi audio-visive per rilevare i punti dove avviene il cambio di argomento. L'analisi può basarsi su diversi fattori tra cui l'estrazione di caratteristiche (feature) audio, come volume, intonazione o rilevamento delle pause, analisi del parlato attraverso tecniche di NLP oppure algoritmi di visione artificiale per il riconoscimento degli oggetti presenti all'interno delle scene.
Come la corrispettiva testuale questa task si dimostra utile in diverse situazioni tra cui l'indicizzazione di meetings o lezioni e l'aiuto della ricerca di uno specifico argomento all'interno di un video.

## 1.4 Metriche di valutazione

La valutazione di algoritmi di "Topic Segmentation" presenta diverse difficoltà e criticità. Il problema maggiore si rifà al concordare quale sia la soluzione esatta del problema. Stabilire dove debba essere posizionato il termine di un segmento è complesso anche per un giudice umano e spesso due soluzioni calcolate da persone differenti possono variare anche notevolmente. Questo rende difficile la scelta per il segmentazione di riferimento quando si va a calcolare l'accuratezza di un risultato ottenuto da un algoritmo.

### Precision & Recall

Questo metodo rappresenta una metrica di valutazione largamente utilizzata all'interno di task di "classificazione" e "information retrival". Questo metrica si compone di due parti principali "Precisione" (Precision) e "Richiamo" (Recall). All'interno dell'ambito della segmentazione di un testo questi due termini possono essere definiti come:

- **Precision**:
  Percentuale dei limiti tra argomenti differenti identificati dall'algoritmo di "Text Segmentation" che sono effettivamente presenti all'interno della soluzione.

- **Recall**:
  Percentuale di veri limiti identificati dal modello.

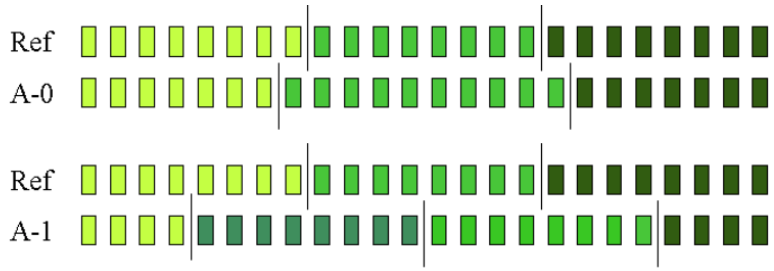



Figura 1.3: Grafico rappresentante in confronto di due differenti risultati (A-0 e A-1) con la stringa di riferimento "base truth" (Ref).
da https://www.assemblyai.com/blog/text-segmentation-approaches-datasets-and-evaluation-metrics/

Questo metodo di valutazione risulta però essere problematico per due motivi [9]. Il primo fa riferimento al fatto che il risultato sia un compromesso tra i parametri di precision e recall. Con l'intenzione di migliorare il parametro recall si va ad aumentare la quantità di segmenti presenti all'interno della soluzione dell'algoritmo, peggiorando cosi il valore di precision. Il secondo problema di questa valutazione è che non tiene conto dei "near miss", cioè dei limiti identificati molto prossimi a quelli all'interno della soluzione, ma che vengono comunque interpretati come sbagliati. Un esempio di questo fenomeno è rappresentano in Figura 1.3. La segmentazione A-0 è molto simile a quella presentata nella sequenza di riferimento, ma il risultato ottenuto è comunque identico a quello dalla sequenza A-1, anche se questa è molto meno precisa.

### Pk Score

Per risolvere i problemi causati da "Precision & Recall" è stata introdotta un'ulteriore metrica chiamata PK Score, presentata all'interno dell'articolo "Statistical Models for Text Segmentation" [10].

Il calcolo di Pk viene eseguito facendo scorrere all'interno dei segmenti una finestra di dimensione k, solitamente pari a metà della media della lunghezza dei segmenti. Ad ogni iterazione la finestra viene fatta spostare di un elemento, e vengono calcolate le penalità confrontando il contenuto della sequenza di riferimento con quella della sequenza calcolata. Se esistono delle differenze un contatore viene incrementato di una unità. Al termine dell'analisi delle sequenze il contenuto del contatore viene scalato tra 0 e 1. Se la sequenza di riferimento e quella calcolata dall'algoritmo sono identiche il risultato della valutazione è 0. Ciò significa che risultati prossimi vicini allo 0 indicano una buona segmentazione mentre al contrario quelli prossimi a 1 indicano una divisione del testo poco efficace.

### WindowDiff

Anche Pk Score peró presenta alcuni problemi, studiati all'interno dell'articolo dal titolo "A Critique and Improvement of an Evaluation Metric for Text Segmentation' [9]. I principali problemi riscontrati dagli autori riguardano lo sbilanciamento tra le penalizzazioni eccessive date ai falsi negativi rispetto ai falsi positivi, la mancata considerazione del numero di limiti trovati dall'algoritmo rispetto a quelli presenti all'interno della sequenza di riferimento, l'eccessiva sensibilità alla variazione della dimensione dei segmenti e l'eccessiva penalizzazione degli errori "near miss".
All'interno dello stesso articolo viene quindi proposta una soluzione che prende il nome di "WindowDiff", con lo scopo di migliorare l'algoritmo Pk Score.
Questo algoritmo calcola il punteggio attraverso la seguente formula:

$$WindowDiff(ref, hyp) = \frac{1}{N-k} \sum_{i=1}^{N-k} (|b(ref_i, ref_{i+k}) - b(hyp, hyp_{i+k})| > 0)$$

Anche in questo caso l'analisi di due sequenze con i limiti negli stessi punti da risultato 0.

# Capitolo 2

# Related works

## 2.1 Panoramica sulle soluzioni preesistenti in letteratura

Il problema della segmentazione automatica di video è un argomento presente in letteratura da diversi anni ma poco dibattuto. Per quanto siano pochi, gli algoritmi proposti hanno performance in continua crescita grazie alle evoluzione nell'ambito della computer vision e del Machine Lerning.

Prima di approfondire la nostra proposta, è essenziale condurre un'analisi approfondita delle soluzioni preesistenti presenti in letteratura. La revisione critica della letteratura ci permetterà di contestualizzare il nostro lavoro all'interno del panorama delle soluzioni esistenti, cercando di evidenziare le nuove opportunità rimaste ancora inesplorate.

Per la scelta dei lavori da analizzare sono state prese in considerazione proposte recenti con approcci anche molto differenti da quelli proposti in questa tesi.

### 2.1.1 A Framework for Lecture Video Segmentation from Extracted Speech Content

Il lavoro intitolato "A Framework for Lecture Video Segmentation from Extracted Speech Content" [2] presenta un framework di recente sviluppo per la segmentazione di video. Il metodo utilizzato prevede una segmentazione in capitoli generata grazie all'analisi delle caratteristiche (feature) audio contenute all'interno del video.

L'intero procedimento della generazione dei segmenti (Figura 2.1) può essere diviso in due macro processi, il primo dedicato all'estrazione delle feature audio mentre il secondo si occupa di utilizzare i dati estratti per la segmentazione del video.

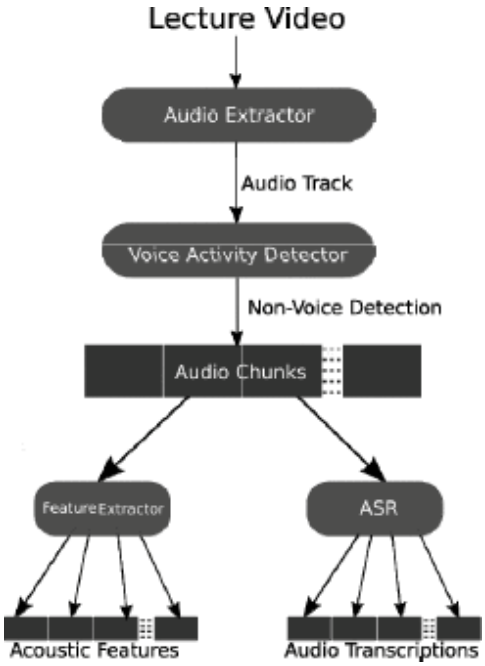



Figura 2.1: Panoramica del processo di estrazione delle feature audio proposto da "A Framework for Lecture Video Segmentation from Ex-tracted Speech Content". [2]

La prima procedura comprende l'estrazione delle caratteristiche audio dal video. Durante questo processo l'audio viene segmentato attraverso l'utilizzo di un Voice Activity Detector (VAD), un dispositivo in grado di distinguere le porzioni di audio dov'è presente del parlato da quelle dove sono presenti silenzi o brani musicali. Il modello VAD utilizzato si basa su un classificatore

GMM pre-addestrato [11]. In questa fase vengono eliminate dall'audio le sezioni indesiderate, cioè le parti prime di parlato o contenenti spezzoni di musica, per poi separarlo in segmenti più brevi e privi di disturbi.

L'idea alla base della frammentazione dell'audio utilizzando come delimitazione le pause viene giustificato dagli autori come metodo per poter ottenere spezzoni di audio dal contenuto coerente, visto che è molto probabile che il cambio di argomento avvenga dopo una pausa [12].
Il passaggio successivo prevede l'analisi di questi spezzoni di audio da parte di due moduli, il Feature Extractor (FE) e L'Automatic Speach Recognition (ASR).

Il componente chiamato Feature Extractor si occupa di estrarre le caratteristiche acustiche, che verranno poi utilizzate in fase di segmentazione.
In questa fase risulta fondamentale il lavoro precedentemente svolto dal modulo VAD. Infatti la pulizia dell'audio permette al modello di estrarre solo le proprietà utili da segmenti privi di disturbi.

Dagli spezzoni di audio viene estratta anche la trascrizione attraverso l'utilizzo di un modello di ASR. Nello specifico questo progetto utilizza lo strumento *Pocketsphinx* [13], un ASR open-source a basso consumo di risorse.

