

Università degli Studi di Milano Bicocca

Dipartimento di Informatica, Sistemistica e Comunicazione

Corso di Laurea Magistrale in Informatica



Labelling di cluster di modelli architetturali

Relatore:

Leonardo Mariani

Candidato:

Marco Di Capua

Correlatore:

Maria Teresa Rossi

ANNO ACCADEMICO 2025/2026

Indice

Introduzione	1
1 Modelli Architetturali	5
1.1 Architetture software	5
1.2 Model-driven Engineering e Architecture Description Languages	7
1.3 Architecture Analysis and Design Language	9
1.3.1 Component	9
1.3.2 Feature	11
1.4 OSATE	11
1.5 Clustering di modelli architetturali	14
2 Tecniche di summarization	16
2.1 Tecniche di sintesi del testo	16
2.2 Approcci adottati in letteratura	18
2.3 Term Frequency - Inverse Document Frequency	20
2.4 Latent Dirichlet Allocation	22
3 Labelling dei cluster	25
3.1 Estrazione delle informazioni	27
3.2 Preprocessing	28
3.2.1 Tokenization	29
3.2.2 Alphanumericals	29
3.2.3 Stopwords	30
3.2.4 Lemmatization	30
3.3 Labelling	31
3.3.1 TF-IDF labelling	31
3.3.2 LDA labelling	33
4 Applicazione delle metodologie	36
4.1 Base di conoscenza e lavori pregressi	36

4.2	Impostazione del progetto	37
4.3	Estrazione delle informazioni dai modelli	39
4.3.1	Analisi della base di conoscenza	39
4.3.2	Estrazione degli elementi chiave dai modelli	43
4.3.3	Selezione dei modelli idonei	45
4.4	Preprocessing dei dati	48
4.4.1	Tokenization e prime normalizzazioni	48
4.4.2	Rimozione di caratteri non alfanumerici	49
4.4.3	Rimozione delle stopwords	49
4.4.4	Lemmatization e normalizzazioni finali	50
4.5	Generazione delle etichette	53
4.5.1	TF-IDF	54
4.5.2	LDA	59
5	Analisi dei risultati	64
5.1	Ground truth	64
5.2	Cosine Similarity	67
5.3	Precision, Recall e F1 score	70
	Conclusioni	76

Elenco delle figure

1.1	Rappresentazione delle possibili astrazioni del sistema in MDE.	8
1.2	Tipologie di componenti AADL. Tratto da [7]	10
1.3	Rappresentazione di un sistema di sicurezza in OSATE.	13
2.1	Esempio di topic individuati da LDA. Tratto da [3].	24
3.1	Fasi che portano alla generazione delle etichette.	26
3.2	Rappresentazione grafica di un modello AADL con OSATE.	28
4.1	Distribuzione dei modelli all'interno dei cluster.	40
4.2	Rappresentazione grafica di un modello AADL di smart parking.	41
4.3	Rappresentazione testuale di un modello AADL di smart parking.	42
4.4	Conteggio dei nomi degli elementi estratti divisi per tipologia.	43
4.5	Classifica dei 25 nomi più ricorrenti per le componenti.	44
4.6	Classifica dei 25 nomi più ricorrenti per le feature.	44
4.7	Classifica dei 25 nomi più ricorrenti per le connection.	45
4.8	Percentuale dei modelli idonei rispetto al numero totale di modelli.	46
4.9	Distribuzione dei modelli idonei all'interno dei cluster.	47
4.10	Classifica dei 25 token più ricorrenti per i nomi di file AADL.	49
4.11	Classifica dei 25 token più ricorrenti per i nomi di file AADL dopo aver rimosso i caratteri non alfanumerici.	50
4.12	Classifica dei 25 token più ricorrenti per i nomi di file AADL dopo aver rimosso le stopwords.	51
4.13	Classifica dei 25 token più ricorrenti per i nomi di file AADL dopo il preprocessing.	52
4.14	Classifica dei 25 token più ricorrenti per i nomi di feature dopo il preprocessing.	52
4.15	Classifica dei 25 token più ricorrenti per i nomi di component dopo il preprocessing.	53
4.16	Andamento degli score TF-IDF per le parole ottenute dai nomi dei file dei modelli.	56

4.17 Andamento degli score TF-IDF per le parole ottenute dai nomi delle componenti e delle feature dei modelli.	56
4.18 Andamento dei valori di perplexity al variare del numero di topic ottenuti dai nomi dei file dei modelli.	60
4.19 Andamento dei valori di perplexity al variare del numero di topic ottenuti dai nomi delle componenti e delle feature dei modelli.	60
4.20 Distribuzione dei topic per ogni cluster	62
5.1 Valore delle similarità coseno per cluster ottenuta confrontando ground truth e etichette di TF-IDF.	69
5.2 Valore delle similarità coseno per cluster ottenuta confrontando ground truth e etichette di LDA.	69
5.3 Valori di precision e recall per cluster ottenuti confrontando ground truth e etichette di TF-IDF.	73
5.4 Valori di precision e recall per cluster ottenuti confrontando ground truth e etichette di LDA.	73
5.5 Boxplot dei valori di precision, recall e F1 per le etichette di TF-IDF e LDA.	75

Abstract

I modelli architetturali rappresentano una risorsa preziosa nell’ambito della progettazione di nuovi sistemi, poiché consentono di riutilizzare delle soluzioni già validate attraverso una rappresentazione standardizzata delle componenti principali di un sistema, consentendo in questo modo di affrontare con maggiore efficienza le sfide progettuali, minimizzando il rischio di errori e ottimizzando le risorse. Tuttavia, con l’aumento della complessità dei sistemi e la crescente quantità di modelli disponibili, diventa fondamentale poterli organizzare in modo efficiente. La mancata catalogazione e strutturazione dei modelli architetturali rende difficile la gestione di grandi collezioni, rallentando la selezione dei più adatti per specifiche esigenze progettuali e aumentando il rischio di errori.

Questo elaborato si propone di affrontare tali difficoltà utilizzando degli approcci basati su tecniche automatiche per l’etichettatura di cluster di modelli architetturali, al fine di renderli facilmente consultabili attraverso una rappresentazione sintetica del loro contenuto.

Il progetto sperimenta l’utilizzo delle tecniche di TF-IDF e LDA, applicate alle informazioni contenute nel nome e nelle relative componenti e feature di modelli AADL, facenti parte di un dataset costituito da 1070 elementi, al fine di individuare delle parole chiave da utilizzare per l’etichettatura. I risultati ottenuti dalla conduzione di questo esperimento sono frutto di un’attenta analisi preventiva del contenuto dei modelli, nonché di una selezione e rielaborazione accurata delle informazioni in essi contenute. La valutazione dei risultati, attraverso l’utilizzo di una ground truth e di diverse metriche di misurazione, ha dimostrato l’efficacia dei metodi proposti nel raggiungimento dello scopo prefissato.

Il contributo di questo progetto risiede nella sua capacità di semplificare e velocizzare l’individuazione dei modelli architetturali, permettendo agli sviluppatori e architetti del software di selezionare facilmente i modelli più adatti alle proprie esigenze progettuali, grazie alla loro etichettatura automatica e alla rappresentazione sintetica dei contenuti.

Introduzione

I modelli architetturali sono strumenti fondamentali nella progettazione di sistemi complessi, poiché forniscono una rappresentazione astratta e dettagliata delle componenti principali di un sistema, delle loro interazioni e dei vincoli che ne governano il comportamento. Uno dei principali linguaggi standardizzati utilizzati per la rappresentazione dei modelli architetturali è il linguaggio **Architecture Analysis and Design Language** (AADL), che consente di modellare sia le caratteristiche hardware che le caratteristiche software di un sistema evidenziandone le interazioni. La capacità di rappresentare in modo chiaro sia gli aspetti funzionali che quelli non funzionali, come le prestazioni, la sicurezza e l'affidabilità, rende questo linguaggio particolarmente adatto alla progettazione e analisi di sistemi *embedded* in tempo reale, tipici dell'ambito dell'avionica, dei sistemi automobilistici e dell'automazione industriale.

L'utilizzo di questi modelli risulta essere sempre più frequente ed utile per gli sviluppatori e architetti del software, poiché essi consentono di affrontare in modo più agevole attività complesse come la progettazione, la validazione e l'ottimizzazione di sistemi. L'utilizzo di soluzioni già validate come punto di partenza per lo sviluppo di nuovi sistemi accelera notevolmente i tempi richiesti per lo sviluppo, diminuisce il rischio di errori e facilita la collaborazione interna tra team di sviluppo attraverso la standardizzazione e la modularizzazione delle componenti del sistema.

Alcune importanti società di sviluppo software, come ad esempio Red Hat¹ e SAP², hanno iniziato a condividere architetture di riferimento utili allo sviluppo di applicazioni in diversi domini; tuttavia, ad oggi esistono poche risorse condivise che individuano delle collezioni di modelli architetturali organizzate e catalogate al fine di consentire un'agevole fruizione del contenuto del sempre crescente numero di tali modelli.

Lo scopo di questa tesi è sviluppare un approccio per l'etichettatura automatica dei cluster di modelli architetturali descritti in linguaggio AADL, utilizzando come input di partenza dei cluster generati da precedenti lavori di ricerca. In particolare, il lavoro di tesi dei colleghi di cui questa ricerca si fa carico, ha permesso di identificare gruppi di modelli architetturali simili attraverso l'utilizzo di tecniche di clustering avanzate che combinano l'uso di metriche

¹<https://www.redhat.com/en/blog/application-enterprise-architecture>

²<https://docs-eam.leanix.net/docs/reference-catalog>

di similarità strutturale e semantica. L'approccio per generare le etichette adottato in questo elaborato si articola in diverse fasi:

1. **Analisi dei modelli AADL** presenti nel dataset iniziale e identificazione dei contenuti informativi più adatti all'etichettatura.
2. **Preprocessing dei dati** selezionati al fine di normalizzarne il contenuto per rendere più agevole l'applicazione di tecniche di labelling automatico.
3. Utilizzo delle tecniche **TF-IDF** e **LDA** per individuare le parole chiave da utilizzare come etichette dei cluster.

Al termine del processo di generazione e attribuzione delle etichette ai cluster, segue una valutazione dei risultati ottenuti attraverso il calcolo della **cosine similarity** e delle misure di **precision**, **recall** e **F1**. Al fine di calcolare queste metriche è stata generata manualmente una *ground truth* che attribuisce ad ogni cluster un set di etichette. L'analisi di queste metriche mostra come entrambe le tecniche adottate ottengano dei risultati soddisfacenti, lasciando però spazio a margini di miglioramento attraverso l'ampliamento della base dati iniziale di modelli architetturali, l'utilizzo di una ground truth più completa e l'adozione di tecniche di labelling che fanno uso di approcci astrattivi come i *GPT*.

Struttura della tesi

L'elaborato si articola nei seguenti capitoli:

- **Capitolo 1 - Modelli architetturali:** Inquadramento riguardante lo scopo dei modelli architetturali nella progettazione di sistemi. Introduzione del linguaggio di modellazione AADL e descrizione del tool OSATE utilizzato per la generazione e l'analisi dei modelli. Sintesi delle tecniche di clustering applicabili ai modelli architetturali concentrandosi su quelle adottate per il clustering di modelli AADL.
- **Capitolo 2 - Tecniche di summarization:** Secondo capitolo introduttivo che analizza le tecniche di summarization utilizzate in letteratura e definisce i metodi TF-IDF e LDA utilizzati nel progetto.
- **Capitolo 3 - Labelling dei cluster:** Questo capitolo descrive in maniera dettagliata le metodologie che sono state implementate per manipolare il contenuto dei modelli AADL

al fine di individuare delle parole chiave da utilizzare come etichette da attribuire ai cluster.

- **Capitolo 4 - Applicazione delle metodologie:** Il capitolo introduce la base di conoscenza iniziale di modelli AADL e dei cluster che li contengono. A seguito dell'estrazione e preprocessing delle informazioni chiave dai modelli, si mostra l'applicazione delle metodologie di labelling definite nel capitolo precedente e le scelte progettuali adottate per la generazione delle etichette.
- **Capitolo 5 - Analisi dei risultati:** Capitolo conclusivo che mostra il processo di validazione delle etichette generate dalle tecniche TF-IDF e LDA. Una volta descritto il processo di definizione della ground truth manuale, si procede al calcolo di diverse metriche di confronto tra le etichette per valutare le prestazioni dei modelli di etichettatura automatica.

1. Modelli Architetturali

Questo capitolo introduce il concetto di **architettura del software**, definendone i punti cardine e focalizzandosi sul possibile riutilizzo delle componenti che costituiscono un sistema. Viene poi presentato l'approccio ingegneristico **Model-Driven Engineering** (MDE), che si basa sul concetto di astrazione per la rappresentazione dei modelli, e i linguaggi formali **Architecture Description Languages** (ADL) progettati per la descrizione dell'architettura dei sistemi. In particolare, si approfondisce il linguaggio **Architecture Analysis and Design Language** (AADL) e il tool di supporto **OSATE**, utilizzato per sviluppare modelli utilizzando questo linguaggio. Infine, si propone un accenno alle tecniche di **clustering di modelli architettuali**, come la similarità strutturale e similarità semantica, illustrandone i vantaggi derivanti dal loro utilizzo.

1.1 Architetture software

L'architettura del software è un aspetto fondamentale dell'ingegneria del software che, attraverso una rappresentazione astratta delle componenti e delle loro interazioni, sia reciproche che con l'ambiente esterno, consente di definire un sistema software ad alto livello in maniera chiara e semplice. Lo scopo delle architetture del software è segmentare il sistema in moduli autonomi ed evidenziare come questi siano progettati per soddisfare determinati requisiti e obiettivi in termini di prestazioni, scalabilità, sicurezza, manutenibilità, ecc..

Le porzioni di sistema in un'architettura del software sono rappresentate tramite **modelli architetturali**, i quali, facendo uso di diagrammi di componenti, sequenza e flusso, sono in grado di documentare e comunicare le scelte progettuali adottate. Queste rappresentazioni consentono una comunicazione chiara ed efficace delle prime decisioni di design verso gli *stakeholder*, permettendo di velocizzare la fase iniziale della progettazione di un sistema software, nota come *fase architettonica*.

In letteratura, sono state fornite diverse definizioni di architettura del software che si sono evolute nel tempo, a seguito degli studi condotti in questo campo. Alcune delle definizioni più rilevanti sono le seguenti:

- "L'architettura del software è una rappresentazione astratta, o modello, di un sistema

software in termini di una struttura che consiste in una raccolta di elementi insieme alle relazioni tra di essi per raggiungere gli obiettivi di progettazione del software e per manifestare un determinato insieme di proprietà progettuali del sistema. I dettagli degli elementi e delle relazioni sono nascosti e sostituiti con entità computazionali astratte, chiamate rispettivamente componenti e connettori. ”[20]

- *”L’architettura riguarda la selezione degli elementi architettonici, le loro interazioni e i vincoli su quegli elementi e le loro interazioni necessari per fornire un quadro nel quale soddisfare i requisiti e fungere da base per la progettazione (dettagliata)”[13]*
- *”L’architettura del software comprende non solo una raccolta di componenti, connessioni e vincoli di sistemi e software, ma raccoglie anche le dichiarazioni dei bisogni degli stakeholder e produce una giustificazione che dimostra che i componenti, le connessioni e i vincoli definiscono un sistema che, se implementato, soddisfarebbe la raccolta delle dichiarazioni dei bisogni degli stakeholder del sistema.”[1]*

In un contesto di architettura software, un *componente* è un’unità modularizzata del sistema che può essere facilmente riutilizzata in diversi progetti o contesti, senza dover essere progettata nuovamente o riscritta. Questo approccio, noto come **riutilizzo di componenti software**, è una caratteristica fondamentale nell’architettura del software che offre numerosi vantaggi, tra cui:

- **Riduzione dei costi e dei tempi di sviluppo:** Il riutilizzo di componenti già esistenti permette di evitarne la riprogettazione da zero per ogni nuovo progetto, velocizzando la fase di sviluppo e migliorando la qualità complessiva del sistema [2].
- **Maggiore affidabilità e qualità:** I componenti testati in precedenza sono più stabili e sicuri, riducendo il rischio di errori [16].
- **Flessibilità e adattabilità:** La possibilità di sostituire o modificare un componente, essendo modulare, senza influenzare altre parti del sistema, risulta particolarmente utile in ambienti dinamici, come quelli dei sistemi embedded [2].
- **Facilitazione della collaborazione e standardizzazione:** Il riutilizzo di componenti comuni consente una migliore collaborazione tra i vari team di sviluppo, facilitando la comunicazione e il coordinamento tra le diverse parti del progetto [4].

1.2 Model-driven Engineering e Architecture Description Languages

L'uso dell'astrazione per rappresentare i modelli all'interno del processo di sviluppo del software è una delle caratteristiche distintive del **Model-Driven Engineering** (MDE). In questo approccio, i modelli sono generati a diversi livelli di astrazione, consentendo di distogliere l'attenzione dai dettagli implementativi e focalizzarsi sulle componenti principali del sistema e sulle loro interazioni. Questo permette di eseguire simulazioni e analisi in fase preliminare, prima che il sistema venga effettivamente sviluppato [15]. I modelli, inoltre, possono essere trasformati automaticamente in codice eseguibile o in altre rappresentazioni necessarie per il sistema, riducendo così il rischio di errori umani e accelerando notevolmente il processo di sviluppo [17].

I livelli di astrazione utilizzati da MDE per la descrizione di un sistema, mostrati in Figura 1.1, sono:

- **Computation Independent Model (CIM)**: il modello descrive il sistema in modo indipendente dalla tecnologia o piattaforma adottata, concentrandosi solo sugli aspetti funzionali e sui requisiti del sistema.
- **Platform Independent Model (PIM)**: il sistema viene descritto dal modello in modo indipendente dalla piattaforma tecnologica su cui sarà realizzato, focalizzandosi sugli aspetti architetturali e di design, come le interazioni tra componenti e le logiche di business.
- **Platform Specific Model (PSM)**: ultimo livello di astrazione che descrive il sistema in modo specifico per una particolare piattaforma tecnologica o un ambiente di esecuzione. Vengono quindi indicati gli aspetti legati all'implementazione che consentono di simulare il comportamento del sistema.

Per ottenere rappresentazioni a diversi livelli di astrazione, come quelle appena descritte, è necessaria l'adozione di strumenti e linguaggi adeguati che permettano di ridurre la complessità dei sistemi, facilitando la gestione e la rappresentazione dei modelli. A tal proposito, si propongono i linguaggi ADL.

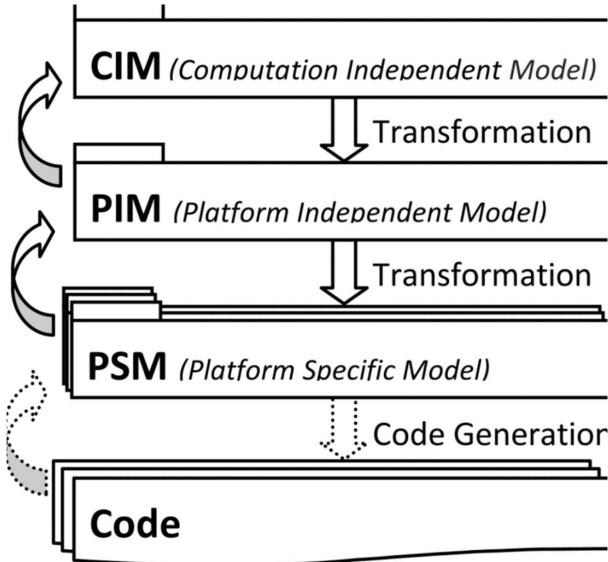


Figura 1.1: Rappresentazione delle possibili astrazioni del sistema in MDE.

