Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica per il Management

ANALISI DI SCENEGGIATURE CINEMATOGRAFICHE CON METODI QUANTITATIVI: APPLICAZIONE WEB E CASO DI STUDIO

Relatore:
Chiar.mo Prof.
ANGELO DI IORIO

Presentata da:
ALESSANDRO
CAMPEDELLI

II Sessione
Anno Accademico 2024/25

Ad Alice,

A te che hai sempre creduto in me,
A te che mi hai donato l'amore che cercavo,
A te che mi dai la forza in ogni momento,
A te che sei la mia guerriera,

A te che sei la mia vita.

Indice

Elenco delle figure

Elenco delle tabelle

Listings

Introduzione

Il tema di questa tesi è la realizzazione di un'applicazione web denominata **CAST** (Cinematic Analysis and Statistics Tool) [1]. L'idea alla base della creazione di questo sistema emerse quando, a seguito di una ricerca approfondita sul web riguardante tool specializzati nel settore cinematografico, si notò che la disponibilità pubblica di migliaia di copioni cinematografici rappresentava un patrimonio informativo ancora poco studiato. Per questo, CAST è pensato per iniziare a sfruttare maggiormente questa risorsa, creando un sistema modulare di analisi in grado di eseguire studi aggregati su un insieme di film, oppure più dettagliatamente su uno soltanto. L'utenza consigliata al sistema implementato comprende tutti gli esperti di dominio interessati ad eseguire analisi su determinate dimensioni rilevanti, oppure, ad utenti appassionati di cinema che vogliono approfondire la loro conoscenza a proposito dei loro film preferiti. Una caratteristica fondamentale di CAST è la sua modularità: il sistema è studiato per essere generico e estendibile in modo tale da soddisfare tutte le richieste eterogenee di analisi possibili, che ogni diverso esperto potrebbe richiedere. In particolare, attualmente sono state implementate solo alcuni esempi di analisi interessanti come ad esempio aspetti legati all'ambientazione e alla temporalità di una scena.

La volontà di sfruttare al meglio la presenza di migliaia di copioni cinematografici liberamente accessibili è soddisfatta implementanto una pipeline di esecuzione completa. Quest'ultima prende in input un insieme di copioni cinematografici in formati eterogenei (scaricati manualmente dagli archivi pubblici) e a seguito di diverse fasi di esecuzione vengono prodotti risultati all'interno di una pagina web. La pipeline implementata è suddivisa in tre fasi intermedie:

- **estrazione** - il sistema estrae il testo del copione da sorgenti in formato pdf e html, salvandone il contenuto in un file testuale;
- **conversione** - successivamente, questi file di testo vengono convertiti in file scritti con il linguaggio di markup TEI-XML. Questo formato permette di strutturare il copione formalmente attraverso dei tag specifici, creandone così uno ben strutturato, suddiviso in varie sezioni (scene, speech, battute, speaker...) e pronto per essere analizzato;

[1] Link repository github: `https://github.com/alessandrocampedelli/CAST`

- **analisi** - infine il copione strutturato viene analizzato per produrre risultati statistici aggregati o specifici al singolo film.

Una volta terminate queste fasi intermedie, i risultati vengono visualizzati all'interno di una pagina web inseriti in una dashbaord. Tramite essa, gli esperti di dominio potranno osservare e interagire con i grafici realizzati per le diverse dimensioni. In questo modo essi possono eseguire analisi in fase di produzione di nuove realizzazioni cinematografiche basandosi su tendenze passate, migliorando così la loro qualità e non incappando in errori di produzione banali.

La tesi è organizzata nei seguenti capitoli:

1. **capitolo 1**: presenta il contesto generale nel quale si posiziona CAST, facendo un focus su due tipologie di tool cinematografici (portali di visualizzazione e strumenti di supporto alla scrittura);
2. **capitolo 2**: illustrazione degli obiettivi di CAST, descrizione ad alto livello della logica del sistema e spiegazione dettagliata dell'utilizzo di quest'ultimo lato utente;
3. **capitolo 3**: descrizione dei dettagli implementativi della fase di estrazione e conversione dei copioni;
4. **capitolo 4**: descrizione dei dettagli implementativi della fase di analisi dei copioni;
5. **capitolo 5**: illustrazione e validazione dei risultati ottenuti;
6. **capitolo 6**: capitolo conclusivo nel quale si fanno delle considerazioni finali sui risultati ottenuti, viene espresso un breve riepilogo di tutto il processo di implementazione, si analizzano i punti deboli e i punti di forza di CAST e infine si propongono dei possibili sviluppi futuri.

Capitolo 1

Piattaforme per analizzare ed estrarre informazioni da copioni di film

Fino a prima della nascita di Internet, come lo intendiamo oggi, accedere a informazioni che ora riputiamo "facilmente accessibili" era molto complicato. Un settore che ha tratto un forte beneficio sotto questo punto di vista è sicuramente l'industria cinematografica. Grazie a Internet e i motori di ricerca, oggi è possibile conoscere ogni singolo dettaglio su qualsiasi film, grazie alla presenza sempre maggiore di repository pubblici che svolgono il ruolo di veri e propri archivi cinematografici.

1.1 Descrizione dei repository fornitori del contributo

Sul web, al giorno d'oggi, sono presenti tantissimi repository pubblici attraverso i quali gli utenti appassionati possono accedere ai copioni integrali dei loro film preferiti. Gli utenti in questo modo possono apprezzare da un punto di vista tecnico come viene scritta la sceneggiatura di un film e catturare dei dettagli che magari in fase di visione della pellicola farebbero fatica a cogliere. Questi database rappresentano oggi un corpus testuale prezioso per applicazioni di text mining e analisi computazionale. La presenza di tali repository ha aperto nuove possibilità di ricerca quantitativa sull'industria cinematografica, permettendo analisi su larga scala prima impossibili.

Sempre più spesso i copioni forniti in questi archivi sono in diversi formati, i più comuni sono:

- **html**: le pagine web che vengono visualizzate sul browser contengono il testo del copione integrale;
- **pdf**: attraverso diversi siti è possibile scaricare i file pdf dei copioni;
- **FDX (Final draft)**: è il formato usato dal programma più famoso per scrivere copioni a Hollywood;

- **CELTX**: simile a Final Draft, è il formato di un altro programma popolare per scrivere copioni.

I principali archivi cinematografici trovati online sono:

- **Script Slug**
- **Screenplays.io**
- **IMSDb**
- **Sprinfield! Springfield!**

Vediamo più nel dettaglio le caratteristiche di ogni repository menzionato.

1.1.1 Script Slug

Script slug [1] è un repository online cinematografico specializzato nella raccolta di migliaia di sceneggiature, nato appositamente per scopi didattici e di studio. La piattaforma offre la possibilità di scaricare più di 1000 copioni differenti distribuiti in formato pdf, principalmente hollywoodiani. Dunque, le caratteristiche principali del sistema sono:

- **formato disponibile**: PDF;
- **tipologia di contenuti**: le sceneggiature presenti sull'archivio sono riferiti a film prodotti professionalmente, ponendo particolare attenzione sulle produzioni più recenti e di successo. Inoltre, sono anche accessibili script riguardanti serie tv e podcast;
- **accessibilità**: la piattaforma è completamente gratuita e non richiede l'ausilio di login per usufruire dei servizi disponibili;
- **organizzazione**: il portale offre la possibilità di visualizzare gli script disponibili ordinati per genere, regista, casa di produzione cinematografica e premi oscar degli anni passati.

In aggiunta, Script Slug mette a disposizione una sezione dedicata a diversi articoli relativi a interviste, approfondimenti e curiosità riguardante il mondo cinematografico. Infine è disponibile anche una pagina relativa alle risorse: l'idea è supportare l'utente nell'espansione del proprio toolkit cinematografico attraverso libri, software di produzione, blog, podcast e competizioni di scrittura di un copione.

1.1.2 Screenplays.io

Screenplays.io [2] è un repository online cinematografico anch'esso specializzato nella raccolta di migliaia di sceneggiature. Nella descrizione del sito viene espresso appositamente il fatto che l'idea della nascita del repository ha come obiettivo quello di fornire un ampio insieme di script. Questo porterà diversi scrittori a migliorare la loro qualità di scrittura, prendendo spunto dai copioni esistenti. Le caratteristiche di Screenplays.io sono molto simili a quelle di Script Slug:

- **formato disponibile**: PDF;
- **tipologia di contenuti**: film di ogni genere ed epoca, serie televisive (episodi pilota e stagioni complete), documentari;
- **numero di film**: migliaia di sceneggiature cinematografiche e numerosi script televisivi;
- **accessibilità**: la piattaforma è completamente gratuita e non richiede l'ausilio di login per usufruire dei servizi disponibili;
- **organizzazione**: gli script sono visualizzabili per popolarità, tipo di script (film o serie tv), genere e studi cinematografici;
- **qualità dei contenuti**: mix di draft versions, shooting scripts e versioni post-produzione; spesso sono disponibili multiple versioni dello stesso film (es. prima bozza vs versione finale).

1.1.3 IMSDb

IMSDb (Internet Movie Script Database) [3] è uno dei primi archivi cinematografici online che è stato reso pubblico. Per questo motivo, sebbene disponga di un interfaccia grafica poco moderna, è ancora oggi uno dei più popolari. A differenza dei repository trattati fino ad adesso, esso mette a disposizioni i copioni in un formato differente: il testo dei vari script è direttamente disponibile sulla pagina web. Questa caratteristica consente all'utente di non dover scaricare nessun file rispetto ai servizi precedenti, ma dall'altra parte lo vincola a visitare la pagina ogni qual volta lo ritenga necessario. Vediamo più nel dettaglio le sue caratteristiche:

- **formato disponibile**: HTML;
- **tipologia di contenuti**: prevalentemente film hollywoodiani, con focus su produzioni dagli anni '80 in poi. Inoltre è presente una collezione limitatissima di serie TV;

- **numero di film**: circa 1000 (quantità minore rispetto ai repository precedenti);
- **accessibilità**: completamente gratuito con possibilità di creazione di un account;
- **organizzazione**: indice alfabetico per titolo, ricerca per genere cinematografico, e sezione "top rated scripts" basata su valutazioni degli utenti.

Una funzionalità aggiuntiva di IMSDb è la possibilità, da parte degli utenti, di inviare e di rispondere a dei commenti relativi a un determinato copione. Inoltre, nel commento è possibile attribuire anche una valutazione da zero a dieci stelle. Queste due caratteristiche consentono a IMSDb di far interagire la propria utenza, creando discussione con un conseguente aumento di popolarità.

1.1.4 Sprinfield! Sprinfield!

Springfield! Springfield! [4] è un repository specializzato in trascrizioni complete di dialoghi cinematografici e televisivi. In origine, l'archivio nacque esclusivamente per le serie tv e in particolare per *The Simpson*, dalla quale deriva il nome. Con il passare del tempo, il repository si espanse includendo migliaia di film e programmi televisivi. Come per IMSDb, tutte le risorse disponibili sono espresse direttamente all'interno delle pagine html. Di seguito vengono illustrate le caratteristiche dettagliate:

- **formato disponibile**: HTML;
- **tipologia di contenuti**: trascrizioni complete dei dialoghi e copioni "dialogue-only" senza stage directions dettagliate;
- **numero di film**: oltre 3.000 trascrizioni cinematografiche e circa 10.000 episodi televisivi;
- **accessibilità**: completamente gratuito ma disponde di un'interfaccia condizionata pesantemente dalle pubblicità;
- **organizzazione**: ricerca alfabetica per titolo.

La particolarità di questo repository è l'enorme sezione dedicata alla serie *The Simpson*. Al suo interno sono presenti diverse sottosezioni, ognuna delle quali è dedicata a: descrizione dettagliata personaggi, archivio dei dialoghi di tutti gli episodi delle 37 stagioni della serie, wallpapers e brevi audio di citazioni dei personaggi.

1.2 Portali di visualizzazione e aggregazione dati

Grazie alla sezione 1.1 abbiamo capito che sul web sono presenti tantissimi database (dati grezzi) tramite i quali gli esperti di dominio e gli appassionati possono prendere

informazioni sui film. Tuttavia, sul web sono anche disponibili dei tool cinematografici che mettono a disposizione funzionalità di alto livello attraverso le quali è possibile eseguire operazioni più complesse e articolate piuttosto che dedurre informazioni dai singoli copioni. Più nel dettaglio, esistono diversi tool che hanno lo scopo di fornire informazioni aggregate tramite la visualizzazione di tabelle e grafici. Attraverso questi elementi gli utenti possono osservare caratteristiche significative sui diversi film. Di seguito verranno presentati esempi a riguardo.

1.2.1 IMDb

IMDb (Internet Movie Database) [5] è un sito Web di proprietà di Amazon che cataloga, archivia film, attori, registi, personale di produzione, programmi televisivi, e anche videogiochi. È considerato il più grande database cinematografico online e viene utilizzato da appassionati di cinema, critici e professionisti del settore per ottenere informazioni dettagliate sulle opere audiovisive. IMDb organizza i suoi contenuti in diverse sezioni, che includono:

- **Film e Serie TV**: raccolta di tutti i dati relativi, tra cui titolo, anno di uscita, durata, genere, registi, sceneggiatori, produttori, cast principale, sinossi, budget, incassi al botteghino e altro.
- **Persone**: Attori, registi, produttori, montatori, sceneggiatori e altri membri dello staff tecnico con biografie dettagliate, filmografia e premi ricevuti.
- **Videogiochi**: Elenco di videogiochi con informazioni sugli sviluppatori, doppiatori e trama.
- **Recensioni e valutazioni**: Gli utenti possono assegnare voti da 1 a 10, che contribuiscono a formare il punteggio IMDb di un film.
- **Curiosità, errori, citazioni**: Sezione dedicata a dettagli dietro le quinte, errori di continuità e citazioni famose dei film.

Tuttavia, non tutte le informazioni sono accessibili gratuitamente: mentre le schede principali dei film e degli attori sono disponibili per tutti gli utenti, i dati più dettagliati, come le informazioni tecniche sulla produzione, i contatti dell'industria cinematografica e alcune statistiche avanzate, richiedono l'iscrizione a IMDbPro.

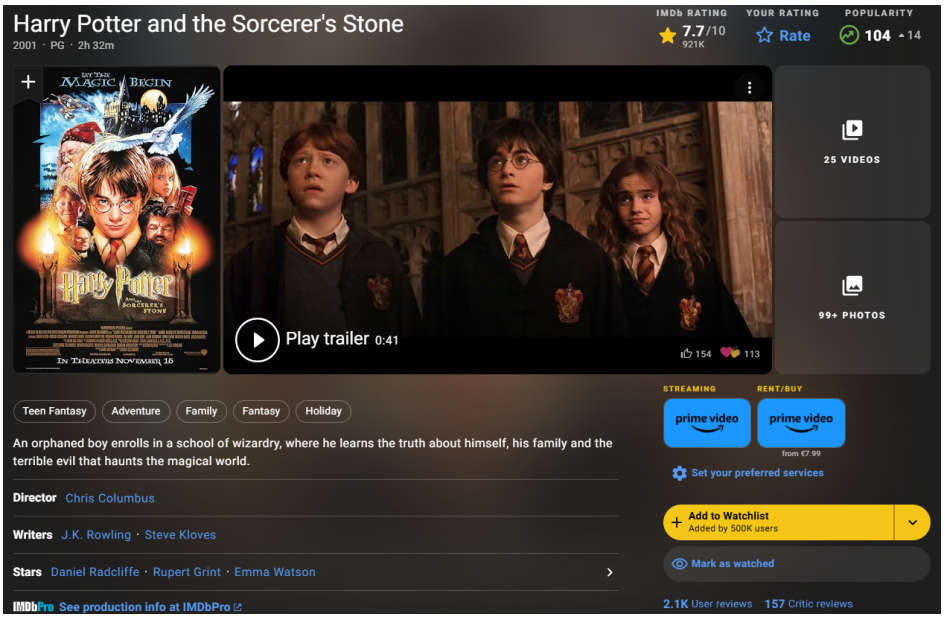


Figura 1.1: Esempio di schermata di IMDb del film Harry Potter and the Sorcerer's Stone (2001)

Oltre all'archiviazione delle informazioni, IMDb offre anche statistiche dettagliate e classifiche basate sulle valutazioni degli utenti. Tra le più note vi sono:

- **Top 250 Movies**: classifica dei 250 film con il punteggio più alto assegnato dagli utenti.
- **Bottom 100**: elenco dei 100 film con le peggiori valutazioni.
- **Trending Movies & TV Shows**: lista dei film e delle serie TV più cercate e discusse sul sito.

Tuttavia, IMDb non fornisce degli strumenti per creare delle visualizzazioni significative utilizzando questi dati. Non presenta la possibilità di creare dei grafici che mettano in relazione tra di loro determinate informazioni come l'ambientazione di vari film per ottenere quella più diffusa. Inoltre, non solo non vengono generate statistiche partendo dai dati a disposizione, ma non è possibile neanche ottenere delle statistiche inserendo in input i copioni di questi film. Questa limitazione rende difficile effettuare delle analisi avanzate delle informazioni, rendendo obbligatorio l'utilizzo di strumenti esterni.

1.2.2 Moviegalaxies

Moviegalaxies [6] è una piattaforma che genera grafi e altre visualizzazioni minori basate sulle relazioni tra personaggi di un film, mostrando come interagiscono tra loro nel corso della trama. I creatori hanno utilizzato un approccio basato sull'apparizione nella stessa

scena per costruire queste reti sociali. Viene analizzata la sceneggiatura e ogni volta che due personaggi appaiono nella stessa scena viene creato un collegamento. Per ogni nodo, Moviegalaxies raccoglie:

- **id**: Identificatore univoco del personaggio.
- **name**: Il nome del personaggio.
- **color**: Il colore del nodo all'interno del grafo. Utilizzato per distinguere personaggi e gruppi.
- **group**: Un numero che indica il gruppo di appartenenza del personaggio.
- **degree**: Il numero totale di connessioni che ogni personaggio ha con altri personaggi. Più è alto questo valore più il personaggio è centrale nella rete.
- **pagerank**: Una misura dell'importanza del nodo nel grafo. Più alto è il valore, maggiore è l'importanza del personaggio rispetto agli altri.
- **triangles**: Il numero di triangoli in cui il personaggio è coinvolto. Un triangolo è formato da tre personaggi che si conoscono reciprocamente.
- **eccentricity**: La distanza massima di un personaggio dagli altri nella rete. Più il valore è basso e più il personaggio è centrale.
- **closeness centrality**: Una misura di quanto sia vicino un nodo agli altri nodi della rete. Valori più alti indicano che il personaggio può interagire rapidamente con altri personaggi.
- **betweenness centrality**: Indica il numero di volte in cui un personaggio funge da ponte tra altri due personaggi. Un valore elevato suggerisce che il personaggio ha un ruolo chiave nella comunicazione tra altri.
- **eigenvector centrality**: Una misura dell'importanza di un nodo basata non solo sul numero di connessioni, ma anche sulla qualità di quelle connessioni. Un personaggio ben connesso a personaggi influenti avrà un'eigenvector centrality più alta.

Di seguito viene riportato un esempio di grafo sociale sul film Pirati dei Caraibi - La maledizione della prima luna.

Figura 1.2: Esempio di un grafo sociale di Moviegalaxies del film - La maledizione della prima luna

Di seguito, invece, sono riportate le informazioni relative al nodo del personaggio di Jack del grafo in figura 1.2.

```
{
  "id": "JACK",
  "name": "JACK",
  "color": "#7A40BB",
  "group": 0,
  "degree": 21,
  "pagerank": 0.15610898819154087,
  "object_id": "3912697",
  "triangles": 57,
  "eccentricity": 2.0,
  "closeness centrality": 1.1923076923076923,
  "betweenness centrality": 110.62747252747252,
  "eigenvector centrality": 0
}
```

Listing 1.1: Esempio di nodo JSON rappresentante il personaggio jack

Tuttavia, a differenza di CAST, Moviegalaxies si concentra esclusivamente sulle connessioni tra personaggi, senza includere altri tipi di analisi, rendendo questo sistema limitato

1.2.3 Tableau Public

Tableau Public [7] è uno strumento di data visualization che permette di creare dashboard interattive basate su dataset. Questo applicativo permette di creare un vasto numero di visualizzazioni utilizzando dei dataset che contengono informazioni su qualsiasi genere di dato, ad esempio film che vengono recuperati tramite IMDb. Dopo un'attenta ricerca su Tableau Public non sono state trovate visualizzazioni che sono invece presenti su CAST. Infatti, non è stata trovata una visualizzazione che mostri delle statistiche aggregate sull'ambientazione interna o esterna delle scene di film, sul luogo di sceneggiatura, sul periodo dell'anno e sul momento della giornata nel quale si svolge la scena. Inoltre non sono state trovate neanche le stesse dimensioni analizzate a livello di un singolo film.

Tra le visualizzazioni su Tableau Public inerenti al mondo cinematografico, ne è stata trovata una che mostra i film migliori raggruppati per genere e mostra anche altre informazioni su registi e attori [8]. In questa visualizzazione è possibile filtrare per data di pubblicazione, lingua originale del film, genere e voto. I dati vengono visualizzati nei tre grafici sottostanti. Il primo (figura 1.3) mostra i film migliori tenendo conto dei filtri indicati. Il secondo diagramma (figura 1.4 in basso a sinistra) mostra, selezionando un film dal primo grafico, una tabella contenente il/i regista/i e gli attori che hanno preso parte alla pellicola. Il terzo (figura 1.4 in basso a destra), mostra le persone che durante l'anno hanno avuto più ruoli come attori e/o registi.

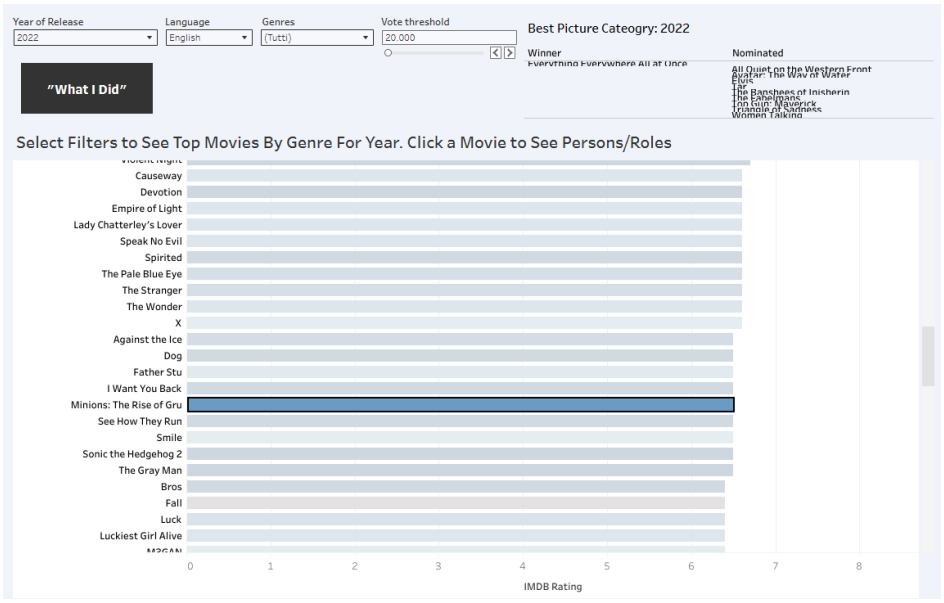


Figura 1.3: Visualizzazione relativa ai generi - prima parte

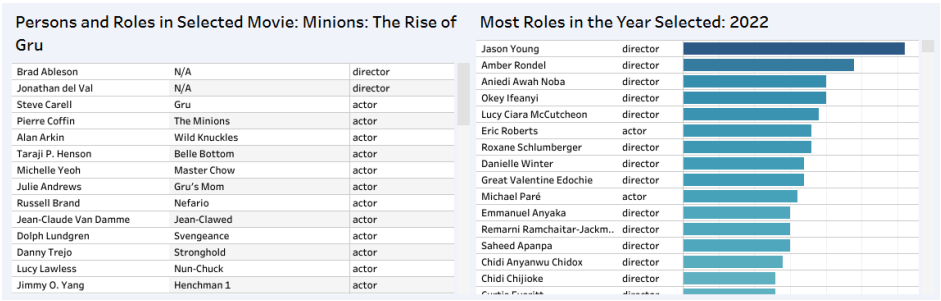


Figura 1.4: Visualizzazione relativa ai generi - seconda parte

Altri grafici molto interessanti trovati durante la ricerca sono quelli che includono una linea del tempo: molti di questi si concentrano sulla data di pubblicazione del film fornendo in aggiunta solo qualche informazione riassuntiva sugli attori. Completamente trascurata è l'analisi sulle dimensioni delle scene. Ad esempio, uno di questi grafici rappresenta i dieci migliori film di Ridley Scott, secondo il rating di IMDb, disposti su una linea del tempo in base all'anno di uscita [9].