Terminata l'estrazione dei dati inizia la parte di segmentazione del video. L'algoritmo di segmentazione utilizzato si ispira all'algoritmo presentato in [14] con l'aggiunta di alcune modifiche che prendono in considerazione le relazioni tra la tonalità e il volume provenienti da [15, 16]

L'algoritmo di segmentazione utilizzato prevede la risoluzione del seguente problema di ottimizzazione:

$$maxT \sum_{i=1}^{n} U_i \cdot X_i - \sum_{i=1}^{n} X_i$$

Dove:

- T è l'insieme di tutte le possibili soluzioni
- La soluzione del problema è il sotto insieme di segmenti che ottimizzano l'equazione

- $X_i$ rappresenta una variabile decisionale definita come

$$X_j = \begin{cases} 1 \text{ if } S_i \in T \\ 0 \text{ if } S_i \notin T \end{cases}$$

- $U_i$ rappresenta l'Utility Score calcolato per ogni segmento, dato dalla funzione:

$$U_i = \alpha(F_i + V_i) + \beta \cdot P_i + \gamma \cdot D$$

  Dove:

  - $\alpha, \beta, \gamma$ sono costanti decise dall'utente
  - $F_i$ è il tono
  - $V_i$ è il volume
  - $P_i$ è la pausa
  - $D_i$ rappresenta la similarità coseno tra le rappresentazioni *word2vect* della trascrizione di del frammento audio Si e dei suoi 2 vicini $S_{i+1}$ $S_{i-1}$ calcolata con la formula:

$$D_i = Dcos(i-1,\ i) + Dcos(i,\ i+1) \tag{2}$$

### 2.1.2 Temporal Lecture Video Fragmentation Using Word Embeddings

Il progetto presentato in questo articolo [3] propone una soluzione al problema della segmentazione in argomenti di un video utilizzando un approccio basato sul Natural Language Processing e la rete neurale *Word2Vec* (Figura 2.2).
Una volta ottenuto la trascrizione di un video tramite l'utilizzo di un ASR, questa verrà utilizzato come input per l'algoritmo di segmentazione.

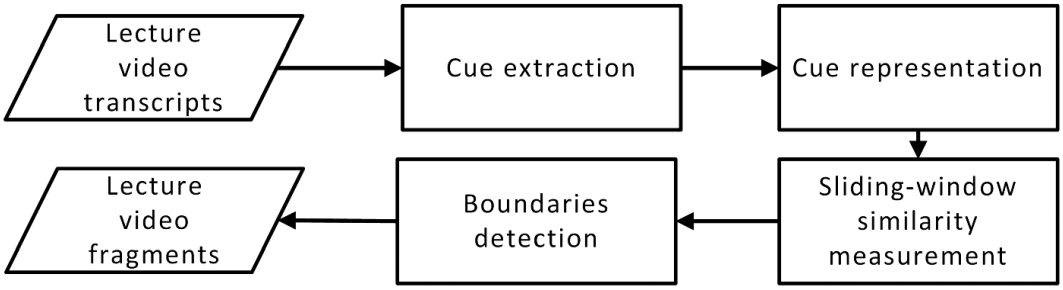



Figura 2.2: Panoramica del processo di segmentazione del video in Temporal Lecture Video Fragmentation Using WordEmbeddings. [3]

.

Una finestra scorrevole viene utilizzata per spezzettare la trascrizione in frammenti di dimensione N parole, per essere poi codificati in rappresentazione vettoriale utilizzando la rete neurale *Word2Vec*. Questo sistema trasforma ogni parola $w_i$ in un vettore $V^{w_i}_{word2vec} = [v_1, v_2, ..., v_n]$ che rappresenta le proprietà lessicali e semantiche della parola.

L'intero frammento di testo viene quindi rappresentato in forma vettoriale seguendo l'approccio detto *"average word vector"* che consiste nel calcolare la media delle rappresentazioni vettoriali delle parole contenute all'interno del frammento. In questo modo si ottiene un unico vettore che rappresenta l'intera frase.

Terminata la fase di rappresentazione del testo in forma vettoriale viene eseguita l'operazione di segmentazione del testo. Questa fase prevede diversi step che come risultato finale forniscono l'elenco dei punti temporale dove avviene il cambio di argomento.

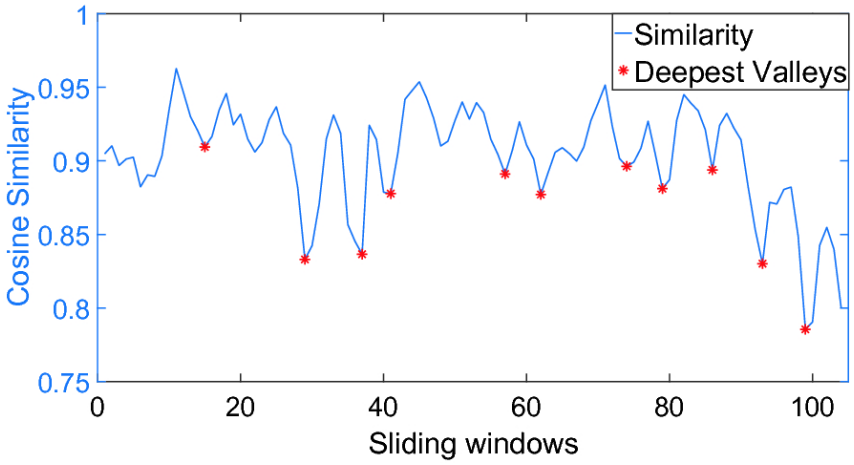



Figura 2.3: Illustrazione della similarità tra segmenti adiacenti attraverso il sistema usato dallo studio "Temporal Lecture Video Fragmentation Using Word Embeddings". [3]

Inizialmente viene utilizzato il calcolo della similarità coseno tra le coppie di segmenti adiacenti, in modo da stimarne la similarità semantica. Dal risultato dell'analisi viene estratta una funzione della forma $y = f(x)$, dove x rappresenta il tempo mentre y rappresenta la similarità calcolata tra 2 segmenti adiacenti (Figura 2.3).

Si procede quindi con un'analisi della funzione, andando a ricercare prima i punti di minimo, e poi le relative differenze dai punti di massimo destro e sinistro più vicino, calcolandone quindi la "profondità della valle". Questo viene fatto perché andare a calcolare la profondità della valle generata dal punto di minimo permette di stimare quanto è variata la similarità semantica tra i 2 segmenti adiacenti. I punti di minimo all'interno delle valli con profondità maggiore rappresentano quindi ottimi candidati per essere scelti come punto di cambio argomento.
La profondità della valle viene calcolata con la seguente formula:

$$depth_{val} = (peak_1 - val) + (peak_1 - val)$$

Da qui vengono proposti 2 possibilità per scegliere i punti delimitatori di argomento. Il primo metodo prevede di estrarre i K punti con il valore di depthVal maggiori, mentre il secondo utilizza un sistema dove vengono scelto tutti i punti con il valore calcolato di depthVal maggiore di una certa soglia (Thr) calcolata come:

$$Thr = m : (\mu - \alpha)$$

dove $\mu$ rappresenta la media dei valori della funzione nei punti di minimo, $\alpha$ è la deviazione standard e $m$ è una costante.

Oltre all'algoritmo di segmentazione questo studio propone anche un dataset, creato artificialmente, pensato appositamente per la valutazione di sistemi per la segmentazione di video a seconda degli argomenti. Questo dataset verrà poi analizzato all'interno della sezione dedicata.

### 2.1.3 Lecture video indexing using boosted margin maximizing neural networks

Questo caso di studio [4] presenta una soluzione molto differente da quelli precedentemente trattate, sia per tecnologie utilizzate che per processo di estrazione dati dal video. L'idea alla base del progetto è quella di utilizzare le immagini contenenti l'inquadratura ed associare il punti di cambio argomento con quelli dove vengono fatte avanzare le slide della presentazione.

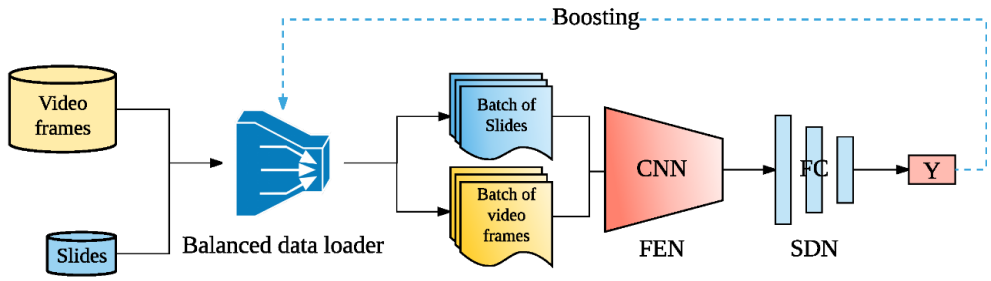



Figura 2.4: Architettura del sistema proposto da Lecture video indexing using boosted margin maximizing neural networks. [4]

La soluzione proposta si basa su una rete neurale il cui scopo è quello di valutare la similarità di 2 frame successivi del video.

La metodologia presentata si compone di due reti neurali connesse tra loro (Figura 2.4). La prima è denominata FEN (Feature Extraction Network) e si occupa di estrarre le caratteristiche dalle immagini in input. La seconda è chiamata SDN (Similarity Discrimination Network) e si occupa di calcolare la similarità tra le caratteristiche estratte da una coppia di immagini.

Le due reti neurali sono ottenute grazie a un addestramento simultaneo. Durante l'addestramento vengono proposte due immagini al modulo FEN per far si che questo ne estragga le caratteristiche, poi riproposte al modulo SDN.