Gli **Architecture Description Languages** (ADL) sono linguaggi formali progettati per descrivere, documentare e analizzare l’architettura di sistemi software. L’obiettivo di un ADL è fornire un linguaggio strutturato e standardizzato per rappresentare le componenti principali di un sistema, le loro interazioni e i vincoli che le regolano. Questi linguaggi possono essere utilizzati per generare delle *“viste architettoniche”* [4], ossia rappresentazioni di porzioni di architettura che si concentrano su diverse prospettive, come le interazioni tra moduli, le componenti fisiche, le prestazioni o la sicurezza. Le rappresentazioni architettoniche generate da un ADL offrono un livello superiore di granularità nella rappresentazione architettonica rispetto al classico utilizzo del *Unified Modeling Language* (UML), che è ampiamente utilizzato per rappresentare la struttura statica (tramite diagrammi di classe) e quella dinamica (tramite diagrammi di sequenza e attività) di un sistema, ricoprendone l’intero ciclo di vita. Esistono diversi linguaggi di descrizione dell’architettura che si specializzano in determinati contesti, tra cui:

- **AADL (Architecture Analysis and Design Language)**: linguaggio sviluppato per descrivere principalmente sistemi embedded in tempo reale, con un forte focus sull’analisi delle prestazioni e sulla gestione delle risorse.
- **Synergy**: ADL progettato per supportare la modellazione e la verifica delle architetture di sistemi distribuiti, concentrandosi sulle interazioni tra i componenti e la loro integrazione.

- **Meta-H**: linguaggio estendibile e specializzato nella progettazione di architetture software gerarchiche, adatto per rappresentare sistemi complessi e modulari.

1.3 Architecture Analysis and Design Language

L'**Architecture Analysis and Design Language** (AADL) [7] è un linguaggio standardizzato sviluppato dalla *Society of Automotive Engineers* (SAE) per la modellazione e l'analisi di sistemi embedded in tempo reale, tipici nell'ambito dell'avionica, automotive e automazione industriale. La notazione è stata progettata come un linguaggio di base estensibile, con semantiché ben definite e una presentazione sia grafica che testuale. Basato su altri ADL quali **Meta-H** e **UML 2.0**, AADL si concentra sulla descrizione di componenti software e hardware e sulle loro interazioni, con l'obiettivo di supportare non solo la modellazione dei sistemi, ma anche l'analisi delle prestazioni e la gestione delle risorse. I meccanismi di estensione consentono di introdurre proprietà specifiche per analisi architetturali aggiuntive, relative ad altri attributi di qualità come affidabilità e sicurezza, oppure di adottare nuovi approcci di analisi.

1.3.1 Component

Per modellare l'architettura di un sistema, sia lato software che hardware, AADL utilizza tre categorie di componenti, come riassunto in Figura 1.2, di cui segue una breve descrizione:

Application software

Gli elementi di questa categoria si concentrano sulla descrizione del comportamento in *runtime* di un sistema e del relativo codice sorgente. Le componenti appartenenti a questa categoria sono:

- **Thread**: unità schedulabile di esecuzione concorrente del codice.
- **Thread Group**: rappresenta un'unità composita per organizzare i thread in gruppi.
- **Data**: dati statici e tipi di dati utilizzabili dal codice.
- **Process**: spazio protetto di indirizzamento.
- **Subprogram**: unità di codice sequenziale eseguibile.

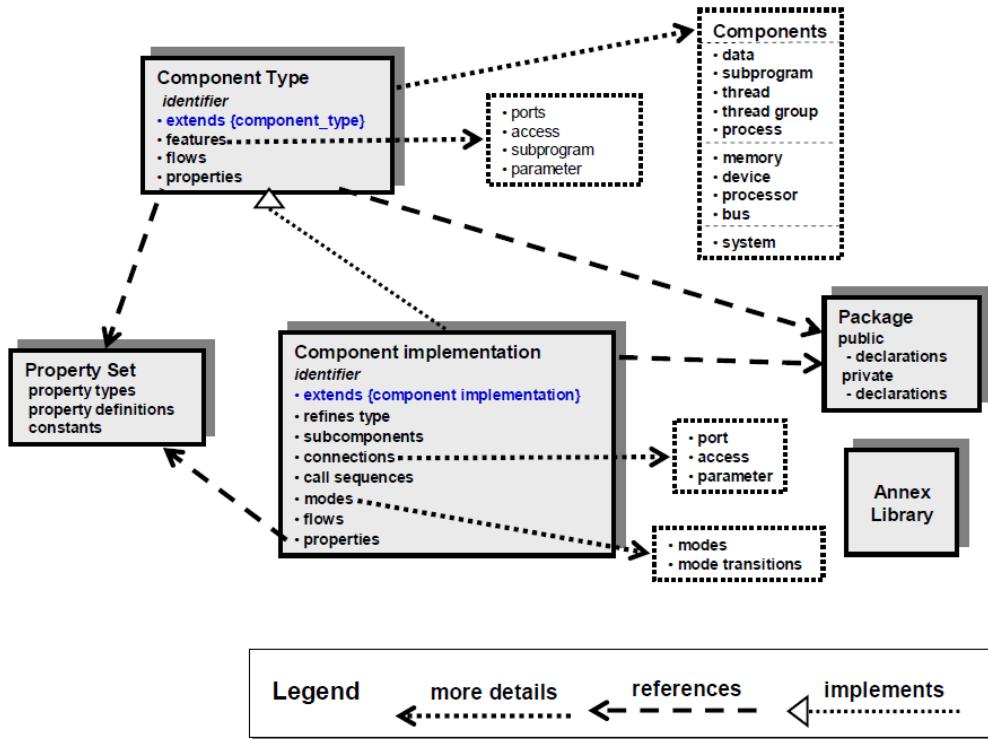


Figura 1.2: Tipologie di componenti AADL. Tratto da [7]

Execution platform

Questa categoria rappresenta i componenti hardware di un sistema. Le *execution platform* possiedono proprietà predefinite, come il tempo di scambio dei *thread* o il protocollo di pianificazione dell'utilizzo dei processori; tuttavia, tramite le estensioni delle proprietà, è possibile introdurre nuovi tipi di pianificazione. Gli elementi di questa categoria sono:

- **Processor**: componente che esegue i thread.
- **Memory**: memoria che contiene il codice e i dati.
- **Device**: componente destinato ad interfacciarsi con l'ambiente esterno.
- **Bus**: elemento che consente la comunicazione tra gli altri componenti della execution platform.

Composite components

L'ultima categoria consiste in componenti composti da elementi sia hardware che software. In genere, si tratta di un componente chiamato **system**, che rappresenta una serie di componenti appartenenti sia alla tipologia execution platform sia alla tipologia application software.

Le astrazioni dei componenti sono suddivise in due elementi principali: *types* (tipi) e *implementations* (implementazioni). I *types* rappresentano l’interfaccia visibile esternamente del componente, mentre le *implementations* definiscono gli aspetti non pubblici, come il codice sorgente o le *subcomponents* che costituiscono il componente stesso [5]. Le componenti interagiscono attraverso interfacce definite, che consistono in flussi direzionali di dati, eventi e porte dati. È possibile definire connessioni fisiche tra porte e flussi logici attraverso catene di porte.

1.3.2 Feature

Per poter definire le modalità di interazione tra le componenti, AADL mette a disposizione le **Feature**. Una feature è un’entità che descrive un punto di accesso o una caratteristica di un componente che consente di stabilire connessioni o comunicazioni con altri componenti.

Le Feature in AADL si suddividono in diverse categorie, a seconda del tipo di interazione che si desidera rappresentare:

- **Data Port**: rappresenta un punto di accesso per il trasferimento di dati. Le porte di dati sono utilizzate per il flusso di dati unidirezionale tra i componenti, consentendo la comunicazione di stato o altre informazioni.
- **Event Port**: consentono la comunicazione asincrona di eventi tra i componenti. Gli eventi sono utilizzati per rappresentare cambiamenti di stato, segnali o condizioni particolari che attivano azioni all’interno del sistema.
- **Parameters**: definisce la trasmissione di valori di parametro tra componenti.
- **Component Access**: collega tra loro due componenti per effettuare trasmissione dati o chiamate a funzionalità specifiche.
- **Subprogram Calls**: consente di effettuare collegamenti con delle procedure o funzioni specifiche per la loro esecuzione.

1.4 OSATE

Uno dei più utilizzati tra i vari tool sviluppati per supportare l’utilizzo del linguaggio AADL è la piattaforma **Open Source AADL Tool Environment** (OSATE)[18]. Lo scopo principale di

OSATE è fornire agli sviluppatori e progettisti di sistemi un ambiente integrato per il supporto della progettazione e dell'analisi di sistemi sia hardware che software utilizzando AADL. Esso fornisce strumenti per la creazione, la visualizzazione e l'analisi di modelli AADL, ed è progettato per supportare tutte le fasi del ciclo di vita del software, dalla progettazione iniziale alla verifica delle prestazioni fino alla generazione di codice.

L'ambiente di sviluppo implementato da OSATE include diverse funzionalità, tra cui:

- **Creazione e modellazione di modelli:** Consente di creare modelli AADL utilizzando un editor di testo sensibile alla sintassi e un editor grafico sincronizzato. Il tool supporta la validazione dei modelli secondo tutte le regole di denominazione e legalità definite nello standard AADL. L'editor di testo fornisce modelli di codice, controllo della sintassi in tempo reale, completamento automatico del codice e proposte per correggere errori.
- **Generazione automatica del codice:** OSATE supporta la generazione automatica del codice a partire dai modelli AADL. Una volta che l'architettura è stata progettata e verificata, è possibile utilizzare OSATE per generare il codice di implementazione che può essere eseguito direttamente su piattaforme hardware target e su un'ampia scelta di sistemi operativi. Questo riduce i tempi di sviluppo e garantisce che il codice generato rispetti le specifiche dell'architettura.
- **Analisi delle prestazioni e verifiche:** OSATE supporta la schedulabilità dei processi, l'analisi della concorrenza tra i thread e la gestione delle risorse temporali, determinando se l'architettura progettata soddisfa i vincoli di tempo reali. Inoltre, è possibile eseguire verifiche formali, come la verifica della correttezza del comportamento dei componenti, assicurandosi che il sistema rispetti i requisiti non funzionali, come quelli relativi alla sicurezza e all'affidabilità.
- **Simulazione e integrazione:** Il tool consente anche la simulazione del comportamento di un sistema modellato o di una sua partizione tramite AADL, permettendo agli sviluppatori di testare la funzionalità del sistema in fase di progettazione, prima che venga implementato fisicamente. OSATE offre anche la possibilità di integrare i modelli AADL con altre tecnologie, migliorando la capacità di testare e simulare sistemi complessi.

La possibilità di creare e analizzare modelli AADL attraverso un editor grafico è una funzionalità fondamentale ai fini degli esperimenti condotti in questo elaborato. A tal proposito,

risulta utile fornire un esempio grafico mostrato in Figura 1.3 che rappresenta un sistema di sicurezza.

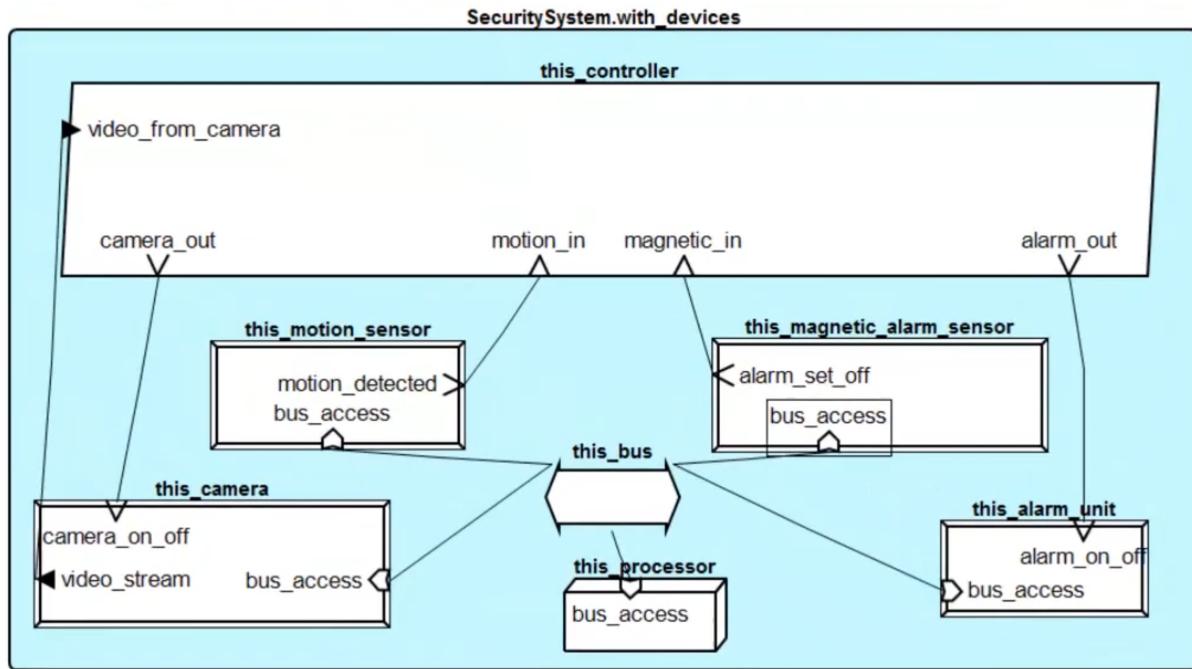


Figura 1.3: Rappresentazione di un sistema di sicurezza in OSATE.

Il componente descritto nell'immagine appartiene alla categoria *application software* e al suo interno contiene una serie di *subcomponents*:

- **`this_motion_sensor`:** sensore di movimento che segnala alla videocamera di iniziare a registrare e all'allarme di attivarsi in caso di intrusione.
- **`this_magnetic_alarm_sensor`:** sensore magnetico installato su una porta o finestra che segnala alla videocamera di iniziare a registrare e all'allarme di attivarsi in caso di intrusione.
- **`this_camera`:** videocamera che su segnalazione inizia a registrare e trasmettere il video al controller.
- **`this_alarm_unit`:** allarme che su segnalazione dei sensori si può attivare.
- **`this_controller`:** componente centrale del sistema di che si occupa di ricevere i segnali da altri dispositivi e di prendere decisioni basate su questi segnali.

- **this_bus**: componente di comunicazione che consente ai vari dispositivi di scambiare dati tra di loro. Funziona come un canale condiviso che collega tutti i componenti principali permettendo la trasmissione di segnali di dati, comandi e stato tra i dispositivi.
- **this_processor**: componente di calcolo del sistema, che si occupa dell’elaborazione delle informazioni. In un sistema embedded, il processore è responsabile dell’esecuzione di logiche complesse e della gestione delle comunicazioni tra i vari componenti.

Le comunicazioni tra le componenti appena descritte avvengono tramite l’utilizzo delle *feature*, in particolare in questo caso vengono utilizzate le *data ports* e *event ports*. Il sensore di movimento di questo sistema possiede una event port chiamata **motion_detected** che identifica il segnale di output emesso dal sensore quando percepisce un movimento. La videocamera, oltre a disporre di una event port che consente di ricevere il segnale per iniziare o terminare la registrazione, ha a disposizione una data port di nome **video_stream** che si occupa della trasmissione dei dati video al controller.

L’ultimo elemento grafico sono le *connections*, ovvero gli archi che graficamente collegano le porte delle componenti. Data la natura semplice di questo tipo di elementi ne si tralascia la descrizione in quanto la rappresentazione grafica risulta sufficientemente esaustiva una volta definite le componenti e le feature del sistema.

1.5 Clustering di modelli architetturali

I modelli architetturali sono strumenti essenziali nel processo di sviluppo dei sistemi software, in quanto consentono di rappresentare in modo astratto e dettagliato l’architettura di un sistema, evidenziando le componenti principali e le loro interazioni. La loro utilità risiede nella capacità di supportare attività cruciali come la progettazione, la validazione, l’analisi delle prestazioni e l’ottimizzazione dei sistemi. Tuttavia, con l’aumento della complessità dei sistemi, la gestione e l’utilizzo di numerosi modelli architetturali diventa una sfida. Raggruppare questi modelli in cluster omogenei permette agli sviluppatori di attingere facilmente a insiemi di modelli già validati e ottimizzati per specifiche esigenze progettuali, riducendo il tempo e gli errori associati alla creazione di nuovi modelli. Le tecniche di **clustering** dei modelli architetturali sono quindi un elemento fondamentale per la gestione di grandi collezioni di modelli, consentendo di organizzarle in gruppi, ognuno dei quali rappresenta una soluzione architetturale simile che può essere facilmente adattata a nuovi progetti.

Tra le più comuni tecniche di clustering di modelli architetturali vi sono quelle che fanno uso di metriche di **similarità strutturale** e **similarità semantica** per identificare gruppi di modelli simili. Queste metriche sono utili per confrontare i modelli in base a due aspetti distinti ma complementari:

- **Similarità strutturale:** Questa metrica si concentra sul confronto della struttura dei modelli, ossia sull’organizzazione dei componenti e sulle loro interazioni. La similarità strutturale tra due modelli è calcolata confrontando le loro rappresentazioni grafiche o formali, come ad esempio una rappresentazione attraverso un *grafo direzionale*. Un approccio comune per calcolare la similarità strutturale tra due modelli rappresentati attraverso dei grafi diretti è l’utilizzo del **Maximum Common Subgraph** (MCS), che identifica la parte condivisa dei due grafi e misura quanto le loro strutture si sovrappongano. Questo approccio è utile per misurare le somiglianze tra i componenti architetturali, come moduli, processi e connettori, e per determinare quanto siano simili le architetture in termini di struttura.
- **Similarità semantica:** Il secondo tipo di similarità, invece, si basa sul confronto dei contenuti e significati dei modelli, piuttosto che sulla loro struttura. La metrica di similarità semantica utilizza tecniche di **Natural Language Processing** (NLP) per confrontare i nomi dei nodi presenti nei grafi che rappresentano i modelli. Un esempio comune è l’uso della *similarità coseno* calcolata su dei vettori semanticici ottenuti attraverso modelli pre-addestrati come **FastText**. Questo approccio è particolarmente utile per misurare le somiglianze concettuali tra i modelli, anche quando le loro strutture fisiche sono differenti. Ad esempio, due modelli possono avere una struttura completamente diversa ma comunque condividere lo stesso scopo funzionale, e la similarità semantica permette di identifierli come simili.

L’utilizzo di queste due metriche può essere combinato per ottenere un approccio di **clustering ibrido** che tiene conto sia della similarità strutturale che di quella semantica dei modelli e delle loro componenti. Attraverso l’attribuzione di pesi ai diversi indici di similarità, è possibile configurare questo tipo di approccio in base al contesto specifico in cui viene applicato, ottenendo così dei risultati bilanciati.

2. Tecniche di summarization

Il capitolo fornisce una panoramica sulle **tecniche di summarization** e sulla loro utilità nell'ambito scientifico, descrivendone le tipologie e caratteristiche principali. Vengono poi presentati approcci presenti in letteratura che utilizzano queste tecniche al fine di estrarre delle parole chiave finalizzate all'individuazione di etichette. Infine, sono descritti nel dettaglio gli algoritmi di **Term Frequency-Inverse Document Frequency** (TF-IDF) e di **Latent Dirichlet Allocation** (LDA), che sono applicati nei capitoli successivi nel contesto del labelling di cluster di modelli architetturali.

2.1 Tecniche di sintesi del testo

A causa della mole in continua crescita di informazioni disponibili online, è diventato sempre più difficile per gli sviluppatori e i professionisti in ambito scientifico individuare e utilizzare efficacemente le risorse che potrebbero essere utili al loro lavoro. Infatti, la crescita esponenziale di testi, documenti tecnici, articoli scientifici e altre fonti di informazioni rende arduo estrarre contenuti pertinenti e rilevanti per un progetto specifico. Per affrontare questa problematica, i ricercatori hanno iniziato a sviluppare le **tecniche di summarization** come una soluzione efficace. Queste tecniche permettono di condensare informazioni da grandi volumi di testo, mantenendo intatti i concetti principali, e facilitano l'estrazione delle informazioni rilevanti, come le parole chiave.