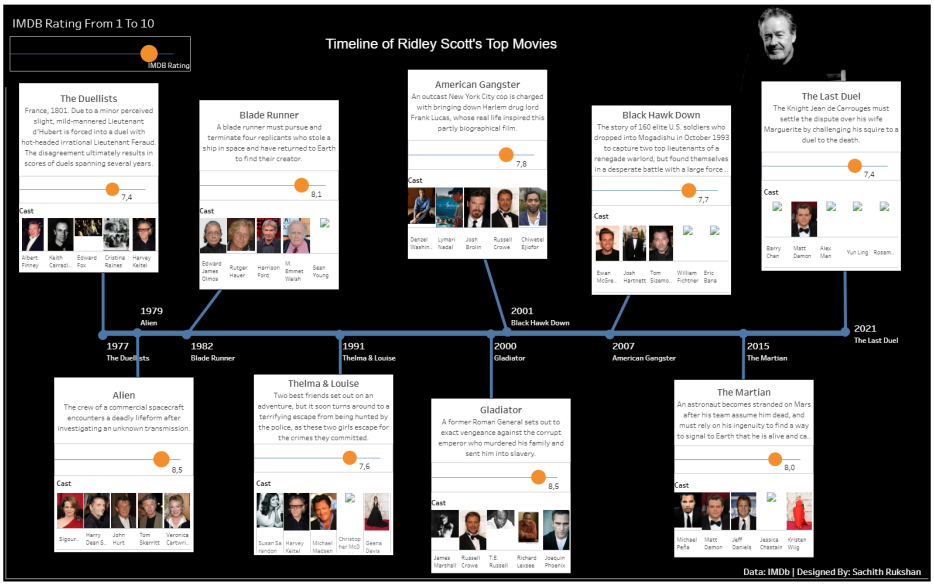


Figura 1.5: Visualizzazione che mostra la timeline dei film di Ridley Scott

L'analisi di Tableau Public evidenzia la potenza dello strumento nella creazione di visualizzazioni, offrendo un'ampia gamma di rappresentazioni grafiche. Tuttavia, non include le visualizzazioni specifiche implementate su CAST, rendendo necessaria, se si desidera, la loro creazione su Tableau Public. Inoltre, le visualizzazioni presenti non analizzano informazioni presenti sul copione del film ma si limitano a lavorare su dati facilmente reperibili come generi, data e paese di pubblicazione.

1.3 Strumenti di Analisi e Supporto alla Scrittura

La seconda categoria di tool presentati, oltre a quelli dedicati alla visualizzazione e l'aggregazione di dati, sono quei servizi che sfruttano sistemi di intelligenza artificiale per fornire strumenti di analisi e supporto alla scrittura dei copioni. Questi sistemi, a differenza dei precedenti, vengono utilizzati in fase di produzione esclusivamente dagli scrittori mentre i primi sono particolarmente utili in fase di post-produzione per poter permettere a qualsiasi utente di eseguire confronti fra film. Di seguito vengono presentati alcuni esempi di questa categoria di strumenti.

1.3.1 ScreenplayIQ

ScreenplayIQ [10] è uno strumento di analisi per sceneggiature sviluppato dall'azienda WriteDuet. Il servizio fornisce un supporto agli scrittori in fase di analisi del copione, consentendo loro di capire quali sono i punti in cui sono presenti margini di miglioramento maggiori.

Il servizio prende in input un copione, lo elabora e produce in output un report in cui vengono riportati tutti i risultati. Essi sono suddivisi in 4 categorie:

- **storia**;
- **immaginare**;
- **personaggi**;
- **valutazioni**.

Iniziando dalla categoria *storia*, il servizio mette a disposizione un brevissimo riassunto del copione consentendo così al lettore di farsi un idea sul film selezionato.

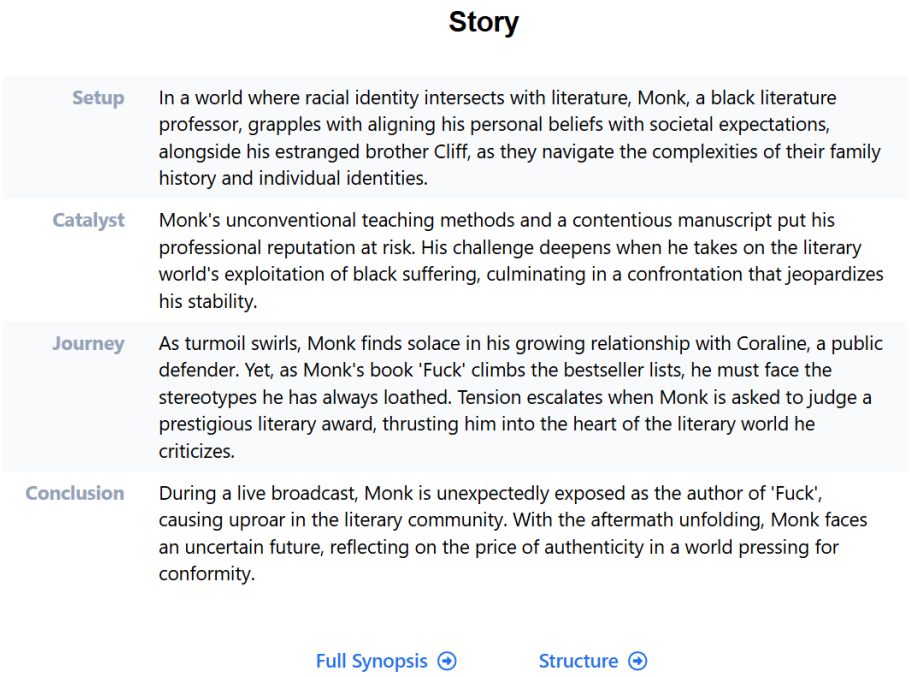


Figura 1.6: Breve riassunto copione

Successivamente, oltre al breve riassunto, è possibile visualizzare una sinossi e la struttura del copione. La sinossi presenta interamente la trama, i personaggi principali, i temi e la struttura narrativa del film. In questo modo il lettore ha la possibilità di conoscere più approfonditamente il film rispetto al breve riassunto precedente senza però venire a conoscenza di dettagli e colpi di scena del film. Invece, la struttura del copione presenta informazioni su atti, sequenze e scene.

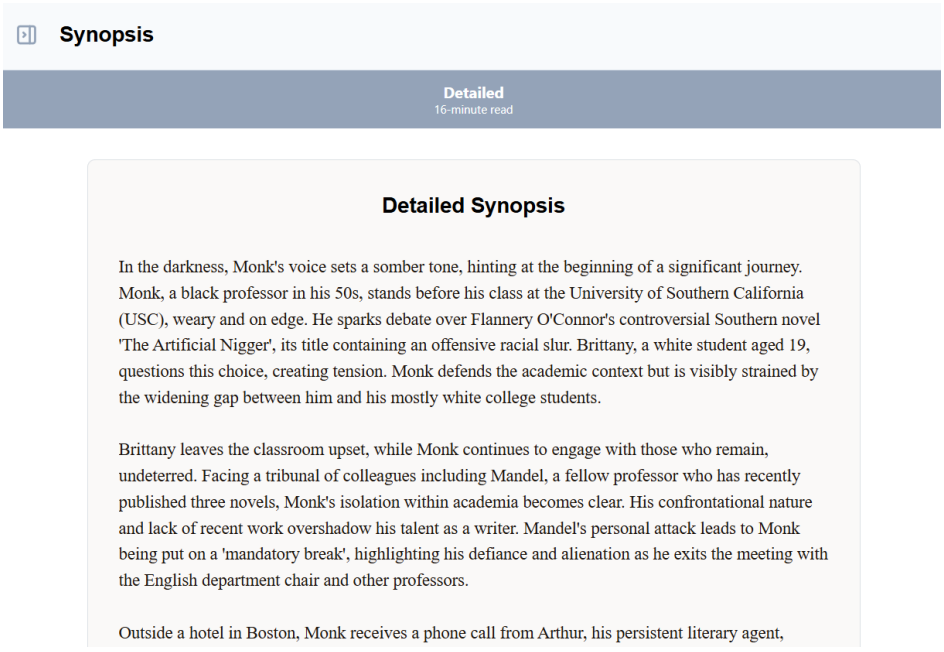


Figura 1.7: Sinossi del copione

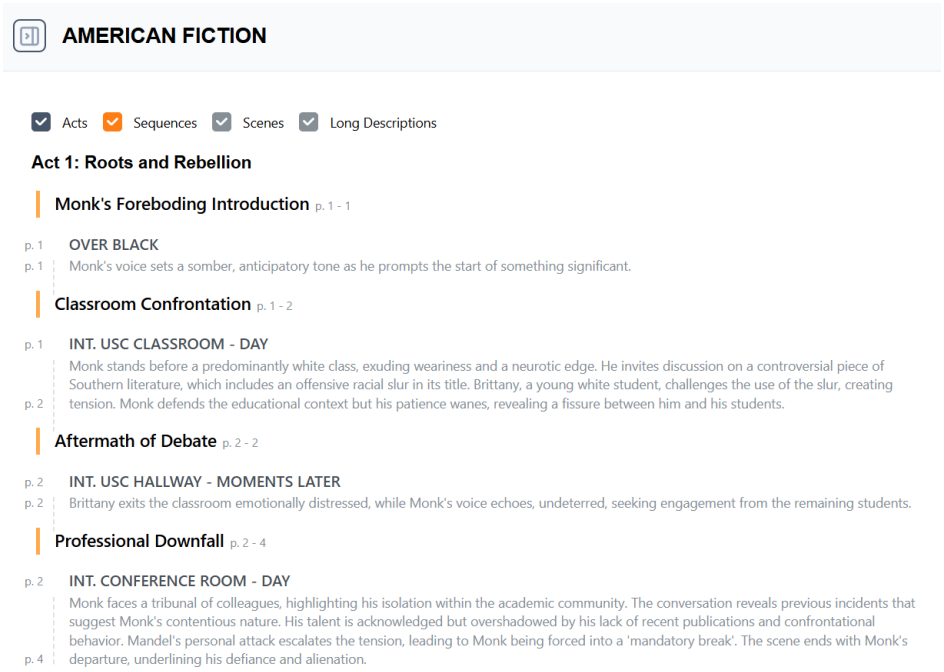


Figura 1.8: Struttura del copione

Spostando l'attenzione sulla categoria *personaggi*, ScreenplayIQ fornisce un analisi dettagliata su ogni personaggio individuato. Per ognuno vengono riportate informazioni riguardante: l'età, il sesso, il numero di scene in cui compare, una breve descrizione ed infine un grafico. Grazie a quest'ultimo sarà possibile capire subito la personalità del personaggio, dal momento che sono riportate le sue principali dimensioni caratteriali.

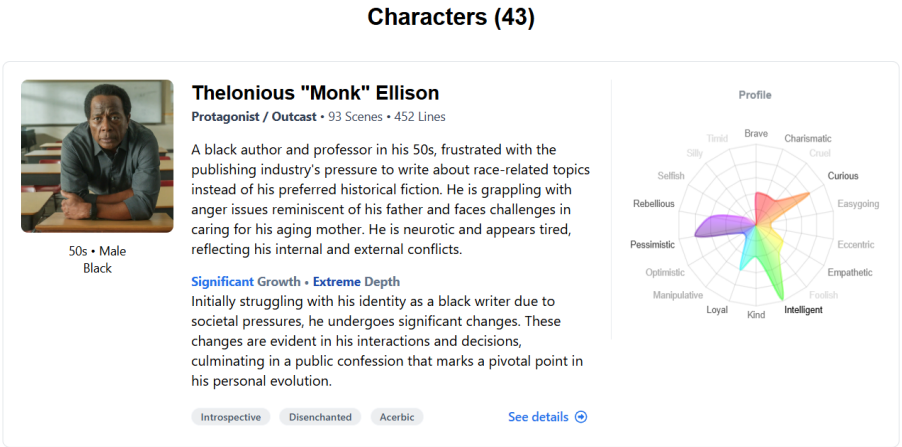


Figura 1.9: Analisi di un personaggio

Infine, osservando la categoria *valutazioni*, il servizio fornisce delle considerazioni finali riguardante l'inclinazione narrativa e i punti chiave del film.

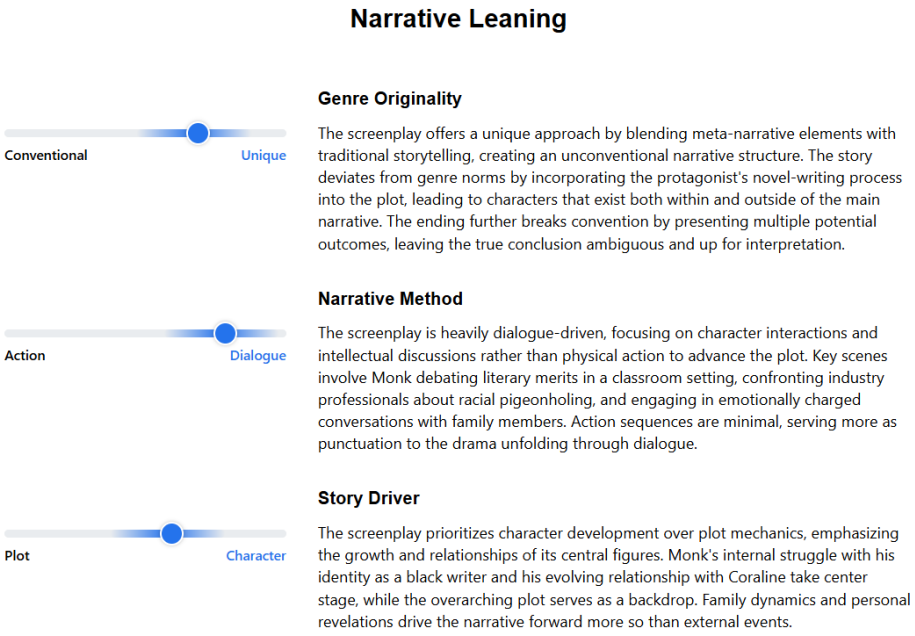


Figura 1.10: Analisi di un personaggio

Grazie a Screenplay quindi, uno scrittore potrà visualizzare come procede la fase di scrittura del suo copione oppure un semplice utente potrà farsi un idea molto approfondita su un film che è magari intenzionato a vedere. Tuttavia Screenplay non fornisce analisi aggregate su dimensioni come ambientazioni di scene, periodi del giorno e periodi dell'anno: l'analisi fornita è esclusivamente limitata a un determinato film.

1.3.2 ScriptReader.ai

Un altro strumento molto utilizzato fra gli scrittori è ScriptReader.ai [11], una piattaforma unica che utilizza l'intelligenza artificiale per analizzare le sceneggiature a livello granulare. Utilizza algoritmi di apprendimento automatico per fornire critiche dettagliate e suggerimenti di miglioramento scena per scena. La piattaforma non si limita a identificare le aree di miglioramento, ma fornisce agli sceneggiatori una comprensione approfondita di moltissimi aspetti e caratteristiche delle loro sceneggiature, aiutandoli a portare il loro lavoro a un livello superiore.

Una volta inserito in input il copione che si vuole analizzare, la piattaforma fornisce analisi scena per scena riguardante:

- **riassunto della scena**;
- **estrapolazione della scena dal copione**;
- **punti di forza e di debolezza**;
- **valutazioni**;

- **suggerimenti**.

Cominciamo ad analizzare più nel dettaglio i singoli elementi sopra citati. In primo luogo la piattaforma consente di selezionare, attraverso un menù a tendina, la scena di cui sarà visualizzata l'analisi. Le prime analisi che emergono all'occhio sono la stampa della *porzione di copione* della scena selezionata: in questo modo l'utente ha la possibilità di leggere dettagliatamente tutte le battute e le restanti informazioni riportate sul copione. Inoltre sarà disponibile un *breve riassunto*, in modo tale che l'utente possa velocemente farsi un' idea di ciò che accade il quella scena.

Successivamente vengono riportati i *punti di forza e i punti di debolezza*: feature che può rivelarsi cruciale in fase di sviluppo per lo scrittore. Egli infatti, indipendentemente dal suo livello di preparazione, avrà sempre la possibilità di scrivere copioni di qualità essendo supportato da feedback pertinenti.

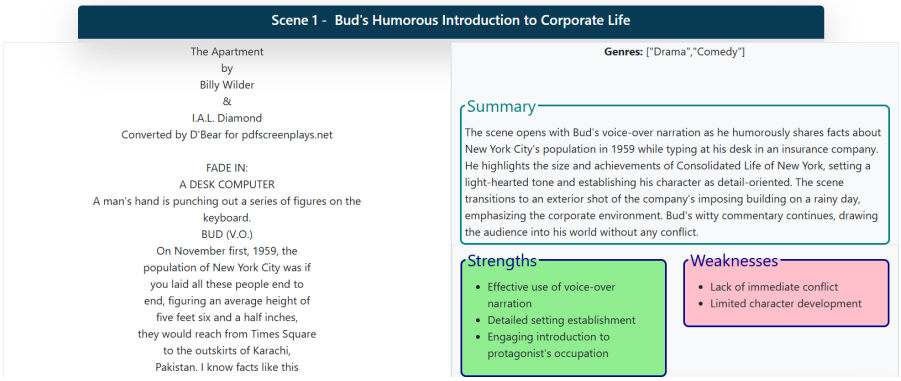


Figura 1.11: Porzione copione, riassunto e punti di forza/debolezza

Proseguendo nell'analisi, la piattaforma fornisce delle valutazioni con annessi piccoli commenti su ogni aspetto e caratteristica della scena. I principali temi sui quali vengono riportate le valutazioni sono:

- **contenuto della trama** (Figura 1.12)
  - concept, idea di base
  - trama
  - originalità
- **sviluppo dei personaggi** (Figura 1.13)
- **elementi di scena**
  - livello di conflitto, intensità del contrasto/tensione narrativa
  - high stakes, posta in gioco/Stakes elevati - L'importanza delle conseguenze per i personaggi.

  - story forward, capacità di far progredire la storia

- **audience engagement**, Capacità di mantenere l'attenzione e l'interesse dello spettatore
  - impatto emotivo
  - engagement, livello di interesse e partecipazione emotiva

- **aspetti tecnici**

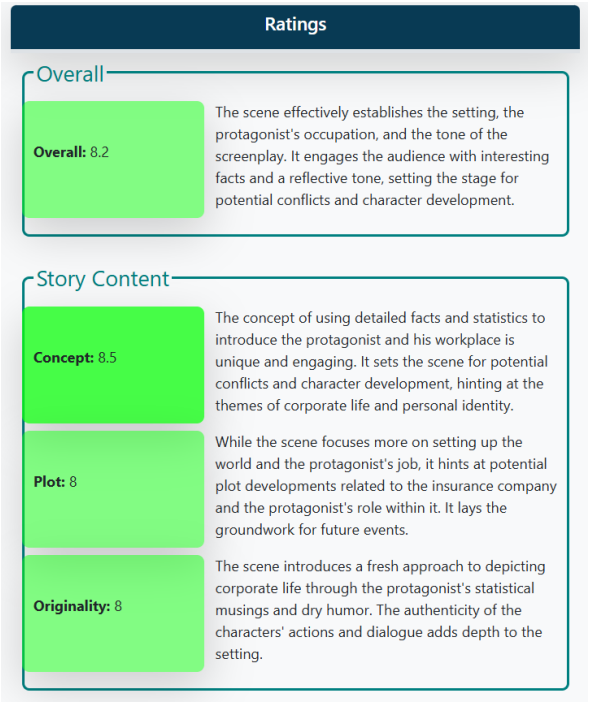


Figura 1.12: Valutazioni generali e valutazioni sulla trama

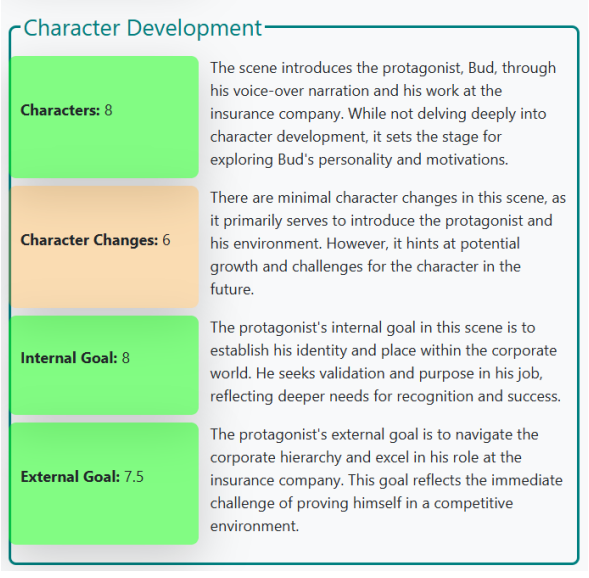


Figura 1.13: Valutazioni sviluppo personaggi

Un altro spunto interessante è lo studio delle *critiche* e i *suggerimenti*. Grazie a queste analisi generate lo scrittore, in fase di sviluppo, ha la possibilità di capire immediatamente quali sono i punti critici sui quali lavorare. Ogni scrittore, non importa quanto esperto, ha dei punti ciechi. Si tratta di schemi ricorrenti o sviste nella scrittura di cui potrebbero non essere consapevoli, ma che potrebbero ostacolare la loro narrazione. ScriptReader.ai agisce come uno specchio magico, riflettendo questi punti ciechi.

In conclusione, si può quindi affermare che ScriptReader.ai è uno strumento di supporto fondamentale alla scrittura di un copione. Non è utilizzabile invece da utenti comuni che vogliono ricevere statistiche aggregate sui loro film preferiti. Per altro, anche questo strumento fornisce analisi relative solamente a un singolo film. Il sistema infatti non è in grado di prendere in input un dataset di copioni di film e generare analisi aggregate.

1.4 Conclusioni

L'analisi dei cinque sistemi sopracitati ha evidenziato una presenza variegata di servizi riguardanti il mondo cinematografico. Una prima categoria è quella riguardante i tool di supporto alla scrittura della sceneggiatura come ad esempio ScreenplayIQ e ScriptReader. Questi tool, per lo più a pagamento, offrono analisi relative a un singolo film. Inoltre sono servizi dedicati principalmente, e in certi casi esclusivamente, a supporto degli scrittori di copioni permettendogli di migliorare la qualità dei loro copioni: non sono perciò utilizzabili dal bacino di utenza più tradizionale.

Altri servizi invece riguardano la visualizzazione di statistiche su dati semplici (anno di uscita, genere, relazione personaggi, ecc) provenienti da grandi database: in questa categoria ritroviamo IMBd considerato come uno degli archivi digitali più completo dell'industria cinematografica che consente agli utenti di valutare e recensire contenuti, creando un sistema di rating aggregato che influenza significativamente le decisioni di visione del pubblico. Altri sistemi, più affini a CAST, utilizzano questi database di dati come input per ottenere analisi grafiche: ad esempio Tableau Public e Moviegalaxies. Nonostante quest'ultimi due sistemi citati si avvicino maggiormente alla logica di business di CAST, essi presentano comunque dei limiti: le analisi non presentano in alcun modo dei grafici che derivano dall'aggregazioni di dati presi in input da un dataset di copioni. Le analisi riportate inoltre non riguardano le dimensioni delle scene come identificazione dell'ambientazione interna o esterna, deduzione del luogo in cui è ambientata la scena, comprensione del periodo del giorno e della stagione in corso.

In conclusione questa ricerca ha evidenziato come ci sia la necessità di implementare un'applicazione che sia in grado di estrarre autonomamente informazioni specifiche e dettagliate sulla sceneggiatura direttamente da un dataset di copioni, per poi elaborarle

e analizzarle in modo tale da fornire statistiche sia a livello di singoli film che analisi aggregate.

Capitolo 2

Architettura e funzionalità del sistema

Come già espresso nell'Introduzione, CAST è un' applicazione web che è nata dall'esigenza di implementare un sistema modulare generico di analisi che consentisse agli esperti di dominio di eseguire studi personalizzati su specifiche dimensioni cinematografiche. Come riportato nel capitolo 1, lo stato dell'arte non presenta un applicativo che nasce con questo particolare scopo. In questo capitolo andremo ad osservare gli obiettivi, il contesto nel quale si inserisce CAST e soprattutto verrà presentata una spiegazione di utilizzo correto per ogni grafico della dashboard.

2.1 Contesto e obiettivi di CAST

L'idea di implementare CAST nasce prendendo spunto da un progetto di tesi antecedente al mio, nel quale si approfondivano tematiche affini alle relazioni tra registri, personaggi e trame di diversi film. Una delle volontà era quella di approfondire gli aspetti cinematografici che non erano stati trattati in quel sistema. L'obiettivo principale di questa tesi è la **realizzazione di un sistema modulare di analisi generico** adatto al mondo cinematografico che possa essere usato da diversi esperti di dominio. L'aspetto interessante di CAST è la sua modularità: data la grande eterogeneità di analisi possibili che ogni esperto di dominio potrebbe voler fare, con CAST ci si pone come obiettivo quello di supportarne il maggior numero possibile. Allo stato attuale, CAST fornisce delle dashboard di dati riferiti all'ambientazione e alla temporalità di un insieme di scene proveniente da un dataset di film iniziale; nulla però vieterà in un secondo momento di estendere queste funzionalità qual ora un esperto di dominio lo ritenesse necessario.

A differenza dei portali di visualizzazione e aggregazione di dati visti nel capitolo 1, che basavano le loro visualizzazioni su dati che ottenevano esternamente già strutturati, CAST si "costruisce" internamente i propri dati. In altre parole CAST implementa una pipeline completa che gli permette di ottenere i risultati partendo direttamente dai copioni integrali dei film prestabiliti. Quest'ultimi sono stati ricavati proprio dai

repository pubblici dei quali ho parlato nella sezione 1.1 del capitolo 1. Nella sezione seguente viene spiegata più nel dettaglio la pipeline completa.