Durante il training delle coppie casuali di frame vengono sottoposte alla rete neurale, che dovrà stabilire se le due immagini sono correlate oppure no. Come riportato nell'articolo allenare la rete neurale con tutte le combinazioni possibili di frame estratti è da evitare, poiché creerebbe un dataset sbilanciato. Per evitare quindi che all'interno della rete neurale si sviluppino dei così detti *"bias"* viene utilizzato un componente chiamato Balanced Data Loader. Il suo scopo è quello di bilanciare la percentuale di probabilità che al sistema venga sottoposto un caso a cui deve dare risposte positive con quelle a cui deve darne di negative.

## 2.2 Analisi dei dataset

### 2.2.1 Artificially-generated Lecture Video Fragmentation Dataset and Ground Truth

Questo dataset viene presentato all'interno dell'articolo [3].
Per la creazione di questo dataset sono state utilizzate 1498 trascrizioni di video a trattanti diversi temi di interesse scientifico provenienti dal sito *VideoLectures.NET*. Poiché sia i video che le trascrizioni risultano sprovviste di un indice gli autori hanno optato per la creazione di un dataset artificiale.

Per prima cosa le trascrizioni sono state divise in frammenti della durata dai 4 agli 8 minuti. In seguito, vari frammenti provenienti da lezioni differenti sono stati combinati tra loro per formare dei video della durata di circa 120 minuti.

In questo modo è stato possibile creare delle trascrizioni di cui è nota la suddivisione in capitoli, dove cioè è noto il punto nel quale avviene il cambio di argomento.

Si è deciso di non utilizzare questo dataset poiché, essendo stato creato artificialmente, non rappresenta delle casistiche reali.

### 2.2.2 AVLecture

Questo dataset venne presentato all'interno dell'articolo scientifico intitolato: "Unsupervised Audio-Visual Lecture Segmentation" [17].

Il dataset proposto è composto da oltre 2,350 registrazioni di lezioni, provenienti da 86 corsi, per un totale di oltre 2,200 ore di video. Per l'acquisizione del materiale è stato utilizzato il sito MIT-OCV[1], nel quale sono raccolte note, lezioni e materiale didattico proveniente dai corsi del MIT.

Del totale di 86 corsi contenuti all'interno dataset solo 15 risultano provvisti di indicizzazione e solamente 5 di loro risultano effettivamente segmentati manualmente. Per i rimanenti 10 sono stati utilizzati i frammenti di video di cui si conosceva l'argomento per ricomporre i video originali.

Anche in questo caso si è scelto di non utilizzare questo dataset. Seppure alcuni dei video proposti siano stati indicizzati manualmente, il numero è comunque

[1] `https://ocw.mit.edu/`

insufficiente per risultare utile nel contesto della metodologia proposta da questa tesi.

### 2.2.3 VT-SSum

Questo dataset venne presentato nell'articolo dal nome "VT-SSum: A Benchmark Dataset for Video Transcript Segmentation and Summarization" [18].

Il dataset contiene 9,616 video di 26 categorie provenienti dal sito VideoLectures.NET[2].

Anche in questo caso in origine i video utilizzati non erano provvisti di indicizzazione. É stato quindi utilizzato il video della presentazione per crearne una. Per prima cosa è stato necessario scaricare le lezioni, per poi procedere a un'analisi delle caratteristiche video. Utilizzando queste ultime si è andato a trovare i punti nel quale il professore fa avanzare le slide all'interno della presentazione. I punti sono stati poi utilizzati come riferimento per l'indicizzazione.

Poiché i video risultavano sprovvisti di trascrizione sono stati trascritti artificialmente, attraverso l'utilizzo di Microsoft Speech to Text[3].

Anche in questo caso a causa dell'assenza di video annotati manualmente si è deciso di non utilizzare questo dataset.

[2] `http://videolectures.net/`

[3] `https://azure.microsoft.com/en-us/products/ai-services/speech-to-text/`

## 2.3 Tecnologie impiegate

### 2.3.1 Whisper

Whisper [19] è un una modello di intelligenza artificiale open source, fornito da OpenAi, il cui scopo è il riconoscimento automatico del parlato (ASR). I risultati ottenuti da questo modello raggiungono lo stato dell'arte in diverse tipologie di task come il rilevare automaticamente la lingua parlata nel video, il tradurre le trascrizioni in inglese e il prevedere i momenti temporali nel quali vengono pronunciate le frasi. Ad esempio per quanto riguarda la trascrittura di un audio in inglese la precisione di Whisper è solamente dell' 1.15% inferiore a quella creata d un essere umano.

Il modello è stato ottenuto grazie ad un addestramento supervisionato utilizzante un dataset molto ampio, composto da 680.000 ore di audio contenenti parlato in varie lingue.
Secondo gli autori, l'utilizzo di un dataset così ampio e diversificato aumenta la robustezza del modello davanti alla presenza di accenti, linguaggio tecnico, del rumore di fondo e della traduzione da altre lingue all'inglese.

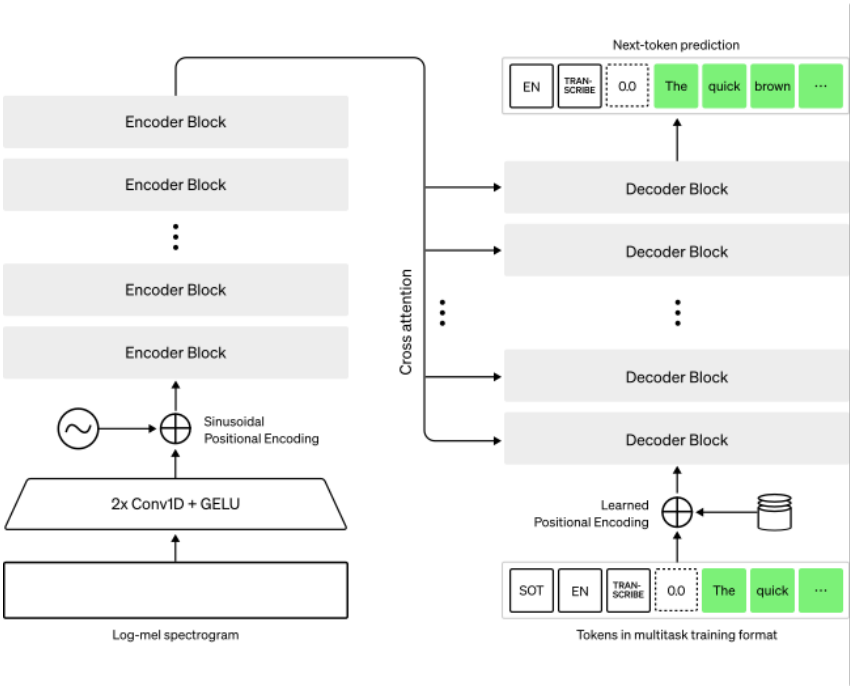



Figura 2.5: Architettura del modello di whisper.
da https://openai.com/research/whisper

L'Architettura di Whisper (Figura 2.5) è stata implementata utilizzando la composizione encoder-decoder tipica dei Transformer. L'audio in input viene diviso in segmenti da 30 secondi, per poi essere codificati dall'encoder. Il decoder viene quindi addestrato a trascrivere il parlato all'interno dell'audio.

| **Dimensione** | **Parametri** | **VRAM** | **Relative Speed** | **WER eng LibriSpeach [19]** |
|---|---|---|---|---|
| `tiny` | 39 M | ∼1 GB | ∼32x | 15.7 |
| `base` | 74 M | ∼1 GB | ∼16x | 11.7 |
| `small` | 244 M | ∼2 GB | ∼6x | 8.3 |
| `medium` | 769 M | ∼5 GB | ∼2x | 6.8 |
| `large` | 1550 M | ∼10 GB | 1x | 6.3 |

Tabella 2.1: Performance relative alle varie distribuzioni di Whisper.
dati presi dalla repository ufficiale: `https://github.com/openai/whisper`

Whisper è stato distribuito in diverse versioni di dimensioni più o meno leggere le cui prestazioni sono visionabili all'interno della Tabella 2.1.

All'interno del progetto, Whisper, è stato utilizzato come ASR per via delle sue prestazioni che raggiungono lo stato dell'arte.
Questo modello è stato utilizzato sia per ottenere le trascrizioni contenute all'interno del dataset proposto, sia come parte della metodologia presentata da questa tesi per la divisione automatica dei video in capitoli.

### 2.3.2 Peft

PEFT[4], acronimo di Parameter-Efficient Fine-Tuning, è una libreria fornita da Hugging Face[5] che offre metodi efficienti per eseguire operazioni di fine-tuning su LLM preaddestrati.
Andare ad eseguire le operazioni di training coinvolgendo tutti i parametri del modello è un'operazione notevolmente costosa in quanto a risorse utilizzate. Per efficientare questo processo vengono utilizzate le metodologie introdotte da LORA [6], concentrandosi quindi sul fine-tuning di un numero limitato di parametri aggiuntivi, rendendo una vasta gamma di LLM accessibili anche su dispositivi dotati di poca memoria video.

[4] `https://huggingface.co/docs/peft/index`
[5] `https://huggingface.co/`

### 2.3.3 Llama 2

LLama 2 [5] rappresenta il nuovo stato dell'arte per quanto riguarda i LLM open source, gratuito sia per uso di ricerca che per scopi commerciali.

Questo modello presenta notevoli avanzamenti rispetto al predecessore Llama 1 come la dimensione della context window quasi raddoppiata e un nuovo metodo per l'attention.