Le tecniche di summarization mirano a ridurre e semplificare il contenuto testuale preservandone però i concetti fondamentali, consentendo di ottenere una rappresentazione più breve e mirata di un testo. Questa operazione è particolarmente utile in contesti come la gestione dei documenti, la ricerca di informazioni, e l'analisi dei dati. In ambito software, ad esempio, la summarization può essere utilizzata per estrarre parole chiave da documenti tecnici o modelli architetturali, permettendo agli sviluppatori di raccogliere rapidamente informazioni pertinenti per la progettazione di sistemi.

Le tecniche di summarization si dividono principalmente in due categorie: **tecniche estrattive** e **tecniche astrattive**.

Tecniche estrattive

Le tecniche estrattive si occupano di riassumere il testo contenuto in uno o più documenti selezionando porzioni, frasi o parole di particolare rilevanza al loro interno. L'importanza delle frasi estratte è ottenuta tramite l'analisi statistica e linguistica del testo complessivo, come la frequenza delle parole o il posizionamento di frasi chiave nel documento [12].

Tra le principali tecniche di summarization estrattive si hanno le tecniche basate sulla frequenza delle parole, che si concentrano sull'analisi di quante volte una parola appare nel testo. Più una parola appare frequentemente, maggiore è la sua importanza nel testo. Questi metodi assumono che le parole più frequenti siano anche quelle più rilevanti e, attraverso l'utilizzo di misurazioni come **TF-IDF**, attribuiscono un punteggio alle parole, consentendo di stilare una classifica delle parole di score più alto, e quindi più rilevanti. **Latent Dirichlet Allocation** (LDA) rappresenta una naturale evoluzione di questo approccio. A differenza di TF-IDF, che si concentra esclusivamente sulla frequenza dei termini, LDA introduce il concetto di topic: una serie di parole che appaiono frequentemente insieme e che identificano una tematica specifica all'interno del testo. In LDA, un documento è visto come una combinazione di diversi topic, ciascuno dei quali è rappresentato da un insieme di parole che ne definiscono il significato. Questo approccio consente una sintesi più profonda rispetto a TF-IDF, poiché identifica non solo le parole frequenti, ma anche i concetti latenti che meglio descrivono i temi principali trattati nel documento [3]. LDA, quindi, è in grado di estrarre temi dai documenti e di utilizzarli per creare riassunti tematici che riflettono meglio la struttura e il contenuto del testo, andando oltre la semplice selezione di parole chiave.

Esistono delle tecniche estrattive basate sui grafi, che trattano il testo come un grafo in cui ogni nodo rappresenta una frase o una parola, e gli archi tra i nodi indicano la similarità o la connessione tra le frasi. I metodi appartenenti a questa categoria, come **TextRank** e **LexRank**, si basano sull'idea che le frasi più centrali nel grafo sono le più importanti e pertinenti per il riassunto. Questi algoritmi utilizzano misure di centralità come la **PageRank** per identificare le frasi più significative, attraverso il concetto di importanza relativo alla posizione e alle connessioni delle frasi nel testo [10].

Tecniche astrattive

In contrasto alle tecniche precedenti, le tecniche astrattive adottano un approccio più complesso, il cui obiettivo è quello di generare un riassunto rielaborando il contenuto del testo originale.

Al fine di non perdere il significato del testo originale, ma di riformularlo in maniera concisa, si adottano modelli linguistici avanzati che possono comprendere il contesto e produrre un riassunto coerente [8]. Alcuni di questi modelli, come le **Recurrent Neural Networks** (RNN), si basano sull'utilizzo di reti neurali per apprendere una rappresentazione sequenziale del testo in input e generare un riassunto.

Una categoria molto popolare di modelli utilizzati per la generazione di riassunti astrattivi sono quelli basati su *transformers* come ad esempio il **Generative Pre-trained Transformer** (GPT) e **Bidirectional Encoder Representations from Transformers** (BERT)[6]. In generale, i modelli appartenenti a questa categoria si basano sull'utilizzo di *encoder* che si occupano di creare una rappresentazione contestualizzata dell'input e *decoder* che generano un riassunto basandosi sulla rappresentazione fornita dall'encoder.

2.2 Approcci adottati in letteratura

Il *clustering* è un processo che raggruppa documenti simili in base a caratteristiche comuni, tuttavia, senza un appropriato *labelling*, il significato e la rilevanza di ciascun cluster possono rimanere poco chiari, limitando l'efficacia del raggruppamento. Le tecniche di summarization si rivelano strumenti adatti per affrontare questa problematica, in quanto consentono di estrarre le parole chiave più significative da ciascun cluster di documenti.

L'applicazione di diverse metodologie per il clustering e labelling di documenti, indipendentemente dalla loro natura, è stata oggetto di numerose pubblicazioni scientifiche negli ultimi anni. Ad esempio, Nasim et al. [11] hanno proposto un approccio di labelling automatico di cluster di tweet in lingua urdu, utilizzando un metodo non supervisionato basato sul **noun phrase chunking**. Il clustering dei tweet è stato attuato attraverso la rappresentazione vettoriale degli stessi, seguita dall'applicazione dell'algoritmo **K-Means** per generare i cluster. Il processo di etichettatura inizia con il noun phrase chunking, ossia l'estrazione di sequenze di parole che formano concetti significativi nei documenti analizzati. Una volta estratti i *noun chunks*, è stato costruito un vocabolario di *n-grams*, ovvero sequenze di n parole consecutive. Il vocabolario, infine, è stato ordinato, individuando gli n-grams più importanti attraverso l'applicazione di **TF-IDF** per il calcolo del punteggio associato ai vettori che li rappresentano.

Nel lavoro di Weng et al. [19], è stato proposto un approccio per l'identificazione e visualizzazione di **topics** nelle pubblicazioni scientifiche, utilizzando modelli di linguaggio basati

su *transformer* e tecniche moderne di *document clustering*. L’obiettivo era identificare e visualizzare i temi emergenti all’interno di pubblicazioni scientifiche per migliorare il processo di ricerca di informazioni in domini come lo studio urbano e l’apprendimento automatico. Gli abstract dei documenti nella base dati sono stati rappresentati tramite **embedding GPT-3** per catturarne il significato semantico. Successivamente, è stato applicato l’algoritmo di clustering non supervisionato **Hierarchical Density-Based Spatial Clustering of Application with Noise** (HDBSCAN) per la generazione dei cluster. Le parole chiave degli abstract sono state estratte attraverso una tecnica avanzata di *tokenizzazione* del testo, che ne identifica gli n-grams, a cui ha fatto seguito l’applicazione dell’algoritmo **Maximal Marginal Relevance** (MMR) per la selezione delle parole chiave più rappresentative, massimizzandone la rilevanza e minimizzandone la ridondanza.

Nello studio di Pham et al. [14], è stato introdotto *TopicGPT*, un approccio che utilizza modelli di **Large Language Modeling** (LLM) per generare e assegnare topic in un corpus di testi. Lo scopo di questo framework è quello di andare a sostituire la classica rappresentazione dei topic, ottenuta attraverso l’utilizzo delle parole più frequenti, che spesso e volentieri generano un set di parole incoerenti tra di loro e difficili da interpretare. La generazione dei topic avviene tramite un processo iterativo che prevede una prima fase di generazione di topic, a partire da un campione di documenti utilizzando gli LLM. Successivamente, per ogni nuovo gruppo di documenti, l’LLM viene fornito con i topic precedentemente generati e i nuovi documenti. L’LLM aggiorna i topic esistenti o ne crea di nuovi, migliorando progressivamente la qualità e la specificità dei topic. Dopo ogni iterazione, i topic vengono analizzati e affinati per garantire che siano distinti, pertinenti e rappresentativi del contenuto del corpus. Questo può comportare la fusione di topic simili o la rimozione di quelli irrilevanti. Per assegnare i topic ai documenti, è stato utilizzato un prompt che fornisce i topic generati e un documento specifico, chiedendo all’LLM di associare il topic più pertinente. Inoltre, per ogni assegnazione, viene fornita una citazione dal documento che supporta l’assegnazione, migliorando la verificabilità del processo.

Nel contesto specifico dei modelli architetturali, il raggruppamento di modelli all’interno di cluster e l’etichettatura degli stessi contribuiscono a una gestione più organizzata e scalabile dei dati. Il clustering e il labelling sono particolarmente utili per gli architetti del software, poiché consentono di raggruppare modelli simili e di assegnare loro etichette pertinenti che riflettono le caratteristiche principali o le funzionalità del sistema rappresentato. Questo processo ren-

de possibile l'identificazione rapida di modelli già esistenti che soddisfano specifici requisiti progettuali, facilitando il loro riutilizzo in nuovi contesti.

Il lavoro proposto in questa tesi si presenta come un tassello da aggiungere al vasto mosaico di pubblicazioni scientifiche che trattano l'argomento del labelling di cluster, con un focus particolare sull'adozione degli algoritmi TF-IDF e LDA per derivare delle etichette da attribuire a cluster di modelli architetturali definiti tramite il linguaggio AADL. Questo approccio mira a ottimizzare la gestione dei modelli architetturali, facilitando il loro riuso e migliorando l'efficacia del processo di progettazione attraverso l'automazione del labelling.

2.3 Term Frequency - Inverse Document Frequency

Una tra le tecniche più utilizzate nell'ambito dell'elaborazione del linguaggio naturale e del *text mining* è la **Term Frequency - Inverse Document Frequency** (TF-IDF) [9]. Questa metodologia consente di valutare l'importanza di una parola all'interno di un documento rispetto a un intero corpus, bilanciando la frequenza di occorrenza locale con la rarità globale del termine. Il suo obiettivo è attribuire un punteggio a ciascuna parola in un documento, basato sia sulla sua frequenza all'interno di quel documento (Term Frequency) sia sulla sua rarità in tutti i documenti del corpus (Inverse Document Frequency). Questo approccio aiuta a identificare le parole chiave che meglio rappresentano il contenuto di un documento o di un insieme di documenti.

Il punteggio calcolato per una parola da TF-IDF è ottenuto dalla composizione di due elementi:

- **Term Frequency (TF)**: Misura la frequenza relativa di una parola all'interno di un documento. Più una parola è frequente in un dato testo, più è probabile che essa sia rilevante. Per calcolare questa misura basta contare le occorrenze di una parola in un documento e normalizzare la sua frequenza dividendola per il numero totale di parole presenti nel documento.
- **Inverse Document Frequency (IDF)**: Misura quantità di documenti contenti una determinata parola rispetto al numero totale di documenti disponibili. Questa misura consente di attribuire un peso minore alle parole che sono presenti in un maggior numero di documenti rispetto al totale.

Il calcolo del punteggio TF-IDF di una parola w_d può essere espresso con la seguente formula:

$$w_d = f_{w,d} \times \log\left(\frac{|D|}{f_{w,D}}\right)$$

dove:

- $f_{w,d}$ è la frequenza della parola w nel documento d ,
- $|D|$ è il numero totale di documenti nel corpus,
- $f_{w,D}$ è la frequenza della parola w nell'intero corpus D .

Esistono diverse situazioni che possono verificarsi per ogni parola, a seconda dei valori di $f_{w,d}$, $|D|$, e $f_{w,D}$.

Supponiamo che la dimensione del corpus $|D|$ sia quasi equivalente alla frequenza della parola nel corpus $f_{w,D}$. Se $1 < \log\left(\frac{|D|}{f_{w,D}}\right)$, anche se per un valore molto piccolo, allora w_d sarà minore di $f_{w,d}$ ma comunque positivo. Ciò implica che la parola w sia relativamente comune nell'intero corpus ma che mantenga comunque una certa importanza all'interno di D . Questa situazione è comune per parole estremamente frequenti come articoli, pronomi e preposizioni, che da sole non possiedono un alto contenuto informativo. Queste parole comuni ricevono così un punteggio TF-IDF molto basso, rendendole praticamente irrilevanti nella ricerca.

Infine, supponiamo che la frequenza di una parola in un documento $f_{w,d}$ sia grande, ma che il numero di documenti del corpus contenenti quella parola $f_{w,D}$ sia piccolo. In questo caso, $\log\left(\frac{|D|}{f_{w,D}}\right)$ sarà abbastanza grande, e quindi anche w_d risulterà grande. Questo è il caso che ci interessa maggiormente, poiché parole con un alto valore di w_d implicano che la parola w sia importante nel documento d ma non comune nell'intero corpus D . Una parola con queste caratteristiche viene considerata ad alto contenuto informativo per la rappresentazione di un documento e quindi riceve un alto punteggio TF-IDF.

La tecnica TF-IDF è semplice ed efficace, adatta per identificare parole rappresentative nei documenti e non richiede una conoscenza specifica del dominio del testo analizzato, in quanto si basa esclusivamente su misure statistiche di frequenza. Tuttavia, questa tecnica presenta alcune carenze, come ad esempio il fatto che due parole con lo stesso significato, ma lessicalmente diverse, vengono trattate come entità differenti tra loro. Per far fronte a questa problematica, spesso si fa uso di tecniche di preprocessing che normalizzano il testo prima di

applicare l’algoritmo TF-IDF, in modo tale da uniformare il più possibile le parole prima di calcolarne il relativo punteggio. Un altro punto debole di TF-IDF risiede nel fatto che non vengono considerate le relazioni semantiche tra le parole, o più in generale, non viene considerato il contesto nel quale una parola è inserita. Esistono tecniche più avanzate che trattano questo aspetto, come ad esempio l’uso di word embeddings, che catturano le relazioni semantiche tra le parole e migliorano la comprensione del contesto in cui appaiono, o modelli come il Latent Dirichlet Allocation (LDA) che riescono a individuare topic e correlazioni semantiche nei testi.

2.4 Latent Dirichlet Allocation

Il **Latent Dirichlet Allocation** (LDA) [3] è una delle tecniche più popolari per il *topic modeling*, che permette di ridurre la dimensionalità dei dati di testo, identificando argomenti o *topic* che rappresentano gruppi di parole correlate in modo significativo. L’obiettivo principale di LDA è identificare una serie di argomenti che spiegano in modo probabilistico la distribuzione delle parole in un corpus di documenti, consentendo di attribuire un’etichetta tematica a ciascun documento senza la necessità di un’etichettatura predefinita. A differenza delle tecniche basate su misure statistiche come TF-IDF, LDA si basa su un approccio probabilistico che assume che ogni documento sia una combinazione di diversi topic, e ogni topic sia a sua volta una distribuzione di parole.

LDA si basa su una struttura gerarchica bayesiana a tre livelli, dove ogni documento è rappresentato come un insieme di argomenti latenti e ogni argomento è caratterizzato da una distribuzione di probabilità sulle parole. La generazione di ogni parola nel documento avviene scegliendo un argomento, quindi selezionando una parola da una distribuzione multinomiale condizionata sull’argomento. LDA assume un processo generativo per ogni documento in un corpus, che consiste nei seguenti passaggi:

1. Si sceglie un numero N di parole appartenenti ad un documento da una **distribuzione Poisson**.
2. Si seleziona una distribuzione di argomenti (topic) θ da una **distribuzione Dirichlet**.
3. Per ognuna delle parole w_n appartenenti al documento si sceglie un **argomento** (topic) z_n da una distribuzione multinomiale condizionata su θ , e successivamente si seleziona una **parola** w_n da una distribuzione multinomiale condizionata su z_n .

I passaggi appena elencati mostrano come, inizialmente, per ogni documento nel corpus, LDA determini quante parole N verranno considerate, seguendo una distribuzione di Poisson, che è una distribuzione di probabilità discreta che modella eventi che accadono in un dato intervallo di tempo o spazio. Una volta stabilito il numero di parole per documento, LDA seleziona un insieme di probabilità θ per ogni documento, che rappresenta la combinazione dei topic presenti in quel documento. La distribuzione Dirichlet, utilizzata per generare θ , è particolarmente utile perché consente di modellare la probabilità di diversi topic all'interno di un singolo documento, tenendo conto che ogni documento può trattare più argomenti. Infine, il processo si conclude associando probabilisticamente ogni parola a uno dei topic individuati nello step precedente. Dopo aver assegnato il topic, una parola viene selezionata in base alla distribuzione di probabilità delle parole per quel topic. Questo processo avviene per ogni parola nel documento, e l'iterazione di questi passaggi consente a LDA di affinare la comprensione di quali parole siano correlate a ciascun topic.

Utilizzando questo approccio, LDA riesce, attraverso l'assegnazione iterativa e probabilistica delle parole ai topic, ad estrarre argomenti significativi e a rappresentare ogni documento come una miscela di questi, come mostrato in Figura 2.1. A partire dal testo in esempio, sono stati individuati i topic *"Arts"*, *"Budgets"*, *"Children"* e *"Education"*, ai quali sono state assegnate le parole che costituiscono il corpus del documento.

L'algoritmo adottato da LDA, per quanto efficace, presenta alcune debolezze che ne limitano l'efficacia in determinati contesti. Ad esempio, nel caso in cui i documenti che costituiscono il corpus siano particolarmente complessi o poco definiti, il numero di topic, che deve essere definito dall'utente prima di applicare il modello, può influenzare negativamente il comportamento dell'algoritmo. Nel caso in cui sia troppo basso, LDA tenterà di comprimere un numero maggiore di argomenti in pochi topic, con il rischio che topic diversi vengano uniti. Se invece il numero di topic è troppo alto, il modello avrà la tendenza a generare topic molto specifici che descrivono aspetti minori del corpus oppure che descrivono dei sotto-argomenti troppo simili tra loro. Anche la distribuzione di probabilità θ , che identifica la combinazione di topic per ogni documento, può causare delle alterazioni sul risultato finale in base alla natura dei documenti analizzati. Infatti, se un documento tratta principalmente un singolo argomento, questo avrà una probabilità molto più alta rispetto agli altri argomenti trattati e, conseguentemente, verranno estratte delle parole che possono essere associate maggiormente al tema principale. Se al contrario un documento copre molti temi con distribuzioni simili, si avranno numerosi

“Arts”	“Budgets”	“Children”	“Education”
NEW	MILLION	CHILDREN	SCHOOL
FILM	TAX	WOMEN	STUDENTS
SHOW	PROGRAM	PEOPLE	SCHOOLS
MUSIC	BUDGET	CHILD	EDUCATION
MOVIE	BILLION	YEARS	TEACHERS
PLAY	FEDERAL	FAMILIES	HIGH
MUSICAL	YEAR	WORK	PUBLIC
BEST	SPENDING	PARENTS	TEACHER
ACTOR	NEW	SAYS	BENNETT
FIRST	STATE	FAMILY	MANIGAT
YORK	PLAN	WELFARE	NAMPHY
OPERA	MONEY	MEN	STATE
THEATER	PROGRAMS	PERCENT	PRESIDENT
ACTRESS	GOVERNMENT	CARE	ELEMENTARY
LOVE	CONGRESS	LIFE	HAITI

The William Randolph Hearst Foundation will give \$1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Juilliard School. “Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important as our traditional areas of support in health, medical research, education and the social services,” Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center’s share will be \$200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York Philharmonic will receive \$400,000 each. The Juilliard School, where music and the performing arts are taught, will get \$250,000. The Hearst Foundation, a leading supporter of the Lincoln Center Consolidated Corporate Fund, will make its usual annual \$100,000 donation, too.

Figura 2.1: Esempio di topic individuati da LDA. Tratto da [3].

argomenti trattati con una conseguente grande varianza tra le parole estratte.