2.2 Architettura e pipelining

L'implementazione CAST è suddivisa principalmente in quattro fasi fondamentali:

- **estrazione dei copioni**: estrazione del testo proveniente da sorgenti di dati eterogenee (PDF e pagina html);
- **conversione dei copioni**: conversione da formato testo ad un formato dedicato;
- **analisi dei copioni**: svolgimento di analisi statistiche su sceneggiatura dei copioni;
- **visualizzazione delle statistiche**: visualizzazione dei file json generati in fase di analisi all'interno di una pagina web.

L'architettura generale del sistema può essere rappresentata tramite il seguente diagramma:

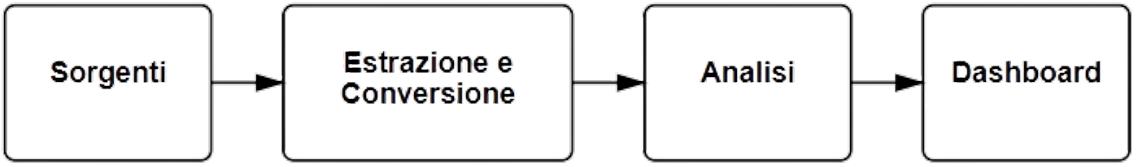


Figura 2.1: Pipeline del progetto

Ora che è stata descritta nel dettaglio l'architettura generale del sistema, vediamo più nel dettaglio i possibili utilizzi di CAST.

2.3 Home dell'applicazione

Una volta aperto sul browser CAST verrà mostrata immediatamente la homepage dell'applicazione, la quale fornisce tutte le statistiche visualizzate attraverso dei grafici.

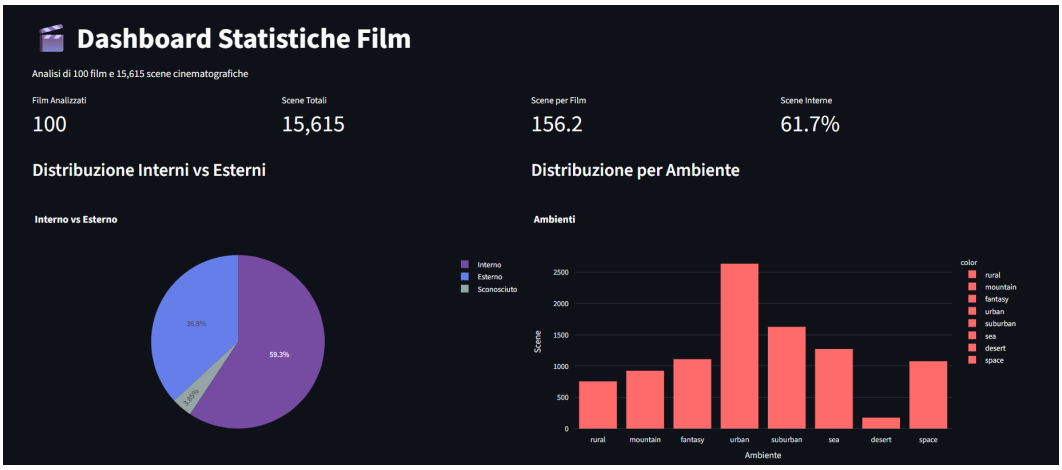


Figura 2.2: Esempio di homepage di CAST

Per aiutare alla comprensione di questa pagina è comodo considerare la pagina suddivisa in 5 porzioni:

1. **Dati generali**;
2. **Analisi grafica aggregata**;
3. **Confronto fra film**;
4. **Analisi grafica per film**;
5. **Insights e confronto con la media**.

Di seguito verranno riportate le spiegazioni di ogni singola sezione.

2.4 Dati generali

In questa prima sezione è possibile visualizzare alcune informazioni generali dell'analisi: in particolare il numero di film analizzati, il numero di scene, la media di scene per film e la percentuale di scene svolte in un luogo al chiuso. Questi valori numerici consentono subito di far capire all'utente la mole di dati che si è presa in esame per poi fare già un primo riscontro sui dati ottenuti.

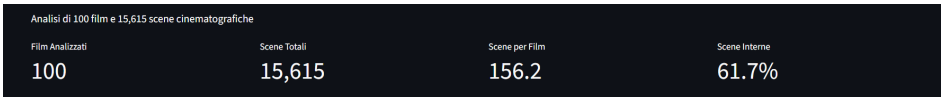


Figura 2.3: Sezione Dati generali della homepage

2.5 Analisi grafica aggregata

La seconda sezione è dedicata alla visualizzazione dei risultati dell'analisi grafica aggregata. In questa sezione CAST svolge delle analisi aggregate a livello di scene indipendentemente dal film di cui fanno parte. In particolare le analisi svolte sono relative a 4 dimensioni di una scena:

- **ambientazione interna/esterna**;
- **luogo**: ambiente specifico in cui si svolge la vicenda;
- **periodo del giorno**;
- **stagione di ambientazione**.

Per ognuna delle dimensioni citate sono stati inseriti diversi grafici. A partire da questo momento verranno spiegati più nel dettaglio i diagrammi presenti, in modo tale che chiunque voglia leggerli sia in grado di comprenderli a pieno e dedurre considerazioni finali coerenti.

2.5.1 Grafico a torta - ambientazione interna/esterna

Il primo diagramma riportato è un grafico a torta che mostra le percentuali delle scene etichettate come *interne* o *esterne*. Inoltre il grafico riporta anche la percentuale di scene che CAST non è riuscito ad analizzare, etichettandole come *sconosciuto*. Questo grafico è molto comodo perchè attraverso di esso si può subito capire le tendenze statistiche degli sceneggiatori, se prediligono creare scene ambientate in luogo al chiuso o in un luogo all'aperto.

Questa analisi risulta particolarmente preziosa per produttori e sceneggiatori che devono valutare i costi di produzione, poiché le scene esterne comportano generalmente maggiori spese logistiche e dipendenza dalle condizioni meteorologiche. Inoltre, la distribuzione INT/EXT può essere utilizzata per identificare pattern stilistici caratteristici di specifici generi cinematografici o registi.

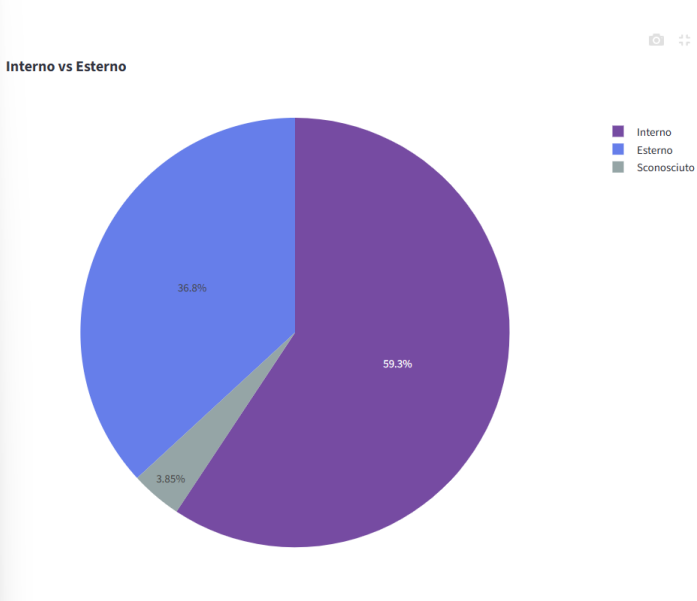


Figura 2.4: Esempio di grafico a torta per scene interne/esterne

2.5.2 Grafico a torta - Periodo del giorno

Il secondo diagramma riportato è anch'esso un grafico a torta che, questa volta, etichetta le scene in base alla fascia oraria in cui sono ambientate. CAST è in grado di interpretare quattro fasce della giornata:

- **Giorno**
- **Mattina**
- **Notte**
- **Sera**

L'analisi dei periodi giornalieri può rivelare tendenze narrative interessanti, come la predilezione per scene notturne nei thriller o scene diurne nelle commedie romantiche. Questi dati possono inoltre supportare le decisioni di pre-produzione relative alla pianificazione delle riprese e alla gestione dell'illuminazione sul set.

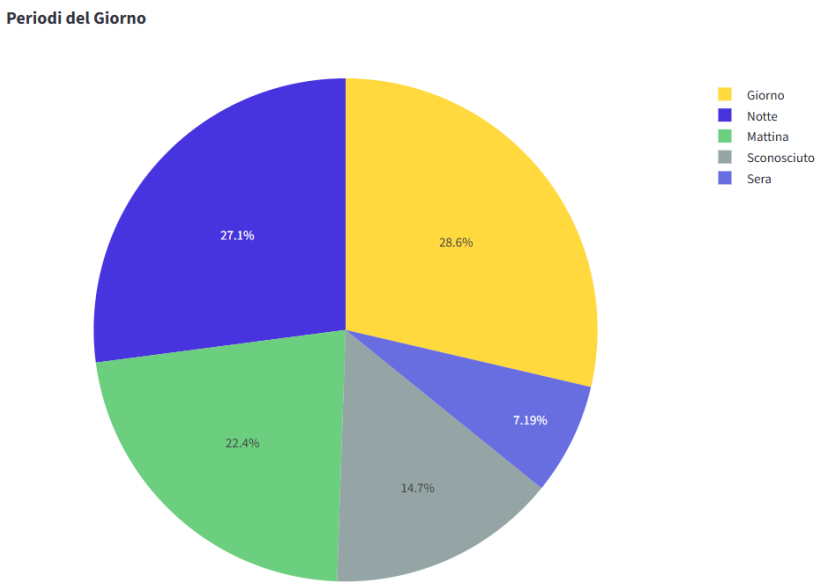


Figura 2.5: Esempio di grafico a torta per periodo del giorno

2.5.3 Grafico a torta - stagione di ambientazione

Il terzo diagramma riportato è ancora una volta un grafico a torta. In questo caso esso etichetta le scene in base alla stagione in cui sono ambientate, catturando ognuna delle quattro stagioni possibili. Come si può notare dalla figura 2.6, anche in questo caso CAST etichetta le scene come *sconosciute* qualora non riesca ad individuare una stagione plausibile.

La distribuzione stagionale offre spunti preziosi per comprendere le strategie narrative e la coerenza temporale dei copioni, evidenziando eventuali preferenze per ambientazioni natalizie o estive che potrebbero influenzare il target di pubblico. Queste informazioni risultano utili anche per la pianificazione delle riprese e la scelta delle location, ottimizzando i tempi di produzione in base alle condizioni climatiche più favorevoli.

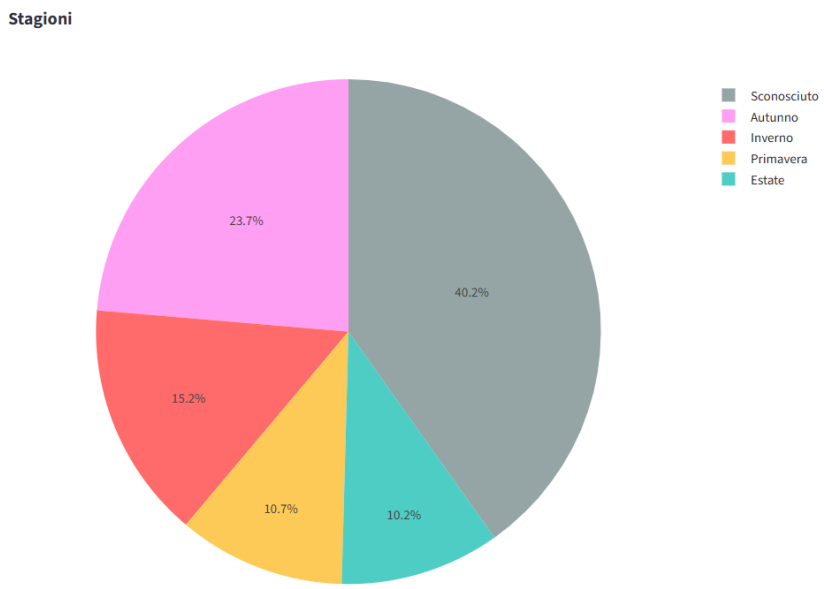


Figura 2.6: Esempio di grafico a torta per stagione

2.5.4 Istogramma - luogo di ambientazione

L'ultimo grafico dell'analisi aggregata per scene è un istogramma che sull'asse x possiede le diverse categorie di luoghi possibili mentre sull'asse y il numero di occorrenze catturate per ognuna. In sostanza, come si può osservare la dalla figura 2.7 sottostante, ogni colonna rappresenta un luogo differente. Di seguito vengono riportati i possibili luoghi catturabili:

- **Rurale**
- **Urbano**
- **Montano**
- **Sub-urbano**
- **Mare**
- **Fantastico**
- **Deserto**
- **Spazio**

In questo caso, a differenza dei grafici in precedenza, è stata omessa la colonna che conteggia le scene etichettate come *sconosciute*.

Grazie a quest'analisi potrebbe essere interessante capire quali sono i setting più popolari del cinema, supportando decisioni strategiche sulla scelta di ambientazioni

future in base alle tendenze passate del settore. Un altro aspetto che l'analisi dei luoghi offre potrebbe essere la comprensione della fattibilità economica della scelta di determinate location piuttosto che altre, considerando l'aspetto economico come fattore fondamentale nella scelta della location.

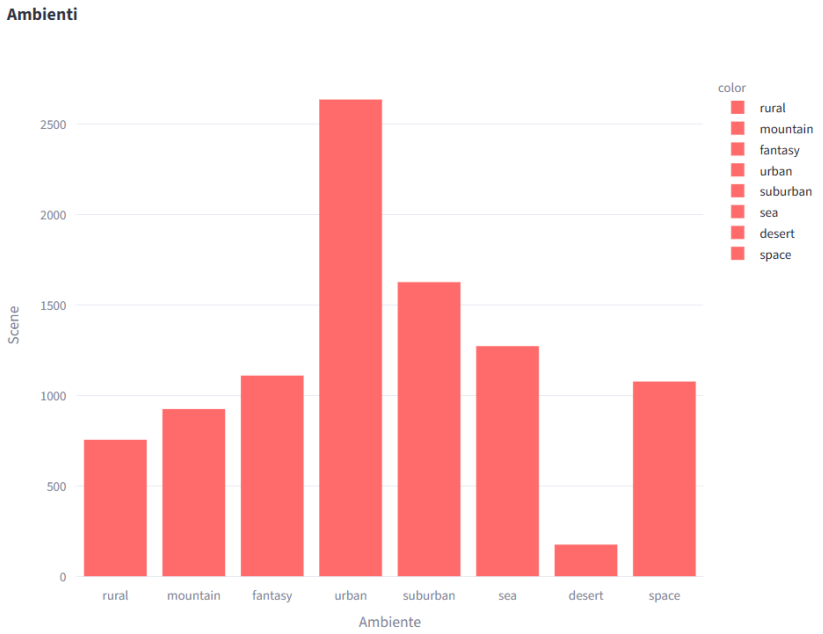


Figura 2.7: Esempio di istogramma dei luoghi

2.6 Confronto fra film

La sezione di homepage *confronto fra film* consente all'utente di svolgere analisi di comparazione tra i diversi film riguardanti le dimensioni di interno/esterno (figura 2.8), periodo del giorno (figura 2.9) e stagione (figura 2.10). Attraverso questa sezione è facilmente possibile fare paragoni e confronti tra particolari film che ci interessa analizzare.

Ogni istogramma di questa sezione è composto da 100 colonne, ognuna delle quali corrisponde a un film differente. Ogni colonna a sua volta è composta da diversi segmenti, ognuno dei quali rappresenta un sotto insieme di scene etichettate in una determinata categoria della dimensione genitore. Inoltre, eseguendo un hover con il mouse su ogni colonna sarà possibile identificare: nome del film, nome della categoria su cui si è fermato il mouse e il relativo numero di scene. Per un'esperienza di utilizzo ancora più gradevole, se si vuole avere una visualizzazione più ingrandita dei grafici sarà sufficiente utilizzare gli appositi bottoni che saranno visibili una volta che ci si avvicina il mouse alla zona della legenda. Di seguito verranno riportati i grafici (esclusivamente istogrammi) che fanno parte di questa sezione.

2.6.1 Istogramma - interno/esterno

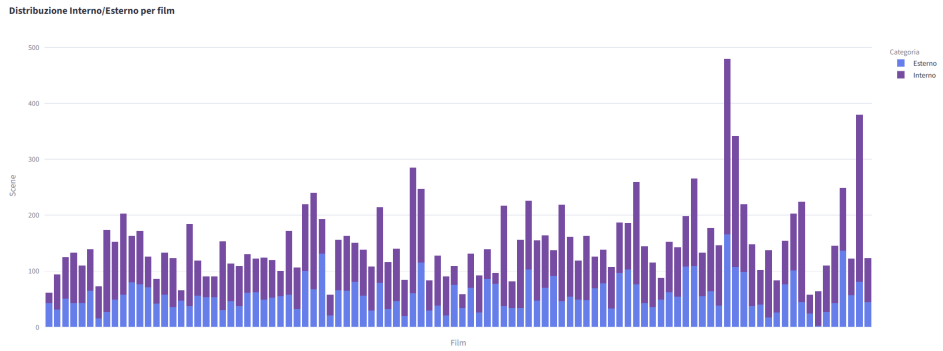


Figura 2.8: Esempio di istogramma interno/esterno

2.6.2 Istogramma - periodo del giorno

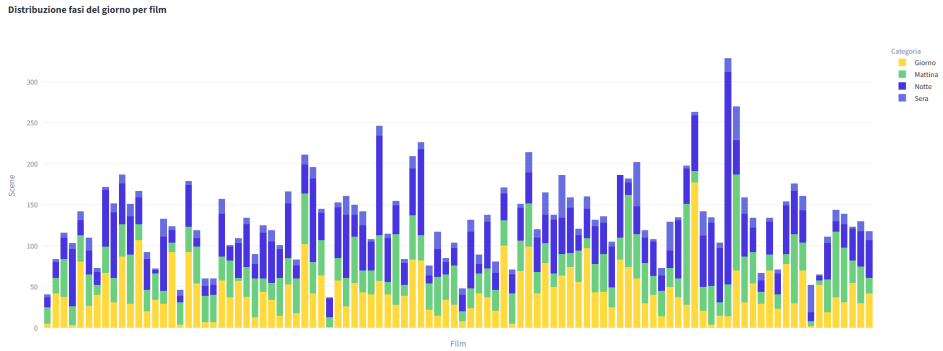


Figura 2.9: Esempio di istogramma interno/esterno

2.6.3 Istogramma - stagioni

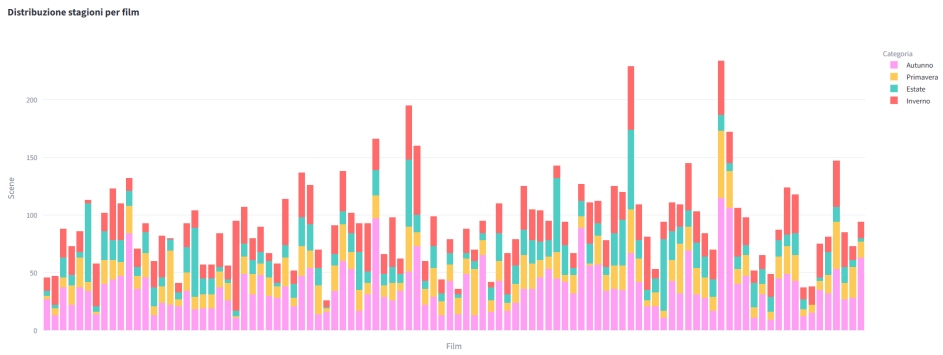


Figura 2.10: Esempio di istogramma interno/esterno

2.7 Analisi grafica per film

Continuando a scorrere verso il basso la *homepage*, la penultima sezione presente è quella dedicata all'*analisi grafica per film*. In questa sezione vengono riportati le stesse

tipologie di grafici presentati nel paragrafo 2.5, nel quale si trattava l'analisi aggregata per scene.

In questo caso, invece, l'analisi viene fatta a livello di film: CAST selezionerà solo le scene di quel copione ed eseguirà le stesse analisi svolte in precedenza. Il film sotto esame viene selezionato e cambiato in un qualsiasi momento tramite il *dropdown* presente all'inizio della sezione. Non appena si cambia il film selezionato al suo interno, i grafici si aggiorneranno automaticamente senza necessità di re-load del browser.

In questa sezione lo studio e le considerazioni fatte su gli stessi grafici nel paragrafo 2.5 possono assumere riflessioni differenti: se in precedenza potevamo osservare solamente tendenze statistiche in generale, ora è possibile invece analizzare più dettagliatamente le stesse dimensioni potendo così andare a fare dei confronti molto più ravvicinati su determinati film che interessano l'utente.

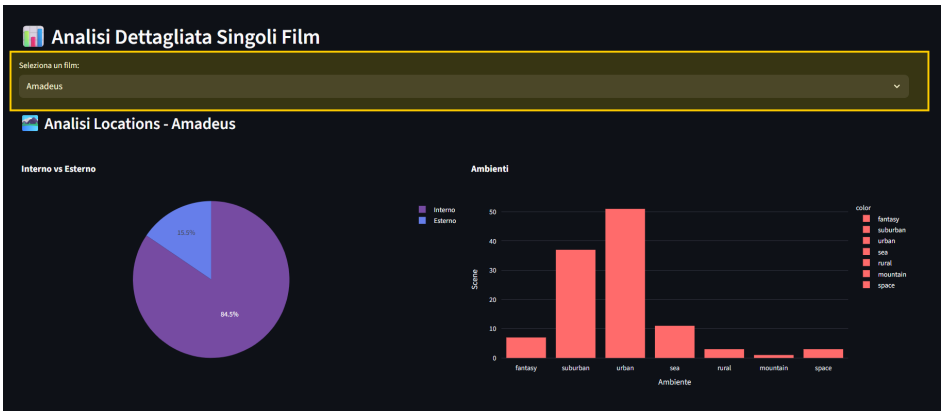


Figura 2.11: Visualizzazione della sezione analisi grafica per film

2.8 Insights

L'ultima sezione della *homepage* è quella relativa agli insights. In questa fase CAST trasforma i dati numerici raccolti in informazioni qualitative interpretabili e utilizzabili per ulteriori analisi.

2.8.1 Insights per singolo film

Per ogni film analizzato CAST fornisce una schermata di riassunto mostrando tre metriche:

- **ambiente dominante**: identifica la tipologia di location più frequente nel film;
- **periodo dominante**: evidenzia la fascia oraria più utilizzata;
- **percentuale delle scene interne**: calcola il rapporto tra scene interne ed esterne.

Gli insights possono essere uno strumento di analisi molto utile: partendo da essi è possibile dedurre pattern stilistici, tendenze di genere e caratteristiche autoriali creando così un ottimo punto di partenza per studi più approfonditi.

2.8.2 Confronto con la media generale

Infine CAST fornisce un piccolo confronto su alcuni parametri del film rispetto alla media generale. In particolare viene osservato:

- **percentuale di scene interne**: quantifica lo scostamento percentuale dalla media generale delle scene interne;
- **numero di scene**: confronta la lunghezza del copione con la media.

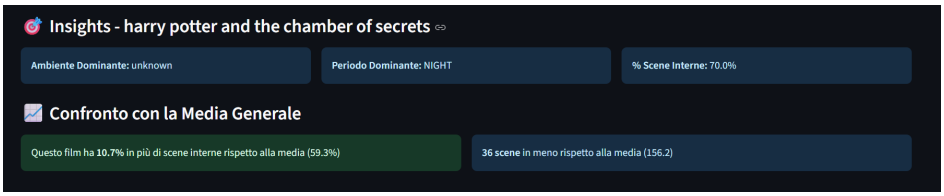


Figura 2.12: Esempio di insights e confronto con la media del film del film Harry Potter e la camera dei segreti

2.9 Dettagli implementativi

Di seguito verrà spiegata brevemente la figura 2.13, che descrive la struttura del progetto. I file e le cartelle saranno suddivisi in sotto categorie rispetto alla loro funzione all'interno della pipeline dell'applicazione (figura 2.1):

- **estrazione copioni**
  - `input/pdf_scripts/`: sub-directory contenenti tutti i file pdf, ognuno dei quali corrisponde a un copione di un film differente;
  - `sites.txt`: file di testo contenente gli url delle pagine html contenenti i copioni;
  - `extract_txt.py`: script python che prende in input le sorgenti eterogenee di copioni e le converte in file di testo;
- **conversione copioni**
  - `txt_scripts/`: sub-directory contenenti tutti i copioni salvati in formato testo (formato intermedio);

  - `txt2tei.py`: script python che prende in input i copioni in formato testo e li converte nel formato finale TEI-XML;
  - `utils.py`: script python con funzioni di utilità a supporto del convertitore;

- **analisi copioni**
  - `tei_scripts/`: sub-directory contenenti tutti i copioni salvati in formato TEI-XML;
  - `TEIAnalyzer.py`: script python che analizza i copioni TEI-XML ottenendo statistiche;

- **visualizzazione dei risultati**
  - `analysis/screenplay_analysis.json`: file json con analisi per film;
  - `analysis/screenplay_analysis_macro_stats.json`: file json contente le analisi aggregate delle scene indipendentemente dal film d'appartenenza;
  - `dashboard.py`: script python che genera la pagina web per la visualizzazione delle statistiche.