Il modello viene distribuito in 3 modelli base di dimensioni differenti, 7, 13 e 70 miliardi di parametri. Oltre a questi sono disponibili anche i corrispettivi modelli in versione "chat", addestrati appositamente per l'utilizzo nei dialoghi. Questi modelli superano ampiamente le prestazioni dei LLM open sourse precedenti in molti benchmark e alla pari con modelli closed source come ChatGPT.

| Model | Size | Code | Commonsense Reasoning | World Knowledge | Reading Comprehension | Math | MMLU | BBH | AGI Eval |
|---|---|---|---|---|---|---|---|---|---|
| MPT | 7B | 20.5 | 54.4 | 41.0 | 57.5 | 4.9 | 26.8 | 31.0 | 23.5 |
| | 30B | 28/9 | 64.9 | 50.0 | 64.7 | 9.1 | 46.9 | 38.0 | 33.8 |
| Falcon | 7B | 5.6 | 56.1 | 42.8 | 36.0 | 4.6 | 26.2 | 28.0 | 21.2 |
| | 40B | 15.2 | 69.2 | 56.7 | 65.7 | 12.6 | 55.4 | 37.1 | 37.0 |
| Llama1 | 7B | 14.1 | 60.8 | 46.2 | 58.5 | 6.95 | 35.1 | 30.3 | 23.9 |
| | 13B | 18.9 | 66.1 | 52.6 | 62.3 | 10.9 | 46.9 | 37.0 | 33.9 |
| | 33B | 26.0 | 70.0 | 58.4 | 67.6 | 21.4 | 57.8 | 39.8 | 41.7 |
| | 65B | 30.7 | 70.7 | 60.5 | 68.6 | 30.8 | 63.4 | 43.5 | 47.6 |
| LLama 2 | 7B | 16.8 | 63.9 | 48.9 | 61.3 | 14.6 | 45.3 | 32.6 | 29.3 |
| | 13B | 24.5 | 66.9 | 55.4 | 65.8 | 28.7 | 54.8 | 39.4 | 39.1 |
| | 34B | 27.8 | 69.9 | 58.7 | 68.0 | 24.2 | 62.6 | 44.1 | 43.4 |
| | 70B | **37.5** | **71.9** | **63.6** | **69.4** | **35.2** | **68.9** | **51.2** | **54.2** |

Tabella 2.2: Confronto tra le prestazioni di Llama 2 e altri LLM open source. [5]

La Tabella 2.2 riporta una comparazione tra le performance di Llama 2 ed altri LLM su diversi dataset, che mirano a stabilirne la generale conoscenza acquisita durante il pretraining (MMLU [20]), la capacità di svolgere compiti linguistici difficili (BBH [21]) e le capacità generali del modello di gestire compiti a livello umano (AGI Eval [22]).

É stato scelto di utilizzare questo modello poiché tra quelli open source disponibili risultava essere quello con le performance base migliori.

### 2.3.4 Weight & Bias

Wight & Bias (W&B)[6] è la piattaforma leader utilizzata per lo sviluppo di applicazioni relative all'intelligenza artificiale. I servizi offerti dalla piattaforma sono numerosi, di seguito verranno descritti quelli utilizzati all'interno del progetto.

#### Controllo di versione e Artefatti

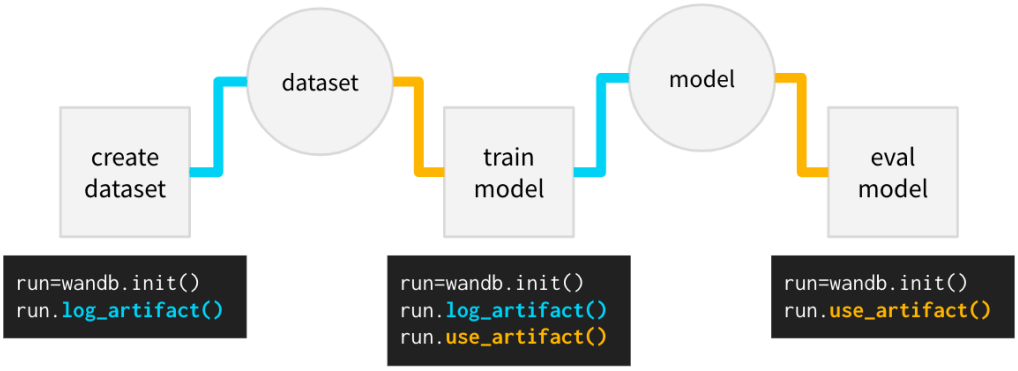



Figura 2.6: Diagramma dimostrante come gli artefatti di W&B possono essere utilizzati all'interno dell'intero processo di sviluppo di un progetto.
da https://docs.wandb.ai/guides/artifacts

Durante lo sviluppo di un progetto di Intelligenza Artificiale è comune dover effettuare molteplici esperimenti (run) relativi a un singolo fine-tuning utilizzando iperparametri o modelli differenti. Tenere traccia dei parametri utilizzati e dei risultati del processo può essere difficoltoso quando il numero di tentativi inizia a diventare consistente. Per risolvere questo problema è stata utilizzata una funzionalità di W&B che permette di effettuare il controllo di versione di ogni tentativo effettuato, salvando le configurazioni iniziali, come iperparametri e modello modello di partenza, il log delle operazioni di fine-tuning e al termine dello stesso salvare una copia (Artefatto) del modello riaddestrato (Figura 2.6).

La copia del modello riaddestrato potrà venire ripristinata in qualsiasi momento ed utilizzata per ulteriori fine-tuning o inferenze.
Per rendere più ordinata l'area di lavoro ogni esecuzione ad ogni run può essere assegnato un nome ed è anche possibile correlarla a un progetto specifico.

[6] `https://wandb.ai/site`

### Report e Tabelle

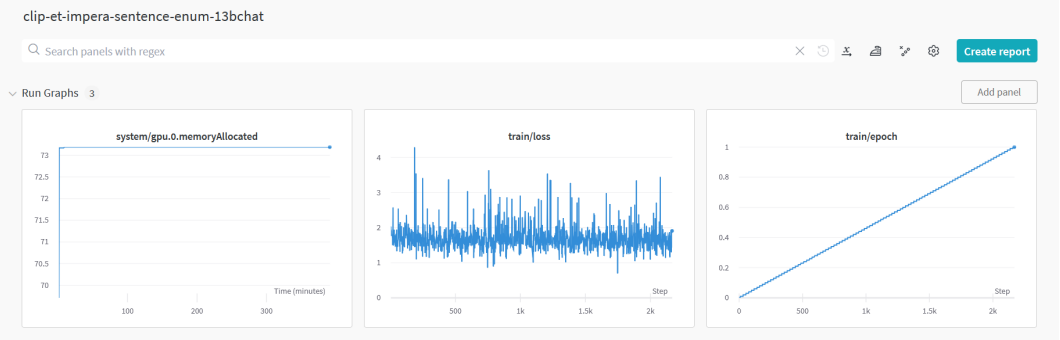



Figura 2.7: Esempio di grafici relativi a una specifica run forniti dalla schermata dedicata sulla piattaforma W&B.

Al termine di ogni run è possibile visitare una pagina dedicata in cui vengono mostrati i risultati delle operazioni effettuate. Ad esempio al termine di una operazione di fine-tuning vengono visualizzati automaticamente i grafici relativi al tracciamento dell'evoluzione dei parametri come learning-rate e loss del modello durante l'addestramento (Figura 2.7). Inoltre vengono anche fornite le statistiche relative all'utilizzo dell'hardware. Queste risultano molto utili per monitorare le risorse utilizzate da un modello durante le fasi di addestramento o inferenza.

### Prompt engineering

All Traces

| | Success | Timestamp | Input | Output |
|---|---|---|---|---|
| 3 | True | - | **0.question:** You are now an expert at splitting text into arguments. The input consists of several lines. Each line contains a phrase from the text and the number of the line. Your task is to identify the sentences that precede a change in an argument. Transcribe the indexes of the sentences in which the argument changes. Format the | **0.response:** , 11, 30, 43, 60, 72] |
| 4 | False | - | **0.question:** You are now an expert at splitting text into arguments. The input consists of several lines. Each line contains a phrase from the text and the number of the line. Your task is to identify the sentences that precede a change in an argument. Transcribe the indexes of the sentences in which the argument changes. Format the | **0.response:** |

3 - 4 of 541

Figura 2.8: Esempio di tabella fornita dallo strumento Traces per lo studio dei prompt.

Al fine di ottenere le performance migliori dal modello è necessario formulare

correttamente il prompt che gli viene fornito. Non esiste però uno schema preciso per sapere a priori se un cerco prompt ottiene o meno i risultati sperati ed è quindi necessario eseguire diversi test. Per facilitare le procedure legate a questi esperimenti W&B fornisce uno strumento chiamato Traces[7]. Nello specifico questo strumento permette di tracciare gli input forniti al modello e la relativa risposta, per poi rappresentare i dati in una comoda tabella all'interno del riassunto della run (Figura 2.8).

[7]https://wandb.ai/site/traces

# Capitolo 3

# Metodologia proposta

## 3.1 Dataset Proposto

### 3.1.1 Creazione del dataset

Nella fase iniziale, dopo un'attenta analisi dei dataset disponibili, ci si è resi conto che questi non soddisfacevano le esigenze specifiche del progetto proposto. Questo perché gran parte dei dataset esistenti si utilizza suddivisioni in capitoli ottenute tramite euristiche poco precise, compromettendo quindi la veridicità dei dati.
Poiché la metodologia proposta richiede un'ampia quantità di dati etichettati manualmente è stato necessario creare un nuovo dataset ad hoc per garantire una buona riuscita delle prossime fasi del progetto.

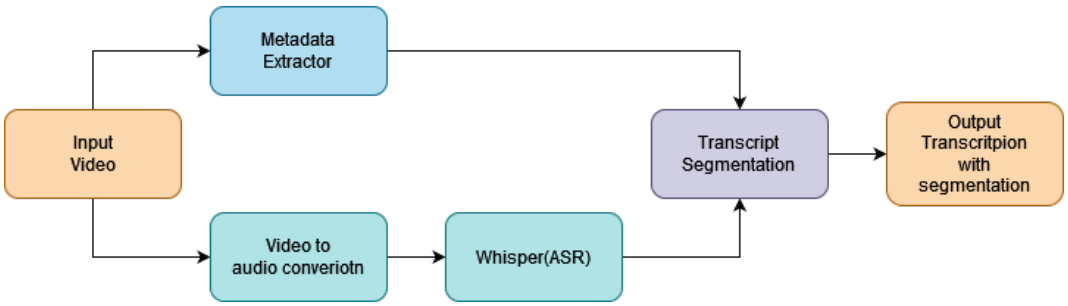



Figura 3.1: Processo di creazione dati proposto.