Per far fronte a queste problematiche ed attenuare le possibili complicazioni derivanti da esse, spesso si ricorre a tecniche di preprocessing che ripuliscono il testo prima di applicare LDA. Queste tecniche escludono le parole poco rilevanti, consentendo così a LDA di identificare meglio i topic. Un altro grande limite di LDA è che non possiede una comprensione semantica del testo che sta analizzando. Ciò significa che LDA non considera le relazioni contestuali tra le parole, come la sinonimia e la polisemia, causando quindi talvolta la creazione di topic distinti che trattano lo stesso argomento, oppure l’attribuzione di parole errate a topic che trattano argomenti totalmente differenti.

3. Labelling dei cluster

Il seguente capitolo illustra le metodologie adottate nel progetto per generare in modo automatico delle etichette da attribuire a dei cluster contenenti modelli architetturali descritti in linguaggio AADL. In particolare, si vuole generare delle etichette che riflettano le caratteristiche comuni dei modelli, come componenti simili e domini applicativi di appartenenza. Queste etichette dovrebbero rappresentare in modo sintetico e chiaro il contenuto dei cluster, facilitandone così la consultazione, l'analisi e il riutilizzo dei modelli.

Il progetto pone le basi sui risultati ottenuti in esperimenti precedenti che hanno portato alla generazione di cluster di modelli architetturali provenienti da repository pubbliche di **GitHub**. Il clustering di questi modelli, ottenuto attraverso la combinazione di metriche di similarità strutturale e semantica, ha individuato per ogni modello della base di conoscenza il relativo cluster associato. L'insieme dei modelli architetturali e l'associazione cluster-modello costituiscono il punto di partenza per il processo di etichettatura.

Il contenuto presentato in Figura 3.1 mostra come il processo si sviluppi in tre fasi sequenziali:

1. **Estrazione delle informazioni:** questa fase iniziale prevede l'estrazione delle informazioni rilevanti contenute nei modelli architetturali, utilizzate in seguito per derivare le etichette da attribuire ai cluster.
2. **Preprocessing dei dati:** il contenuto estratto dai modelli architetturali attraversa diverse fasi di normalizzazione, che rendono i dati più adatti al processo di labelling automatico.
3. **Labelling dei cluster:** in questa fase, si applicano le tecniche di labelling alle informazioni normalizzate, individuando le parole chiave da utilizzare come etichette per i cluster.

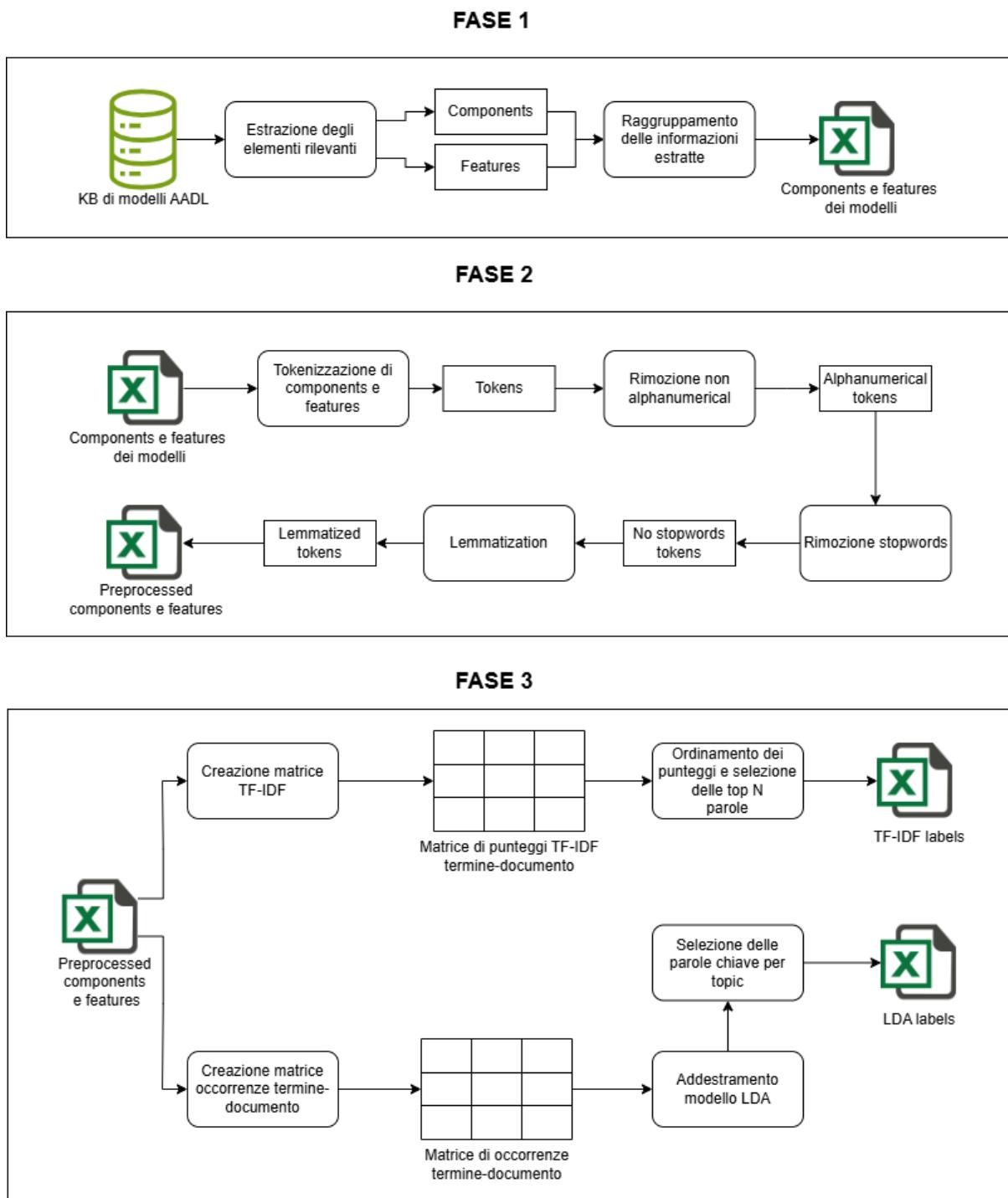


Figura 3.1: Fasi che portano alla generazione delle etichette.

3.1 Estrazione delle informazioni

La prima fase, propedeutica alla generazione delle etichette da attribuire ai cluster, prevede l'estrazione delle informazioni rilevanti per il processo di labelling. L'estrazione, modellazione e raggruppamento di specifiche informazioni contenute all'interno dei modelli, consentono di ottenere una solida base di partenza, facilitando le fasi di preprocessing e labelling. In questo modo, vengono isolate le parole che definiscono il contenuto dei modelli. Gli elementi estratti dai modelli sono:

- **Component:** i nomi delle componenti dei modelli consentono di identificare le principali funzionalità di un modello, sia lato hardware che software. L'astrazione dell'implementazione che caratterizza il linguaggio AADL consente di rappresentare funzionalità ad alto livello, come sensori, allarmi, gps, ecc. I nomi delle componenti, che identificano questi aspetti, forniscono informazioni utili al contesto di labelling.
- **Feature:** le feature rappresentano le interazioni tra le componenti di un modello. I loro nomi possono identificare le informazioni trasferite, gli eventi segnalati oppure la richiesta di esecuzione di azioni specifiche da un componente ad un altro.

All'interno dei modelli AADL della base di conoscenza sono presenti diversi altri elementi che però sono stati esclusi dal processo di estrazione, poiché non contribuiscono significativamente alla generazione di etichette utili per i cluster. Tra gli elementi esclusi vi sono:

- **Connection instances:** le informazioni relative alle connessioni tra componenti, pur essendo rilevanti per la descrizione del comportamento del sistema, sono state escluse poiché identificano i nomi delle componenti interconnesse andando quindi a duplicare il contenuto già estratto dalle stesse.
- **Modes:** i modes rappresentano stati operativi alternativi di un sistema o di un componente, ma la loro inclusione non è utile per la generazione di etichette che devono sintetizzare le caratteristiche principali del sistema. I modes tendono a rappresentare configurazioni temporanee o specifiche di esecuzione.
- **Flow specifications:** questi elementi descrivono i flussi di dati tra i componenti del sistema. Sebbene importanti per l'analisi del comportamento, queste informazioni non

sono state considerate rilevanti per l’etichettatura dei cluster, poiché i flussi di dati non forniscono una comprensione concettuale e astratta del contenuto dei modelli.

L’esclusione di queste informazioni consente di rimuovere un numero considerevole di parole dal basso contenuto informativo, migliorando la qualità delle etichette generate e riducendo il rumore tra i cluster.

Esempio di estrazione

Si procede con l’estrazione delle informazioni di un modello AADL, la cui rappresentazione grafica, ottenuta da OSATE, è mostrata in Figura 3.2. Il modello descrive un sistema di interfaccia operatore implementato nel contesto di un modello *Isolette* per il monitoraggio della temperatura. Osservando il grafico, si può notare la presenza di diverse componenti, come ad esempio il *sensore di temperatura* e l’*allarme*. Sono presenti anche numerose feature come quelle che identificano le impostazioni di *temperatura desiderata*.

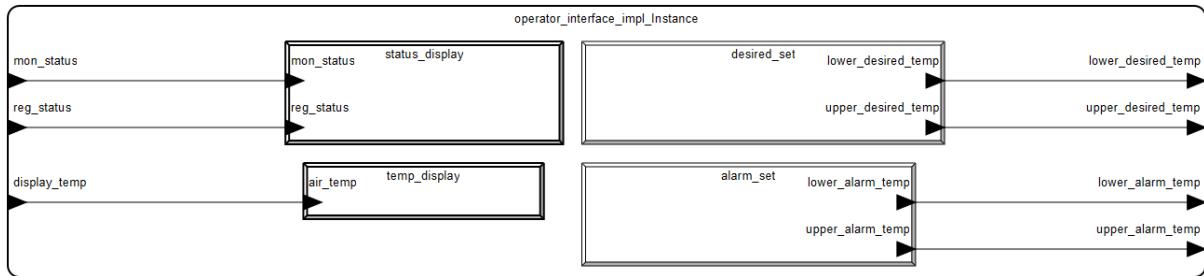


Figura 3.2: Rappresentazione grafica di un modello AADL con OSATE.

Le componenti e le feature del modello estratte, sono riportate nella Tabella 3.1. Il contenuto della tabella mostra come siano stati catturati gli elementi chiave del modello descritto in Figura 3.2 attraverso una semplice rappresentazione testuale. Questo tipo di rappresentazione predispone le informazioni a essere normalizzate nella fase di preprocessing, per poi essere utilizzate dalle tecniche di labelling nella generazione delle etichette.

3.2 Preprocessing

L’estrazione mirata dei nomi delle componenti e delle feature di un modello consente di isolare le informazioni che meglio rappresentano il comportamento e le funzionalità di un sistema. Tuttavia, la forma in cui queste informazioni vengono presentate potrebbe influenzare

<i>Component</i>	<i>Feature</i>
temp_display, status_display, desired_set, alarm_set	display_temp, reg_status, mon_status, lower_desired_temp, upper_desired_temp, lower_alarm_temp, upper_alarm_temp, air_temp reg_status, mon_status, lower_desired_temp, upper_desired_temp, lower_alarm_temp, upper_alarm_temp

Tabella 3.1: Esempio informazioni estratte dalle componenti e feature di un modello AADL.

sensibilmente i risultati ottenuti dai modelli di individuazione delle parole chiave ai fini dell’etichettatura. Per uniformare il più possibile il contenuto testuale da fornire in ingresso agli algoritmi di labelling, in genere, vengono adottate diverse tecniche di preprocessing. Queste tecniche mirano a ridurre la variabilità e a migliorare la qualità dei dati, consentendo di ottenere rappresentazioni più coerenti e rilevanti per la generazione delle etichette.

3.2.1 Tokenization

Il primo passo fondamentale del preprocessing è la **tokenizzazione**, ovvero la suddivisione del testo in unità più piccole, chiamate token. Un token può essere una parola, una sequenza di caratteri, o una singola entità, a seconda del tipo di analisi che si intende effettuare. In questo contesto, ogni parola contenuta nel nome delle componenti e feature estratte dai modelli AADL viene trattata come un token. Ad esempio, nel caso di una feature come **"temp_display"**, la tokenizzazione separa il termine in due unità: **"temp"** e **"display"**. Questo passaggio è cruciale per gestire parole composite o per identificare concetti distinti all’interno di una singola stringa. La tokenizzazione consente di suddividere il testo, rendendolo più strutturato e pronto per l’applicazione di altre tecniche di preprocessing. Essa rappresenta il primo step fondamentale per scomporre un testo complesso in unità semanticamente significative, che possono essere successivamente analizzate e trattate singolarmente.

3.2.2 Alphanumericals

La rimozione dei caratteri alfanumerici è una tecnica volta a eliminare simboli, numeri e caratteri speciali non utili all’analisi del testo. Sebbene questi elementi possano essere necessari per il funzionamento interno del modello AADL, non hanno valore semantico e potrebbero

influenzare negativamente il processo di labelling. Se un modello che rappresenta un sistema di sicurezza è dotato di diversi sensori di movimento, è possibile che questi siano identificati come **"Sensor1"**, **"Sensor2"** ecc. Questa rappresentazione, pur essendo utile per distinguere le componenti del modello, genera nomi diversi per componenti identiche. Applicando la rimozione dei caratteri alfanumerici, entrambi i sensori vengono rappresentati con la parola **"Sensor"** che, fornita in ingresso alle tecniche di labelling, consente di considerare in maniera equivalente le due componenti, migliorando la qualità delle etichette generate.

3.2.3 Stopwords

Le **stopwords** sono parole che, pur essendo molto comuni, non portano informazioni significative per il processo di analisi semantica di un testo. Esempi di stopwords includono articoli, preposizioni, congiunzioni, ecc. Queste parole sono generalmente rimosse durante il preprocessing in quanto non contribuiscono al significato complessivo del testo. Il processo di rimozione delle stopwords avviene confrontando ogni parola del testo con un elenco predefinito di parole comuni. Se la parola è presente nella lista, viene eliminata dal testo. Questo passo riduce il rumore nei dati, migliorando la qualità del processo di analisi. Nel contesto dei modelli AADL, spesso capita che alcune componenti come bus e thread vengano rappresentate con il nome **"this_bus"** e **"this_thread"**. Se molti modelli analizzati contengono componenti con questa struttura, a seguito della fase di tokenizzazione, si avrà una proliferazione di token della parola **"this"**. Questi token, se forniti in ingresso agli algoritmi di labelling, potrebbero alterare sensibilmente il processo di generazione delle etichette in quanto introducono delle parole generiche e prive di contenuto informativo nel processo di labelling. L'eliminazione di queste parole comuni consente di concentrarsi sulle informazioni effettivamente rilevanti, migliorando la qualità delle etichette generate.

3.2.4 Lemmatization

La **lemmatizzazione** è il processo di riduzione delle parole alla loro forma base o lemma. A differenza dello **stemming**, che taglia le parole semplicemente alla radice, la lemmatizzazione fa uso di un dizionario linguistico per determinare la forma canonica di una parola. I dizionari linguistici come *WordNet* o gli strumenti di analisi morfologica sono utilizzati per individuare la forma base di ogni parola contenuta nei token. Varianti della stessa parola come **"alarm"** e **alarms** vengono normalizzate a **"alarm"**, mentre i verbi coniugati in differenti tem-

pi verbali come **"running"** vengono ricondotti alla loro forma base **"run"**. Le normalizzazioni effettuate da questa tecnica riducono la variabilità linguistica trattando varianti grammaticali come un'unica entità, migliorando la coerenza dei dati e semplificando l'analisi, il che facilita l'assegnazione di etichette più precise e coerenti durante il processo di labelling.

3.3 Labelling

La fase finale del processo di etichettatura dei cluster di modelli architetturali si fonda sull'utilizzo delle tecniche di labelling, ovvero metodi utilizzati per associare etichette tematiche a documenti o cluster di modelli, basandosi sull'analisi del contenuto testuale. Queste tecniche estraggono **parole chiave** dai dati testuali, che rappresentano gli aspetti più rilevanti o distintivi del contenuto analizzato. Nel contesto dei modelli architetturali AADL, l'obiettivo del labelling è identificare termini significativi che possano descrivere le caratteristiche principali dei cluster di modelli, permettendo di attribuire a ogni cluster un'etichetta che riassume le sue caratteristiche principali.

I dati testuali forniti in ingresso agli algoritmi di labelling sono i nomi delle componenti e feature dei modelli AADL a cui sono state applicate le tecniche di preprocessing definite nelle fasi precedenti. Le etichette vengono generate utilizzando due differenti tecniche di labelling: **Term Frequency - Inverse Document Frequency** (TF-IDF) e **Latent Dirichlet Allocation** (LDA). L'applicazione separata di queste due tecniche sullo stesso dataset consente di generare etichette attraverso approcci complementari, offrendo l'opportunità di valutare quale delle due tecniche fornisca i risultati migliori o se entrambe siano equivalenti nell'assegnazione delle etichette.

3.3.1 TF-IDF labelling

La tecnica **Term Frequency - Inverse Document Frequency** (TF-IDF) è ampiamente utilizzata nell'elaborazione del linguaggio naturale per determinare l'importanza di una parola all'interno di un documento rispetto a un corpus di documenti. Nel contesto dei modelli AADL, TF-IDF rappresenta una delle metodologie più efficaci per l'identificazione delle parole chiave che meglio descrivono ciascun cluster di modelli, basandosi sulla frequenza e sulla distribuzione delle parole nel corpus. Questa tecnica consente di individuare le parole che sono significative

e rappresentative di un determinato gruppo di modelli, facilitando l’assegnazione di etichette significative ai cluster.

A seguito delle normalizzazioni ottenute dalla fase precedente, ogni cluster è rappresentato come una collezione di documenti, dove ogni documento corrisponde a un singolo modello. Il contenuto di ogni documento è costituito da un elenco di parole estratte dalle sue componenti e feature, sulle quali sono state applicate le tecniche di preprocessing. Per applicare TF-IDF ai dati estratti dai modelli AADL, prima di tutto viene creata una **matrice TF-IDF**, che contiene i punteggi calcolati per ogni parola in ogni documento appartenente ai singoli cluster. Ogni riga della matrice corrisponde a un documento, mentre ogni colonna rappresenta una parola unica estratta dal corpus di modelli del cluster. Il valore di ciascuna cella della matrice rappresenta il punteggio TF-IDF di una parola per un documento specifico. La matrice risultante sarà sparsa, poiché non tutte le parole sono presenti in ogni modello, ma ognuna di esse ha un punteggio che indica la sua rilevanza.

Calcolo dei punteggi TF-IDF

Per ogni parola w contenuta in un documento d , il punteggio TF-IDF viene calcolato come:

$$TF - IDF_{(w,d)} = TF_{(w,d)} \times IDF_w$$

dove:

$$TF_{(w,d)} = \frac{f_{w,d}}{|d|}$$

$$IDF_w = \log \left(\frac{|D|}{f_{w,D}} \right)$$

Per calcolare la **Term Frequency** (TF), si calcola la frequenza della parola w nel documento d , diviso per il numero totale di parole nel documento $|d|$. La **Inverse Document Frequency** (IDF), invece, si calcola come il numero totale di documenti nel cluster $|D|$ diviso il numero di documenti in cui la parola w appare, moltiplicato per il logaritmo naturale, che permette di normalizzare il punteggio.

Nel calcolo di TF-IDF, le parole con alta frequenza all’interno di un singolo documento tendono a ricevere un punteggio elevato grazie al valore della *TF*, mentre le parole che sono presenti in numerosi documenti ricevono un punteggio ridotto a causa del termine *IDF*, che ne penalizza l’importanza. Tuttavia, l’impatto di queste parole può variare notevolmente a seconda della loro distribuzione all’interno del corpus.