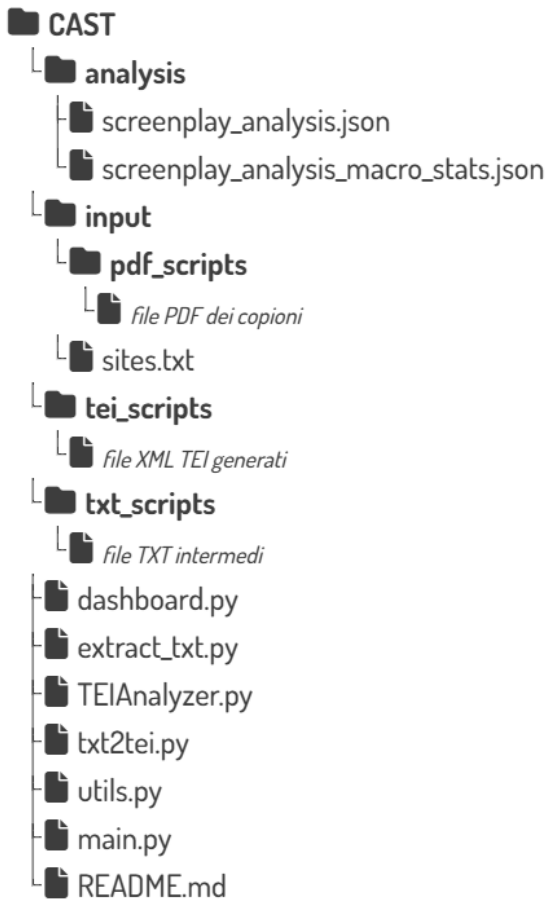


Figura 2.13: Struttura del progetto CAST

Capitolo 3

Estrazione e conversione dei copioni

CAST è un tool di estrazione e analisi di informazioni testuali provenienti da copioni cinematografici. Il sistema è progettato come una pipeline modulare composta da quattro fasi distinte, ognuna delle quali si occupa di una funzionalità ben precisa che porta poi al risultato finale. Grazie a questo approccio, sarà possibile in un futuro modificare ed estendere funzionalità garantendo quindi ottima manuntenibilità. La pipeline è implementata da una serie di script Python, ognuno dei quali svolge una determinata funzionalità rispettando al meglio i pattern suggeriti dalla letteratura. L'implementazione attuale prevede la presenza esclusiva del frontend, infatti tutti i dati necessari per poter essere utilizzati devono trovarsi localmente al sistema nel quale viene eseguito l'applicativo.

In questo capitolo entreremo nel dettaglio dell'implementazione delle funzionalità di estrazione e conversione dei copioni, mentre invece nel capitolo 4 il focus passerà sull'analisi.

3.1 Sorgenti

La peculiarità di CAST sta nel fatto che sia in grado di accettare input eterogonei. Questa caratteristica permetterà, ipoteticamente, un utilizzo per un bacino di utenza maggiore. L'implementazione di CAST prevede il supporto per due tipologie di input:

- **file pdf**: tutti i copioni utilizzati sono stati scaricati dal sito web Script Slug e salvati all'interno della cartella `input/pdf_scripts`;
- **pagina html**: in questo caso durante lo svolgimento del progetto sono stati utilizzati due siti web che offrivano i copioni stampati direttamente nella pagina;
  - **IMSDb**
  - **Springfield! Springfield!**

Per poter permettere a CAST di analizzare tutti gli url delle pagine web contenenti, essi devono essere copiati all'interno del file `sites.txt` inserendone ognuno su una riga differente.

Ai fini dell'implementazione è importante sottolineare che le analisi sono state svolte esclusivamente sui copioni in formato pdf e sui copioni provenienti dal sito web IMSDb. *Springfield! Springfield!* non è stato utilizzato in quanto la qualità, in termini di informazioni riportati sui copioni, era insufficiente ai fini degli obiettivi dell'analisi posti in fase di progettazione.

3.2 Estrazione

L'implementazione della fase di *estrazione* viene svolta dallo script `extract_txt.py`. L'esecuzione inizia dalla funzione `main()`, la quale gestisce tutta il corebusiness dello script.

3.2.1 Estrazione da pagina html

La prima attività svolta è l'estrazione del testo del copione. Inizialmente, il main estrae dall'url il nome del copione che verrà poi utilizzato come nome del file di testo. Successivamente viene richiamata la funzione `estrai_da_imbsdb`: essa utilizza la libreria **BeatifulSoap4** [12] , la quale permette in maniera semplice e veloce di estrarre del testo da un tag html impostato da noi che si trova all'interno della pagina web sotto osservazione. Come riportato nel codice 3.1, viene indicato il tag `<pre>`: esso di norma viene utilizzato nelle pagine web come contenitore per inserire tutto il testo del copione. Di seguito viene riportato il codice di questa funzione.

```
1  def estrai_da_imsdb(url, output_path):
2      print(f"[INFO] Scaricamento da IMSDB: {url}")
3      response = requests.get(url)
4      soup = BeautifulSoup(response.text, 'html.parser')
5
6      pre_tag = soup.find('pre')
7      if not pre_tag:
8          raise ValueError("Tag <pre> non trovato nella pagina.
           ")
9
10     testo = pre_tag.get_text(separator="\n", strip=True)
11
12     with open(output_path, 'w', encoding='utf-8') as f:
13         f.write(testo)
```

Listing 3.1: Funzione di estrazione da IMSDB

3.2.2 Estrazione da pdf

Proseguendo la sua esecuzione, la funzione `main()` esegue l'estrazione del testo dei copioni dalle sorgenti pdf. In questo caso l'estrazione avviene grazie alla libreria **pdfminer.six** [13]: fork moderno usato nei progetti Python 3.x che ha sostituito la libreria PDFMiner. Dal momento che questa libreria non estrae semplicemente il testo in ordine, ma cerca di sostituire la struttura logica della pagina (righe, paragrafi, colonne), è stato necessario l'utilizzo della classe `LAParams` (Layout Analysis Parameters). Questa classe, appartenente sempre alla libreria scelta, configura dei parametri necessari per analizzare al meglio il layout del file pdf.

Una volta definiti i parametri, si può eseguire l'estrazione vera e propria tramite la funzione `extract_text_to_fp()`.

Infine, come per l'altra estrazione, viene salvato il testo estratto all'interno di un file di testo.

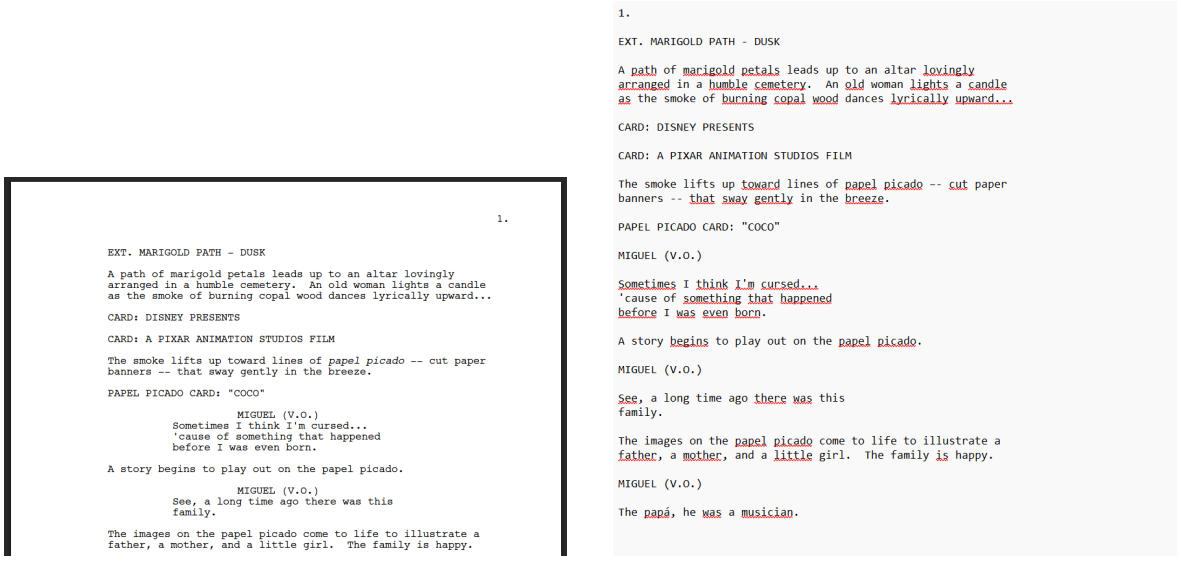
```
def estrai_pdf(percorso_pdf, output_path):
    print(f"[INFO] Estrazione testo da PDF: {percorso_pdf}")

    # Parametri di parsing pdfminer per mantenere una buona
       struttura
    laparams = LAParams(
        line_margin=0.1,
        char_margin=2.0,
        word_margin=0.1,
        boxes_flow=None
    )

    # Legge PDF ed estrae testo
    output_string = io.StringIO()
    with open(percorso_pdf, 'rb') as f:
        extract_text_to_fp(
            f,
            output_string,
            laparams=laparams,
            output_type='text',
            codec='utf-8'
        )

    testo = output_string.getvalue()

    # Salva il testo estratto in .txt
    with open(output_path, 'w', encoding='utf-8') as f:
```

```
27            f.write(testo)
```

Listing 3.2: Funzione di estrazione da pdf

3.2.3 Risultato in output

Al termine dell'esecuzione, lo script `extract_txt.py` memorizza ogni file testuale relativo a un copione, contenenti il flusso di testo estratto, all'interno della cartella `txt_scripts`. Di seguito la figura 3.1 mostra la comparazione tra il file di input (PDF) e l'output finale (TXT).



(a) Input: Copione PDF (b) Output: File TXT estratto

Figura 3.1: Confronto tra formato di input e output dell'estrazione

Come si puo osservare dal confronto in figura 3.1, il risultato ottenuto è grezzo. Lo script infatti si occupa semplicemente di estrapolare il testo, senza interessarsi al significato. Per questo motivo la fase successiva di conversione in TEI-XML sarà molto importante non solo per convertire il testo nel formato di output, ma anche per filtrare solo gli elementi che sono interessanti ai fini dell'analisi.

3.3 Conversione in TEI-XML

La fase di *conversione in TEI-XML* dei copioni è una delle fasi più importanti dell'intera implementazione: si crea il dataset di copioni che verranno poi utilizzati per creare le analisi statistiche. Prima di scendere nei dettagli implementativi della soluzione è bene capire cos'è il linguaggio TEI.

3.3.1 Linguaggio TEI-XML

La Text Encoding Initiative (TEI) [14] è un consorzio di istituzioni internazionali, di ambito linguistico e letterario, che ha sviluppato uno standard per la rappresentazione dei testi in formato digitale. Lo scopo della TEI è quello di sviluppare, mantenere una serie di linee guida di alta qualità per la codifica di testi umanistici e sostenere il loro uso da parte di istituzioni, comunità di progetti e singoli individui.

Attraverso le cosiddette Guidelines for Electronic Text Encoding and Interchange, la TEI definisce un linguaggio di markup (in XML) per la digitalizzazione dei testi, utile in particolar modo per coloro che intendono costituire archivi e banche dati testuali. L'idea di fondo era quella di creare uno standard per digitalizzare i testi letterari (in particolare testi antichi), per poterli conservare nel tempo in maniera efficiente. Dalla codifica digitale si ottengono anche altri benefici, come la portabilità dei testi, la facilità di archiviazione e la facilità di gestione attraverso gli strumenti informatici.

Il linguaggio vede la sua nascita nel 1990 con la sua versione P1. Nel tempo sono state rilasciate numerose versioni aggiornate arrivando ad oggi con la versione P5. Con il passare del tempo il linguaggio ha avuto un utilizzo sempre maggiore, ad oggi è utilizzato principalmente per:

- Manoscritti antichi e medievali
- Opere letterarie classiche e moderne
- Documenti storici
- Copioni cinematografici
- Corrispondenze epistolari
- Testi religiosi e filosofici
- Opere teatrali e drammatiche

Quindi, CAST utilizza il linguaggio di markup TEI-XML per strutturare i copioni cinematografici in modo che possano essere analizzati computazionalmente più facilmente. Il sistema sfrutta la capacità del formato di distinguere elementi come scene, personaggi, dialoghi e didascalie in forma machine-readable.

3.3.2 Struttura del copione TEI

TEI mette a disposizioni numerosi tag che caratterizzano in maniera univoca il testo della sceneggiatura di un film. Per l'implementazione di CAST sono stati utilizzati solo alcuni di questi tag, solo quelli necessari a mappare il copione con i dati necessari da sottoporre poi alle analisi. Prima di entrare nel dettaglio del codice cerchiamo di capire come può essere strutturato il copione nel formato XML-TEI:

```xml
<?xml version="1.0" encoding="utf-8"?>
<TEI xmlns="http://www.tei-c.org/ns/1.0">
  <teiHeader>
    <fileDesc>
      <titleStmt>
        <title>Il Grande Inganno</title>
      </titleStmt>
    </fileDesc>
  </teiHeader>
  <text>
    <body>
      <div type="scene" n="1">
        <stage type="location">INT. UFFICIO DETECTIVE - NOTTE
           </stage>
        <stage>La pioggia batte contro le finestre. MARCO
           ROSSI, 40 anni,
               detective privato, e' seduto alla scrivania
                  illuminata solo
               da una lampada da tavolo.</stage>
        <sp>
          <speaker>MARCO</speaker>
          <p>Ancora nessuna traccia di Sarah. E' scomparsa
             nel nulla.</p>
        </sp>
        <stage>Il telefono squilla. Marco esita prima di
           rispondere.</stage>
        <sp>
          <speaker>MARCO</speaker>
          <p>Detective Rossi.</p>
        </sp>
        <sp>
          <speaker>VOCE MISTERIOSA</speaker>
          <p>Smetti di cercarla se tieni alla tua vita.</p>
        </sp>
        <stage>La linea cade. Marco guarda il telefono con
           preoccupazione.</stage>
      </div>

      <div type="scene" n="2">
        <stage type="location">EXT. STRADA BUIA - NOTTE</
           stage>
```

```
35          <stage>Marco esce dall'edificio e cammina verso la
               sua auto.
36                  I lampioni creano ombre inquietanti.</stage>
37          <sp>
38            <speaker>MARCO</speaker>
39            <p>Chi diavolo era?</p>
40          </sp>
41          <stage>Un'auto nera si avvicina lentamente. Marco
               accelera
42                  il passo.</stage>
43        </div>
44      </body>
45    </text>
46  </TEI>
```

Listing 3.3: Esempio (generato da IA) di copione codificato in TEI-XML

Vediamo ora più nel dettaglio tutti i vari tag utilizzati nel codice di esempio 3.3:

- `<TEI>`: elenco radice del documento;
- `<teiHeader>`: contenitore dei metadati del documento (titolo, autore, ecc);
- `<text><body>`: corpo principale del testo;
- `<div type="scene" n="X">`: contenitore di una scena, div di tipo *scene* con attributo *n* per enumerare la scena contenuta;
- `<stage type="location">`: tag contenente l'informazione del tipo di ambientazione della scena (interna o esterna);
- `<stage>`: tag contenente informazioni di *contesto e didascalie* annesse alla scena;
- `<sp>`: speech - contenitore per i dialoghi di un personaggio in una scena;
- `<speaker>`: nome del personaggio che parla;
- `<p>`: paragrafo di dialogo.

3.3.3 Implementazione del convertitore

Il cuore dell'implementazione della funzionalità di *conversione* viene svolta dallo script `txt2tei.py`: il suo compito è trasformare il testo grezzo estratto dai copioni "testuali" in documenti XML strutturati secondo lo standard TEI. Durante la conversione lo script utilizza dei metodi di utilità inseriti all'interno dello script `utils.py`. Questa suddivisione fra logica di business e metodì di utilità si è resa necessaria in fase di sviluppo per garantire una migliore manuntenibilità del codice.

Preambolo

Iniziando a scendere più nel dettaglio del codice dello script `txt2tei.py`, esso gestisce tutta la logica di conversione di un copione testuale nel metodo `converti_in_tei`. Per prima cosa la funzione legge tutto il contenuto del file di testo, salvando tutte le righe del documento all'interno di una lista. Dopo di che viene estratto il titolo del film tramite la funzione `utils.extract_title_from_filename`, definita la struttura TEI di base e inizializzate delle variabili che sono necessarie alla gestione della conversione vera e propria.

```
1  def converti_in_tei(percorso_txt):
2      with open(percorso_txt, 'r', encoding='utf-8') as f:
3          righe = [r.strip() for r in f if r.strip()]
4
5      # Estrae il titolo dal nome del file
6      filename = os.path.basename(percorso_txt)
7      titolo = utils.extract_title_from_filename(filename)
8
9      corpo_righe = righe
10
11     root = ET.Element("TEI", xmlns="http://www.tei-c.org/ns
           /1.0")
12
13     # HEADER
14     teiHeader = ET.SubElement(root, "teiHeader")
15     fileDesc = ET.SubElement(teiHeader, "fileDesc")
16     titleStmt = ET.SubElement(fileDesc, "titleStmt")
17     ET.SubElement(titleStmt, "title").text = titolo
18
19     # CORPO TESTO
20     text = ET.SubElement(root, "text")
21     body = ET.SubElement(text, "body")
22
23     # variabili utili
24     scena_corrente = None
25     scene_counter = 1
26     speaker_corrente = None
27     ultimo_sp_element = None
28     speech_in_continuazione = False
29     i = 0
30
31     # ... continua con il ciclo di parsing principale
```

Listing 3.4: Inizializzazione della funzione converti_in_tei

Da questo momento in avanti l'esecuzione entra nel vivo: viene eseguito un ciclo while che scorre tutte le righe memorizzate. Per ogni riga, CAST applica una **gerarchia di controlli a priorità decrescente**. Questa architettura garantisce che gli elementi più critici vengano identificati prima di quelli più generici, evitando false classificazioni. La logica segue questo ordine di priorità:

1. **filtri di esclusione**: Righe `CONTINUED`, numeri di pagina, intestazioni e transizioni cinematografiche vengono immediatamente scartate;
2. **rilevamento scene**: Pattern di location (`INT./EXT.`) identificano l'inizio di nuove scene e attivano la creazione di elementi `<div type="scene">`;
3. **identificazione speaker**: Nomi in maiuscolo determinano l'inizio di nuovi dialoghi e la creazione di elementi `<sp>`;
4. **classificazione residuale**: Tutto il testo rimanente viene categorizzato come didascalia (`<stage>`).

Vediamo ora nel dettaglio i controlli effettuati all'interno del ciclo `while`.

Priorità 1 - filtri di esclusione, righe CONTINUED

Il primo controllo che CAST esegue su una riga è verificare se essa indica la continuità di uno speech, all'interno di una scena, da una pagina all'altra nel file pdf. Queste tipologie di righe vengono inserite dagli scrittori per facilitare la comprensione al lettore di una pagina del copione. Qual ora quest'informazione fosse assente dal copione, si formerebbe un'ambiguità perchè il lettore non capirebbe se lo speech o la scena stiano proseguendo il loro svolgimento a discapito della loro ri-definizione nella nuova pagina.

Il metodo utilizzato è `utils.is_continued_line`, il quale utilizza espressioni regolari progressive (dalle più semplici alle più complesse) per identificare i pattern più comuni utilizzati tipicamente nei copioni formattati.

Qual ora CAST identifichi quella riga come `CONTINUED` verrebbe ignorata e si passerebbe alla riga successiva. Altrimenti se non lo è si va avanti nei controlli.

Priorità 2 - rilevamento scene, location-line

Il secondo controllo che CAST esegue su una riga è identificare se essa rappresenta una *location di scena* (come "INT. CASA - GIORNO" o "EXT. STRADA - NOTTE") utilizzando anche in questo caso un sistema di priorità crescente. Questa logica consente

di garantire un livello di precisione molto alto distinguendo vere location da descrizioni simili.

Il controllo viene effettuato dalla funzione `utils.is_location_line`, la quale verifica se la riga in esame appartiene a diversi pattern aventi priorità differenti:

- **match ad alta priorità**: pattern cinematografici standard (`INT.` , `EXT.` , ecc. . .) con varianti di pattern numerici e con punteggiatura. Il sistema predilige a scegliere queste righe come righe di localizzazione;
- **esclusioni a media priorità**: CAST non considera quelle righe che contengono indicatori narrativi (verbi di azione), descrizioni (tipicamente identificati con righe che iniziano con articoli - `THE` , `A`);
- **match condizionali a bassa priorità**:
  - **shot keywords**: il sistema cerca di individuare righe di localizzazione controllando la presenza di parole chiave tipicamente usate nei copioni inserendo criteri aggiuntivi di selezione per aumentare la precisione (righe corte, non contenere punteggiatura complessa e non avere indicatori narrativi);
  - **pattern temporali**: CAST ricerca righe che terminano ad esempio con `- DAY` e `- NIGHT` ;
  - **location specifiche**: il sistema ricerca parole chiave che indicano un luogo ben preciso come ad esempio `SETTING:` o `SCENE LOCATION:` .

Qual ora il metodo verifichi che la riga sia effettivamente una *location-line*, l'esecuzione dello script `txt2tei.py` procede a pulire la riga da eventuali spazi multipli e identificare il pattern di locazione. Inoltre, essendomi accorto in fase di sviluppo che ogni nuova scena viene inizializzata da una riga di localizzazione (identificata dal sistema come *location-line*), CAST tutte le volte che incontra questa tipologia di righe **crea una nuova scena.** Per fare questo CAST utilizza il metodo `SubElement` della classe `xml.etree.ElementTree` [15]. Infine il sistema è in grado di enumerare la scena corrente sia con un contatore a runtime oppure estrapolando il numero di scena dalla riga del copione. Grazie a questo, in fase di analisi sarà possibile fare considerazioni sul numero di scene di media presente nei film. Terminata questa sezione di codice viene incrementato il numero di scene per poi passare all'analisi della riga successiva. (Codice 3.5)

Qual ora invece il metodo `utils.is_location_line` restituisca *false*, si passerebbe ai controlli successivi in ordine di priorità.

```
1 def converti_in_tei(percorso_txt):
2     # ... Preambolo
3
```

```
    while i < len(corpo_righe):
        # Priorita 1 ...

        # Priorita 2
        if utils.is_location_line(riga):
            #1) pulizia della riga: normalizzo tutti gli spazi multipli in un singolo spazio
            riga_clean = riga.strip().replace("\xa0", " ").replace("\u00A0", " ")
            riga_clean = re.sub(r'\s+', ' ', riga_clean)

            # Pattern 1: numero/alfanumerico + descrizione + numero/alfanumerico
            numero_desc_numero_match = re.match(r'^([A-Za-z]*\d+[A-Za-z]*)\s+(.+?)\s+([A-Za-z]*\d+[A-Za-z]*)$',
                                                riga_clean)

            # Pattern 2: numero + INT./EXT./I/E. + descrizione
            numero_location_match = re.match(r'^(\d+[A-Za-z]*)\s+((?:INT\.?|EXT\.?|I/E\.?)\s+.+)$',
                                             riga_clean,
                                                re.
                                                IGNORECASE
                                                )

            if numero_desc_numero_match:
                # Estrae SOLO la descrizione, ignora il numero originale
                location_description = numero_desc_numero_match.group(2).strip()

                numero_scena = str(scene_counter)

                # Crea la scena usando numerazione automatica
                scena_corrente = ET.SubElement(body, "div", type="scene")
                scena_corrente.set("n", numero_scena)

                # Aggiunge la location estratta
```

```
32                  ET.SubElement(scena_corrente, "stage", type="
                        location").text = location_description
33
34              elif numero_location_match:
35                  location_description = numero_location_match.
                        group(2).strip()
36
37                  numero_scena = str(scene_counter)
38
39                  scena_corrente = ET.SubElement(body, "div",
                        type="scene")
40                  scena_corrente.set("n", numero_scena)
41
42                  ET.SubElement(scena_corrente, "stage", type="
                        location").text = location_description
43              else:
44                  #location standard
45                  location_line = riga
46
47                  numero_scena = str(scene_counter)
48
49                  scena_corrente = ET.SubElement(body, "div",
                        type="scene")
50                  scena_corrente.set("n", numero_scena)
51
52                  ET.SubElement(scena_corrente, "stage", type="
                        location").text = location_line
```

Listing 3.5: Blocco *location-line* nel metodo `converti_in_tei`

Priorità 3 - filtri di esclusione, numeri di pagina

Il terzo controllo che CAST esegue in ordine di priorità è verificare se la riga in esame contenga il numero di pagina del file pdf originario. Per fare questo, il metodo `utils.is_page_number` restituisce *true* o *false* se rispettivamente identifica dei match con pattern cinematografici oppure no.

Qual ora il risultato restituito ci confermi la presenza di un numero di pagina, la riga viene ignorata e si passa all'iterazione succesiva del ciclo con una nuova riga; viceversa si prosegue con i controlli di priorità minore.

Priorità 4 - filtri di esclusione, numeri di scena

Il quarto controllo che CAST esegue in ordine di priorità è verificare se la riga in esame contenga il numero della scena. Per fare questo, il metodo `utils.is_scene_number` restituisce *true* o *false* se rispettivamente identifica dei match con pattern cinematografici oppure no.

Qual ora il risultato restituito ci confermi la presenza di un numero di scena, la riga viene ignorata e si passa all'iterazione succesiva del ciclo con una nuova riga; viceversa si prosegue con i controlli di priorità minore.

Priorità 5 - filtri di esclusione, *more-line*

Il quinto controllo che CAST esegue in ordine di priorità è verificare se la riga in esame sia una *more-line*. In CAST viene usato questo termine per indicare quelle righe che indicano che la battuta di uno speaker non si interrompe con la fine della pagina ma che prosegue anche in quella successiva. Questa tipologia di riga insieme alle righe CONTINUED trattate nella sezione 3.3.3 creano un pattern di scrittura del copione che ho personalmente chiamato *pattern more-continued*. Di seguito viene mostrata una figura per capire meglio il problema.