Per la creazione del dataset, è stato adottato un approccio metodologico rigoroso che ha coinvolto diverse fasi cruciali (Figura 3.1). Inizialmente è stato selezionato un insieme di video da fonti pubblicamente accessibili in internet che coprissero argomenti a carattere scientifico.

I video collezionati sono tutti accompagnati da un'indicizzazione effettuata manualmente, contenente i minuti di inizio, di fine e un titolo per ogni capitolo. Questi dati verranno poi utilizzati per suddividere la trascrizione in segmenti.

Trascrivere manualmente ogni video raccolto sarebbe stato impossibile e, per questo, è stato utilizzato un ASR.
Nello specifico è stato utilizzato Whisper in quanto offre un'ottima combinazione tra accuratezza e funzionalità avanzate. Whisper si è dimostrato estremamente preciso nel trascrivere l'audio, il che è stato di fondamentale importanza per garantire l'integrità delle trascrizioni. Inoltre, una caratteristica chiave di questo ASR è l'abilità di prevedere i timestamp a livello di frase, consentendo una marcatura temporale precisa di quelle presenti all'interno del video. Questo è stato fondamentale per la creazione del dataset, poiché ci ha permesso di collegare i punti temporali di inizio e fine dei segmenti con il punto esatto all'interno della trascrizione.

Poiché Whisper è in grado di operare solamente su file audio si è reso necessario convertire i video collezionati. Terminata la conversione ognuno di questi è stato dato in input a Whisper che in output ha restituito le trascrizioni, comprendenti di timestamp a livello di frase.
Seppure indicizzati manualmente i video presentavano delle piccole irregolarità tra il punto temporale indicato come delimitatore del segmento e l'effettivo termine di esso. Per evitare che una frase potesse essere troncata e divisa in due capitoli differenti si è scelto di dividere il video in prossimità della fine della frase pronunciata al minuto temporale designato come termine del segmento.

Terminata la fase estrattiva dei dati sono state eseguite delle operazioni di pulizia. Questa fase risulta fondamentale poiché l'eliminazione dei dati inconsistenti da un dataset è un'operazione necessaria per garantire la validità e l'affidabilità dei risultati ottenuti da qualsiasi derivato, come ad esempio un modello addestrato su questi dati.
Sono stati quindi esclusi i dati incompleti o ritenuti poco significativi. Alcune operazioni fondamentali svolte in questa fase sono state la rimozioni di dati riguardanti video con audio non presente, la rimozione dei video con lingua diversa da quella inglese e la rimozione di video con segmenti vuoti.

### 3.1.2 Analisi del dataset

Al termine delle operazioni di creazione è stata svolta un'attività di analisi del dataset atta a descriverne il contenuto.
Il principale motivo di estrarre grafici e statistiche è quella di cercare di comprendere a pieno il dataset con il quale si andrà a lavorare nelle fasi successive del progetto.

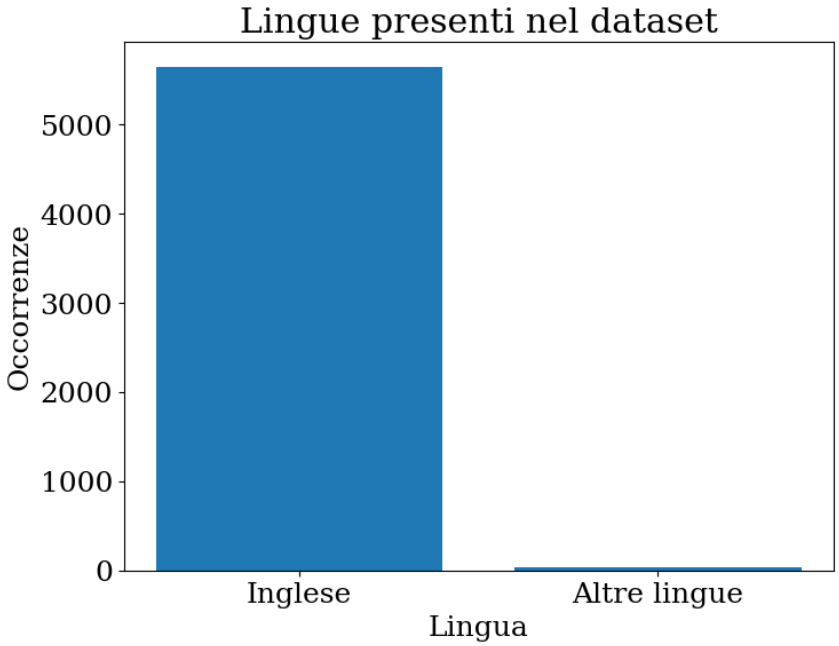



Figura 3.2: Differenza tra le occorrenze dei video in lingua inglese e quelli in altre lingue all'interno dei dati raccolti

Com'è facilmente visibile dall'istogramma presente in Figura 3.2 la stragrande maggioranza dei video analizzati contengono del parlato in inglese. In effetti, i video in questa lingua costituiscono una percentuale pari al 99.3% dell'intero dataset. Alla luce di ciò e in virtù della necessità di garantire la coerenza e l'omogeneità dei dati, è stata presa la decisione di escludere i video in lingua diversa dall'inglese. È inoltre importante notare che la percentuale di tali video è trascurabile rispetto al totale, e pertanto la loro esclusione non avrà un impatto significativo.

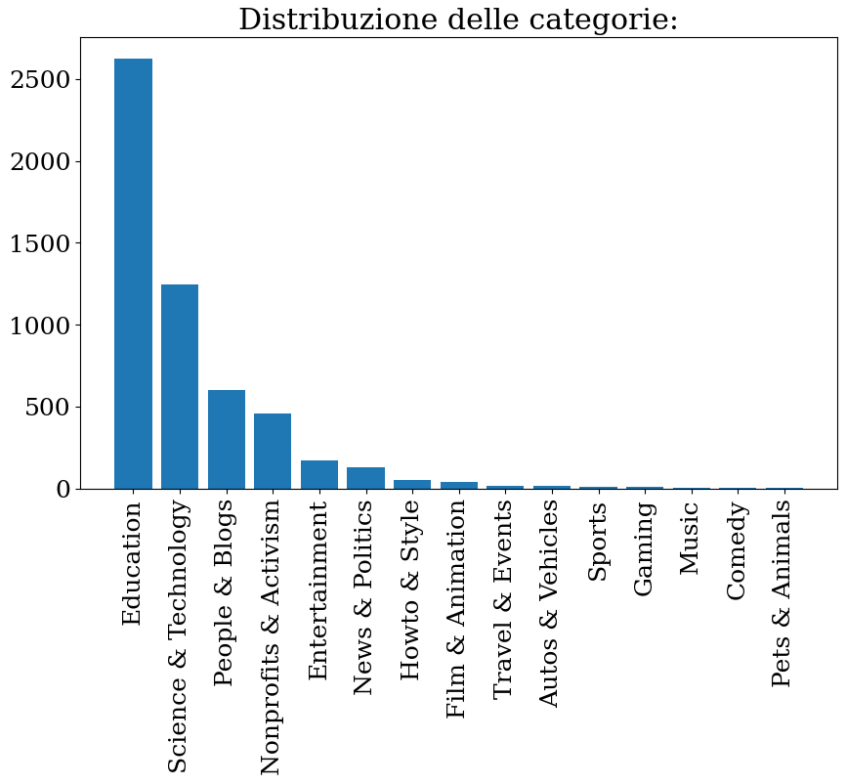



Figura 3.3: Grafico mostrante la distribuzione degli argomenti trattati all'interno dei video contenuti nel dataset proposto.

Il passo successivo ha comportato l'elaborazione statistica dei dati relativi agli argomenti al fine di acquisire una comprensione dettagliata delle categorie di video presenti nel dataset in esame. Il grafico risultante (Figura 3.3) ha chiaramente evidenziato che la maggior parte dei video tratta argomenti relativi a educazione e scienze e tecnologia. Solamente i video che trattano queste 2 categorie rappresentano il 71.37% del totale presente dei video analizzati.

Nel contesto del progetto in questione, è fondamentale sottolineare che la categoria specifica dei video non riveste alcuna importanza cruciale. Di conseguenza, non è prevista alcuna azione di eliminazione dei dati relativi alle categorie, poiché esse non influiscono significativamente sugli obiettivi e sui risultati del progetto.

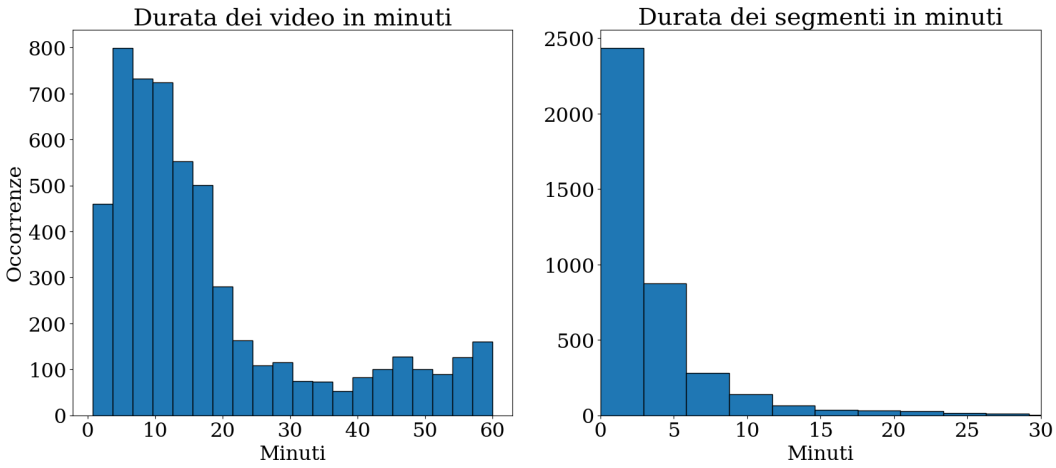



Figura 3.4: Durata dei video all'interno del dataset proposto.