Ad esempio, supponiamo di analizzare i documenti di un cluster che contiene modelli simili a quello presentato in Figura 3.2. I modelli in questo cluster, pur non essendo solamente dei modelli di Isolette, condivideranno gran parte delle proprie caratteristiche strutturali e semantiche, a causa delle tecniche di clustering applicate. Un tipo di componente comune tra questi modelli potrebbe essere il **”sensor”**, che identifica un sensore di temperatura nel caso di incubatrici, ma che potrebbe anche riferirsi a un sensore di movimento, ad esempio, in un sistema di sicurezza come quello mostrato in Figura 1.3. Questa parola, quindi, avrà un alto valore di TF , poiché è molto frequente nei modelli del cluster. Tuttavia, questo valore viene bilanciato dal punteggio IDF , dato che questa componente appare frequentemente anche in altri modelli del cluster. Al contrario, componenti come **”temperature”** e **”motion”**, che identificano rispettivamente la tipologia specifica di sensore nei due tipi di modelli presentati nell’esempio, avranno un punteggio TF molto alto nei modelli in cui sono presenti e un punteggio IDF più basso, data la loro assenza nei modelli complementari. Questo consente di attribuire dei punteggi complessivamente più alti a queste parole, rendendole più significative per la rappresentazione del contenuto dei modelli.

Una volta calcolati i punteggi TF-IDF per tutte le parole di ogni documento, per poter individuare le parole chiave del cluster si sommano verticalmente i punteggi TF-IDF di ogni parola su tutti i documenti del cluster. In questo modo si ottiene un punteggio cumulativo per ogni parola presente nel cluster, consentendo l’ordinamento di tutte le parole in base al loro punteggio. A seguito di questo ordinamento, si selezionano le N parole di punteggio più alto come etichette da attribuire al cluster, in quanto meglio descrivono il contenuto dei modelli presenti al suo interno.

3.3.2 LDA labelling

La **Latent Dirichlet Allocation** (LDA) è un metodo di **topic modeling** ampiamente utilizzato per scoprire argomenti latenti all’interno di un corpus di documenti. LDA si basa sull’assunzione che ogni documento sia una combinazione di vari topic, e che ogni topic sia rappresentato come una distribuzione di parole. Per applicare questa tecnica ai dati estratti dai modelli AADL, LDA analizza le co-occorrenze delle parole nei documenti del cluster, identificando i topic latenti che meglio rappresentano le caratteristiche comuni di quel gruppo.

Per applicare l’algoritmo di LDA, per prima cosa è necessario generare una rappresentazione del contenuto di tutti i documenti presenti nel cluster attraverso una **matrice di frequenza**

termine-documento che conta le occorrenze con cui una parola appare in un dato documento. Il secondo elemento necessario per applicare LDA è il valore che identifica il numero di topic da estrarre tramite il modello; questo valore può essere scelto arbitrariamente o determinato in modo più accurato attraverso misurazioni specifiche. Una di queste misurazioni è la **perplexity** [3], che consente di valutare come il modello riesca a rappresentare i dati al variare del numero di topic selezionati. Minore è la perplexity di un modello, maggiore è la sua capacità di adattarsi ai dati analizzati.

Le informazioni contenute nella matrice termine-documento, assieme al parametro che identifica il numero di topic da rilevare, costituiscono i dati di input forniti al modello di addestramento di LDA. Tramite l'uso di un algoritmo di inferenza bayesiana, LDA determina la distribuzione dei topic per ciascun documento e la distribuzione delle parole per ciascun topic. Il modello si allena iterativamente per trovare i topic latenti che meglio spiegano la co-occorrenza delle parole nel corpus analizzato. Durante il processo, le parole che compaiono frequentemente insieme nei documenti vengono raggruppate sotto lo stesso topic, mentre le parole meno frequenti o non correlate ai temi principali del cluster vengono assegnate a topic distinti.

Una volta identificati i topic, il passo successivo consiste nell'ordinare le parole per ciascun topic in base alla loro probabilità di appartenenza a quel topic. Le parole più rappresentative di ogni topic avranno delle probabilità più alte; viceversa, le parole meno rappresentative avranno delle probabilità minori. La selezione delle parole da includere in ogni topic avviene generalmente selezionando le parole con i punteggi più alti, spesso applicando un valore di soglia.

Le parole selezionate rappresentano i temi principali dei modelli all'interno di ciascun cluster. Ogni topic rappresenta un argomento latente, e le parole con la probabilità più alta di appartenere a un topic sono quelle che meglio descrivono il contenuto del cluster. Queste parole chiave vengono quindi utilizzate per etichettare il cluster, con l'obiettivo di riassumere in modo conciso i temi trattati nei modelli.

Riprendendo gli esempi di modelli AADL presentati nelle Figure 1.3 e 3.2, supponiamo di avere un cluster che contiene diversi modelli simili tra di loro, caratterizzati dalla presenza della componente **"sensor"**. Dopo aver applicato LDA ai dati preprocessati di questi modelli, i topic principali individuati potrebbero essere:

- **Topic 1:** *"sensor"*, *"alarm"*, *"temperature"*

- **Topic 2:** *"sensor"*, *"alarm"*, *"motion"*

In questo caso, LDA ha identificato i principali temi che emergono nei documenti del cluster. Le parole chiave per Topic 1 sono quelle relative ai sensori di temperatura, mentre per Topic 2 sono quelle legate ai sensori di movimento. Le etichette generate consentono quindi di rappresentare chiaramente i principali argomenti trattati nel cluster, fornendo informazioni sintetiche e significative sul contenuto dei modelli presenti al suo interno.

4. Applicazione delle metodologie

In questo capitolo vengono descritte le operazioni di applicazione delle metodologie eseguite dal codice *Python*, sviluppato per analizzare, etichettare e validare cluster di modelli architetturali descritti in linguaggio AADL. Dopo aver introdotto il contenuto della base dati di partenza, derivante da lavori pregressi, si illustrano nel dettaglio le scelte operative adottate nello sviluppo del codice. Il codice, suddiviso in moduli separati, inizialmente si occupa di analizzare il contenuto dei modelli AADL presenti nei cluster e di selezionare gli **elementi fondamentali** da estrarre. In seguito, si procede con la normalizzazione del contenuto estratto, utilizzando diverse tecniche di **preprocessing**, per generare un input adeguato alle tecniche di labelling. Infine, viene mostrato come le tecniche **TF-IDF** e **LDA** siano implementate per l'individuazione di parole chiave all'interno dei modelli e, in conclusione, come queste parole vengano selezionate per la generazione delle etichette.

4.1 Base di conoscenza e lavori pregressi

I dati utilizzati come input per lo sviluppo di questo progetto derivano da lavori precedenti condotti da colleghi che si sono occupati della generazione della base di conoscenza iniziale di modelli AADL e del clustering degli stessi.

La *Knowledge Base* di partenza su cui il progetto si fonda è costituita da una raccolta di **1078** modelli architetturali descritti in linguaggio AADL. La fonte dati utilizzata per la creazione della base di conoscenza è **GitHub**, da cui sono stati estratti i modelli presenti in repository pubblici. I modelli estratti massivamente sono in seguito validati tramite un processo che elimina tutti gli artefatti privi di contenuto informativo. Infine, i modelli AADL sono convertiti in formato **AAXL2**, che è una rappresentazione XML dei modelli. Questo formato è fondamentale, in quanto consente una facile valutazione del contenuto dei modelli e, tramite l'utilizzo di **OSATE**, una rappresentazione grafica degli stessi.

Per poter organizzare i modelli AADL simili all'interno di cluster, si è deciso di adottare un approccio ibrido che sfrutta le metriche di **similarità strutturale** e **similarità semantica**. La similarità strutturale, calcolata dopo aver convertito i modelli in grafi diretti, identifica gli elementi comuni tra i modelli utilizzando il **Maximum Common Subgraph** (MCS). La si-

milarità semantica, invece, sfrutta i nomi dei nodi dei grafi che rappresentano i modelli per generare, attraverso l'uso di **FastText**, dei vettori semantici che consentono il confronto tra modelli differenti. Il calcolo della similarità strutturale e semantica tra tutti i modelli della base di conoscenza ha portato alla generazione di matrici di similarità, utilizzate nel processo di clustering.

Il clustering adottato è di tipo **gerarchico agglomerativo**, il che porta alla creazione di una struttura ad albero, dove le foglie rappresentano i singoli modelli e i nodi rappresentano i cluster. La tecnica iterativa prevede che i due modelli più simili vengano uniti in un cluster, ripetendo il processo fino a ottenere un unico grande cluster che include tutti i modelli della base di conoscenza. Il taglio a un'altezza specifica del dendrogramma generato dal processo di clustering ha determinato la creazione di **44** cluster. Questi cluster, insieme ai modelli AADL in formato AAXL2 contenuti al loro interno, costituiscono le informazioni di partenza per lo sviluppo di questo progetto.

4.2 Impostazione del progetto

Il progetto si sviluppa attraverso una serie di fasi sequenziali che mirano ad analizzare e elaborare i modelli AADL, partendo dall'estrazione delle informazioni principali e dal preprocessing delle stesse fino ad arrivare alla generazione delle etichette e valutazione dei risultati ottenuti. Queste fasi sono suddivise come segue:

1. **Estrazione delle informazioni dai modelli:** In questa fase, i modelli AADL vengono analizzati per estrarre le informazioni rilevanti, come componenti e feature. Viene effettuata una selezione accurata per garantire che solo i modelli pertinenti e completi vengano utilizzati nel processo successivo.
2. **Preprocessing dei dati:** I dati estratti dai modelli vengono sottoposti a un processo di normalizzazione e pulizia. Questa fase include la rimozione di informazioni non rilevanti e la trasformazione del contenuto in un formato adatto per le tecniche di etichettatura.
3. **Generazione delle etichette:** Una volta normalizzati i dati, vengono applicate le tecniche di etichettatura per identificare e assegnare le parole chiave ai cluster di modelli. In questa fase vengono utilizzate due tecniche principali: *TF-IDF* e *LDA*, per migliorare l'accuratezza delle etichette assegnate.

4. Valutazione dei risultati: Dopo aver generato le etichette, il processo viene validato mediante il calcolo di diverse metriche di performance, come la similarità coseno, la precisione, il recall e l'F1 score. Queste metriche permettono di misurare l'efficacia del processo di etichettatura.

Ogni fase è supportata da specifici moduli Python, ognuno dei quali è stato progettato per risolvere particolari criticità e ottimizzare il processo complessivo. Di seguito viene fornita una panoramica dei principali file del progetto e delle operazioni che ciascuno di essi esegue:

- `main.py`: questo modulo rappresenta il punto di ingresso del programma. Gestisce l'elaborazione dei dati e la generazione dei report per ogni fase del progetto. Il file coordina il flusso delle operazioni e assicura la corretta esecuzione sequenziale delle fasi.
- `AADLManager.py`: il modulo si occupa dell'analisi del contenuto dei modelli AADL e della estrazione delle informazioni rilevanti presenti al loro interno. Le informazioni estratte sono salvate nel file **suitable_models_data.csv** che contiene anche il nome di ogni modello idoneo e il cluster di appartenenza.
- `labelling.py`: questo modulo, suddiviso nelle classi `TextPreprocessing` e `Labelling`, si occupa della normalizzazione del contenuto estratto dai modelli e in seguito utilizza le informazioni processate per effettuare il labelling dei cluster utilizzando in modo parallelo le tecniche TF-IDF e LDA. I token normalizzati, ottenuti attraverso le fasi di preprocessing, vengono salvati in file CSV separati. Il file **preprocessed_clusters.csv** contiene i token normalizzati per i nomi dei file dei modelli AADL, mentre il file **preprocessed_suitable_models_data.csv** contiene i token relativi ai nomi delle componenti e delle feature estratte dai modelli idonei. Per quanto riguarda le etichette generate attraverso le tecniche TF-IDF e LDA, queste vengono salvate nei file **TFIDF_Labels.csv** e **LDA_Labels.csv**.
- `validation.py`: quest'ultimo modulo è dedicato alla validazione del processo di etichettatura. Calcola diverse misure di performance per analizzare la qualità delle etichette generate e si occupa anche della generazione di grafici e report utili all'analisi dei risultati ottenuti.

Gli strumenti sviluppati sono pubblicamente disponibili su repository GitHub¹ per consentire di riprodurre i risultati presentati in seguito.

4.3 Estrazione delle informazioni dai modelli

La prima fase, fondamentale per poter garantire una corretta generazione di etichette per i cluster di modelli architetturali, prevede l’analisi dei modelli AADL e la selezione delle principali informazioni di interesse per garantire che i dati estratti siano coerenti e utili per le fasi successive del progetto.

Dopo l’analisi manuale iniziale di alcuni modelli disponibili eseguita con l’aiuto di OSATE, si procede con il processo di analisi massiva del dataset e di selezione ed estrazione degli elementi rilevanti dei modelli AADL. Una parte dei metodi implementati in questo modulo si occupa della generazione di report relativi al contenuto dei modelli dell’intera base di conoscenza, la cui analisi ha portato alle prime scelte progettuali. La restante parte dei metodi implementati si occupa dell’estrazione delle componenti più importanti dei modelli, immagazzinandole in un unico file che sarà utilizzato per le fasi successive di preprocessing e labelling.

4.3.1 Analisi della base di conoscenza

La base di conoscenza iniziale è composta da **1078 modelli AADL**, provenienti da repository pubblici. Ogni modello rappresenta un sistema complesso, descritto in modo astratto attraverso componenti, feature e relazioni. I modelli sono raggruppati all’interno di **44 cluster**; l’associazione modello-cluster è definita dal contenuto di una tabella di cui viene mostrato parzialmente il contenuto in Tabella 4.1.

Il contenuto della tabella è stato utilizzato per generare il grafico in Figura 4.1, che mostra la distribuzione dei modelli all’interno dei cluster. Analizzando il grafico, si nota che la distribuzione dei modelli non è omogenea: alcuni cluster, in particolare i cluster numero 30, 33 e 34, contengono un numero significativamente inferiore di modelli rispetto alla media.

Per ottenere una conoscenza più dettagliata dei modelli AADL contenuti nei vari cluster, vengono eseguite delle analisi puntuali su modelli appartenenti a cluster differenti. I modelli AADL possono essere analizzati tramite due modalità principali:

¹https://github.com/MarcoDiCapua/AADL_Labelling

<i>Modello</i>	<i>Cluster</i>
6UCubeSat_Decision_Module_Decision_Module_2_Controller	5
6UCubeSat_Rate_Limited_PID_Controller_Rate_Limited_Controller	5
6UCubeSat_RTA_Controller_RTA_Controller_Impl_1	7
6UCubeSat_Unverified_Controller_PID_controller_Unverified	7
6UCubeSat_Verified_Controller_Rate_Limited_PID_Controller	7
890-isolette_KSU_Isolette_isollette_dual_sensor_1	2
890-isolette_KSU_Isolette_isollette_dual_sensor_10	2
890-isolette_KSU_Isolette_isollette_dual_sensor_11	2
890-isolette_KSU_Isolette_isollette_dual_sensor_2	2
890-isolette_KSU_Isolette_isollette_dual_sensor_3	2

Tabella 4.1: Estratto della tabella contenente le associazioni modello-cluster.

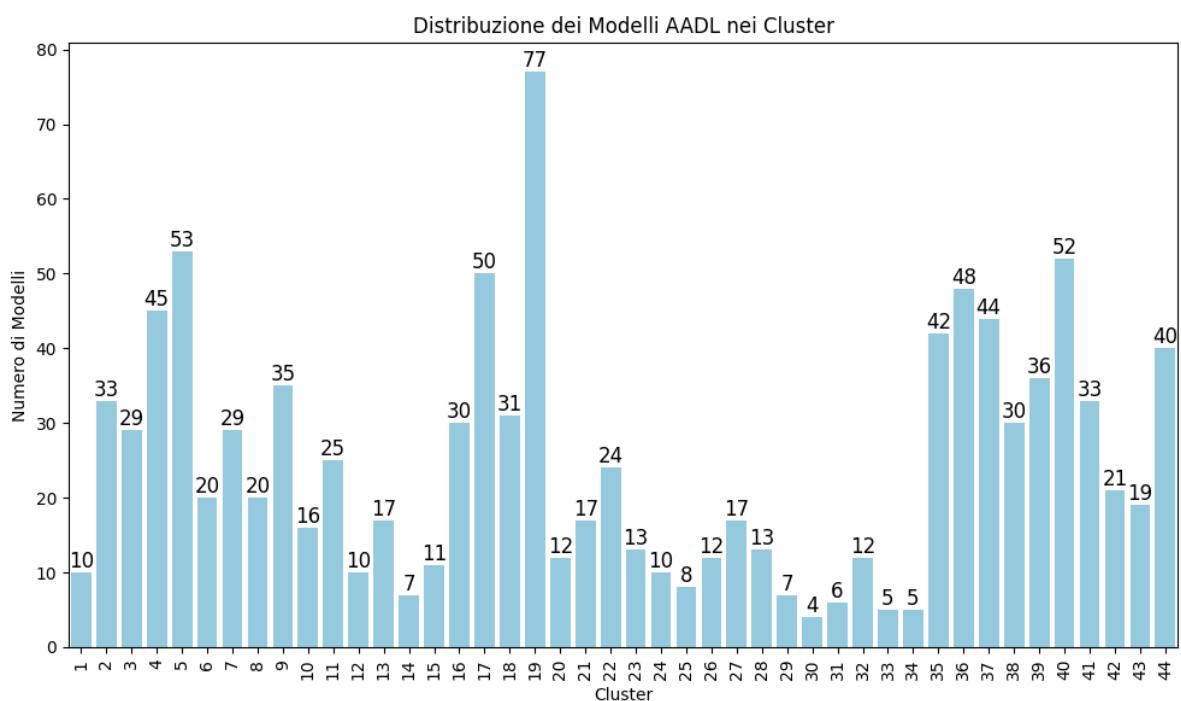


Figura 4.1: Distribuzione dei modelli all'interno dei cluster.

- **Rappresentazione grafica:** utilizzando **OSATE**, è possibile visualizzare i modelli AADL in forma grafica. Questo approccio facilita la comprensione della struttura e delle interazioni tra i componenti del sistema, fornendo una rappresentazione visiva chiara e intuitiva.
- **Rappresentazione testuale:** i modelli AADL possono essere anche visualizzati in formato testuale, attraverso una rappresentazione **XML**. Questa modalità consente una visione dettagliata dei dati strutturati del modello, utile per analisi più approfondite e manipolazioni automatizzate.