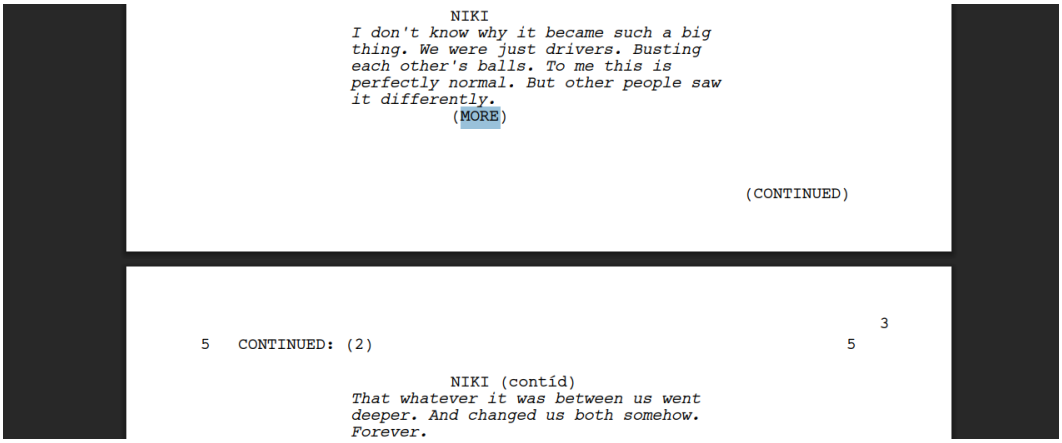


Figura 3.2: Esempio di gestione delle righe continued e more nel copione Rush

Il metodo `utils.is_more_line` restituisce *true* o *false* se rispettivamente identifica dei match con pattern cinematografici oppure no.

Qual ora il risultato restituito ci confermi la presenza di una riga che indica la continuazione dello speech, viene impostato il flag booleano `speech_in_continuazione` a *true*. In questo modo, all'iterazione successiva CAST saprà operare in maniera specializzata. A questo punto l'esecuzione va ad analizzare la riga successiva. Viceversa si prosegue con i controlli di priorità minore.

Priorità 6 - filtri di esclusione, righe di transizione

Il sesto controllo che CAST esegue in ordine di priorità è verificare se la riga in esame sia una riga di transizione.

Molto spesso gli scrittori all'interno dei loro copioni inseriscono moltissime informazioni aggiuntive rispetto alle semplici battute, unico elemento che un non addetto ai lavori potrebbe immaginare che sia presente nel copione. Per identificare righe di transizione, il metodo `utils.is_transition_line` restituisce *true* o *false* se rispettivamente identifica dei match con pattern cinematografici oppure no. Le keyword più utilizzate in questi casi sono: `"CUT TO:"`, `"FREEZE FRAME:"`, `"RESUMING:"`, `"THE END"`, `"FADE TO BLACK"`, `"FADE OUT"`, `"FADE IN:"`, ecc.

Qual ora il risultato restituito ci confermi la presenza di una riga di transizione, la riga viene ignorata e si passa all'iterazione succesiva del ciclo con una nuova riga; viceversa si prosegue con i controlli di priorità minore.

Priorità 7 - filtri di esclusione, righe in intestazione o pie di pagina

Il settimo controllo che CAST esegue in ordine di priorità è verificare se la riga in esame sia una riga di intestazione o pie di pagina del documento pdf originale. All'interno degli archivi digitali esistono copioni di diversa natura: alcuni contenenti le minime informazioni necessarie mentre altri molto più ricchi di metadati. Essi tipicamente sono inseriti nelle righe di intestazione o pie di pagina, proprio per questo motivo CAST tramite la funzione `utils.is_header_line` ne va alla ricerca con l'obiettivo di trascurarli dal risultato finale.

Il metodo sopra citato è in grado di identificare i seguenti pattern:

- **date**: date relative molto spesso al rilascio del copione;
- **titolo by autore**: tipicamente nelle intestazioni vengono inseriti il titolo del film e l'autore;
- **revisioni e draft**: il sistema individua keywords come `SHOOTING`, `FIRST`, `Rev` intuendo che nelle intestazioni si sta facendo riferimento alle versioni del copione;
- **copyright**: il sistema individua keywords affini al mondo della sicurezza e protezione dei dati;
- **nomi di case cinematografiche**: il sistema rileva i nomi delle principali case cinematografiche;
- **caratteri speciali**: il sistema rileva caratteri speciali, elementi tipici di intestazioni di copioni cinematografici.

Qual ora il risultato restituito ci confermi la presenza di una riga di intestazione o pie di pagina, la riga viene ignorata e si passa all'iterazione succesiva del ciclo con una nuova riga; viceversa si prosegue con i controlli di priorità minore.

Priorità 8 - Gestione dialoghi e speaker

L'ottavo controllo che CAST esegue in ordine di priorità rappresenta il blocco più complesso dell'intero sistema di conversione: la gestione dei dialoghi. Questa sezione si articola in quattro fasi distinte che garantiscono la corretta identificazione, raccolta e strutturazione dei discorsi dei personaggi.

Identificazione speaker La prima fase consiste nel verificare se la riga contiene la definizione dello speaker di un nuovo speech, distinguendolo da descrizioni narrative che potrebbero essere simili. L'implementazione di questa funzionalità viene svolta dal metodo `utils.is_speaker`. La logica del metodo si basa su tre fasi distinte:

1. **applicazione di filtri**: vengono applicati filtri che eseguono esclusioni drastiche per eliminare falsi positivi, in particolare
   - **punteggiatura non ammessa**: righe con virgole o punteggiatura complessa sono tipiche delle descrizioni narrative;
   - **numeri di scena isolati**;
   - **marcatori CONTINUED**: righe che contengono solo `CONTINUED` vengono scartate;
   - **righe troppo lunghe**: oltre 60 caratteri indica che probabilmente non si sta trattando in un nome di speaker. Esso si trova sempre da solo su una riga;
   - **articoli iniziali**: righe che iniziano con "THE", "A", "AN" sono descrizioni;
   - **verbi d'azione**: i verbi indicano che si sta descrivendo qualcosa, quindi non sono nomi di speaker;
2. **ricerca di pattern positivi**: vengono ricercati pattern che deducano la presenza di nomi di speaker
   - **nomi in maiuscolo**: tutti i nomi di speaker sono tipicamente inseriti in maiuscolo;
   - **nomi con continuazione**: nomi come `MARIO (CONT'D)` usato nel pattern *more-line* spiegato nella figura 3.2;
   - **titoli professionali**: nomi come `MR. ROSSI` o `DR. SMITH`;
   - **speaker numerati**: `COP #1` ...;

- **nomi descrittivi**: `BIG JOHN`, `OLD MARY` che sono pattern tipici dei copioni;
- **nomi stranieri**: gestisce prefissi come `Mc`, `O'`, `De` ...;

3. **verifica che le parole siano prevalentemente maiuscole**.

Qual ora l'esito della funzione `utils.is_speaker` sia *true*, CAST inizierà a iterare le righe successive con uno scopo più preciso: memorizzare le righe per comporre lo speech e capire quando interromperlo. In particolare per prima cosa vengono estrapolati il nome dello speaker (`utils.extract_speaker_name`) e intuito se il suo nome corrisponde a una continuazione di uno speech precedente o meno (`utils.is_continuation_speaker`). Di seguito verrà spiegato nel dettaglio il codice di questa parte.

```
if utils.is_speaker(riga) and scena_corrente is not None:
    speaker_name = utils.extract_speaker_name(riga)
    is_continuation = utils.is_continuation_speaker(riga)

    # Determina se continuare la speech precedente
    continua_speech = (
        (speech_in_continuazione and speaker_name ==
           speaker_corrente) or
        (is_continuation and speaker_name == speaker_corrente
           )
    ) and ultimo_sp_element is not None
```

Listing 3.6: Identificazione speaker e gestione continuazioni

Il sistema riconosce due scenari di continuazione:

1. **Continuazione implicita**: Il dialogo precedente terminava con `(MORE)` e il personaggio riappare nella pagina successiva.
2. **Continuazione esplicita**: Il nome del personaggio è seguito da `(CONT'D)` o `(CONTINUED)`.

La variabile `continua_speech` determina se aggiungere le battute a un elemento `<sp>` esistente o crearne uno nuovo.

Raccolta battute Una volta identificato lo speaker, il sistema entra in un ciclo di raccolta che aggrega tutte le righe di dialogo consecutive fino a incontrare un elemento di interruzione. Il listato 3.7 illustra questo processo.

```
if utils.is_speaker(riga) and scena_corrente is not None:
    # identificazione speaker

    # Salto la riga dello speaker
    i += 1
    battute = []

    # Loop per raccogliere tutte le righe di dialogo del
       personaggio corrente
    while i < len(corpo_righe):
        next_riga = corpo_righe[i].strip()

        # Stop conditions: il loop si ferma quando incontra
           elementi strutturali
        if (utils.is_speaker(next_riga) or utils.
           is_location_line(next_riga) or
            utils.is_page_number(next_riga) or utils.
               is_continued_line(next_riga) or
            utils.is_more_line(next_riga) or utils.
               is_scene_number(next_riga)):
            break

        # Ignora header e transizioni durante la raccolta
        if utils.is_header_line(next_riga):
            i += 1
            continue

        if utils.is_transition_line(next_riga):
            i += 1
            continue

        # Raccolta effettiva: salva la riga nella lista
           battute
        if next_riga:
            battute.append(next_riga)
        i += 1
```

Listing 3.7: Ciclo di raccolta delle battute di dialogo

Il ciclo si interrompe quando incontra uno dei seguenti elementi strutturali:

- **nuovo speaker**: un altro personaggio che inizia a parlare ciò significa che inizia

un nuovo speech;

- **nuova location**: indica il cambio di scena;
- **numeri di pagina o elementi di continuazione**;
- **marcatore** `(MORE)`: che indica prosecuzione su pagina successiva.

Durante la raccolta, elementi non rilevanti come intestazioni e transizioni vengono filtrati per mantenere solo il contenuto effettivo del dialogo.

Gestione continuazioni (MORE) I copioni cinematografici utilizzano il marcatore `(MORE)` per indicare che il dialogo di un personaggio continua sulla pagina successiva, vedi figura 3.2. Il sistema deve riconoscere questo pattern e gestirlo correttamente. Il listato 3.8 mostra l'implementazione.

```
if utils.is_speaker(riga) and scena_corrente is not None:
    # identificazione speaker

    # raccolta delle righe di dialogo

    # Verifica se c'e (MORE) alla fine delle battute raccolte
    if battute and utils.is_more_line(battute[-1]):
        # Rimuove (MORE) dalle battute
        battute.pop()
        speech_in_continuazione = True
    else:
        speech_in_continuazione = False
```

Listing 3.8: Gestione del marcatore (MORE)

Se l'ultima riga raccolta è identificata come `(MORE)`:

1. La riga viene rimossa dalla lista delle battute (non deve apparire nell'XML finale).
2. Il flag `speech_in_continuazione` viene impostato a *True*.
3. Quando il parser incontrerà nuovamente lo stesso personaggio, aggiungerà le battute all'elemento `<sp>` esistente invece di crearne uno nuovo.

Questo meccanismo garantisce che i dialoghi spezzati su più pagine vengano ricomposti in un unico elemento XML coerente.

Generazione XML L'ultima fase del processo converte le battute raccolte nella struttura TEI-XML appropriata, distinguendo tra nuovi dialoghi e continuazioni. Il listato 3.9 mostra la logica implementata.

```
if utils.is_speaker(riga) and scena_corrente is not None:
    # identificazione speaker

    # raccolta delle righe di dialogo

    # gestione delle Continuazioni con (MORE)

    # Creazione/aggiornamento XML aggiungendo le battute
    if battute:
        if continua_speech:
            # Continua la speech precedente
            for battuta in battute:
                # Inserisce tutti gli elementi <p> nell'<sp>
                    esistente
                ultimo_sp_element.append(utils.
                    crea_elemento_testo("p", battuta))
        else:
            # Crea nuova speech
            sp = ET.SubElement(scena_corrente, "sp")
            ET.SubElement(sp, "speaker").text = speaker_name
            for battuta in battute:
                # Inserisce tutti gli elementi <p> nel nuovo
                    <sp>
                sp.append(utils.crea_elemento_testo("p",
                    battuta))
            # Aggiorna il riferimento all'ultimo elemento <sp
                >
            ultimo_sp_element = sp
            speaker_corrente = speaker_name
```

Listing 3.9: Generazione elementi XML per i dialoghi

Il sistema adotta due strategie distinte:

- **Continuazione di dialogo esistente**: `continua_speech == True`
  - Le nuove battute vengono aggiunte come elementi `<p>` all'interno dell'elemento `<sp>` precedente;
  - Mantiene l'integrità del dialogo anche quando spezzato su più pagine;

- **Nuovo dialogo**: `continua_speech == False`
  - Viene creato un nuovo elemento `<sp>` come figlio della scena corrente;
  - Si aggiunge l'elemento `<speaker>` con il nome del personaggio;
  - Ogni battuta diventa un elemento `<p>` all'interno di `<sp>`;
  - Si aggiornano le variabili di stato per tracciare l'ultimo speaker.

Questa architettura garantisce che la struttura XML risultante sia coerente con lo standard TEI e mantenga la corretta associazione tra personaggi e dialoghi.

Priorità 9 - salvataggio elementi stage

Infine, arrivati a questo punto dell'esecuzione del ciclo while principale, CAST non fa altro che considerare tutto ciò che non è stato catturato dalle analisi precedenti come elementi generici di tipo `<div type="stage">`. Come spiegato nella sezione 3.3.2 questi elementi contengono informazioni di contesto e didascalie annesse alla scene.

Formattazione e salvataggio

Una volta terminata l'esecuzione del ciclo while principale che gestisce tutta la logica di business della funzione `converti_in_tei` dello script `txt2tei.py`, CAST si occupa della formattazione e salvataggio del documento TEI-XML generato. Il listato seguente mostrerà le funzionalità appena descritte.

```
1  def converti_in_tei(percorso_txt)
2
3      # priorita 1-9 descritte in precedenza
4
5      # Formattazione e salvataggio
6      utils.indent(root)
7      tree = ET.ElementTree(root)
8      nome_file = os.path.basename(percorso_txt).replace(".txt"
           , ".xml")
9      os.makedirs(OUTPUT_DIR, exist_ok=True)
10     percorso_finale = os.path.join(OUTPUT_DIR, nome_file)
11     tree.write(percorso_finale, encoding="utf-8",
           xml_declaration=True)
```

Listing 3.10: Formattazione e salvataggio tei

3.3.4 Esempio completo di conversione

CAST memorizza tutti i copioni TEI-XML all'interno della directory `tei_scripts/`. Di seguito viene fornito un esempio di copione TEI realizzato dal sistema. La sezione di copione fornita è la stessa del testo presentato in figura 3.1, può essere interessante confrontarle per vedere concretamente i risultati ottenuti.

```
<?xml version='1.0' encoding='utf-8'?>
<TEI xmlns="http://www.tei-c.org/ns/1.0">
  <teiHeader>
    <fileDesc>
      <titleStmt>
        <title>coco</title>
      </titleStmt>
    </fileDesc>
  </teiHeader>
  <text>
    <body>
      <div type="scene" n="1">
        <stage type="location">EXT. MARIGOLD PATH - DUSK</stage>
        <stage>A path of marigold petals leads up to an altar
            lovingly</stage>
        <stage>arranged in a humble cemetery. An old woman lights
            a candle</stage>
        <stage>as the smoke of burning copal wood dances lyrically
             upward...</stage>
        <stage>CARD: DISNEY PRESENTS</stage>
        <sp>
          <speaker>MIGUEL</speaker>
          <p>Sometimes I think I'm cursed...</p>
          <p>'cause of something that happened</p>
          <p>before I was even born.</p>
        </sp>
        <sp>
          <speaker>MIGUEL</speaker>
          <p>See, a long time ago there was this family.</p>
          <p>The images on the papel picado come to life to
              illustrate a</p>
          <p>father, a mother, and a little girl.  The family is
              happy.</p>
        </sp>
      </div>
    </body>
  </text>
</TEI>
```

Listing 3.11: Esempio di output TEI-XML generato da txt2tei.py

Come si può osservare dal listato 3.11, il documento presenta una struttura gerarchica ben definita dove ogni elemento del copione originale è stato identificato e marcato secondo lo standard TEI, rendendo il documento pronto per le successive fasi di analisi computazionale. Tuttavia, se si guarda il risultato con occhio attento si osserva che l'output non è esattamente come quello che ci si aspetterebbe. Tratteremo più nel dettaglio i limiti di CAST nel capitolo 6.

3.4 Conclusioni

Riepilogando, in questo capitolo siamo entrati nei dettagli implementativi dello script `extract_txt` e `txt2tei.py` che rispettivamente gestiscono l'estrazione del testo da sorgenti eterogenee e la loro conversione in formato TEI-XML. Questo capitolo ha trattato uno dei due pilastri fondamentali dell'intera implementazione di CAST. Il prossimo argomento che verrà trattato nel capitolo 4 sarà la fase di analisi: secondo cardine fondamentale dell'intera implementazione.

Capitolo 4

Analisi dei copioni

Durante la ricerca svolta, della quale ho parlato approfonditamente nel capitolo 1, sono stati individuati esclusivamente dei tool che svolgono analisi cinematografiche relative ai personaggi con le rispettive relazioni sociali, riassunti, strutture di scene e grafici temporali di pubblicazioni di film di uno stesso autore. CAST ha un obiettivo più ambizioso: generare statistiche da copioni cinematografici relative a dimensioni di scene fino ad ora tralasciate dallo stato dell'arte.

4.1 Dimensioni analizzate

CAST è nato con l'idea di generare analisi su dimensioni che finora non sono state ancora trattate dai tool ricercati. In particolare il sistema si concentra sulla rilevazione di:

- **distinzione tra scene ambientate in luogo al chiuso o all'aperto**: `INT` oppure `EXT`
- **intuizione del luogo preciso nel quale si svolge la scena**: in particolare i luoghi che il sistema è in grado di rilevare sono
  - `'sea'`
  - `'mountain'`
  - `'urban'`
  - `'suburban'`
  - `'rural'`
  - `'desert'`
  - `'space'`
  - `'fantasy'`

- **periodo del giorno di ambientazione**:
  - `'morning'`
  - `'day'`
  - `'evening'`
  - `'night'`
- **stagione di ambientazione**:
  - `'winter'`
  - `'spring'`
  - `'summer'`
  - `'autumn'`

CAST tutte le volte che non riesce a etichettare le scene con una di queste dimensioni utilizza la categoria `UNKNOWN`.

4.2 Evoluzione dell'analizzatore: dal prototipo alla soluzione finale

L'implementazione della fase di analisi in CAST viene gestita esclusivamente dallo script `TEIAnalyzer.py`. Nel corso dell'implementazione, prima di arrivare alla soluzione finale, si sono susseguite diverse versioni dello script nate tutte con con comune denominatore: risolvere i limiti e fornire un'analisi di qualità superiore rispetto alla versione precedente. Il cuore di questo capitolo sarà quindi ripercorrere il cammino svolto in fase di sviluppo, capire il perchè di determinati cambiamenti e di scelte implementative e infine compredere l'implementazione della soluzione finale.

Prima di procedere con la spiegazione delle diverse versioni bisogna però sottolineare una differenza sostanziale di approccio tra l'individuazione del tipo di ambientazione della scena (se interna (`INT`) o esterna (`EXT`)) e le restanti implementazioni di individuazione delle dimensioni implementate. Essendo che l'informazione relativa al tipo di ambientazione della scena (`INT` o `EXT`) è già contenuta nel copione TEI, più in particolare all'interno del tag `<stage type="location"></stage>`, lo script si limita semplicemente a leggere quel valore e memorizzarlo. Ciò significa che l'informazione non è dedotta da un algoritmo di analisi, bensì il risultato è ottenuto mediante estrazione diretta dal testo strutturato del copione TEI-XML.

4.2.1 Versione base: approccio con dizionari statici

In ordine cronologico, la prima versione realizzata dello script `TEIAnalyzer` si basava su un approccio con dizionari statistici: per ogni dimensione citata nella sezione 4.1 CAST associa un dizionario in cui il valore di *chiave* è il nome della categoria mentre il *valore* è una lista molto limitata di parole affini ad essa. Il listato seguente mostra un esempio.

```
self.environment_keywords = {
    'sea': ['ocean', 'sea', 'beach', 'coast', 'waves', '
       harbor', 'port', 'ship', 'boat', 'pier', 'marina',
            'wharf', 'dock', 'seaside', 'shoreline', 'naval',
                'yacht', 'submarine', 'lighthouse'],

    'mountain': ['mountain', 'hill', 'peak', 'cliff', 'valley
       ', 'cave', 'rock', 'ridge', 'summit', 'alpine',
                    'slope', 'canyon', 'gorge', 'boulder', '
                       crag'],

    'urban': ['city', 'street', 'building', 'apartment', '
       office', 'shop', 'mall', 'downtown', 'skyscraper',
                'plaza', 'square', 'avenue', 'boulevard', '
                   metropolis', 'sidewalk', 'crosswalk', '
                   intersection',
                'alley', 'penthouse', 'loft'],

    #approccio identico...
    'suburban': [...],

    'rural': [...],

    'desert': [...],

    'space': [...],

    'fantasy': [...]
}
```

Listing 4.1: dizionario `environment_keywords` nella versione base

Architettura e funzionamento

I due metodi fondamentali dell'implementazione della versione base dell'analisi CAST sono:

- `_analyze_location`: identifica le scene `INT` o `EXT` e intuisce il luogo preciso della scena;
- `_analyze_temporal`: intuisce il periodo stagionale e il periodo del giorno della scena.

La logica dei metodi `_analyze_location` e `_analyze_temporal` per **intuire** le rispettive dimensioni è molto semplice:

1. viene eseguita un'iterazione sulle chiavi dei dizionari delle dimensioni;
2. per ogni chiave (*env_ type*) viene eseguita un'ulteriore iterazione che scorre la lista di parole affini a quella categoria (*keywords*). All'interno del ciclo:
   (a) vengono contate le occorrenze di ogni keyword nel testo;
   (b) viene stabilito un peso per ogni keyword, premiando quelle multi parola considerate più specifiche e meno ambigue delle parole singole (peso x2 rispetto al numero di parole che compone il termine - es: "late afternoon" $\mapsto$ 2 parole $\mapsto$ peso $= 4$);
   (c) si calcola lo *score* come `occurences * weight`;
   (d) assegno lo *score* come valore all'interno di un dizionario, che per ogni dimensione possiede una chiave (*env_ type*) con associato il suo score in quella scena;
   (e) infine si ricava la *env_ type* con score maggiore;
   (f) qual ora lo *score* fosse uguale a zero si imposterebbe la catogoria `"UNKNOWN"`.

Di seguito viene riportato un listato del codice per capire meglio la logica implementata.

```
environment_scores = {}

# env_type: stringa che rappresenta il tipo di ambiente
# keywords: lista di parole associate a quell'ambiente
for env_type, keywords in self.environment_keywords.items():
    score = 0
    for keyword in keywords:
        occurrences = all_text.count(keyword)
        if occurrences > 0:
            # Peso maggiore per parole piu specifiche
```

```
11              weight = len(keyword.split()) * 2 if ' ' in
                    keyword else 1
12              score += occurrences * weight
13
14      if score > 0:
15          environment_scores[env_type] = score
16
17  if environment_scores:
18      location_info.environment = max(environment_scores, key=
            environment_scores.get)
19  else:
20      location_info.environment = 'unknown'
21  }
```

Listing 4.2: logica del metodo `analyze_location` per la dimensione `environment`

La logica soprastante, come si può immaginare, viene replicata anche per le altre dimensioni analizzate. I due metodi restituiscono rispettivamente un'oggetto `LocationInfo` e `TemporalInfo`, i quali vengono trattati e aggregati dal metodo `analyze_scene`: metodo richiamante dei due metodi specifi per svolgere l'analisi. Questi dettagli verranno spiegati con maggiore dettaglio nel momento nella sezione dedicata alla versione finale implementata (sezione 4.2.3).

Valutazione critica

Facendo un valutazione critica e oggettiva di questa implementazione ho trovato punti di forza ma anche parecchi punti deboli.

Punti di forza I punti di forza che ho identificato in questa prima versione sono:

- **semplicità implementativa**: la logica utilizzata e la pipeline dello script sono facilmente comprensibili e implementabili;
- **classificazione accurata**: inserendo *keyword* specifiche nei dizionari posso ottenere una *precision* accurata (anche di scene ambientate in luoghi non abituali);
- **controllo totale sul vocabolario**: stabilendo a priori un insieme chiuso di parole che compongono il dizionario e conoscendo il set di copioni in input, posso presupporre i risultati che otterrò dall'analisi.

Limiti identificati Osservando i risultati ottenuti tuttavia mi sono subito accorto che la versione implementata aveva diversi limiti. Ad esempio:

- **scarsa *recall***: l'elevato numero di scene etichettate come `UNKNOWN` dall'analisi diminuiscono la qualità del soluzione ottenuta;
- **assenza di varianti lessicali**: il sistema è in grado di etichettare solo le parole che fanno match con le keyword; qual ora invece dovessero essere presenti sinonimi, che farebbero comunque ricondurre alla categoria in questione, verrebbero trascurati;
- **coverage limitata del vocabolario cinematografico**: il mondo cinematografico utilizza un vocabolario molto ricco non compreso interamente da CAST.