Procediamo con l'analizzare la durata dei video e dei segmenti.
Com'è possibile notare dai dati rappresentati nei grafici (Figura 3.4), la stragrande maggioranza dei video all'interno nel dataset ha durata inferiore a 40 minuti. Solo il 14.1% dei video analizzati supera questa soglia, raggiungendo un massimo di un'ora di durata. Questo limite non è casuale, ma riflette una scelta precisa. Infatti, durante la fase di acquisizione dei dati, è stata applicata una limitazione sulla durata dei video, collezionando solo quelli con una durata massima di un'ora. Questo è stato fatto per facilitare la gestione dell'input fornito al LLM utilizzato nelle fasi successive del progetto. Limitare la durata dei video mira a evitare problemi relativi alla dimensione della context window del LLM, garantendo un'elaborazione efficiente e accurata delle informazioni.

Per quanto riguarda l'analisi della durata dei segmenti all'interno del dataset, emergono alcuni dati interessanti. In particolare, si osserva che la durata dei segmenti è omogenea, con la stragrande maggioranza di essi che rientra in un intervallo di tempo inferiore ai 10 minuti.
Si può quindi costatare che la durata dei segmenti risulta quasi sempre breve e che quindi non esista una dipendenza diretta tra la durata totale del video e quella dei segmenti

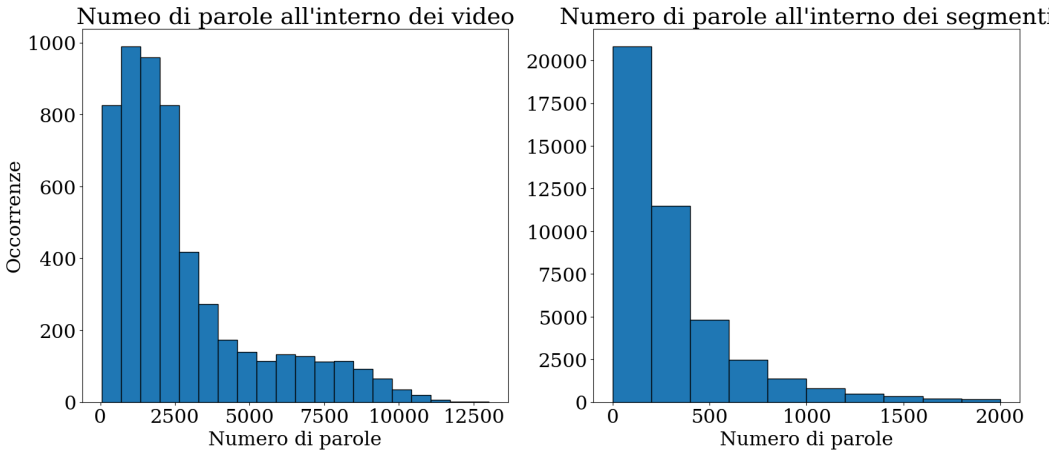



Figura 3.5: Numero di parole pronunciate all'interno dei video contenuti del dataset proposto.

Come annunciato precedentemente, date le limitazioni presenti con l'utilizzo di un LLM, un dato di cruciale importanza è quello dalla quantità di parole contenute all'interno del video. Grazie alla trascrizione e suddivisione del testo è stato possibile condurre uno studio del numero di parole presenti all'interno di ogni video e segmento.

Anche in questo caso ci troviamo davanti a due istogrammi (Figura 3.5) che presentato dati molto raggruppati tra loro.
Per quanto riguarda il grafico relativo al numero di parole presente all'interno di un video possiamo facilmente notare che la maggior parte di questi ne contiene meno di 4000.
Questa osservazione è di particolarmente rilevante in quanto rende il dataset adatto all'uso con un LLM moderno come Llama2. Questo infatti dispone di una context window predefinita di 4096 tokens. Ciò significa che la stragrande maggioranza dei video analizzati può essere completamente inserita all'interno dell'input del modello senza dover ricorrere ad strategie di suddivisione del testo o ad altre operazioni complesse per gestire la dimensione del testo in input.

Anche nel contesto del secondo grafico, si osserva una notevole densità dei dati, la cui disposizione era prevedibile in quanto, come precedentemente sottolineato, i segmenti presentano una distribuzione temporale uniforme e omogenea.

## 3.2 Preparazione del modello Clip-et-impera

Il modello "Clip-et-impera" costituisce il nucleo centrale di questo progetto e assume la responsabilità di individuare i confini tra gli argomenti all'interno del testo. Come punto di partenza, abbiamo selezionato il modello Llama2. Esistono diverse distribuzioni di questo LLM. Al fine di scegliere la più adatta è stata effettuata una analisi delle risposte base in modalità "zero shot". Questa analisi è stata svolta grazie all'utilizzo dello strumento "Trace" fornito da Weights and Biases che permette di tracciare facilmente i prompt e le relative risposte generate da parte di un determinato modello. Dopo una analisi qualitativa delle risposte delle varie versioni è stata selezionata la verso Llama-2-13b-chat[1], caratterizzato da 13 miliardi di parametri, con un preaddestramento specifico all'utilizzo conversazionale.

Le risposte del modello base risultavano coerenti con il prompt assegnato ma avevano alcuni problemi quali la precisione e la mancanza di un formato standard.
Come primo tentativo per risolvere questo problema si è tentato di utilizzare la metodologia "In Context Learning" in modalità "one shot". Da subito però si è reso evidente che la presenza di un esempio occupava troppo spazio all'interno del prompt, limitando notevolmente la massima lunghezza del testo che poteva essere analizzata dal modello. Si è quindi reso necessario effettuare il fine-tuning del modello.

### 3.2.1 fine-tuning di LLama2

Inizialmente si è effettuato uno studio sulle istruzioni da impartire al modello. Questa procedura richiede una valutazione attenta e sistematica del testo contenuto nel prompt fornito.
Durante questa fase sono stati provati diversi prompt il cui risultato è stato tracciato ancora una volta grazie all'utilizzo di Trace.

Dopo aver selezionato il prompt, il passo successivo coinvolge la preparazione dei dati in modo tale che possano essere compresi dal modello Llama2. Per massimizzare l'efficacia del fine-tuning su modelli come questo è fondamentale organizzare l'input in un formato coerente con quello atteso dal modello e che includa sia il prompt che il risultato desiderato.

[1] `https://huggingface.co/meta-llama/Llama-2-13b-chat`

```
"<s>[INST] {{prompt}} + {{transcription}} [/INST]
{{expected_model_answer}}</s>"
```

Frammento di codice 3.1: Esempio di input per l'addestramento con formato coerente a Llama2 chat.

Una volta pronto l'input si continua con la preparazione modello alla procedura di fine-tuning. Per limitare le risorse necessarie e il tempo di addestramento sono state applicate le tecniche Lora e la quantizzazione in 4-bit.

La procedura di fine-tuning è composta da diversi step. All'interno di ognuno di essi al modello viene fornita una stringa, contenente l'insieme indicizzato delle frasi pronunciate nel video e quello che gli viene chiesto di prevedere è la lista degli indici delle frasi nel quale avviene il cambio di argomento.
Al termine di ogni step i pesi del modello vengono aggiornati in modo da minimizzare l'errore nella risposta.

Terminato il fine-tuning il modello addestrato è stato salvato tramite l'utilizzo della funzionalità di "Weight and Bias" per il controllo di versione nel modello.

## 3.3 Clip-et-impera, metodologia proposta

In questa sezione andremo a descrivere il funzionamento della soluzione proposta (Figura 3.6), partendo prima da una generale introduzione per poi passare alla descrizione dettagliata di ogni componente del progetto.

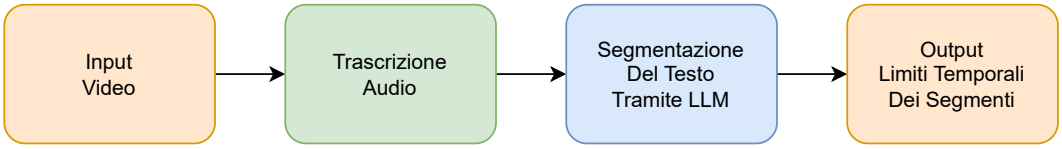



Figura 3.6: Schema generale della soluzione proposta.

Come prima cosa al programma viene fornito in input il video di cui si vogliono ottenere i segmenti. Una volta ottenuto si procederà automaticamente con l'estrazione l'audio, per poi passarlo a Whisper [19], l'ASR scelto. In questa fase viene sfruttata la caratteristica di Whisper nel prevedere il minutaggio nel quale viene pronunciata una certa frase in modo da costruire una struttura dati contenente le associazioni minutaggio - frase necessarie per ricostruire la soluzione. Il passo successivo prevede di formattare il testo estratto, per poi passarlo a "Clip-et-impera", il LLM ottenuto tramite fine-tuning di LLama2, addestrato specificatamente per la segmentare il testo in input. Una volta che il modello avrà generato la soluzione il programma procederà con la ricostruzione della soluzione, per poi fornire in output i limiti temporali di ogni segmento.