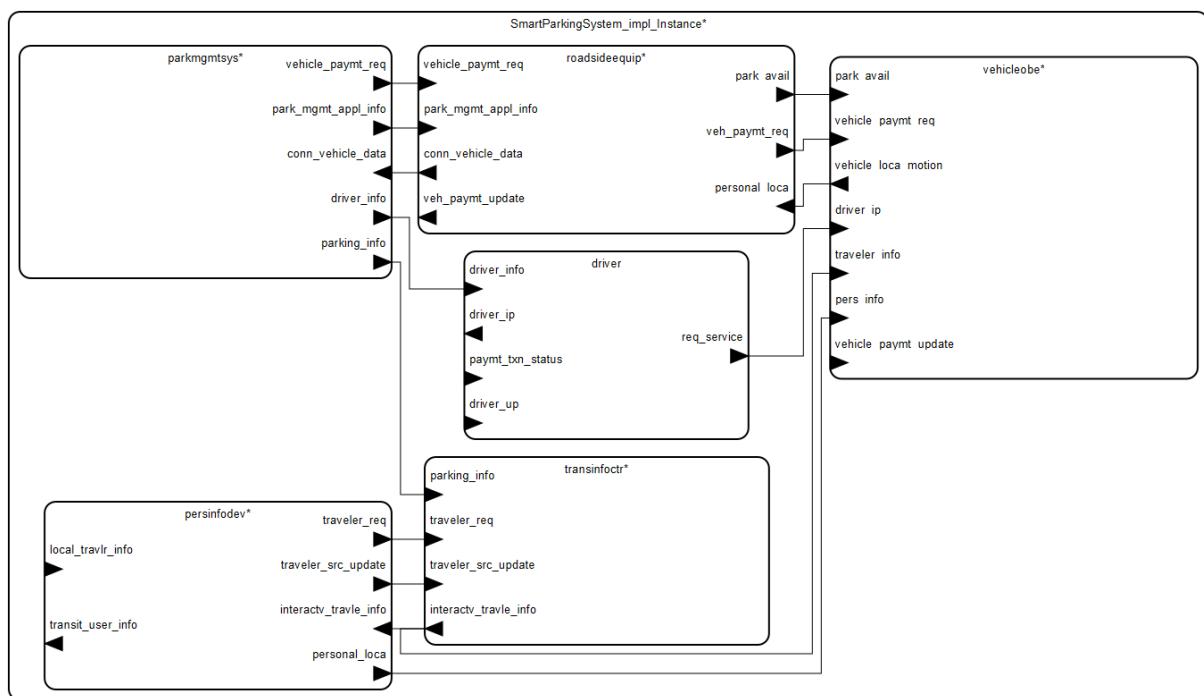


Figura 4.2: Rappresentazione grafica di un modello AADL di smart parking.

```

1 <?xml version="1.0" encoding="ASCII"?>
2 <instance:SystemInstance xmi:version="2.0"
3   xmlns:xmi="http://www.omg.org/XMI"
4   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5   xmlns:aadl2="http://aadl.info/AADL/2.0"
6   xmlns:instance="http://aadl.info/AADL/2.0/instance" name="SmartParkingSystem_impl_Instance" category="system">
7 <componentInstance name="parkmgmtsys" category="system">
8   <featureInstance name="vehicle_paymt_req" direction="out">
9     <feature xsi:type="aadl2:DataPort"/>
10  </featureInstance>
11  <featureInstance name="conn_vehicle_data">
12    <feature xsi:type="aadl2:DataPort"/>
13  </featureInstance>
14  <featureInstance name="parking_info" direction="out">
15    <feature xsi:type="aadl2:DataPort"/>
16    <type href="../../github/AadlProjects/smартpark/instances/smартparking_SmartParkingSystem_impl_Instance.aaxl2#/6"/>
17  </featureInstance>
18  <featureInstance name="park_mgmt_appl_info" direction="out">
19    <feature xsi:type="aadl2:DataPort" />
20  </featureInstance>
21  <featureInstance name="driver_info" direction="out">
22    <feature xsi:type="aadl2:DataPort" />
23  </featureInstance>
24  <componentInstance name="parkmgmt" category="system">
25    <featureInstance name="occupancy" direction="out">
26      <feature xsi:type="aadl2:DataPort" />
27      <type href="../../github/AadlProjects/smартpark/instances/smартparking_SmartParkingSystem_impl_Instance.aaxl2#/4"/>
28  </featureInstance>
29  <featureInstance name="park_spaces" direction="out">
30    <feature xsi:type="aadl2:DataPort" />
31    <type href="../../github/AadlProjects/smартpark/instances/smартparking_SmartParkingSystem_impl_Instance.aaxl2#/5"/>
32  </featureInstance>
33  <componentInstance name="dynamarklot" category="process">
34    <featureInstance name="occupancy" direction="out">
35      <feature xsi:type="aadl2:DataPort" />
36      <type href="../../github/AadlProjects/smартpark/instances/smартparking_SmartParkingSystem_impl_Instance.aaxl2#/1"/>
37    </featureInstance>
38    <componentInstance name="park_lot_occupancy" category="thread">
39      <featureInstance name="park_occupancy" direction="out">

```

Figura 4.3: Rappresentazione testuale di un modello AADL di smart parking.

Il modello di esempio, rappresentato graficamente in Figura 4.2 e testualmente in Figura 4.3, riguarda un sistema di *smart parking*. L’analisi della rappresentazione grafica del modello consente di comprendere agevolmente la sua struttura in termini di componenti e come queste si relazionano tra loro attraverso le feature. Analizzando la rappresentazione testuale, invece, si nota come i singoli componenti e feature siano identificati da tag come `componentInstance` e `featureInstance`. All’interno di questi tag sono contenute informazioni fondamentali, come il nome dell’elemento, che spesso descrive una funzionalità chiave del sistema.

A seguito di questa analisi, è emerso che i nomi degli elementi costitutivi di un sistema AADL sono informazioni chiave per rappresentarne le funzionalità. Pertanto, si è deciso di procedere con l’estrazione dei nomi di tutti gli elementi presenti in tutti i modelli AADL della base dati.

4.3.2 Estrazione degli elementi chiave dai modelli

L'analisi manuale del contenuto dei modelli AADL ha portato alla scelta progettuale di estrarre i nomi di tutti gli elementi che compongono i modelli della base dati. Le tipologie di elementi presenti in questi modelli sono: *component*, *feature*, *connections*, *modes* e *flow specification*. Per ciascuna di queste tipologie è stato implementato del codice che esegue l'estrazione massiva dei nomi di queste componenti per poterle analizzare.

Il conteggio degli elementi estratti, come rappresentato in Figura 4.4, evidenzia che i quantitativi di nomi appartenenti alle tipologie modes e flow specification sono insufficienti per essere utilizzati nel processo di labelling.

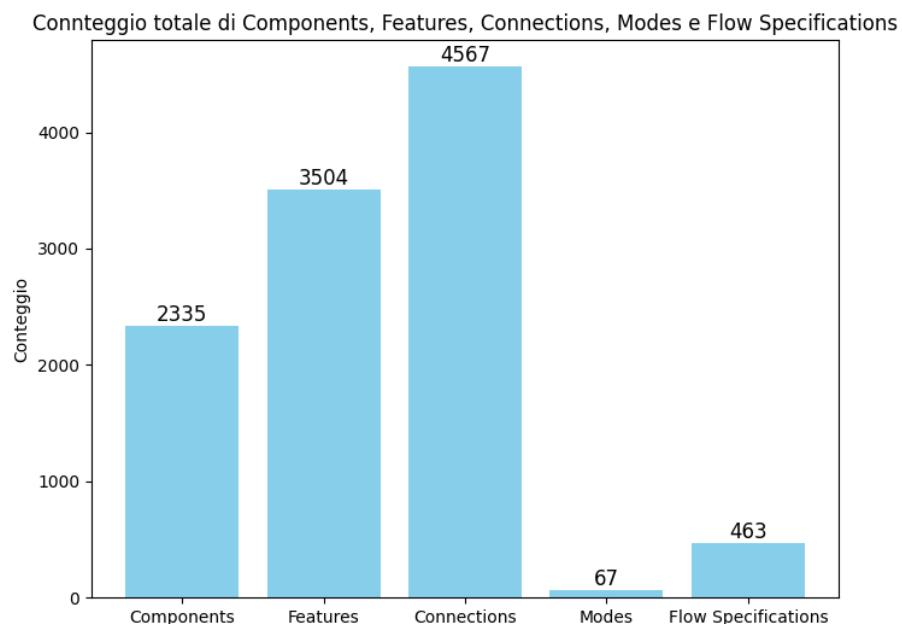


Figura 4.4: Conteggio dei nomi degli elementi estratti divisi per tipologia.

Gli elementi restanti a livello quantitativo risultano adeguati, pertanto si procede con un'analisi dettagliata del loro contenuto. In particolare, sono stati prodotti dei grafici che mostrano la classifica delle parole più ricorrenti per le tre tipologie. Queste classifiche, mostrate in Figura 4.5 e Figura 4.6, dimostrano come l'estrazione abbia efficientemente catturato le informazioni chiave utili alla descrizione dei modelli AADL.

Il grafico in Figura 4.7, rappresentante i nomi più ricorrenti delle connessioni, denota come, nonostante il quantitativo complessivo di nomi estratti dalle connections sia il più alto tra le tipologie, i nomi stessi siano in gran parte costituiti dai nomi delle componenti collegate. Poiché

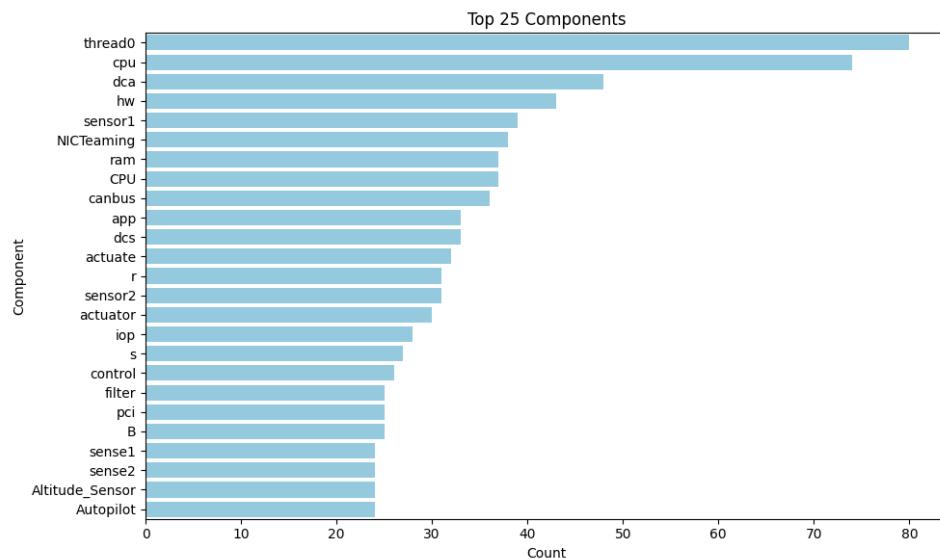


Figura 4.5: Classifica dei 25 nomi più ricorrenti per le componenti.

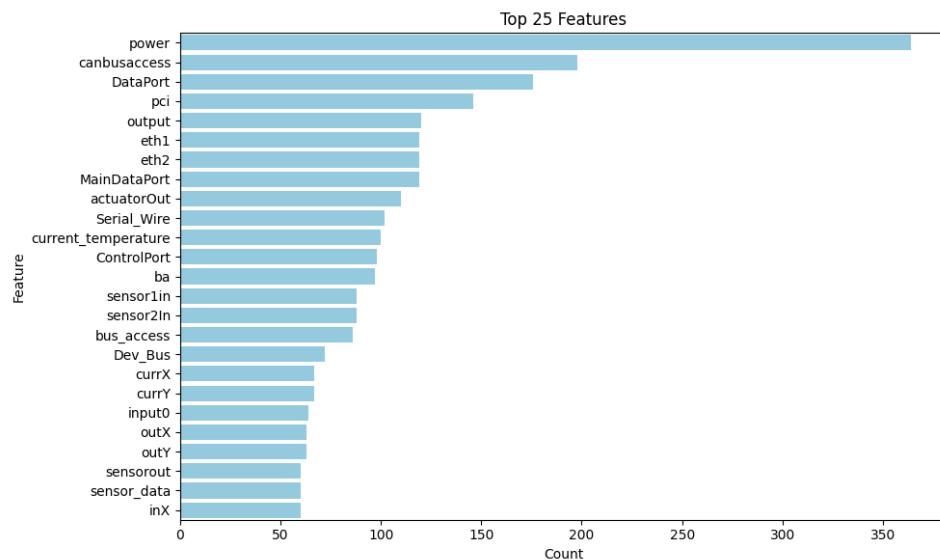


Figura 4.6: Classifica dei 25 nomi più ricorrenti per le feature.

questi elementi non aggiungono nuove informazioni al modello, l'inclusione dei nomi delle connections nel dataset per la generazione delle etichette porterebbe a una ridondanza, senza migliorare la qualità del labelling. Per questo motivo, si è deciso di escludere le connections dal processo di etichettatura.

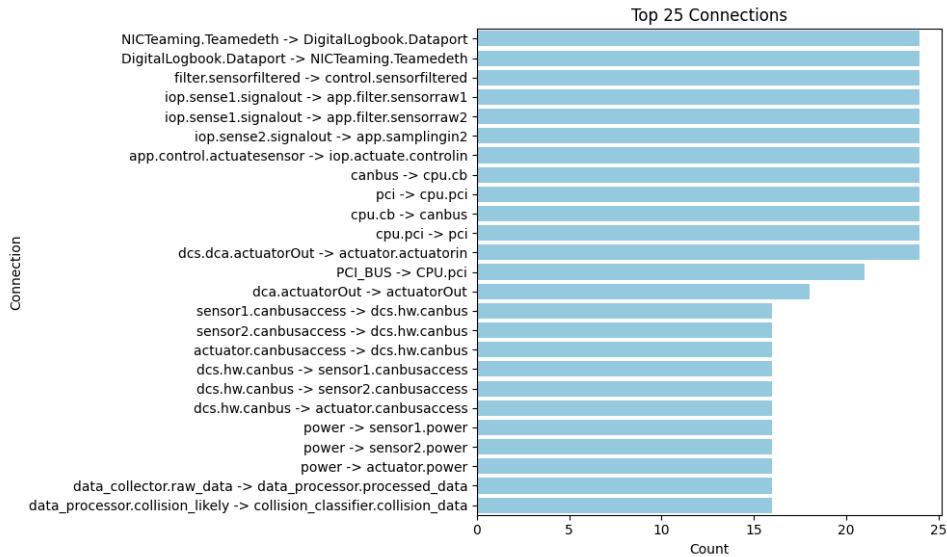


Figura 4.7: Classifica dei 25 nomi più ricorrenti per le connection.

4.3.3 Selezione dei modelli idonei

Come descritto nelle sezioni precedenti, l’analisi dei modelli AADL ha evidenziato che i componenti e le feature sono gli elementi più rilevanti per rappresentare un sistema in modo efficace, poiché riflettono le sue caratteristiche principali e le sue funzionalità. Di conseguenza, si è deciso di estrarre esclusivamente i nomi di questi elementi per il processo di labelling. Tuttavia, non tutti i modelli AADL contengono entrambi questi elementi, il che rende necessario un processo di selezione.

Il codice implementato si occupa di verificare che, per ogni modello appartenente alla base dati, siano presenti almeno una componente e almeno una feature. Se questo criterio è rispettato, il modello viene considerato idoneo: i nomi delle sue componenti e feature vengono estratti e le loro informazioni vengono salvate in una tabella. I modelli che non rispettano questo requisito, invece, sono considerati non idonei e le loro informazioni non vengono estratte.

Il criterio di estrazione adottato, come mostrato in Figura 4.8, ha portato all’esclusione di numerosi modelli appartenenti alla base dati. Su un totale di 1078 modelli, **705** dispongono di almeno una componente e una feature. I modelli scartati rientrano quindi in una delle seguenti tre categorie:

- **Modelli senza componenti:** se un modello non ha alcuna componente, non sono definiti gli elementi essenziali, sia hardware che software, per la rappresentazione di un sistema. Pertanto, non è utile includerli nel processo di labelling.

- **Modelli senza feature:** i modelli che non hanno feature sono quelli in cui, seppur presenti componenti, queste non dispongono delle modalità di interazione necessarie tra di loro. Ciò le rende elementi statici, incapaci di rappresentare in modo completo un sistema complesso.
- **Modelli senza componenti e feature:** questi modelli, privi sia di componenti che di feature, sono da considerarsi vuoti, in quanto carenti degli elementi principali che definiscono un sistema in linguaggio AADL.

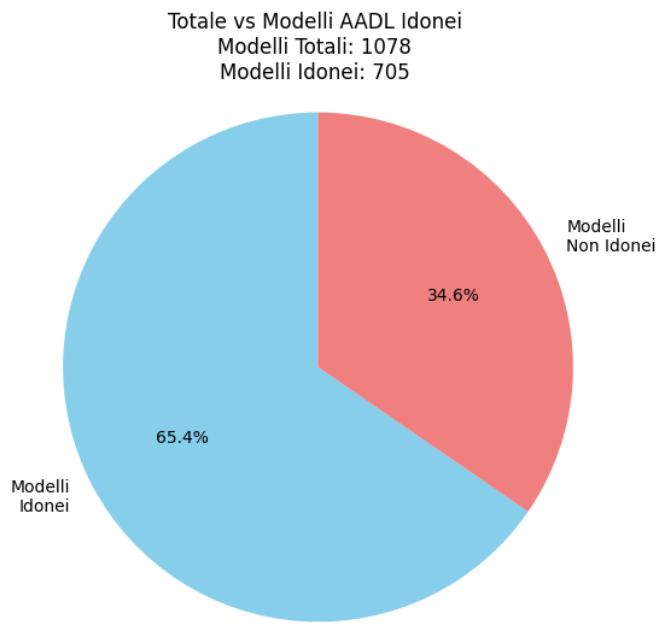


Figura 4.8: Percentuale dei modelli idonei rispetto al numero totale di modelli.

Con questa selezione, il processo di labelling è garantito di basarsi esclusivamente su modelli che contengono le informazioni fondamentali per una rappresentazione semantica significativa e utile, ottimizzando così l'efficacia del processo complessivo.

Una volta identificati i modelli adeguati ed estratte le loro informazioni, sono stati generati dei grafici di distribuzione dei modelli all'interno dei cluster. Questi grafici evidenziano il numero di modelli adeguati in rapporto al totale dei modelli del cluster. Il grafico in Figura 4.9 mostra una potenziale criticità dovuta alla selezione dei modelli. Alcuni cluster contengono un numero considerevole di modelli non idonei al processo di labelling secondo i criteri appena definiti. In particolare, su un totale di 44 cluster, solo **34** contengono almeno un modello idoneo.

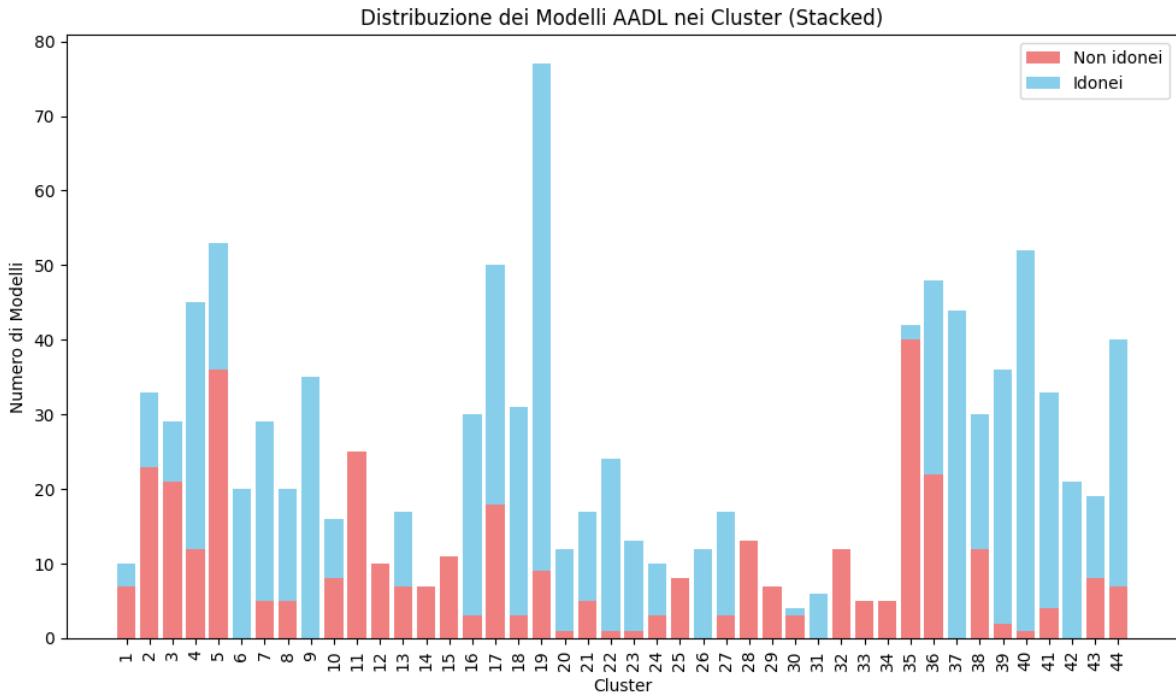


Figura 4.9: Distribuzione dei modelli idonei all'interno dei cluster.