A seguito della presa visione di questi limiti, era necessario trovare una soluzione che mi permettesse di aumentare la recall, in modo da migliorare la qualità delle mie analisi e ridurre di fatto le scene "perse". L'intuizione che mi ha poi portato alla creazione della versione successiva, è stata quella di utilizzare una libreria esterna che mi avrebbe consentito di ampliare a *runtime* i miei dizionari, nella speranza di aumentare il più possibile la copertura delle scene.

4.2.2 Versione espansa: integrazione di WordNet

Come evidenziato nella sezione 4.2.1 il limite maggiore della versione 1.0 di `TEIAnalyzer` è la scarsa recall del sistema di analisi. Per risolvere questo problema ho utilizzato una libreria python che mi consentisse di espandere i miei limitati dizionari precedentemente implementati. La libreria di cui sto parlando è NLTK (**Natural Language Toolkit**) [16]. Questa libreria è stata utilizzata perchè fornisce un'interfaccia python per accedere al database WordNet, il quale verrà utilizzato per espandere i dizionari di CAST. In sostanza **l'approccio all'analisi rimane invariato**, l'unica differenza è che CAST lavorerà su dizionari di parole espansi rispetto al prototipo iniziale.

Informazioni su WordNet

WordNet [17] è un ampio database lessicale della lingua inglese, sviluppato dall'Università di Princeton. Nomi, verbi, aggettivi e avverbi sono raggruppati in insiemi di sinonimi cognitivi (**synset**), ognuno dei quali esprime un concetto distinto. I synset sono interconnessi tramite relazioni concettuali-semantiche e lessicali. La struttura di WordNet lo rende uno strumento utile per la linguistica computazionale e l'elaborazione del linguaggio naturale.

WordNet assomiglia superficialmente a un *thesaurus* (dizionario dei sinonimi), in quanto raggruppa le parole in base al loro significato. Tuttavia, presenta alcune importanti distinzioni. In primo luogo WordNet collega non solo le forme delle parole (stringhe di lettere), ma anche i significati specifici delle parole. Di conseguenza, le parole che si trovano in stretta prossimità tra loro nella rete vengono disambiguate

semanticamente. In secondo luogo, WordNet etichetta le relazioni semantiche tra le parole, mentre i raggruppamenti di parole in un thesaurus non seguono alcuno schema esplicito se non la somiglianza di significato.

Implementazione tecnica

Per implementare WordNet con NLTK [18] all'interno di CAST è stata importata l'interfaccia `WORDNET` (interfaccia specifica per accedere al database) dal modulo `nltk.corpus`, modulo di NLTK che contiene interfacce per vari corpora linguistici. Nel listato seguente viene mostrato come viene importato.

```
1 import nltk
2 from nltk.corpus import wordnet as wn
```

Listing 4.3: import Wordnet

Una volta importato il database esterno all'interno di CAST e definiti i dizionari iniziali viene richiamata la funzione `_expand_keywords_with_wordnet`. Questa funzione è suddivisa in tre parti:

1. **funzione interna** `expand_list`: funzione che concretamente espande i dizionari;
2. **applicazione della funzione** per `environment_keywords`;
3. **applicazione della funzione** per `time_keywords`.

La funzione `expand_list` in primo luogo crea un set contentente le parole originali. Dopo di che vengono eseguiti tre loop innestati:

1. **Primo loop**: scorre ogni parola della lista originale.
2. **Secondo loop**: per ogni singola parola recupera tutti i synset attraverso un l'API `wn.synsets(word)`. Un synset è il significato della parola analizzata.
3. **Terzo loop**: per ogni synset vengono estratti tutti i **lemmi** tramite l'API `yn.lemmas()`. I lemmi sono parole concrete che fanno riferimento a un determinato synset; ognuno di questi lemmi viene inserito nell'insieme espanso `expanded`.

```
1 def _expand_keywords_with_wordnet(self):
2 """Espande i dizionari di keyword con sinonimi WordNet"""
3 def expand_list(words: List[str]) -> List[str]:
4     expanded = set(words)
5     for word in words:
```

```
            for syn in wn.synsets(word):
                for lemma in syn.lemmas():
                    expanded.add(lemma.name().replace("_", " "))
        return list(expanded)

for env, keywords in self.environment_keywords.items():
    self.environment_keywords[env] = expand_list(keywords)

for time_key, keywords in self.time_keywords.items():
    self.time_keywords[time_key] = expand_list(keywords)
```

Listing 4.4: Espansione dei dizionari tramite Wordnet

Il risultato che otteniamo al termine dell'esecuzione della funzione presente nel listato 4.4 sarà che per ogni categoria definita all'interno dei dizionari `environment_keywords` e `time_keywords`, la sua lista di parole affini sarà espansa.

Ora che è stato raggiunto l'obiettivo prefissato (espandere i dizionari per ridurre il numero di scene etichettate come `UNKNOWN`) viene implementata la medesima logica di analisi illustrata nella sezione 4.2.1.

Analisi critica dei risultati

Osservando i risultati raggiunti dalla versione 2.0 dello script `TEIAnalyzer`, emergono subito all'occhio degli aspetti positivi ma anche dei nuovi problemi emersi.

Miglioramenti ottenuti Il grande miglioramento emerso da questa versione è sicuramente l'incredibile diminuzione di scene etichettate da CAST come `UNKNOWN`: la **recall è stata migliorata**. Vediamo i dati di questo miglioramento, dimensione per dimensione.

Dimensione	Versione 1	Versione 2
Luoghi	6015 (38.5%)	852 (5.5%)
Stagioni	10680 (68.4%)	6074 (38.9%)
Periodi del giorno	3659 (23.4%)	1744 (11.2%)

Tabella 4.1: Confronto del numero di scene classificate come "unknown" tra versione 1 e la versione 2

Se ci soffermiamo esclusivamente ad osservare questo dato, si potrebbe concludere che quest'ultima versione analizza i copioni in maniera accurata e affidabile avendo così raggiunto l'obiettivo finale. Tuttavia, osservando meglio i dati analizzati, si osserva chiaramente l'avvento di un nuovo fenomeno indesiderato e non presente nella prima versione.

Problema emerso: il fenomeno del rumore semantico Con l'introduzione della libreria NTLK che estende i dizionari iniziali tramite WordNet è nato un nuovo problema relativo alla fase di analisi. Il problema nasce dal fatto che la funzione `_expand_keywords_with_wordnet` si limita semplicemente ad aggiungere tutti i lemmi appartenenti ad un specifico synset data una keyword, senza preoccuparsi di:

- **eccessiva generalità**: il sistema non controlla se il sinonimo che sta inserendo è effettivamente appropriato o meno al contesto;
- **termini meta-linguistici**: WordNet può includere termini che descrivono il linguaggio piuttosto che concetti concreti.

Queste sono le principali cause del **rumore semantico**. Questo fenomeno crea una **perdita di specificità**, molte parole troppo comuni compaiono in troppe categorie diverse annullando così la capacità discriminante; questo potrebbe provocare **falsi positivi semantici**: scene che non dovrebbero essere etichettate con una categoria invece lo diventano a causa del rumore. Questo fenomeno è chiaramente osservabile se si analizza nel dettaglio le analisi sull'identificazione del luogo di una scena. Vediamo ora un confronto dettaglio dei due risultati per evidenziare il problema.

Categoria	Versione 1	Versione 2
rural	**556**	**8195**
unknown	6015	852
urban	3093	2009
mountain	1198	1222
suburban	1557	1126
sea	1435	606
fantasy	902	1198
desert	154	38
space	705	369

Tabella 4.2: Confronto delle distribuzioni degli ambienti tra v1 e v2: evidenza del rumore semantico nella categoria "rural"

Come evidenziato dalla tabella 4.2, il fenomeno del rumore semantico si verifica particolarmente nell'etichettatura di una scena come `rural`. Per risolvere il problema è stata pensata l'implementazione di una versione finale, la quale cercasse di mantenere gli aspetti positivi di questa versione e che contemporaneamente risolvesse il problema del rumore semantico.

4.2.3 Versione finale: approccio ibrido con filtri anti-rumore

Per risolvere il problema del rumore semantico, provocato dall'espansione troppo generalizzata dei dizionari tramite WordNet, si è pensato di mantenere invariato l'approccio

inserendo però dei **filtri anti-rumore**. L'obiettivo dell'inserimento di questi di filtri è cercare di diminuire quanto più possibile la perdità di specificità, diminuire i falsi positivi e quindi di conseguenza rimuovere il rumore semantico. Per fare ciò è stato necessario rivedere la definizione dei dizionari iniziali.

Architettura a tre livelli

Il dizionario iniziale dei luoghi, rispetto ai primi due prototipi, ha subito una modifica importante dal punto di vista strutturale. Se in precedenza tutte le parole di ogni categoria avevano la stessa importanza, ora non ce l'hanno più. Di seguito viene riportata la nuova struttura:

- **dizionario primario**: `core_keywords`;
- **dizionario secondario**: `environment_keywords`;
- **dizionario blacklist**: `blacklist`.

Core keywords Il dizionario primario contiene per ogni categoria i lemmi più appropriati: sono quei termini che, incontrati in fase di analisi, danno più garanzie sull'affidabilità dell'etichettatura di una scena. Proprio per questo motivo, durante il calcolo dello *score*, saranno pesati con un moltiplicatore x3.

Environment keywords Il dizionario secondario contiene i termini più generali. Essi, non garantendo una grande affidabilità, durante il calcolo dello *score* gli verranno assegnati un moltiplicatore x1. Nel listato seguente viene riportato un esempio di come è stato ristrutturato concretamente il dizionario per etichettare i luoghi di una scena.

```
# CORE KEYWORDS - Alta priorita, peso maggiore
self.core_keywords = {
    'sea': ['ocean', 'sea', 'beach', 'ship', 'boat', '
       submarine'],
    'mountain': ['mountain', 'hill', 'peak', 'cliff', 'cave'
       ],
    #restanti keywords...
}

# EXTENDED KEYWORDS - Bassa priorita, peso minore
self.environment_keywords = {
    'sea': ['waves', 'harbor', 'port', 'pier', 'marina', '
       wharf', 'dock',
            'seaside', 'shoreline', 'naval', 'yacht', '
               lighthouse'],
```

```
12      'mountain': ['valley', 'rock', 'ridge', 'summit', 'alpine
            ', 'slope',
13                          'canyon', 'gorge', 'boulder', 'crag'],
14      #restanti keywords...
15  }
```

Listing 4.5: Modifica struttura dizionari per analisi finale

Blacklist Il dizionario `blacklist` è stato introdotto con lo scopo di risolvere il problema principale del secondo prototipo (scarsa *precision*). Il suo compito è filtrare e escludere in fase di analisi tutti quei sinonimi troppo generali, che provocherebbero rumore e falsi positivi. Ecco come è implementato il dizionario `blacklist`.

```
1  self.blacklist = {
2      'general': ['place', 'area', 'location', 'site', 'spot',
           'position',
3                  'room', 'space', 'time', 'day', 'way', 'thing
                       ', 'part'],
4      'urban_noise': ['center', 'point', 'line', 'side', 'end',
            'back',
5                      'front', 'top', 'bottom', 'inside', '
                           outside'],
6      'false_positives': ['character', 'scene', 'act', 'stage']
7  }
```

Listing 4.6: Dizionario blacklist

Regole di disambiguazione e soglie di confidenza

Per aumentare ancora di più la qualità dei risultati ottenuti sono state introdotte:

- **regole di disambiguazione**: sistema che risolve eventuali conflitti fra diverse categorie della stessa dimensione (funzione `_apply_disambiguation_rules`);
- **soglie di confidenza**: sistema che verifica se lo score massimo individuato per una scena è sufficientemente alto per attribuire quella categoria alla dimensione considerata.

Maggiori dettagli implementativi saranno spiegati all'interno della sezione dedicata (sezione 4.3).

Parametri configurabili

Un ultima caratteristica della versione finale dello script `TEIAnalyzer` è la possibilità di configurare manualmente due parametri fondamentali per lo svolgimento dell'analisi:

- `expand_with_wordnet`: valore booleano che, se impostato a *True*, permette a CAST di eseguire l'analisi con l'espansione dei dizionari iniziali tramite WordNet. Viceversa, se impostato a *false*, l'analisi avverrà senza l'espansione dei dizionari;
- `max_synonyms`: valore intero che mi stabilisce il livello di espansione dei dizionari. Più nel dettaglio, il valore indica il numero di sinonimi che viene fornito per ogni lemma di ogni synsets.

Questa possibilità di configurazione dei parametri prima dell'avvio dell'analisi permette di far scegliere all'utente che tipo di analisi verrà svolta da CAST: se un'analisi che predilige l'accuratezza (**precision**) riducendo l'espansione dei dizionari, oppure una che predilige la copertura (**recall**) aumentando il numero di sinonimi considerati. Di seguito viene fornito il setup di default eseguito da CAST.

```
analyzer = TEIAnalyzer(
    expand_with_wordnet=True,
    max_synonyms=50
)
```

Listing 4.7: Configurazione iniziale di default `TEIAnalyzer.py`

4.3 Implementazione dell'analizzatore

All'interno di questa sezione entreremo più nel dettaglio della spiegazione del codice della versione finale dello script `TEIAnalyzer.py`, responsabile di gestire tutta la logica di business dell'analisi di CAST.

4.3.1 Pipeline di analisi

L'analisi di CAST prevede l'implementazione di una pipeline suddivisa principalmente in due macro categorie:

1. **Espansione**: fase in cui vengono estratti i dizionari di partenza tramite NLTK con i dizionari WordNet.
2. **Analisi**: fase nella quale avviene lo svolgimento concreto dell'analisi.

Di seguito verranno riportate due figure che illustrano le pipeline e quindi la logica di queste due fasi.

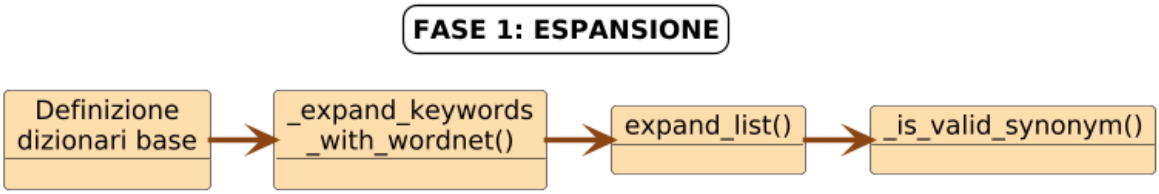


Figura 4.1: Illustrazione pipeline fase di espansione di TEIAnalyzer

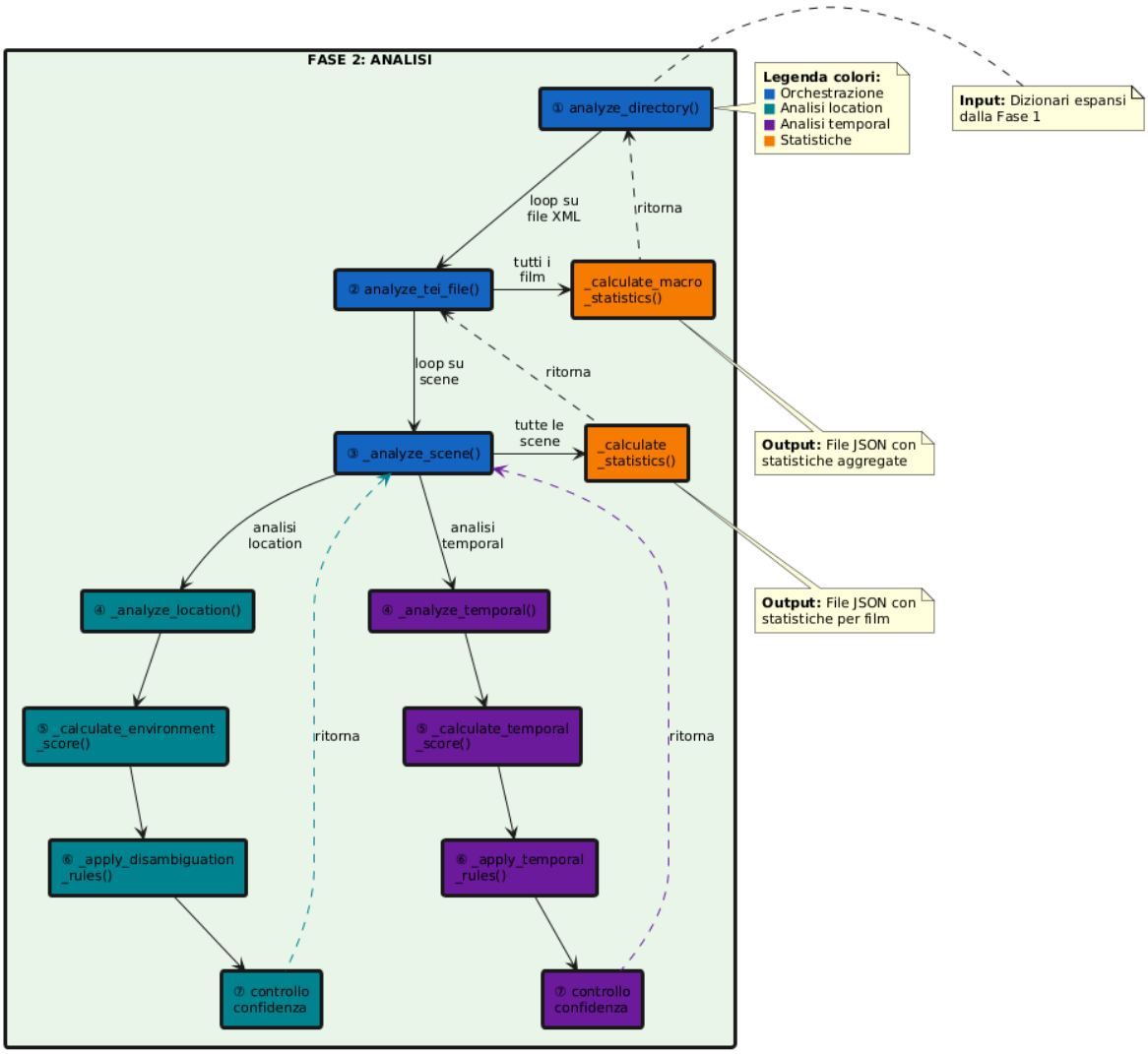


Figura 4.2: Illustrazione pipeline fase di analisi di TEIAnalyzer

4.3.2 Dettagli implementativi

Di seguito verrà descritto il cuore della logica relativa allo script `TEIAnalyzer`, gestendo separatamente le fasi di espansione e analisi.

Espansione

La logica di analisi CAST inizia con la definizione dei seguenti dizionari:

- `core_keywords`: dizionario contenente le parole chiave più rilevanti per le categorie della dimensione dei luoghi di una scena;
- `environment_keywords`: dizionario contenente le parole più generiche per le categorie della dimensione dei luoghi di una scena;
- `blacklist`: dizionario contenente le parole che potrebbero verificare problemi di eccessiva generalità, rumore semantico e falsi positivi;
- `time_keywords`: dizionario contenente le parole relative alle dimensioni di periodi del giorno e delle stagioni di ambientazione delle scene.

Successivamente viene richiamato il metodo `_expand_keywords_with_wordnet`. La logica del metodo è esattamente la stessa a quella descritta nel listato 4.4 ma con una piccola differenza: all'interno del ciclo più interno viene richiamato il metodo `_is_valid_synonym`, il quale ha il compito di filtrare il sinonimo estratto in base a diversi criteri:

- **check blacklist**: controlla se il sinonimo fa parte della lista di parole da escludere definite inizialmente;
- **lunghezza parola**: se la parola è inferiore a tre caratteri è troppo generica e quindi viene trascurata;
- **validazione alfabetica**: se la parola contiene numeri o caratteri speciali viene ignorata;
- **diversità**: se la parola è troppo diversa da quella di partenza viene ignorata.

Da sottolineare un fattore importante: solo il dizionario `environment_keywords` è stato esteso mentre il `core_keywords` no. La motivazione che ha spinto a questa scelta è il tentativo di mantenere quanto più contenuto possibile il rumore semantico. Al termine dello svolgimento del metodo `_expand_keywords_with_wordnet`, la fase di espansione dei dizionari può considerarsi conclusa.

Analisi

Osservando la figura 4.2, la fase di analisi vera e propria inizia con l'invocazione del metodo `analyze_directory` che itera sulla lista file xml e per ognuno richiama il metodo `analyze_tei_file`.

```
1 tei_files = [f for f in os.listdir(tei_dir) if f.endswith('.
    xml')]
2 for tei_file in tei_files:
3     file_path = os.path.join(tei_dir, tei_file)
```

```
4       print(f"Analizzando: {tei_file}")
5
6       analysis = self.analyze_tei_file(file_path)
7       results.append(analysis)
```

Listing 4.8: invocazione metodo analyze_tei_file

Successivamente il metodo `analyze_tei_file` tramite la classe `xml.etree.ElementTree` estrae tutte le scene e su ognuna richiama il metodo `_analyze_scene`.

```
1 scenes = []
2 scene_divs = root.findall('.//{http://www.tei-c.org/ns/1.0}
      div[@type="scene"]')
3
4 for scene_div in scene_divs:
5       scene_analysis = self._analyze_scene(scene_div)
6       if scene_analysis:
7           scenes.append(asdict(scene_analysis))
```

Listing 4.9: invocazione metodo _analyze_scene

A questo punto il metodo `_analyze_scene` analizza ogni singola scena estraendo il contenuto dei tag `<stage type="location">` e `<stage>`, i quali vengono passati come parametri nell'invocazione dei metodi:

- `_analyze_location`: metodo che esegue le analisi sulla localizzazione delle scene (scene interne o esterne e luogo specifico);
- `_analyze_temporal`: metodo che esegue le analisi sulla dimensione temporale delle scene (periodo del giorno e stagione dell'anno).

Analisi di localizzazione In questo paragrafo osserveremo il cuore della logica del metodo `_analyze_location`. Esso è suddiviso principalmente in tre fasi.

Fase 1: individuazione scene `INT` e `EXT` CAST dispone di questa informazione in maniera esplicita all'interno della scena che sta analizzando, di conseguenza dovrà solamente individuarla nel tag specifico.

```
1 location_upper = location_text.upper()
2 if location_upper.startswith('INT') or 'INT.' in
      location_upper:
3       location_info.type = 'INT'
4 elif location_upper.startswith('EXT') or 'EXT.' in
      location_upper:
5       location_info.type = 'EXT'
```

```
else:
    location_info.type = 'UNKNOWN'
```

Listing 4.10: Script di analisi scene interne o esterne

Fase 2: Analisi semantica con score pesati In questa seconda fase CAST itera la lista di chiavi del dizionario `core_keywords` e per ognuna calcola lo score attraverso il metodo `_calculate_environment_score`. Quest'ultimo assegna il peso (*weight*) a seconda di due criteri:

- **importanza lemma**: se la parola appartiene al `core_keywords` significa che la parola è fortenemente rilevante per la determinazione della dimensione, per cui CAST gli assegna un moltiplicatore x3. Viceversa, se appartiene al dizionario secondario `environment_keywords` non verrà applicato nessun moltiplicatore;
- **lunghezza lemma**: se il lemma è composto da più parole CAST gli attribuisce un peso maggiore inserendo un moltiplicatore *numero_parole_lemma * 2*.

A uno stesso lemma possono essere assegnati entrambi i moltiplicatori. Una volta contato il numero di occorrenze del lemma all'interno della sezione dei tag predefiniti, lo score si calcola come `score += occurences * weight`.

```
def _calculate_environment_score(self, text: str, env_type:
    str) -> float:
     score = 0.0

     # CORE KEYWORDS - Peso 3x
     for keyword in self.core_keywords.get(env_type, []):
         occurrences = text.count(keyword)
         if occurrences > 0:
             weight = 3.0 * (len(keyword.split()) * 2 if ' '
                 in keyword else 1)
             score += occurrences * weight

     # EXTENDED KEYWORDS - Peso 1x
     for keyword in self.environment_keywords.get(env_type,
        []):
         occurrences = text.count(keyword)
         if occurrences > 0:
             weight = len(keyword.split()) * 2 if ' ' in
                 keyword else 1
             score += occurrences * weight

```

```
18      return score
```

Listing 4.11: Funzione _calculate_environment_score

Successivamente, proseguendo il flusso del metodo `_analyze_location`, CAST salva il valore dello `score` di ogni categoria all'interno del set `environment_scores`.

Una volta terminato lo scorrimento del copione avendo assegnato uno score ad ogni categoria viene richiamato il metodo `_apply_disambiguation_rules`. Lo scopo generale del metodo è risolvere conflitti e ambiguità quando più categorie ambientali hanno punteggi simili. Applica regole euristiche basate su co-occorrenze semantiche per favorire la categoria più appropriata. In particolare:

- il metodo assegna dei **boost** del 50% agli score delle categorie che possiedono parole chiave altamente specifiche. In altre parole la filosofia è **dare segnali forti e certi** rispetto alla somma di molti segnali deboli. Questa regola avviene per categorie specifiche come il mare, la montagna e lo spazio;
- il metodo **penalizza** la categoria `urban` del 20% se fra le altre categorie ne esiste almeno una con uno score maggiore del 70% rispetto alla categoria `urban`.