### 3.3.1 Trascrizione dell'audio

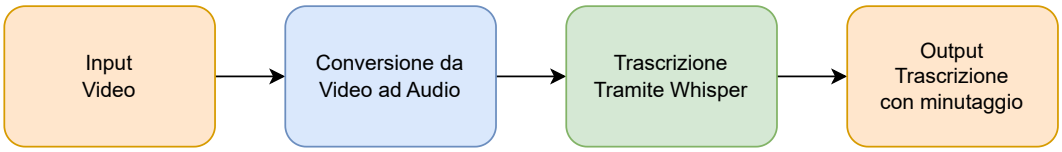



Figura 3.7: Schema del processo di trascrizione del video.

Come precedentemente annunciato per la parte di trascrizione dell'audio contenuto all'interno del video si è deciso di utilizzare Whisper sia a causa della sua alta accuratezza ma soprattutto per la sua capacità di riuscire a prevedere con precisione il minutaggio di ogni frammento di audio.

Prima di poter utilizzare Whisper è però necessario convertire il video in audio. Esistono diverse librerie in linguaggio python che offrono questa funzionalità, nel nostro caso è stata utilizzata moviepy[2].

```
import moviepy.editor as mp

video_path = "path/to/the/video.mp4"

clip = mp.VideoFileClip(video_path)
# Extract video name from the path and create the new audio path
video_name = video_path.split('/')[-1]
audio_path = 'data/' + video_name.replace("mp4","mp3")

clip.audio.write_audiofile(audio_path)
```

Frammento di codice 3.2: Conversione da video ad audio usando la libreria moviepy.

Una volta ottenuto il file audio sarà possibile procedere con l'estrazione del testo tramite Whisper. Per integrare questo modello all'interno del programma è stata utilizzata la libreria whisper[3] che ne permette un facile utilizzo.
Una volta importata la libreria e caricato il modello sarà possibile ottenere la trascrizione tramite la funzione creata appositamente.

```
import whisper
# Load the base version of the whisper model
model = whisper.load_model("base")

def get_whisper_output(audio_file_path):
    with open(audio_file_path, 'rb') as audio_file:
        transcription = model.transcribe(audio_file_path)
    return transcription

whisper_transcription = get_whisper_output(audio_path)
```

Frammento di codice 3.3: Estrazione della trascrizione utilizzando Whisper.

Per le prossime fasi sarà indispensabile formattare le frasi contenute all'interno del video, separandole tra loro e annotando il punto temporale nel quale

[2] https://zulko.github.io/moviepy/
[3] https://github.com/openai/whisper

vengono pronunciate. Occorrerà quindi estrarre queste informazioni dall'output di Whisper.
Questo si compone di un array a due elementi. Il primo contiene la trascrizione completa del video mentre il secondo un array di dizionari python contenenti informazioni su quando una specifica porzione di testo è stata pronunciata. Attraverso questo secondo elemento è possibile costruire una struttura dati che rispetti le nostre esigenze.

Dopo le operazioni di analisi dell'ouput viene quindi creato un array di dizionari python contenente le informazioni di una specifica frase.

```
{'sentence': ' But people have a hard time agreeing on exactly how.',
  'start': 12.88,
  'end': 16.02}
```

Frammento di codice 3.4: Esempio di contenuto all'interno una cella dell'array.

### 3.3.2 Segmentazione del testo usando il modello Clip-et-impera

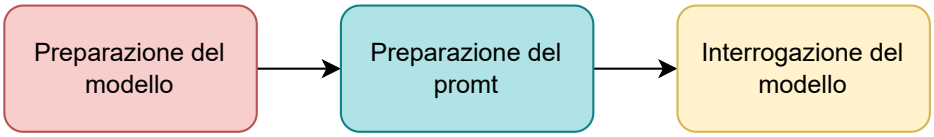



Figura 3.8: Schema del processo di interrogazione del modello ed ottenere i limiti per la segmentazione del video.

Si procede quindi con la procedura di segmentazione del testo (Figura 3.8). Per prima cosa andiamo a preparare il modello. Per farlo sono necessarie due librerie, wandb[4] per scaricare il checkpoint del modello e Transformers[5] per caricarlo all'interno della scheda video. Durante questo processo viene anche utilizzata la libreria bitsandbytes per la sua quantizzazione. Preparato il modello si procede con il creare il tokenizer che verrà poi utilizzato durante il processo di interrogazione.

[4] https://docs.wandb.ai/ref/python/
[5] https://huggingface.co/docs/transformers/index

```
# Download model weights to a folder and return the path
model_dir = my_model_artifact.download()

# Configure the quantization parameters
bnb_config = BitsAndBytesConfig(
  load_in_4bit=True,
  bnb_4bit_quant_type="nf4",
  bnb_4bit_compute_dtype=torch.bfloat16,
)

# Import the model
model = AutoModelForCausalLM.from_pretrained(
  model_dir,
  quantization_config=bnb_config
)

# Import the tokenizer
tokenizer = AutoTokenizer.from_pretrained(model_dir)
```

Frammento di codice 3.5: Preparazione del modello.

Prima di poter procedere con l'interrogare il modello è necessario formattare correttamente l'input, in modo che sia coerente con quello visto durante l'addestramento. É stata creata quindi una nuova stringa con all'interno le frase contenute nel video, una per riga preceduta dell'indice di posizione. A questa è stato poi aggiunto il prompt base e codificata con il pattern tipico per le inferenze a LLama2.

```
"""
...
30.  What happens when general purpose AI hits the real economy?
31.  How do things change?
32.  Can we adapt?
...
"""
```

Frammento di codice 3.6: Esempio di input formattato.

Una volta preparato l'input si procede con l'interrogare il modello per ricevere l'elenco di indici delle frasi nel quale avviane il cambio di argomento. Per farlo prima è necessario codificare il prompt in token comprensibili al modello tramite il tokenizer. Nel caso in cui il numero di token in input dovesse superare

la dimensione della contex window il codice solleverebbe un'eccezione. Una volta ricevuta la risposta dal modello il tokenizer si occuperà di convertire i token ricevuti in parole.

```
# Tokenize the prompt
input_ids = tokenizer.encode(
    text,
    return_tensors="pt",
)
# If the input is longer than the context lenght the program raise an
    ecception
if len(input_ids[0]) > 4096:
    raise TooMuchTokens(len(input_ids[0]), 4096)

# Generate the solution using the LLM
device = model.device
generated_tokens_with_prompt = model.generate(
    input_ids=input_ids.to(device),
)

# Decode the answer
generated_text_with_prompt =
    tokenizer.batch_decode(generated_tokens_with_prompt,
    skip_special_tokens=True)

# Strip the prompt
generated_text_answer = generated_text_with_prompt[0][len(text):]
```

Frammento di codice 3.7: Processo di interrogazione del modello.

### 3.3.3 Ricostruzione della soluzione

Una volta resa utilizzabile la risposta del modello si passa alla fase di ricostruzione della soluzione. L'output del modello consiste in un array di numeri interi, rappresentanti l'indice della frase corrispondente al cambio di argomento.
Tramite la struttura dati precedentemente creata vengono riconvertiti gli indici delle frasi presenti all'interno della risposta come punti temporali all'interno del video di partenza. L'output di questo programma sarà quindi l'insieme di questi punti temporali raccolti.

# Capitolo 4

# Configurazioni utilizzate, esperimenti e risultati

In questo capitolo si andranno ad analizzare i diversi dettagli relativi all'addestramento, le performance ottenute, gli iperparametri utilizzati e l'ambiente di esecuzione del modello.
Nello specifico riguardo alle performance si andrà a discutere come sono state effettuate le valutazioni del modello e verranno citate le metriche utilizzate.

Per quanto riguarda gli iperparametri verranno analizzati i risultati ottenuti con diverse combinazioni di valori.

Infine per garantire la riproducibilità dei risultati verrà descritto l'ambiente di sviluppo, citando l'hardware e le principali librerie utilizzate.

## 4.1 Processo di valutazione del modello

Terminata la procedura di addestramento è stato chiesto al modello addestrato di prevedere i punti di cambio argomento nelle trascrizioni contenute nel validation set. Per ogni previsione è stato calcolato il punteggio ottenuto con le metriche Pk Score e WindowDiff.
Fatto questo è stato possibile calcolare le performance del modello facendo la media di tutti i punteggi ottenuti.

Di seguito vengono riportati alcuni esempi di risposte del modello comparate alla risposta attesa e con corrispettivo valore di Pk Score e WindowDiff.

| Text | Model Prediction | Reference Segmentation | Pk Score | Window Diff |
|---|---|---|---|---|
| 1. Earth's interconnected cycles.<br>2. Air, land, water, and life are all interconnected.<br>3. The different systems of Earth are known as the atmosphere, lithosphere, hydrosphere, and biosphere.<br>4. These systems work together to create our world.<br>....<br>31. Breathe the air, the atmosphere, and drink the water, the hydrosphere.<br>32. We could not exist without any one of these systems. | [6, 12, 15, 18, 20, 25, 30] | [8, 12, 15] | 0.571 | 0.5 |

Continued on next page

Tabella 4.0: (Continued)

| | | | | |
|---|---|---|---|---|
| 1. The discussion section is where you summarize what you found and what it means.<br>2. In this section, you should emphasize what you learned, relate it to any questions raised in the introduction section, and know any implications of your results.<br>3. First, summarize your results.<br>4. Repeat information from the results section, this time using much simpler language and without any statistics.<br>...<br>22. It could even be a study that addresses some of the<br>limitations of your work.<br>23. And that's it. | [11, 16, 21] | [7, 11, 16, 21] | 0.125 | 0.125 |

Continued on next page

Tabella 4.0: (Continued)

| | | | | |
|---|---|---|---|---|
| 1. What would life be without blood?<br>2. Well that's a silly question because<br>we wouldn't be here.<br>3. Nor would any mammals, or actually<br>any birthabrasin fact.<br>...<br>29. They may only be 1% of our blood, but our white blood cells can really peg a punch.<br>30. You just need to remember that phagocytes and gulf pathogens and lymphocytes produce antibodies and antitoxans. | [10, 13, 22, 26] | [6, 10, 13, 22, 26, 27] | 0.103 | 0.214 |