La presenza di cluster che non dispongono di modelli idonei rappresenta una criticità per il processo di etichettatura. Infatti, se le informazioni relative ai nomi delle componenti e delle feature vengono estratte solo dai modelli idonei, i cluster privi di questi elementi non potranno essere etichettati.

Per far fronte a questa problematica, si è deciso di adottare un processo di **labelling composito**. Una prima fase di etichettatura prevede l'utilizzo dei nomi delle componenti e delle feature estratti dai modelli idonei; una seconda fase prevede l'utilizzo dei nomi dei file dei modelli contenuti nei cluster. In questo modo, sarà possibile ottenere delle etichette per ogni modello e, conseguentemente, per ogni cluster, poiché ne verrà generata almeno una a partire dal nome dei file.

4.4 Preprocessing dei dati

Il preprocessing dei dati è una fase cruciale per la preparazione delle informazioni estratte dai modelli AADL prima di procedere con il processo di labelling. L'obiettivo del preprocessing è normalizzare il contenuto delle parole, riducendole a rappresentazioni standardizzate che facilitano l'individuazione delle parole chiave da parte delle tecniche di labelling. In particolare, i nomi delle componenti e delle feature, pur rappresentando concetti simili, possono differire sintatticamente. Se queste differenze non vengono trattate, potrebbero essere identificate come entità distinte nel processo di labelling. L'introduzione di varianti sintattiche simili, ma non identiche, potrebbe compromettere la qualità del processo di labelling, ostacolando l'individuazione delle parole chiave e la generazione delle etichette.

La necessità di applicare il preprocessing ai nomi delle componenti e delle feature estratte è evidente, come mostrato in Figura 4.5, dove, ad esempio, si osservano componenti come **"sensor1"** e **"sensor2"** tra i nomi più frequenti. Questi nomi, pur riferendosi allo stesso concetto, potrebbero facilmente essere ricondotti a un unico termine, **"sensor"**, attraverso le tecniche di preprocessing.

4.4.1 Tokenization e prime normalizzazioni

Il processo di tokenizzazione è alla base dell'applicazione delle tecniche di preprocessing implementate. Prima di generare i token, si procede con alcuni passaggi preliminari. Inizialmente, tutti i caratteri degli elementi vengono convertiti in minuscolo per evitare che differenze tra maiuscole e minuscole influenzino il risultato. Successivamente, i caratteri speciali come **"_"** e **"."**, che fungono da separatori tra le parole nei nomi degli elementi, come in **"altitude_sensor"** o nel nome di file come **"AADL-Designer_CruiseControl"**, vengono sostituiti con uno spazio vuoto. L'ultima normalizzazione implementata prima della generazione dei token prevede la rimozione di un set predefinito di parole estremamente comuni nei nomi dei file, come il termine **"AADL"**, o nei nomi delle componenti e feature, come ad esempio **"system"**.

Il grafico in Figura 4.10 mostra i token più ricorrenti ottenuti a seguito delle manipolazioni appena descritte sui nomi dei file dei modelli AADL.

Dal grafico emerge chiaramente che, nonostante le prime normalizzazioni, la classifica dei token più ricorrenti è ancora composta prevalentemente da termini privi di contenuto in-

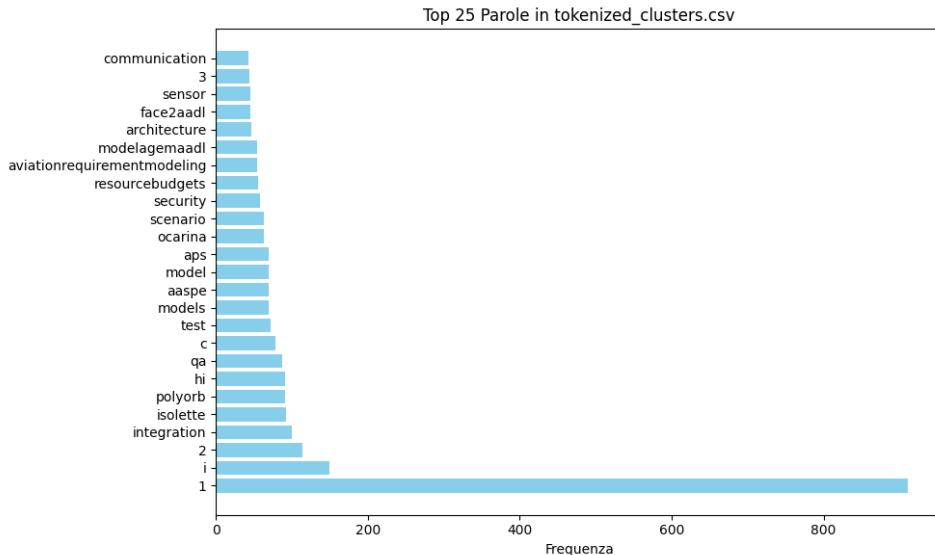


Figura 4.10: Classifica dei 25 token più ricorrenti per i nomi di file AADL.

formativo, come numeri e lettere singole. Questo evidenzia la necessità di ulteriori fasi di normalizzazione.

4.4.2 Rimozione di caratteri non alfanumerici

Una volta generati i token, è possibile applicare su di essi delle tecniche di preprocessing mirate basandosi sulle ricorrenze nel loro contenuto. In particolare, analizzando la Figura 4.10, si nota la presenza di caratteri numerici o, più in generale, di caratteri non alfanumerici che non sono utili ai fini di generare delle etichette e pertanto devono essere rimossi. Questa normalizzazione elimina i token costituiti esclusivamente da numeri e riconduce a una forma comune tutti i nomi di componenti e feature di lunghezza maggiore di due che includono numeri, come ad esempio **"sensor1"** e **"sensor2"**.

Il risultato ottenuto da questa normalizzazione, mostrato in Figura 4.11, evidenzia l'efficacia delle modifiche apportate, poiché i token più frequenti sono ora composti principalmente da parole di contenuto informativo rilevante.

4.4.3 Rimozione delle stopwords

La rimozione dei caratteri non alfanumerici ha rinnovato l'aspetto dei token più frequenti, tuttavia, questi possono ancora contenere parole che non apportano valore semantico per il labelling. La rimozione delle stopwords è una tecnica utilizzata nell'elaborazione del linguaggio naturale

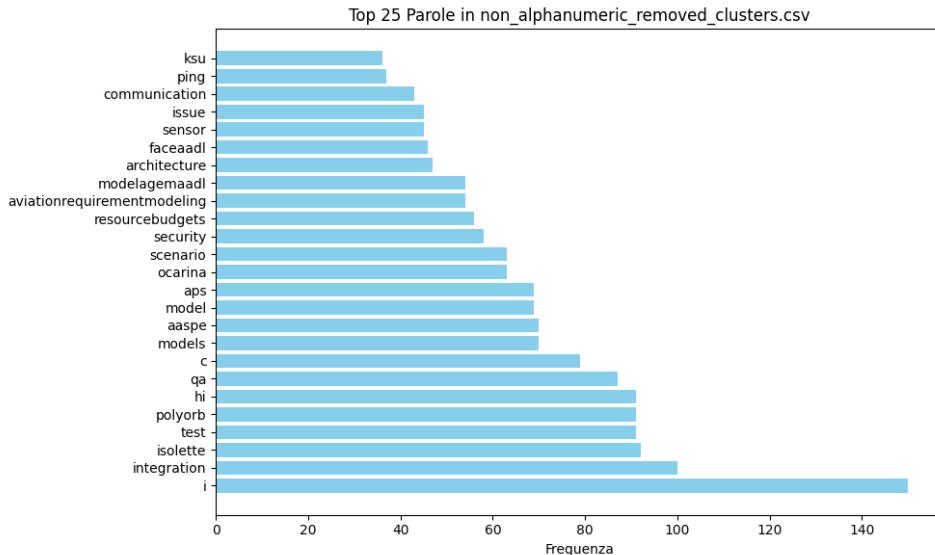


Figura 4.11: Classifica dei 25 token più ricorrenti per i nomi di file AADL dopo aver rimosso i caratteri non alfanumerici.

che può trovare la sua applicazione anche nel contesto dei modelli AADL. Nel caso dei nomi delle componenti e delle feature, ad esempio, può accadere che i nomi siano preceduti dalla parola **"this"**. Preposizioni, articoli e pronomi non sono rilevanti per il labelling e, se non rimossi, potrebbero influire negativamente sui risultati finali.

Il set di stopwords utilizzato per rimuovere parole irrilevanti durante il preprocessing viene caricato dal dizionario predefinito in **Natural Language Toolkit** (NLTK), che offre una lista di parole comuni in diverse lingue.

La rimozione delle stopwords, come osservabile in Figura 4.12, ha comportato l'eliminazione della parola più frequente all'interno dei token dei nomi dei file dei modelli AADL, ossia il termine **"i"**. Sebbene nel contesto del progetto non assuma il significato di pronome, la sua rimozione è vantaggiosa in quanto non apporta alcun valore informativo utile alla generazione delle etichette.

4.4.4 Lemmatization e normalizzazioni finali

Il processo di normalizzazione dei token si conclude con l'applicazione della lemmatizzazione. Il processo linguistico adottato da questa tecnica riduce le parole alla loro forma base o lemma. In questo modo, varianti grammaticali come coniugazioni verbali o plurali, come il plurale **"models"** della parola **"model"**, osservabile in Figura 4.12, vengono ricondotte alla forma di

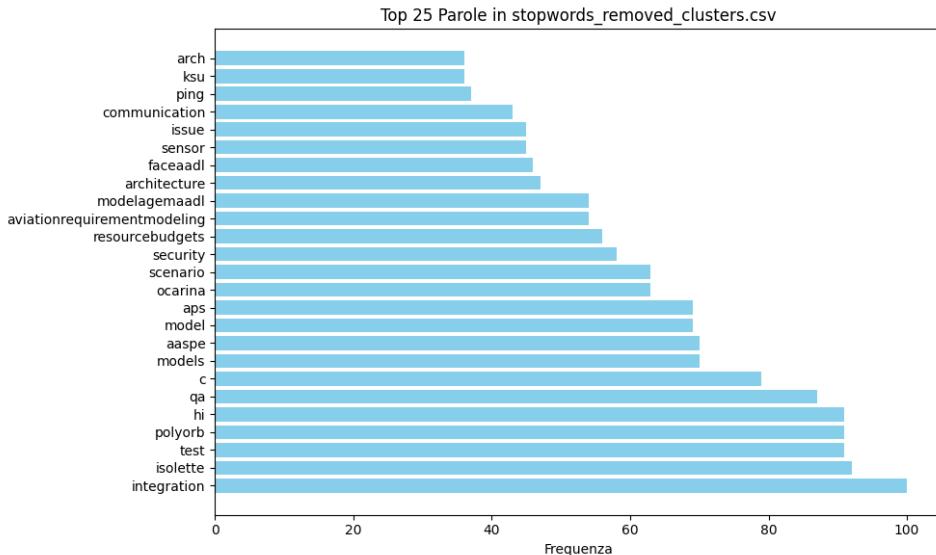


Figura 4.12: Classifica dei 25 token più ricorrenti per i nomi di file AADL dopo aver rimosso le stopwords.

base. Questo passaggio è essenziale per ridurre la variabilità dei dati, garantendo che parole che rappresentano lo stesso concetto siano trattate come entità equivalenti.

La lemmatizzazione viene effettuata utilizzando **WordNet**, una vasta base di dati lessicale che fornisce il lemma di ogni parola, facilitando la riduzione delle varianti. WordNet è un dizionario semantico utilizzato per associare parole con il loro lemma e per identificare la relazione tra di esse. Questo approccio è applicato ai token, che vengono ridotti alla loro forma fondamentale, migliorando la coerenza semantica dei dati.

La fase di preprocessing si conclude con un’ultima normalizzazione che prevede la rimozione di tutti i token la cui lunghezza è pari a uno. La presenza di token di questo tipo introduce degli elementi di rumore all’interno del processo di labelling, pertanto vengono esclusi.

I risultati ottenuti dalle normalizzazioni sono mostrati nelle Figure 4.13, 4.14 e 4.15, che evidenziano come le modifiche applicate abbiano migliorato la qualità dei dati per il processo di labelling.

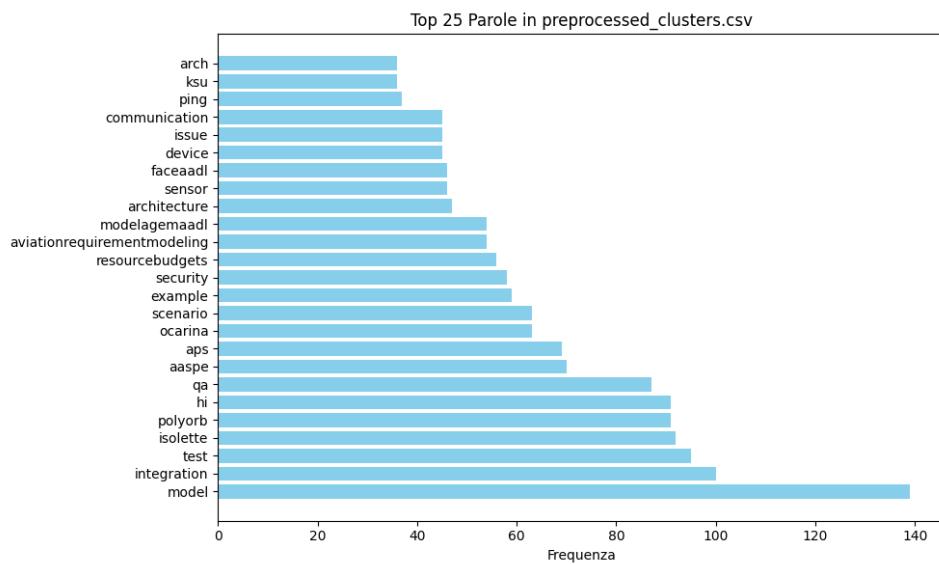


Figura 4.13: Classifica dei 25 token più ricorrenti per i nomi di file AADL dopo il preprocessing.

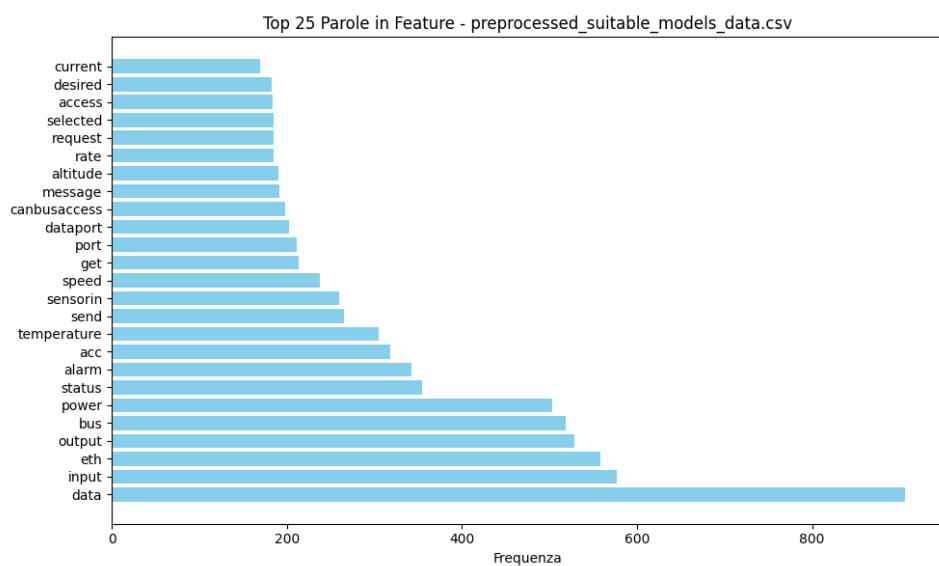


Figura 4.14: Classifica dei 25 token più ricorrenti per i nomi di feature dopo il preprocessing.

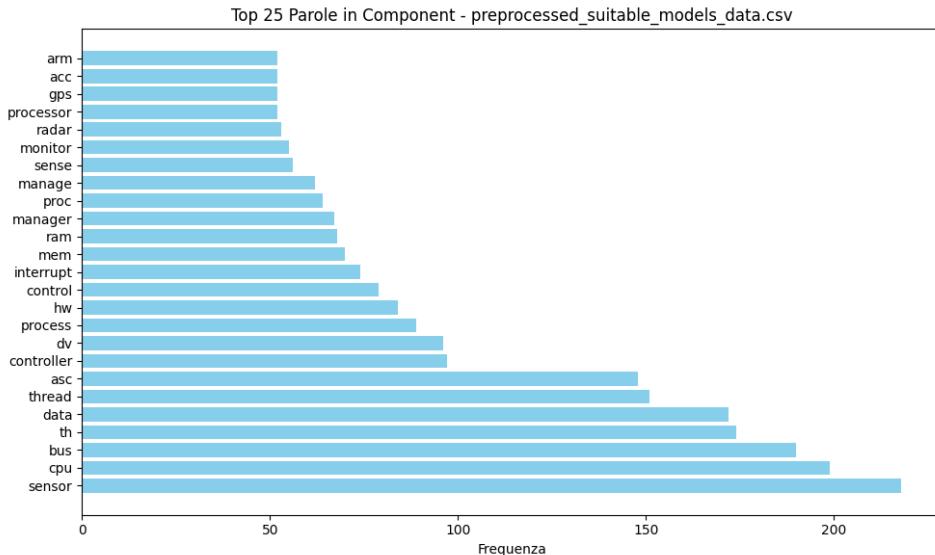


Figura 4.15: Classifica dei 25 token più ricorrenti per i nomi di componenti dopo il preprocessing.

Le modifiche apportate al contenuto estratto dai nomi dei componenti e delle feature dei modelli hanno causato, per alcuni modelli, la rimozione totale delle parole associate a questi elementi. In particolare, sono stati esclusi 7 modelli che non contengono parole normalizzate né per le componenti né per le feature. Questi modelli, ormai privi di contenuto, erano in precedenza caratterizzati dalla presenza di componenti e feature i cui nomi rientravano all'interno di una delle casistiche di normalizzazione adottate che prevedeva la rimozione delle parole. Di conseguenza, questi modelli sono stati esclusi dal processo di labelling che utilizza i nomi delle componenti e delle feature per individuare le parole chiave. Tuttavia, sono stati comunque mantenuti nel processo di generazione delle etichette, che si basa sul nome del file del modello.

4.5 Generazione delle etichette

Ora che il contenuto estratto dai modelli è stato normalizzato, è possibile procedere con l'applicazione delle tecniche di labelling per individuare le parole chiave da utilizzare come etichette da associare ai cluster. Il processo di generazione delle etichette applica le tecniche **Term Frequency-Inverse Document Frequency** (TF-IDF) e **Latent Dirichlet Allocation** (LDA) per individuare le parole chiave per ogni cluster e utilizzarle per la generazione delle etichette da associare ai cluster stessi.

Queste tecniche vengono applicate sia ai nomi dei file dei modelli che ai nomi delle componenti e delle feature estratte dai modelli. Nel caso delle componenti e delle feature, si è scelto di combinarne il contenuto in un'unica stringa prima di generare le etichette, poiché, per alcuni modelli, la presenza simultanea di componenti e feature non è garantita a seguito del preprocessing. Se per un dato cluster non sono disponibili informazioni relative sia ai nomi delle componenti che delle feature, tale cluster viene escluso dalla generazione delle etichette basate su questi elementi.

A seguito dell'individuazione delle parole più rilevanti, vengono utilizzati dei valori di soglia per selezionare i termini da utilizzare come etichette da attribuire a ogni cluster.