La motivazione dell'inserimento di quest'ultimo punto è dovuto al fatto che essendo `urban` la categoria più "rumorosa", in quanto ha molti termini generici, tende a dominare anche quando non è appropriata. Di conseguenza, se un'altra categoria ha un punteggio competitivo (>70%), `urban` viene penalizzata per dare spazio alla categoria più specifica.

```
1  def _apply_disambiguation_rules(self, scores: Dict[str, float
       ], text: str) -> Dict[str, float]:
2      # Regola 1: Se "sea" e "beach" presenti -> favorisce sea
3      if 'sea' in text or 'beach' in text:
4          scores['sea'] = scores.get('sea', 0) * 1.5
5
6      # Regola 2: Se "mountain" e "cave" presenti -> favorisce
           mountain
7      if 'mountain' in text or 'cave' in text:
8          scores['mountain'] = scores.get('mountain', 0) * 1.5
9
10     # Regola 3: Se "space" e "planet" presenti -> favorisce
           space
11     if 'space' in text or 'planet' in text:
12         scores['space'] = scores.get('space', 0) * 1.5
13
14     # Regola 4: Penalizza urban se score troppo vicino ad
           altri
```

```
15      if 'urban' in scores:
16          other_scores = [s for e, s in scores.items() if e !=
                'urban' and s > 0]
17          if other_scores and max(other_scores) > scores['urban
                '] * 0.7:
18              scores['urban'] *= 0.8
19
20      return scores
```

Listing 4.12: Funzione _apply_disambiguation_rules

Fase 3: soglia minima di classificazione L'ultimo step del metodo di analisi degli ambienti implementa l'ultimo filtro applicato ai risultati. Di seguito viene spiegato nel dettaglio questo filtro:

1. **soglia minima**: CAST controlla che la categoria con lo score massimo non abbia un valore inferiore a un valore di soglia minima. Qual ora questa condizione risulti falsa, il sistema attribuirà l'etichetta `UNKNOWN` alla categoria. Questo controllo viene svolto per evitare classificazioni basate su match casuali o troppo vaghi;

2. **predominanza**: se la condizione del punto precedente risulta soddisfatta, CAST controlla se i due valori di score differiscono di almeno del 20%. Se la condizione risulta vera, il risultato finale resta invariato; viceversa significa che c'è molta incertezza su quale sia il valore corretto, di conseguenza il sistema applica l'etichetta `UNKNOWN`.

Come si può capire dalla logica implementata, **CAST predilige ammettere l'incertezza del risultato** piuttosto che dare classificazioni non sicure.

```
1  def _apply_disambiguation_rules(self, scores: Dict[str, float
       ], text: str) -> Dict[str, float]:
2  # STEP 1 ...
3  # STEP 2 ...
4  # STEP 3: Soglia minima per classificazione
5      MIN_SCORE_THRESHOLD = 2.0
6
7      if environment_scores:
8          max_score = max(environment_scores.values())
9
10         if max_score < MIN_SCORE_THRESHOLD:
11             location_info.environment = 'unknown'
12         else:
```

```
13              sorted_scores = sorted(environment_scores.items()
                    , key=lambda x: x[1], reverse=True)
14
15              if len(sorted_scores) > 1:
16                  first_score = sorted_scores[0][1]
17                  second_score = sorted_scores[1][1]
18
19                  # Se la differenza e' < 20% -> troppo ambiguo
                        -> unknown
20                  if second_score > first_score * 0.8:
21                      location_info.environment = 'unknown'
22                  else:
23                      location_info.environment = sorted_scores
                            [0][0]
24              else:
25                  location_info.environment = sorted_scores
                        [0][0]
26      else:
27          location_info.environment = 'unknown'
```

Listing 4.13: Filtro finale - soglia di confidenza

Analisi temporale In questo paragrafo viene spiegato nel dettaglio la logica del metodo `_analyze_temporal`, dedicato all'individuazione del periodo del giorno e della stagione nella quale è ambientata la scena. La logica di base è molto simile a quella del metodo `_analyze_location`; tuttavia, per queste ultime due dimensioni, CAST ne semplifica un po' l'implementazione.

```
1       def _analyze_temporal(self, location_text: str,
            stage_texts: List[str]) -> TemporalInfo:
2           """Analisi temporale (invariata dalla versione
                precedente)"""
3           temporal_info = TemporalInfo()
4           all_text = (location_text + ' ' + ' '.join(
                stage_texts)).lower()
5
6           # Periodo del giorno
7           time_scores = {}
8           for time_type, keywords in self.time_keywords.items()
                :
9               if time_type in ['morning', 'day', 'evening', '
                    night']:
```

```
                score = 0
                for keyword in keywords:
                    occurrences = all_text.count(keyword)
                    if occurrences > 0:
                        weight = 2 if len(keyword) > 5 else 1
                        score += occurrences * weight
                if score > 0:
                    time_scores[time_type] = score

        temporal_info.period = max(time_scores, key=
            time_scores.get).upper() if time_scores else '
            UNKNOWN'

        #stesso approccio per la dimensione stagione
```

Listing 4.14: Metodo _analyze_temporal

Una volta terminata l'esecuzione del metodo `_analyze_temporal`, il cuore dell'analisi è terminata. Quello che resta da fare, visualizzando la pipeline in figura 4.2, è generare le statistiche per singolo film e le statistiche aggregate di sistema. Rispettivamente queste due tipologie differenti di statistiche vengono generate dai metodi `_calculate_statistics` e `_calculate_macro_statistics`.

Aggregazione delle statistiche

Il metodo `_calculate_statistics` riceve in input la lista di tutte le scene analizzate di un singolo film e genera statistiche aggregate per esso. Per ogni dimensione analizzata (location type, environment, period, season), il metodo esegue le seguenti operazioni:

1. estrae i valori classificati da tutte le scene tramite list comprehension;
2. utilizza `Counter` (dalla libreria `collections`) per contare le occorrenze di ogni categoria;
3. identifica la categoria dominante attraverso il metodo `most_common(1)`.

Il risultato è un dizionario strutturato che contiene le distribuzioni per ogni dimensione e le categorie più frequenti del film.

Il metodo `_calculate_macro_statistics`, invece, opera su scala più ampia: riceve la lista di tutti i risultati dei singoli film e produce statistiche aggregate dell'intero dataset. Il processo si articola in:

1. **filtraggio**: esclude film con errori di analisi (`valid_results`);

2. **aggregazione**: costruisce liste globali contenenti tutte le classificazioni di tutti i film;
    - per ogni film, estrae le distribuzioni delle categorie;
    - replica ogni categoria un numero di volte pari al suo conteggio;
    - ottiene così liste "piatte" che rappresentano tutte le scene analizzate;
3. **calcolo statistiche globali**: applica `Counter` alle liste aggregate per ottenere:
    - conteggi totali per ogni categoria;
    - percentuali sul totale delle scene;
    - categorie dominanti a livello di sistema;
4. **metadati**: aggiunge informazioni di sintesi (numero film analizzati, scene totali, media scene per film, timestamp).

Entrambi i metodi producono output in formato JSON, rispettivamente salvati nei file:

- `screenplay_analysis.json` (statistiche individuali);
- `screenplay_analysis_macro_stats.json` (statistiche aggregate).

4.3.3 Output generati

In questa sezione sono visualizzate le strutture degli output in formato JSON generati dai metodi `_calculate_statistics` e `_calculate_macro_statistics`.

`_calculate_statistics`


```
{
  "film": "1917",
  "total_scenes": 61,
  "statistics": {
    "locations": {
      "int_ext_distribution": {
        "EXT": 43,
        "INT": 18
      },
      "environment_distribution": {
        "rural": 42,
        "unknown": 9,
        "mountain": 1,
```

```
          "fantasy": 1,
          "urban": 5,
          "suburban": 2,
          "sea": 1
        },
        "most_common_environment": "rural"
      },
      "temporal": {
        "day_night_distribution": {
          "DAY": 5,
          "UNKNOWN": 20,
          "MORNING": 20,
          "NIGHT": 12,
          "EVENING": 4
        },
        "season_distribution": {
          "autumn": 27,
          "unknown": 15,
          "spring": 3,
          "summer": 4,
          "winter": 12
        },
        "most_common_period": "UNKNOWN"
      }
    },
    "analysis_timestamp": "2025-10-06T14:44:46.576771"
}
```


Listing 4.15: Struttura JSON delle statistiche per singolo film (1917)

`_calculate_macro_statistics`


```
{
  "analysis_summary": {
    "total_films_analyzed": 100,
    "total_scenes_analyzed": 15615,
    "average_scenes_per_film": 156.2,
    "analysis_timestamp": "2025-10-06T14:44:51.828140"
  },
  "aggregated_statistics": {
    "locations": {
      "int_ext_totals": {
        "EXT": 5753,
        "INT": 9261,
        "UNKNOWN": 601
      },
```

```
      "int_ext_percentages": {
        "EXT": 36.8,
        "INT": 59.3,
        "UNKNOWN": 3.8
      },
      "environment_totals": {
        "rural": 755,
        "unknown": 6036,
        "mountain": 925,
        "fantasy": 1110,
        "urban": 2637,
        "suburban": 1627,
        "sea": 1273,
        "desert": 175,
        "space": 1077
      },
      "most_common_overall": {
        "location_type": "INT",
        "environment": "unknown"
      }
    },
    "temporal": {
      "period_totals": {
        "DAY": 4473,
        "MORNING": 3495,
        "NIGHT": 4228,
        "EVENING": 1122,
        "UNKNOWN": 2297
      },
      "season_totals": {
        "autumn": 3694,
        "unknown": 6275,
        "spring": 1672,
        "summer": 1599,
        "winter": 2375
      },
      "most_common_overall": {
        "period": "DAY",
        "season": "unknown"
      }
    }
  }
}
```


Listing 4.16: Struttura JSON delle statistiche aggregate del sistema

4.4 Confronto sintetico delle tre versioni

Ora che è stato descritto tutto il processso cronologico che ha portato alla creazione della versione finale dello script `TEIAnalyzer.py` si può fare un breve riassunto degli aspetti più significativi per ognuna delle versioni presentate.

Caratteristica	V1	V2	V3
Dimensione dizionari	Piccola (manuale)	Grande (espansione automatica)	Media (ibrida con filtri)
Espansione WordNet	No	Sì (illimitata)	Sì (controllata, max 50 sinonimi)
Filtri anti-rumore	Non necessari	Assenti	Blacklist + validazione sinonimi
Sistema scoring	Semplice (occorrenze)	Semplice (occorrenze)	Pesato (core 3x, extended 1x)
Disambiguazione	No	No	Sì (regole + soglie)
Precision	Alta	Bassa	Alta
Recall	Bassa	Alta	Alta
Rumore semantico	Assente	Elevato	Controllato
Manutenibilità	Alta	Bassa	Media
Scalabilità	Bassa	Alta	Alta

Tabella 4.3: Confronto sintetico delle caratteristiche delle tre versioni dell'analizzatore

La tabella 4.3 mette chiaramente in evidenza come la versione 3 sia quella più bilanciata e che offre un compromesso migliore. In particolare, il valore di **recall** e **precision** sono entrambi soddisfacenti mentre nei prototipi precedenti era molto difficile trovare un trade-off soddisfacente. In conclusione si può dire che è stato raggiunto un discreto livello di qualità delle analisi.

4.5 Conclusioni

In questo capitolo abbiamo affrontato in maniera molto approfondita l'analisi di CAST. In primo luogo è stata descritta l'evoluzione dell'analizzatore attraverso tre versioni successive, ciascuna progettata per risolvere le limitazioni della precedente mantenendo i punti di forza acquisiti. In generale, è stato descritto un sistema complesso che prende in input dei copioni in formato xml-tei, svolge delle analisi e su di esse calcola delle statistiche sul singolo film e delle statistiche aggregate sull'intero dataset.

Con la chiusura di questo capitolo si chiude anche la descrizione dei dettagli implementativi del codice sorgente. Nel capitolo successivo andremo ad osservare i risultati ottenuti da questa fase di analisi, in modo da poter fare alcune considerazioni e deduzioni sugli aspetti scenografici del mondo cinematografico.

Capitolo 5

Risultati e valutazione

Il mondo cinematografico è un settore in costante evoluzione nel quale le scelte e le decisioni, prese in fase di produzione dagli sceneggiatori e dai registi, hanno un effetto determinante sul prodotto finale. In questo capitolo verrà posta l'attezione sui risultati effettivi prodotti dall'analisi di CAST, in modo da poter fare deduzioni e considerazioni sulle tendenze degli sceneggiatori, comprendere quali sono gli ambienti dominanti nel cinema moderno, cercare di creare relazioni per i diversi generi e infine verificare la correttezza di CAST.

5.1 Panoramica dei Risultati Aggregati

CAST fornisce un primo insieme di risultati basati sull'analisi aggregata del dataset iniziale. In questa sezione verranno espressi i risultati ottenuti da questa prima fase. Tuttavia per poterli capire è necessario prima di tutto conoscere il dataset iniziale utilizzato per l'analisi.

5.1.1 Dataset analizzato

Il dataset analizzato è composto da **100 copioni cinematografici** estratti dal database Internet Movie Script Database (IMSDB) e da un archivio digitale Script Slug contenente copioni in formato pdf. I film non sono stati scelti casualmente, bensì sono stati scelti con l'obiettivo di coprire quanto meglio possibile gli ultimi cinquant'anni del panorama cinematografico. In questo modo CAST sarà in grado di fornire risultati equilibrati, rappresentando equamente non solo i film più moderni ma anche quelli meno recenti. Di seguito sarà inserito un elenco di tutti i generi di film utilizzati in CAST, esplicitando anche alcuni titoli di pellicole per ogni categoria:

- **Film d'azione e avventura**: Pirates of the Caribbean, Raiders of the Lost Ark, Guardians of the Galaxy Vol 2.

- **Fantascienza e space opera**: Star Wars: A New Hope, Interstellar, Star Trek II: The Wrath of Khan.
- **Supereroi**: Avengers: Endgame, Spider-Man 2, Fantastic Four.
- **Fantasy**: Lord of the Rings: Fellowship of the Ring, Beauty and the Beast, Shrek the Third.
- **Drammi storici**: Amadeus, Napoleon, The Pianist, 1917.
- **Drammi contemporanei**: Fight Club, Inception, The Matrix, Forrest Gump.
- **Commedie romantiche**: Pretty Woman, Notting Hill (Sex and the City), Bridget Jones's Baby.
- **Film biografici**: The Theory of Everything, Hidden Figures, King Richard.
- **Film d'animazione**: Coco, Elemental, Inside Out, Kung Fu Panda.
- **Thriller psicologici**: Psycho, The Prestige, A Haunting in Venice.

Questa varietà di generi e diversità temporale permette a CAST di gestire convenzioni di scrittura di copioni modificate nel tempo, variabilità stilistiche e approcci narrattivi differenti.

5.1.2 Metriche generali

L'analisi automatica di CAST, eseguita sul dataset descritto nella sezione precedente, ha prodotto le seguenti metriche aggregate:

- **film Analizzati:** 100;
- **scene Totali:** 15.615;
- **scene per Film (media):** 156.2;
- **scene Interne:** 59.3%.

La media di **156.2 scene per film** è in linea con i risultati attesi, in quanto una struttura tipica di un copione professionale si aggira tra le 90 e 250 scene. Il loro numero può variare a seconda della durata del film, del numero di transizioni presenti nella pellicola e infine, non per importanza, dal registra.

Come riportato nella sezione 5.1.1, il dataset è molto eterogeneo e comprende una grande vastità di generi e stili differenti di film. Questo fatto viene confermato dalla variabilità del numero di scene tra i diversi film:

- **Film con meno scene**: The Menu (59 scene), Kung Fu Panda (60 scene), Ghostbusters (61 scene).
- **Film con più scene**: Star Wars: A New Hope (481 scene), Inside Out (381 scene), TGM FSS Cover_for Academy (343 scene).

Questi risultati riflettono perfettamente le **differenze di genere** presenti nel dataset: i film d'azione o di fantascienza tendono ad avere un numero di scene maggiori inserendo un ritmo e una rapidità molto alta mentre film intimisti tendono ad avere scene lunghe con cambiamento di scena molto minore. Un'altra differenza che si riflette è lo **stile registico**: un registra può avere la tendenza a un montaggio rapido con poche scene mentre altri potrebbero preferire sceneggiature più lunghe. Infine anche la **durata del film** è molto significativa: film "epici" (Lord of the Rings, Interstellar) hanno naturalmente più scene.

5.2 Analisi delle Distribuzioni Aggregate

Dopo aver presentato i risultati ottenuti dall'analisi aggregata nella sezione precedente, in questa sezione procederemo a svolgere un'analisi per interpretare, fare deduzioni, comprendere le motivazioni sottostanti ai risultati osservati e valutarli verificando se sono coerenti con le convenzioni narrative e produttive del cinema contemporaneo.

5.2.1 Interno vs Esterno: interpretazione dei risultati

La predominanza di scene interne (59.3%) rispetto alle scene esterne (36.8%) rappresenta uno dei risultati più interessanti e contemporaneamente il più preciso fornito da CAST. La motivazione legata a questa affermazione è dovuta al fatto che CAST riesce in maniera molto efficace a estrapolare questa informazione **contenuta nel copione**. Viceversa, per le altre dimensioni analizzate non è così: i luoghi dettagliati, la fase oraria del giorno e la stagione dell'anno sono dimensioni che hanno molti meno riferimenti espliciti nel copione.

Il rapporto di circa **3:2 in favore degli interni** non è causale, al contrario mette in luce precise scelte produttive e narrative consolidate nell'industria cinematografica.

Motivazioni produttive

Se guardiamo il punto di vista di una casa cinematografica, scegliere una scena interna offre vantaggi non indifferenti:

Controllo ambientale Gli ambienti controllati permettono agli sceneggiatori e ai registi di gestire in maniera molto più precisa e accurata l'illuminazione, eliminado

variabili imprevedibili come il passaggio di nuvole, cambiamenti di luce naturale o presenza di fenomeni metereologici che potrebbero complicare di molto la registrazione. Tutto ciò provoca risparmio in termini di tempo, in quanto non è necessario attendere il verificarsi di condizioni ottimali.

Efficienza economica Le riprese in edifici interni sono genericamente più economiche. Non richiedono permessi straordinari, gestione del traffico e controllo dei passanti sul luogo della scena. Proprio per queste ragioni è possibile ridurre la dimensione della crew al minimo necessario e anche la logistica sarà la più semplice possibile. Possiamo concludere che per produzioni con budget limitati, massimizzare il numero di scene interne potrebbe provocare un grosso risparmio in termini economici.

Qualità tecnica Il controllo acustico in luoghi chiusi è sicuramente migliore. Nei luoghi all'aperto sono presenti costanti rumori di sottofondo indisiderati, i quali richiedono un maggior lavoro in fase di post produzione per essere eliminati. Mantenendo interne le scene si può risparmiare tempo e costi anche sotto questo punto di vista.

Motivazioni narrative

Dal punto di vista narrativo, la prevalenza di interni è dovuta a diversi fattori.

Intimità emotiva Le scene svolte in un luogo chiuso offrono la possibilità di aumentare l'intimità, il dialogo e lo sviluppo dei personaggi. La narrazione cinematografica, prevalentemente, si basa sulle relazioni interpersonali. Possiamo dire quindi che lo svolgimento di scene interne favorisce lo sviluppo della narrazione del film in maniera più semplice.

Significato del 3.8% `UNKNOWN`

CAST è in grado di etichettare esclusivamente scene interne o esterne. Tuttavia, in fase di sviluppo, mi sono imbattuto raramente in scene che contenevano l'informazione `INT/EXT` o viceversa segnalando quindi che la scena si evolveva nel tempo: passando da un luogo interno ad uno esterno. Posso quindi concludere che quel 3.8% è dovuto principalmente a questo fatto.

5.2.2 Ambienti prevalenti nel cinema contemporaneo

L'analisi semantica relativa all'individuazione delle ambientazioni precise delle scene fornisce risultati interessanti sulla geografia narrativa del cinema moderno.

Ambiente	Occorrenze	Percentuale
Unknown	6.036	38.7%
Urban	2.637	16.9%
Suburban	1.627	10.4%
Sea	1.273	8.2%
Fantasy	1.110	7.1%
Space	1.077	6.9%
Mountain	925	5.9%
Rural	755	4.8%
Desert	175	1.1%
Totale	**15.615**	**100%**

Tabella 5.1: Distribuzione delle scene per ambiente semantico

Dominanza urbana/suburbana (27.3%)

Da quanto emerge dall'analisi, si evidenzia come la combinazione di ambienti urban (16.9%) e suburban (10.4%) costituisce più di un quarto delle scene etichettate con successo da CAST, confermando che **il cinema contemporaneo è prevalentemente urbano**.

Urban (16.9%) L'ambiente urbano, al di fuori della categoria `UNKNOWN`, è il più rappresentato. Questo dato evidenzia in maniera netta:

- **rilevanza culturale**: oggi la maggiorparte della popolazione mondiale vive in aree urbane, di conseguenza queste ambientazioni sono facilmente riconoscibili e relazionabili;
- **generi urbani dominanti**: Thriller, crime, drammi romantici, commedie sono naturalmente ambientati in città;
- **iconografia cinematografica**: la presenza di Hollywood marca ancora di più il divario fra la categoria `urban` e le restanti. Infatti città come New York, Los Angeles, Londra, Parigi sono diventate "personaggi" cinematografici a sé stanti.

Inoltre, Le analisi fornite confermano una forte tendenza urbanistica (come ci si aspettava) dai seguenti film: *Sex and the City*, *Joker* (Gotham/New York distopica), *The Pianist* (Varsavia durante la WWII), *Inception*, *Spider-Man 2*, *Fantastic Beasts* (New York anni '20)

Suburban (10.4%) L'ambiente suburbano rappresenta il contesto "normale" della popolazione americana/occidentale. La suburbia è lo spazio dove la classe media vive e passa la maggiorparte del tempo. Molto spesso è l'ambientazione di partenza da cui molti protagonisti iniziano il loro viaggio (letterale o metaforico).

Ambienti naturali significativi (25.0%)

La somma di `sea` (8.2%), `mountain` (5.9%), `rural` (4.8%), `desert` (1.1%) costituisce un quarto del corpus, indicando una presenza robusta di ambientazioni naturali nonostante la dominanza urbana. Una nota di merito va fatta per la categoria `desert`: lo scarso risultato ottenuto delle scene ambientate nel deserto denota la grande difficoltà logistica di riprese desertiche e la minore rilevanza narrativa di questo ambiente nelle generazioni moderne rispetto per esempio alle grandi città.

Fantascienza e Fantasy (14.0%)

L'unione dei risultati di `space` **(6.9%)** e `fantasy` **(7.1%)** rappresenta circa un settimo dell'intero dataset, testimoniando la notà popolarità dei generi fantascientifici e fantasiosi. Il genere `space` è sostenuto da contratti commerciali di successo che garantiscono presenza costante di ambientazioni spaziali nel cinema mainstream. Inoltre il genere `fantasy` ha conosciuto una rinascita negli ultimi venticinque anni grazie al successo del *Signore degli Anelli*, *Harry Potter* e universi Marvel, giustificando quindi la presenza significativa di questi due luoghi nella classificazione di CAST.

L'enigma dell'`UNKNOWN` (38.7%)

La percentuale più alta di scene appartiene alla categoria `UNKNOWN` (38.7%). Tuttavia, questa classificazione non rappresenta un fallimento per CAST, ma mette in luce delle caratteristiche inequivocabili dei copioni moderni:

- **scene generiche**: molto spesso nei film moderni molte scene sono ambientate appositamente in ambientazioni neutre;
- **ambientazioni ibride**: CAST attualmente non è in grado di etichettare altri luoghi che sono utilizzati di frequente nelle sceneggiature dei film;
- **mancanza di elementi descrittivi**: gli screenplay professionali sono spesso minimali nelle descrizioni e quindi per CAST è difficile etichettare dei luoghi.

5.2.3 Pattern temporali nelle narrazioni cinematografiche

L'analisi dei periodi del giorno e delle stagioni dell'anno evidenziano pattern interessanti sulle scelte temporali narrative.

Fasi del giorno

Equilibrio day/night Se per quanto riguardava la classificazione ambientale le scene interne primeggiavano in maniera abbastanza netta, l'analisi temporale sulle fasi della

Periodo	Occorrenze	Percentuale
Day	4.473	28.6%
Night	4.228	27.1%
Morning	3.495	22.4%
Unknown	2.297	14.7%
Evening	1.122	7.2%
Totale	**15.615**	**100%**

Tabella 5.2: Distribuzione delle scene per periodo del giorno

giornata fornisce una sostanziale parità tra i periodi `Day` (28.6%) e `Night` (27.1%). Le principali deduzioni che possiamo ottenere sono che il `day` viene tipicamente scelto perchè è più economico girare alla luce naturale, c'è una maggiore sicurezza sui set esterni e c'è una migliore visibilità per l'audience. Le motivazioni che spingono invece alle scene `night` derivano dalla necessità per specifici generi (horror, thriller, noir) di creare l'atmosfera, la tensione e i giochi di luce drammatici richiesti.

`Morning` **significativo (22.4%)** Il periodo mattutino è sorprendentemente ben rappresentato. Questo perchè:

- **inizio narrativo**: molte scene si aprono con il protagonista che si sveglia;
- **transizione temporale**: "The next morning..." è un dispositivo narrativo comune;
- **atmosfera specifica**: la mattina esprime nuovi inizi, speranza e routine quotidiana.

`Evening` **sottoutilizzato (7.2%)** Il periodo serale è il meno rappresentato in CAST. Questo può derivare da molteplici spiegazioni, come per esempio il fatto che questa fase della giornata è quella che dura di meno, è più difficile distinguere `evening` rispetto al giorno e alla notte e infine è tipicamente la fase della giornata che tradizionalmente viene più trascurata nei film.