## 4.2 Risultati quantitativi e iperparametri

Le operazioni di fine-tuning sono state svolte utilizzando diversi iperparametri in modo da cercare di ottenere il modello che produca i risultati migliori.

| Model Version | Epoch | Gradient Accumulation | Learning Rate | Pk Score | Pk Score min | Pk Score max | WindowDiff | WindowDiff min | WindowDiff max |
|---|---|---|---|---|---|---|---|---|---|
| Llama2 7b | 1 | 2 | 2e-4 | 0.403 | **0** | **0.918** | 0.272 | **0** | **1** |
| | 2 | 2 | 2e-4 | 0.394 | 0.026 | 1 | 0.254 | 0.025 | **1** |
| | 1 | 2 | 1e-4 | 0.414 | **0** | **0.918** | 0.283 | **0** | **1** |
| | 1 | 4 | 2e-4 | 0.439 | 0.026 | 1 | 0.918 | 0.019 | **1** |
| Llama2 13b | 1 | 2 | 2e-4 | 0.390 | **0** | 1 | 0.254 | **0** | **1** |
| | 2 | 2 | 2e-4 | **0.374** | **0** | **0.918** | **0.251** | **0** | **1** |
| | 1 | 2 | 1e-4 | 0.413 | **0** | **0.918** | 0.287 | **0** | **1** |
| | 1 | 4 | 2e-4 | 0.421 | **0** | 0.941 | 0.314 | **0** | **1** |
| Llama2 13b chat | 1 | 2 | 2e-4 | 0.415 | 0.029 | 1 | 0.273 | 0.025 | **1** |
| | 3 | 2 | 2e-4 | 0.394 | **0** | **0.918** | 0.262 | **0** | **1** |

Tabella 4.1: Risultati dei fine-tuning con diverse configurazioni di iperparametri.

Come è possibile notare i risultati sono molto buoni, prova quindi che il modello sia in grado di identificare con buona precisione i limiti tra gli argomenti.
Per quanto riguarda la sperimentazione con gli iperparametri di training, a causa della contex window limitata, la presenza di input molto lunghi e la limitata quantità di memoria grafica, non è stato possibile esplorare alcuni di questi. Ad esempio non è stato possibile sperimentare con il parametro "batch size" poiché il suo incremento causava un'istantanea saturazione delle risorse GPU disponibili.

Alcuni esempi di iperparametri esplorati sono invece il "learning rate, "gradient accumulation" e "epoch number" in diverse configurazioni e applicati a diversi modelli base.
Uno di questi il cui incremento ha creato un miglioramento di performance è il valore di "epoch number". Generalmente infatti si può notare che aumentando questo valore è stato possibile ridurre l'errore sugli elementi forniti in test. Una modifica che ha invece determinato una degradazione delle performance è quella associata all'iperparametro "gradient accumulation". Come si puó notare dalla Tabella 4.1 raddoppiando il valore da 2 a 4 l'errore è aumentato significativamente.

La configurazione con le prestazioni migliori è stata trovata con le seguenti scelte di modello di partenza e iperparametri:

- Modello: Llama2 13b
- Quantizzazione: 4 bit
- Lora:
  - r: 8
  - alpha: 32
- Max sequence length: 3500
- Gradient accumulation steps: 2
- Learning rate: 2e-4
- Epoch number: 2
- Batch size: 1

## 4.3 Dettagli implementativi

Le run relative all'addestramento del nostro modello sono state eseguite su una GPU Nvidia GeForce RTX 3090 con 24 GB di memoria dedicata, 64 GB di VRAM e un processore Intel® Core™ i9-10900X a 3,70 GHz.

Per garantire la riproducibilità è stato creato un container Docker.
Il codice si basa sulle seguenti librerie:

- É stata utilizzata la libreria **PyTorch** (versione 1.12.1) in combinazione con il toolkit **CUDA** (versione 113) per l'accelerazione GPU. Questa configurazione permette l'esecuzione del codice per il fine-tuning all'interno della scheda video.
- Per importare i modelli preaddestrati è stata utilizzata la libreria **HuggingFace Transformers** (versione 4.34.1).
- Per effettuare le operazioni di fine-tuning è stata usata la libreria **TRL** (versione 0.7.2) che offre un'interfaccia facilitata per effettuare le operazioni di fine-tuning.

- Per quantizzare il modello è stata utilizzata la libreria **bitsandbytes** (versione 0.41.1).
- Per utilizzare le metodologie Lora è stata utilizzata la libreria **Peft** (versione 0.5.0).
- Per interfacciarsi con la piattaforma di W&B è stata utilizzata la libreria **wandb** (versione 0.15.7).

# Conclusioni e sviluppi futuri

In sintesi, questa tesi ha indagato nel dettaglio una metodologia innovativa per la segmentazione di video in capitoli, sfruttando tecnologie all'avanguardia che rappresentano lo stato dell'arte nei rispettivi settori. Il contenuto della tesi ha contribuito significativamente all'avanzamento della ricerca in questo campo grazie ai due principali contributi.

Il primo contributo di rilievo è stato la creazione di un nuovo dataset, comprendente oltre 4000 video opportunamente suddivisi in capitoli. Questa proposta ha colmato la carenza di dati disponibili per la ricerca, fornendo una risorsa preziosa per il training di nuovi modelli e l'analisi delle performance di quelli già esistenti. Il dataset rappresenta un ottimo strumento per future indagini nel campo della segmentazione video, offrendo la possibilità di eseguire valutazioni più approfondite e consentendo lo sviluppo di algoritmi sempre più sofisticati.

Il secondo contributo chiave è stato l'implementazione di un sistema inedito per la segmentazione di video. La metodologia proposta, composta dall'analisi del testo da parte di un Large Language Model, si è dimostrata efficace nell'analizzare e suddividere video in capitoli e ha aperto la strada a nuove prospettive di sviluppo.
Tuttavia, il fine-tuning del modello può essere ulteriormente indagato, ricorrendo ad hardware più prestante per una esplorazione più approfondita dello spazio di iperparametri.

In conclusione, i risultati ottenuti suggeriscono che la metodologia proposta ha un buon potenziale e rappresenta un punto di partenza per ulteriori sviluppi. Con la continua innovazione delle tecnologie legate ai Large Language Model e al miglioramento delle dimensioni della context window ci si auspica la possibilità di utilizzare questo metodo anche in video contenenti un maggiore numero di parole, contribuendo così alla crescita continua del campo della segmentazione video.

# Ringraziamenti

Desidero ringraziare il Professore Gianluca Moro per avermi dato la possibilità di svolgere questo progetto di tesi, fornendomi anche le conoscenze e risorse necessarie per lo sviluppo. Un ringraziamento speciale lo dedico al Co-Relatore Giacomo Frisoni che mi ha affiancato durante lo sviluppo del progetto.

Un ringraziamento particolarmente sentito lo dedico alla mia famiglia, che ha permesso il mio percorso universitario e che tuttora continua a sostenermi e incoraggiarmi.

Desidero ringraziare tutti gli amici che ci sono sempre stati e che hanno riempito di gioia le mie giornate.

Un sincero ringraziamento va anche alle persone che hanno condiviso con me le sfide di questo percorso. La vostra amicizia ha reso questo viaggio universitario un'esperienza indimenticabile.

Infine, desidero ringraziare tutti i professori che durante il mio intero percorso scolastico mi hanno ispirato e guidato. La vostra dedizione all'insegnamento sono state fonti di motivazione e apprendimento.

*Luca, 30/11/23*

# Bibliografia


[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[2] Dipesh Chand and Hasan Oğul. A framework for lecture video segmentation from extracted speech content. In *2021 IEEE 19th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pages 000299–000304, 2021.

[3] Mezaris V. Galanopoulos, D. Temporal lecture video fragmentation using word embeddings. 12 2018.

[4] Di Ma, Xi Zhang, Xu Ouyang, and Gady Agam. Lecture vdeo indexing using boosted margin maximizing neural networks. pages 221–227, 12 2017.

[5] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[6] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

[7] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration, 2023.

[8] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023.

[9] Lev Pevzner and Marti A. Hearst. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28(1):19–36, 2002.

[10] Doug Beeferman, Adam Berger, and John Lafferty. Statistical models for text segmentation. *Mach. Learn.*, 34(1–3):177–210, feb 1999.

[11] Barabanov A. Kocharov D. Skrelin P. Moiseev M. Salishev, S. Voice activity detector (vad) based on long-term mel frequency band features. 2016.

[12] Moens MF. Dekens T. Poulisse, GJ. News story segmentation in multiple modalities. 2009.

[13] Abhishek Dhankar. Study of deep learning and cmu sphinx in automatic speech recognition. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 2296–2301, 2017.

[14] Eduardo R. Soares and Eduardo Barrére. An optimization model for temporal video lecture segmentation using word2vec and acoustic features. In *Proceedings of the 25th Brazillian Symposium on Multimedia and the Web*, WebMedia '19, page 513–520, New York, NY, USA, 2019. Association for Computing Machinery.

[15] S. Ternstrom R. Leanderson P. Gramming, J. Sundberg and W. H. Perkins. Relationship between changes in voice pitch and loudness. 1988.

[16] John Neuhoff, Gregory Kramer, and Joseph Wayand. Pitch and loudness interact in auditory displays: Can the data get lost in the map? *Journal of experimental psychology. Applied*, 8:17–25, 04 2002.

[17] Darshan Singh S, Anchit Gupta, C. V. Jawahar, and Makarand Tapaswi. Unsupervised audio-visual lecture segmentation, 2022.

[18] Tengchao Lv, Lei Cui, Momcilo Vasilijevic, and Furu Wei. Vt-ssum: A benchmark dataset for video transcript segmentation and summarization, 2021.

[19] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision, 2022.

[20] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021.

[21] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them, 2022.

[22] Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. Agieval: A human-centric benchmark for evaluating foundation models, 2023.