`UNKNOWN` **moderato (14.7%)** La percentuale di periodi che CAST non è riuscito a classificare è significatamente minore rispetto all'analisi ambientale. Questo indica che i copioni tendono a specificare il momento della giornata in cui avviene la scena e che le *keyword* temporali sono più frequenti e standard.

Distribuzione stagionale complessa

Da come viene evidenziato nella tabella 5.3, la classificazione stagionale presenta i pattern più complicati e quindi un alta percentuale di `UNKNOWN` (40.2%).

Stagione	Occorrenze	Percentuale
Unknown	6.275	40.2%
Autumn	3.694	23.7%
Winter	2.375	15.2%
Spring	1.672	10.7%
Summer	1.599	10.2%
Totale	**15.615**	**100%**

Tabella 5.3: Distribuzione delle scene per stagione

`Autumn` **dominante (23.7%)** Un risultato inaspettato è sicuramente il fatto che la stagione dell'autunno sia quella più rappresentata in CAST. Questo risultato parzialmente sorprendente può essere spiegato da:

1. **Artefatto WordNet/keyword**: termini come "fall" (ambiguo: cadere/autunno), "leaves" (foglie ma anche "lascia"), "harvest" possono generare falsi positivi.
2. **Tematiche narrative**: l'autunno è tipicamente associato a emozioni come il declino, la malinconia e la riflessione. Inoltre l'autunno può essere visto come la transizione tra estate e inverno oppure tra la gioventù e la maturità.

`Winter` **distintivo (15.2%)** L'inverno è ben identificato grazie a keyword molto specifiche (snow, ice, christmas, cold).

`Spring` **e** `Summer` **sottorappresentate (10.7% e 10.2%)** Diversamente da quello che ci potevamo aspettare le stagioni più calde non sono quelle che hanno etichettato più scene. Questo fenomeno può essere spiegato dal fatto che sono presenti *keywords* meno distintive, è più difficile distinguere semanticamente l'estate dalla primavera e infine perchè la stagionalità è spesso implicita piuttosto che esplicita.

`UNKNOWN` **elevato (40.2%)** La classificazione stagione è senza dubbio la **classificazione più complicata** per CAST. Questo perchè:

- **molti film sono "stagione-agnostici"**: moltissimi film urbani contemporanei raramente specificano e fanno intendere la stagione;
- **scene interne**: non forniscono indicazioni stagionali visibili;
- **climi non stagionali**: film ambientati in deserti, tropici o spazio non hanno stagioni marcate;
- **temporalità narrative compresse**: film che coprono pochi giorni non enfatizzano la stagione.

5.2.4 Correlazioni tra ambientazione e periodo temporale

Se osserviamo più nel dettaglio i risultati ottenuti, si evidenziano alcune correlazioni interessanti tra l'ambientazione e il periodo temporale.

Correlazioni periodi del giorno - ambientazione

Urban + Night Esiste una correlazione forte tra ambientazioni urbane e scene notturne. Questo pattern è osservabile in diversi film fra cui:

- thriller urbani: *Fight Club, Joker*;
- noir: *The Matrix*;
- crime: *Fantastic Beasts*.

La città ambientata di notte, infatti, è un'abitudine cinematografica consolidata nel tempo che suscita sensazioni particolari e uniche negli occhi dello spettatore.

Suburban + Day Le ambientazioni suburbane tendono ad essere prevalentemente diurne. La spiegazione dietro a questo pattern potrebbe essere che scene di questo genere vengono tipicamente inserite per rappresentare la quotidianità diurna delle persone nella quale si svolge la vita familiare. Alcuni film che adottano questo pattern sono *Barbie, Pretty Woman, Little Women*.

Sea + Morning/Day Le scene ambientate in luoghi marittimi sono anch'esse tipicamente diurne. In questo caso la spiegazione è più semplica e logica: girare scene sul mare di notte è complesso e pericoloso ed inoltre l'immagine dell'alba in spiaggia è considerata un'icona che fa riferimento ad un nuovo inizio.

Correlazioni stagionali

Winter + Mountain/Rural Esiste una forte associazione tra gli ambienti montani e la stagione invernale. Questo è un risultato che ci aspettavamo infatti tipicamente la montagna viene visitata l'inverno. Inoltre le ambientazioni innevate molto spesso vengono usate per creare l'atmosfera ideale del Natale.

Summer + Sea Un altro risultato atteso è la forte relazione che c'è fra l'estate e il mare. Non a caso la stagione balneare si svolge durante l'estate e tanti film che vogliono basare la loro narrazione sulle vacanze replicano questa ambientazione.

Autumn + Suburban/Rural Infine, un altro risultato preventivabile è la correlazione esistente fra la stagione autunnale e l'ambiente suburbano. Infatti, l'autunno è spesso associato alla suburbia americana tramite eventi come il *back to school*, *Halloween*, e il *Thanksgiving*. Inoltre, l'autunno viene utilizzato molto spesso anche per ricreare ambientazioni drammatiche nella campagna.

Implicazioni narrative

Queste correlazioni non sono casuali ma riflettono:

- **convenzioni di genere**: ogni genere ha le sue combinazioni preferite di spazio-tempo;
- **simbolismo visivo**: certe combinazioni hanno significati culturali consolidati (città notturna = pericolo, spiaggia estiva = libertà);
- **necessità produttive**: in molte circostanze è necessario utilizzare certi pattern spazio temporali per motivi economici;
- **aspettative del pubblico**: gli spettatori hanno aspettative inconsce su quali ambientazioni "appartengono" a quali momenti.

Il sistema CAST, identificando queste distribuzioni, conferma la propria capacità di catturare pattern narrativi reali del cinema contemporaneo.

5.3 Validazione del Sistema tramite Analisi Singolo Film

La **validazione dei dati** prodotti da un sistema software è uno dei requisiti più importanti da soddisfare quando si progetta una soluzione come CAST. Per questo, in questa sezione verrà svolta una prima e approssimata validazione sui risultati ottenuti per determinare la bontà e la qualità del sistema implementato, traendo così delle deduzioni finali.

5.3.1 Metodologia di validazione

La metodologia di validazione scelta per verificare l'accuratezza di CAST si basa su una prima **valutazione svolta su un singolo film**. Essa è un'analisi limitata rispetto a un golden standard, tuttavia è un buon punto di partenza per trarre conclusioni precise e accurate. Nella pratica la metodologia prevede le seguenti fasi:

1. **selezione del caso di studio**: in questa prima fase è stato selezionato il film *Pirati dei Caraibi - La maledizione della prima luna*. Il film non è stato scelto casualmente, infatti presenta dei risultati molto interessanti da essere analizzati;

2. **esecuzione di CAST**: verrà eseguita un'istanza di CAST con solo il copione del film scelto all'interno del dataset in modo da ottenere le analisi aggregate specifiche;

3. **costruzione dei dati di confronto**: è necessario ottenere i dati di confronto il più affidabile possibili in modo da certificare l'accuratezza del risultato;

   (a) **analisi assistita da AI**: in primo luogo è stato chiesto ad un LLM di produrre le stesse analisi implementate in CAST, passandogli in input il copione del film che verrà utilizzato nella validazione, ottenendo così risultati aggregati sul film;

   (b) **verifica manuale con fonti esterne**: successivamente i risultati ottenuti sono stati confrontati consultato Wikipedia alla pagina relativa del film [19], nella quale sono presenti tutte le varie location del film;

4. **confronto e analisi degli scostamenti**: in questa fase avviene il vero e proprio confronto fra i dati ottenuti dal sistema e quelli ottenuti dall'ai, in modo da poter eseguire delle analisi sugli eventuali scostamenti presenti. Solo al termine di questa fase potremmo fare considerazioni finali sull'accuratezza di CAST.

5.3.2 Caso studio: Pirati dei Caraibi - La maledizione della prima luna

Analisi Interno vs Esterno

La tabella 5.4 presenta il confronto tra i risultati di CAST e i dati di confronto per la dimensione interno/esterno.

Sistema	INT	EXT	UNKNOWN	Totale
CAST	46 (33.3%)	91 (65.9%)	1 (0.7%)	138
Dati di confronto	46 (33.6%)	91 (66.4%)	0 (0.0%)	137

Tabella 5.4: Confronto INT/EXT per Pirates of the Caribbean

I risultati mostrano un'accuratezza praticamente perfetta. La singola scena classificata come `UNKNOWN` rappresenta probabilmente un caso limite con indicatori ambigui `INT/EXT` nel copione originale.

Analisi Ambientale

La tabella 5.5 confronta la distribuzione ambientale rilevata da CAST con i dati di confronto.

Ambiente	CAST	Dati di confronto
Sea	61 (44.2%)	65 (47.4%)
Urban	7 (5.1%)	27 (19.7%)
Mountain	17 (12.3%)	1 (0.7%)
Fantasy	2 (1.4%)	14 (10.2%)
Suburban	-	6 (4.4%)
Desert	1 (0.7%)	1 (0.7%)
Unknown	46 (33.3%)	18 (13.1%)

Tabella 5.5: Confronto distribuzione ambientale per Pirates of the Caribbean

L'analisi ambientale evidenzia risultati più complessi. CAST identifica correttamente `sea` come ambiente dominante. Questo **conferma la capacità del sistema di riconoscere l'ambientazione marittima** centrale del film.

Emergono tuttavia alcune discrepanze significative. Per la categoria `urban` CAST rileva solo 7 scene contro le 27 espresse dai dati di confronto. Questa sottostima potrebbe derivare dal fatto che molte scene portuali di Port Royal, pur essendo ambientate in una città coloniale, vengono descritte nei copioni con enfasi sugli elementi marittimi piuttosto che urbani, confondendo il classificatore semantico.

Un falso positivo rilevante riguarda la categoria `mountain`, dove CAST identifica 17 scene contro una sola espressa nei dati di confronto. Questo errore è probabilmente causato da keyword ambigue come "rock", "cliff" o "cave" che appaiono nelle descrizioni delle grotte dell'Isla de Muerta, erroneamente associate all'ambiente montano anziché a quello fantastico/sotterraneo.

La sottostima della categoria `fantasy` (2 scene contro 14) rappresenta un altro limite significativo. Le scene nelle grotte maledette, elemento centrale della trama e della componente fantastica del film, non vengono riconosciute adeguatamente dal sistema. Questo suggerisce che le keyword fantasy utilizzate da CAST potrebbero essere troppo orientate verso il genere fantasy tradizionale (castelli, maghi) piuttosto che verso il fantasy avventuroso/piratesco.

La percentuale elevata di scene `unknown` indica che circa un terzo delle scene **non presenta keyword sufficientemente distintive per permettere una classificazione affidabile**.

Analisi Temporale - Periodo del Giorno

La tabella 5.6 mostra il confronto per i periodi della giornata.

Periodo	CAST	Dati di confronto
Day	50 (36.2%)	54 (39.4%)
Night	66 (47.8%)	67 (48.9%)
Morning	16 (11.6%)	10 (7.3%)
Evening	6 (4.3%)	5 (3.6%)
Unknown	-	1 (0.7%)

Tabella 5.6: Confronto periodi del giorno per Pirates of the Caribbean

L'analisi temporale del periodo del giorno mostra un'accuratezza elevata, infatti CAST rileva 66 scene notturne contro le 67 dei dati di confronto. Analogamente, le scene diurne mostrano una differenza contenuta del 3.2%.

La predominanza di scene notturne è una scelta stilistica deliberata che contribuisce all'atmosfera misteriosa e avventurosa del film. La notte amplifica la dimensione fantastica della maledizione dei pirati scheletrici e crea tensione nelle scene d'azione e nei combattimenti navali.

La leggera sovrastima delle scene `morning` potrebbe derivare da keyword come "dawn" o "sunrise" utilizzate per descrivere scene all'alba, periodo transizionale che CAST classifica come mattina mentre i dati di confronto potrebbero considerare ancora notte.

Analisi Temporale - Stagioni

L'analisi stagionale evidenzia una **discrepanza totale** tra CAST e i dati di confronto. Se essi classificano tutte le 137 scene come `unknown`, CAST distribuisce le classificazioni tra tutte e quattro le stagioni, con una prevalenza di `autumn`.

Questa discrepanza non rappresenta un errore tecnico di CAST, ma evidenzia un **limite intrinseco del copione**. Pirates of the Caribbean è ambientato nei Caraibi tropicali, un'area geografica dove non esistono stagioni marcate come quelle delle zone temperate. Il clima caraibico è relativamente costante durante tutto l'anno, senza le transizioni tipiche primavera-estate-autunno-inverno. Le classificazioni stagionali di CAST sono quindi falsi positivi causati da keyword ambigue.

Questo caso di studio dimostra che **l'analisi stagionale è la dimensione più problematica** per CAST, particolarmente per film con ambientazioni climaticamente neutre o tropicali dove la stagionalità non è narrativamente rilevante.

Capitolo 6

Conclusioni e sviluppi futuri

Lavorare in prima persona alla realizzazione di CAST è stato molto stimolante e arricchente, concedendomi la possibilità di migliorare le mie skills di progettazione, sviluppo e ricerca. Il progetto è stato suddiviso in quattro fasi indipendenti creando così un sistema modulare e generico: obiettivo primario della tesi. Inizialmente, la fase di **estrazione** ha subito presentato le prime criticità: la volontà è stata quella di creare un sistema che accettasse input eterogenei in modo da ampliare quanto più possibile il bacino di utenza. I due formati che attualmente CAST supporta sono file PDF e file HTML, due dei principali formati più diffusi in ambito cinematografico.

Una volta terminata l'estrazione del testo, CAST salva il contenuto di ogni copione all'interno di un file testo. Quest'ultimo è il file di input che viene utilizzato in fase di **conversione**. Essa è parte del cuore della logica di CAST avendo di fatto un compito fondamentale: analizzare il testo e costruire un copione strutturato nel linguaggio TEI-XML. Attraverso questo linguaggio di markup, viene suddiviso tutto il flusso di testo estratto in base alla sua semantica costruendo diverse sezioni tra cui: scene, speech, singole battute, informazioni di contesto e speaker. La sfida più grande di questa fase è stata la realizzazione di un flusso di esecuzione deterministico che, con le adeguate regole e priorità, riuscisse a etichettare correttamente le varie sezioni del testo dei file in formato TEI-XML.

Terminata la conversione, la fase successiva è l'**analisi** vera e propria dei copioni costruiti. Durante l'analisi, CAST analizza ogni scena di ogni copione estraendo informazioni relative all'ambientazione della scena e a fattori temporali della stessa. Anche questa funzionalità è stata particolarmente impegnativa: è stato complicato trovare una logica che fosse in grado di analizzare correttamente i copioni, minimizzando falsi positivi e gestendo correttamente recall e precision. Proprio per questo, in fase di sviluppo, sono state implementate diverse versioni della stessa funzionalità, nate con l'obiettivo di migliorare i limiti di quelle precedenti. Alla fine, l'approccio basato sull'utilizzo di dizionari espansi con filtri si è rilevato il più indicato. Infine, i risultati sono stati riportati all'interno di una pagina web. Essa presenta una **dashboard con**

diversi grafici per ognuna delle diverse dimensioni analizzate, dando la possibilità ai vari esperti di dominio di eseguire le loro analisi.

I risultati ottenuti da CAST sulle analisi relative alle ambientazioni e a fattori temporali delle scene sono da interpretare con particolare attenzione. Osservando una dimensione per volta e confrontando i risultati ottenuti rispetto al modello comparativo, inizialmente notiamo l'assoluta precisione del sistema nell'etichettare una scena come interna o esterna. Come già spiegato, ciò si verifica perchè l'informazione relativa all'internalità o esternalità della scena è esplicita per ognuna di esse. Di conseguenza, CAST ha semplicemente il compito di estrapolare questa informazione e etichettarla in quanto tale.

Proseguendo nella valutazione, CAST etichetta discretamente bene anche il luogo specifico di ambientazione della scena soprattutto per quanto riguarda le ambientazioni più generiche come `urban`, `suburban` e `sea`. Per altre ambientazioni più specifiche l'accurattezza è molto minore, a causa probabilmente dell'eccessiva generalità di *keywords* utilizzate nei dizionari. Guardando invece gli elementi temporali delle scene, l'analisi risulta non accurata. Se osserviamo i risultati ottenuti nella classificazione del periodo del giorno, CAST sembra etichettare con adeguata accuratezza le scene. In realtà, le criticità emergono se si esegue una riflessione importante in merito al tema. CAST non è in grado di tenere traccia della transizione, di un'evoluzione passando da una scena all'altra: per il sistema ogni scena è a sè stante. Ciò significa che se le informazioni, relative ai fattori temporali, sono descritte nella scena x e la scena $x+1$ è un proseguimento della scena precedente nella quale non sono fornite ulteriori informazioni a riguardo, CAST etichetterà in maniera sbagliata quella scena.

Per quanto riguarda l'etichettatura delle stagioni le problematiche sono diverse: il modello comparativo evidenzia l'impossibilità di catalogare le stagioni nelle scene. Queste difficoltà non sono da attribuire a delle scarse capacità di CAST, piuttosto alle assenze molto frequenti di elementi "stagionali" all'interno dei copioni. In conclusione possiamo dire che le analisi riportate dal sistema non sono eccezionali, soprattutto se si considera il fatto che la validazione è preliminare e poco approfondita; tuttavia l'obiettivo non era quello di creare il sistema più preciso possibile ma di creare un sistema modulare che consentisse agli esperti del settore di eseguire analisi personalizzate su un personale dataset di film.

I punti di forza di CAST rispetto ad altri sistemi presenti sul mercato sono molteplici. Innanzitutto, la gestione completa della pipeline consente all'utilizzatore di CAST di non dover usufruire di servizi di terze parti per svolgere le sue analisi: deve solo preoccuparsi di fornire il dataset di film in input al sistema.

Successivamente, anche la scalabilità la e modularità sono un punto a favore di CAST: esso è in grado di analizzare una grande mole di film molto velocemente ed inoltre è facilmente modulabile alle esigenze dell'esperto di dominio.

Inoltre, CAST adotta lo standard TEI-XML. Questa adozione gli consente di risultare attraente agli occhi degli esperti, essendo lo standard TEI molto utilizzato sia in ambito cinematografico ma adatto anche alla scrittura di libri, articoli e copioni di teatro. Ancora, CAST dispone di un algoritmo di parsing complesso in grado di etichettare molte informazioni diverse inserite nei copioni.

Infine, l'eterogeneità del formato di input e l'implementazione di una dashboard user-friendly permettono al sistema di raggiungere un panorama di utenti molto ampio.

Per quanto riguarda i punti deboli, oltre ai risultati delle analisi analizzati nei paragrafi precedenti, CAST dispone di un grosso limite in fase di parsing dei copioni. Per capire il problema è necessario prima capire come è strutturata una scena generica in un copione. Ognuna disponde di una prima serie di informazioni di *location*, successivamente delle informazioni di contesto generale (*stage*), i vari speech e infine (opzionalmente) ulteriori informazioni di contesto. Il limite sta nel fatto che CAST momentaneamente non è in grado di capire quando, all'interno di una scena, viene interrotto uno speech e iniziata una sezione *stage*. Il risultato errato che otteniamo è che tantissime scene contengono del testo etichettato come battute quando in realtà non lo sono. Da sottolineare è che questo limite non ha condizionato in alcun modo le analisi implementate attualmente in CAST, ciò perchè il sistema non esegue analisi sulle battute ma solamente sulle informazioni di *location* e *stage*.

Infine osserviamo quelli che sono i possibili sviluppi futuri in CAST. Sicuramente sarà necessario risolvere il problema di parsing appena descritto in modo tale che, qual ora vengano eseguite analisi sulle battute, il risultato ottenuto sia accurato. Inoltre, potrebbe essere molto interessante cambiare approccio di analisi introducendo sistemi intelligenti sostituendo la logica dei dizionari. Ancora, è opportuno aggiornare CAST in modo tale che sia in grado di tenere traccia dell'evoluzione e del progresso tramandato scena per scena. Questa nuova funzionalità permetterebbe di avere analisi molto più accurate per quelle dimensioni che sono per definizione transitorie (periodo del giorno e stagioni). Per rendere sempre più attaente CAST a nuovi esperti di dominio è importante ampliare quanto più possibili le analisi, introducendone di nuove riguardanti tematiche non ancora trattate come ad esempio i personaggi e le relazioni tra di essi. Infine, sulla base delle nuove analisi, dovranno essere apportate delle modifiche alla dashboard inserendo nuovi grafici appropriati.

Bibliografia

[1] Script Slug. Script slug. URL https://www.scriptslug.com/.

[2] Screenplays.io. Screenplays.io - free screenplays for education. URL https://screenplays.io/.

[3] IMSDb. The Internet Movie Script Database (IMSDb). URL https://imsdb.com/.

[4] Springfield! Springfield! Springfield! springfield! - tv show episode scripts. URL https://www.springfieldspringfield.co.uk/.

[5] IMDb.com. Imdb - internet movie database. URL https://www.imdb.com/.

[6] Jermain Kaminski, Michael Schober, Oleksandr Zastupailo, and Cesar Hidalgo. Moviegalaxies - social networks in movies, Dec 2012. URL http://moviegalaxies.com/.

[7] Tableau Software. Tableau public - about, 2025. URL https://public.tableau.com/app/about.

[8] Benjamin Nemceff. Data movies - movie analysis. Tableau Public Visualization, 2025. URL https://public.tableau.com/app/profile/benjamin.nemceff/viz/DataMovies_17010360628300/MovieAnalysis.

[9] Rukshan. Complete timeline of ridley scott movies - top movies timeline. Tableau Public Visualization, 2025. URL https://public.tableau.com/app/profile/rukshan/viz/CompleteTimelineofRidleyScottMovies/TimelineofRidleyScottsTopMovies.

[10] WriterDuet. Screenplayiq + pitchtrailer - visualize your story from a new perspective. URL https://screenplayiq.com/.

[11] ScriptReader.ai. Scriptreader.ai - ai-powered screenplay analysis. URL https://scriptreader.ai/.

[12] Leonard Richardson. beautifulsoup4. Python Package Index (PyPI). URL https://pypi.org/project/beautifulsoup4/.

[13] pdfminer.six contributors. pdfminer.six documentation (latest), 2025. URL https://pdfminersix.readthedocs.io/en/latest/.

[14] Wikipedia contributors. Text encoding initiative — wikipedia, . URL https://it.wikipedia.org/wiki/Text_Encoding_Initiative.

[15] Python Software Foundation. xml.etree.elementtree — the elementtree xml api. Python Documentation. URL https://docs.python.org/3/library/xml.etree.elementtree.html.

[16] Steven Bird, Ewan Klein, and Edward Loper. Natural language toolkit (nltk), . URL https://www.nltk.org/.

[17] Princeton University. Wordnet - a lexical database for english, 2010. URL https://wordnet.princeton.edu/.

[18] Steven Bird, Ewan Klein, and Edward Loper. Nltk howto: Wordnet interface. NLTK Documentation, . URL https://www.nltk.org/howto/wordnet.html.

[19] Wikipedia contributors. List of locations in pirates of the caribbean — wikipedia, . URL https://en.wikipedia.org/wiki/List_of_locations_in_Pirates_of_the_Caribbean.

Ringraziamenti

Desidero ringraziare innanzitutto il Prof. Angelo Di Iorio per la sua disponibilità e per il prezioso aiuto che mi ha fornito durante questa esperienza.

Un ringraziamento speciale va ai miei genitori, Sabrina e Cristian, e alle mie sorelle Aurora e Vittoria. Grazie a voi ho potuto intraprendere questo percorso che mi ha portato al raggiungimento del tanto atteso traguardo finale. Mi avete sempre dato massima libertà, avete sempre creduto in me sostenendomi in ogni situazione e di questo vi sono molto grado.

Non posso non ringraziare anche i miei zii Cristina e Umberto, e mia nonna Maria che mi hanno sempre sostenuto non solo emotivamente ma supportandomi anche economicamente.

Infine, un profondo e sentito ringraziamento va alla mia ragazza Alice. Si amore, si, proprio tu, che da quando sei entrata nella mia vita hai sempre messo in primo piano di più il sottoscritto piuttosto che te stessa. Anche in questa occasione ti sei comportata a tal proposito, svolgendo un compito determinante nella buona riuscita di questa tesi. Ricordo ancora le svariate volte che ci siamo messi insieme alla ricerca di nuovi copioni di film da inserire nel dataset: io alla tastiera pronto a scrivere e tu che da grande esperta mi suggerivi i titoli dei tuoi film preferiti, dei film di maggior successo e dei cult che secondo te non potevano mancare. Inoltre, anche in fase di scrittura sei stata fondamentale. Senza di te amore mio questo progetto non sarebbe stato la stessa cosa.

Da quando ci siamo conosciuti mi hai reso una persona migliore: mi hai insegnato tantissime cose che porto sempre con me nel mio bagaglio personale. Tu mi hai sempre valorizzato al massimo come mai nessuno prima aveva fatto facendo risaltare i miei lati positivi e di questo te ne sono veramente grato.

Infine, colgo l'occasione per ringraziarti ancora una volta della persona che sei: la più bella, dolcissima, solare, carina con tutti, disponibile ad aiutare il prossimo e mille altri aggentivi che fanno di te una persona speciale. Tu per me amore sei tutto, ti ammiro come persona e sono profondamente orgoglioso di averti al mio fianco. Grazie infinite, amore della mia vita.