4.5.1 TF-IDF

Il calcolo dei punteggi TF-IDF per le parole avviene separatamente per ciascun cluster. In questo modo, vengono estratte le parole chiave che appartengono esclusivamente ai modelli contenuti all'interno di un dato cluster, ottenendo così i termini che meglio descrivono il contenuto del cluster stesso. La tecnica applicata si occupa di generare la matrice TF-IDF, che contiene i punteggi attribuiti a ciascuna parola presente nel cluster, in relazione ai modelli che ne fanno parte.

Il risultato di questa operazione è una lista di termini ordinati in base al loro punteggio TF-IDF, che rappresentano i termini più distintivi e informativi di ogni cluster. Le prime dieci parole, ordinate per punteggio, per ogni cluster sono poi salvate all'interno di un file, il cui contenuto per i primi cinque cluster è mostrato in Tabella 4.3 e 4.5.

<i>Cluster</i>	<i>Top 10 Parole per Cluster(TF-IDF)</i>	<i>Punteggi delle parole</i>
1	isolette, ksu, sensor, single, thermostat	7.239, 3.331, 3.331, 3.331, 3.306
2	isolette, qa, agree, aviationrequirementmodeling, temperature, connectioninheritance, dual, ksu, sensor, nameresolution	6.761, 4.289, 2.615, 2.615, 2.615, 2.479, 2.254, 2.254, 2.254, 2.08
3	gca, singleton, aps, aviationrequirementmodeling, tank, aviation, isolette, vtv, pressurized, aviate	2.868, 2.137, 2.055, 2.008, 1.946, 1.624, 1.512, 1.33, 1.313, 1.087
4	smartparking, temperature, aps, aviationrequirementmodeling, aviation, architecture, communication, agree, sensor, isolette	5.424, 3.7, 3.68, 2.968, 2.453, 2.359, 2.359, 2.321, 2.256, 2.21
5	tcas, aviationrequirementmodeling, aviation, singleton, single, block, ev, lgs, tpaadl, refuel	9.449, 8.474, 3.96, 3.675, 2.387, 2.141, 1.862, 1.862, 1.862, 1.741

Tabella 4.3: Porzione di classifica delle prime 10 parole per score TF-IDF ottenute dai nomi dei file dei modelli e i relativi punteggi.

<i>Cluster</i>	<i>Top 10 Parole per Cluster (TF-IDF)</i>	<i>Punteggi delle parole</i>
1	temp, heat, alarm, heater, loss, sensor, temperature, thermostat, air, source	1.097, 1.096, 0.844, 0.759, 0.651, 0.651, 0.651, 0.651, 0.59, 0.59
2	pin, sys, dataport, pout, tin, datain, dataout, nan, internetaccess, aiscommunication	2.265, 1.746, 1.601, 1.307, 1.172, 1.155, 1.155, 1.155, 1.057, 1.001
3	temp, dataport, eth, gnss, datain, mph, dataout, distance, latitude, longitude	1.391, 0.95, 0.95, 0.95, 0.914, 0.817, 0.697, 0.671, 0.671, 0.671
4	temperature, req, info, traveler, asc, maindataport, loca, update, alarm, personal	2.628, 2.409, 2.066, 2.035, 1.937, 1.737, 1.635, 1.613, 1.564, 1.508
5	link, bus, mph, access, timetofailure, timetorecovery, nearestintruderaltitude, cc, fpadata, nearestintrudergrounddistance	1.357, 1.327, 1.252, 1.24, 1.211, 1.211, 0.962, 0.846, 0.838, 0.794

Tabella 4.5: Porzione di classifica delle prime 10 parole per score TF-IDF ottenute dai nomi delle componenti e delle feature con i relativi punteggi.

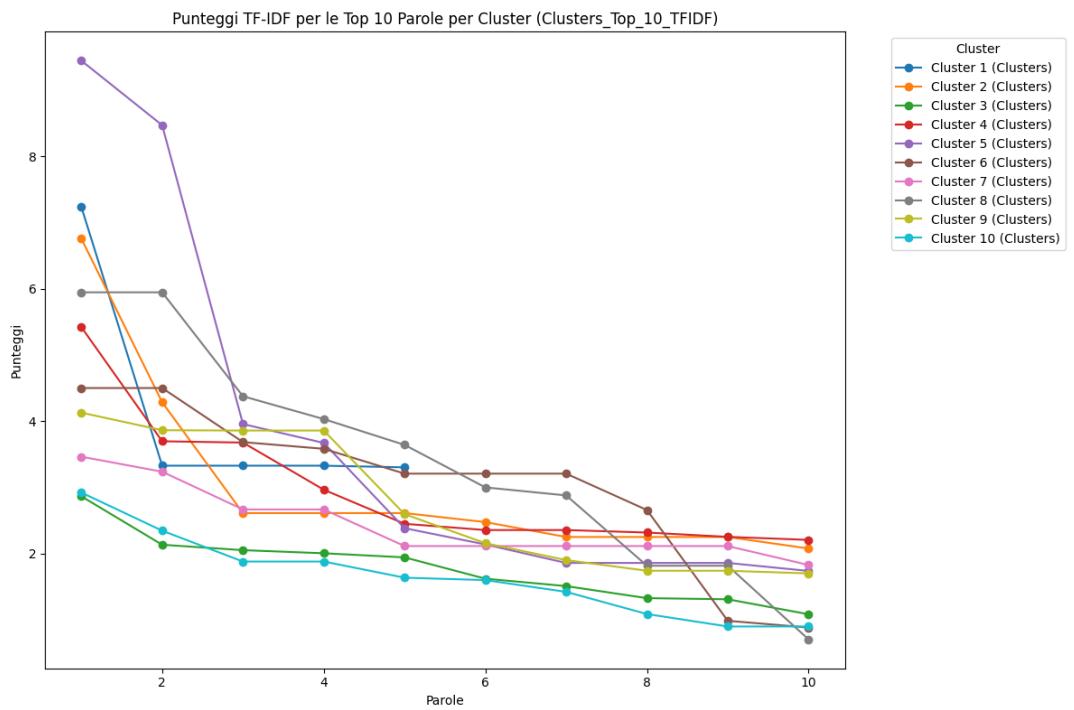


Figura 4.16: Andamento degli score TF-IDF per le parole ottenute dai nomi dei file dei modelli.

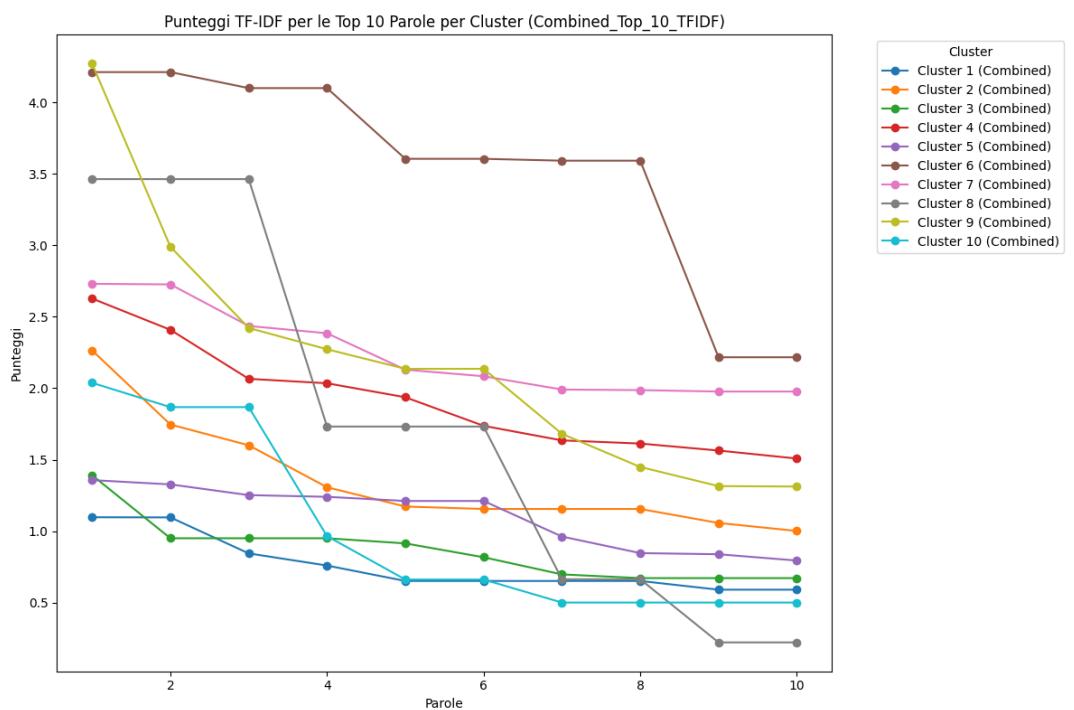


Figura 4.17: Andamento degli score TF-IDF per le parole ottenute dai nomi delle componenti e delle feature dei modelli.

I grafici in Figura 4.16 e 4.17, invece, mostrano l’andamento dei punteggi attribuiti alle parole per cluster di appartenenza. Il contenuto mostrato nelle figure evidenzia come per alcuni cluster vi sia un picco significativo di punteggio per il primo termine, che indica che la prima parola chiave di questo cluster è particolarmente distintiva e rilevante. Per altri cluster, invece, la distribuzione dei punteggi è più uniforme, suggerendo una minore variabilità nella rilevanza delle parole chiave. I punteggi per i nomi dei file dei modelli sono generalmente più variabili tra i cluster rispetto a quelli ottenuti dalle componenti e feature. Questo indica che i nomi dei file tendono a fornire informazioni più specifiche e variegate, mentre la combinazione di componenti e feature risulta in una distribuzione di punteggi meno variabile, suggerendo una rappresentazione più uniforme dei cluster.

L’analisi della classifica delle parole con i punteggi TF-IDF più alti ha portato alla decisione di applicare un ulteriore filtro prima di selezionare le parole da utilizzare come etichette da assegnare ai cluster. In particolare, per essere ritenute adeguate, le parole devono superare un valore di soglia impostato manualmente. Tale valore, determinato a seguito dell’analisi dei punteggi medi di TF-IDF per ogni cluster, è stato fissato a 1. Il numero massimo di parole selezionabili come etichette per ciascun cluster è limitato a 5. Pertanto, se più di cinque parole superano il valore di soglia, vengono selezionate solo le cinque parole con il punteggio più alto. Nel caso in cui per un dato cluster non fossero presenti parole il cui punteggio superi la soglia, si è deciso di estrarre comunque la parola con il punteggio più alto, garantendo così che ogni cluster abbia almeno un’etichetta associata.

Le etichette estratte secondo i criteri appena descritti vengono poi salvate in un file di cui viene mostrata una porzione nelle Tabelle 4.7 e 4.9. Queste etichette rappresentano le parole chiave per ciascun cluster, ottenute tramite l’applicazione delle tecniche TF-IDF sui nomi dei modelli e sulla combinazione dei nomi delle componenti e feature normalizzati.

<i>Cluster</i>	<i>Etichette per Cluster (TF-IDF)</i>	<i>Punteggi delle parole</i>
1	isolette, ksu, sensor, single, thermostat	7.239, 3.331, 3.331, 3.331, 3.306
2	isolette, qa, agree, aviationrequirementmodeling, temperature	6.761, 4.289, 2.615, 2.615, 2.615
3	gca, singleton, aps, aviationrequirementmodeling, tank	2.868, 2.137, 2.055, 2.008, 1.946
4	smartparking, temperature, aps, aviationrequirementmodeling, aviation	5.424, 3.7, 3.68, 2.968, 2.453
5	tcas, aviationrequirementmodeling, aviation, singleton, single	9.449, 8.474, 3.96, 3.675, 2.387

Tabella 4.7: Porzione di etichette ottenute applicando TF-IDF ai nomi dei file dei modelli.

<i>Cluster</i>	<i>Etichette per Cluster (TF-IDF)</i>	<i>Punteggi delle parole</i>
1	temp, heat	1.097, 1.096
2	pin, sys, dataport, pout, tin	2.265, 1.746, 1.601, 1.307, 1.172
3	temp	1.391
4	temperature, req, info, traveler, asc	2.628, 2.409, 2.066, 2.035, 1.937
5	link, bus, mph, access, timetofailure	1.357, 1.327, 1.252, 1.24, 1.211

Tabella 4.9: Porzione di etichette ottenute applicando TF-IDF ai nomi delle componenti e delle feature dei modelli.

4.5.2 LDA

La seconda tecnica applicata per la generazione delle etichette da attribuire ai cluster è la **Latent Dirichlet Allocation**. LDA è un modello generativo utilizzato per identificare i topic latenti all'interno di un insieme di documenti, ovvero una distribuzione di parole che permette di identificare i principali temi o concetti che caratterizzano, in questo caso, i cluster di modelli AADL. La generazione di etichette attraverso l'utilizzo di topic propone un approccio sensibilmente differente rispetto a quello adottato da TF-IDF, che si limita alla selezione di parole singole. Applicando LDA, è possibile ottenere etichette più strutturate, che rappresentano concetti più ampi e facilmente comprensibili all'utente. Questo approccio rende le etichette generate più adatte a descrivere i cluster in modo significativo.

Come per la tecnica precedente, LDA viene applicato separatamente per ciascun cluster, sia ai nomi dei file dei modelli che alle combinazioni di componenti e feature estratte e preprocescate. Il codice che implementa LDA si occupa della gestione delle operazioni di vettorizzazione del contenuto dei modelli e per l'applicazione dell'algoritmo LDA.

Uno degli aspetti più critici nell'applicazione di LDA è la scelta dei parametri, in particolare il numero di topic da identificare. Il numero di topic deve essere scelto in modo tale da rappresentare adeguatamente i diversi aspetti dei modelli AADL, senza che questi siano troppo generici o troppo specifici. Per questo motivo, nel codice, il numero ottimale di topic viene determinato dinamicamente utilizzando il parametro **perplexity**. Questa misura è utilizzata per valutare la qualità del modello LDA. Un valore più basso di perplexity indica un modello migliore, poiché significa che il modello è in grado di descrivere meglio i dati. Il codice implementato esegue una ricerca per determinare il numero ottimale di topic testandone i valori da 2 a 10 e selezionando il numero che minimizza la perplexity. In generale, la perplexity tende a diminuire con l'aumento del numero di topic, ma il calcolo di questo valore eseguito su un numero variabile di topic consente di evitare il rischio di sovradimensionare il numero di topic e ottenere una rappresentazione più precisa e utile dei modelli. Come mostrato nelle Figure 4.18 e 4.19, in alcuni casi il numero ottimale di topic non corrisponde al valore massimo testato, ma piuttosto al punto in cui la perplexity è minimizzata, migliorando così la qualità del modello. L'utilizzo di questa misura consente quindi di individuare il numero ottimale di topic per ogni singolo cluster del dataset, migliorando notevolmente i risultati ottenuti rispetto all'utilizzo di un parametro fisso.

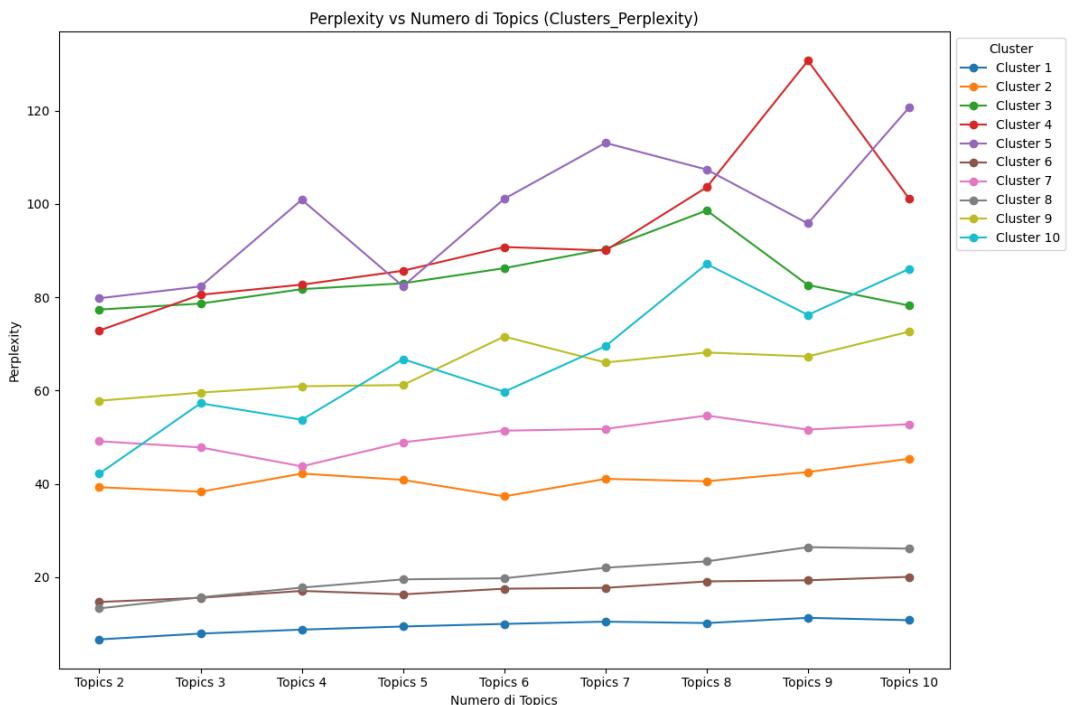


Figura 4.18: Andamento dei valori di perplexity al variare del numero di topic ottenuti dai nomi dei file dei modelli.

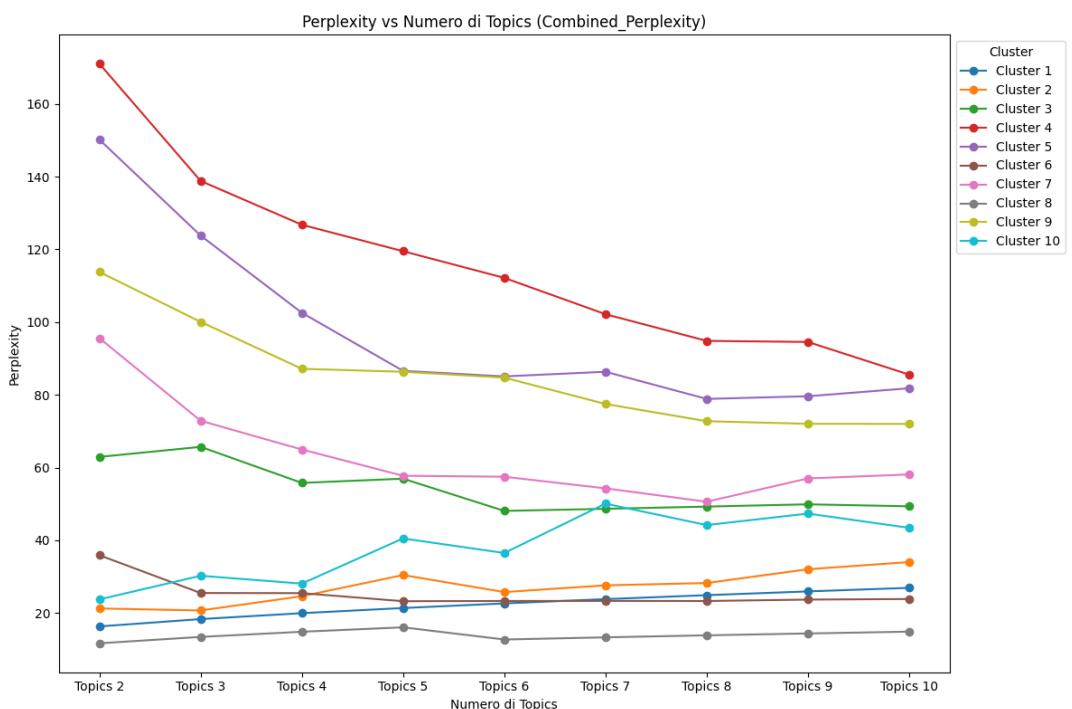


Figura 4.19: Andamento dei valori di perplexity al variare del numero di topic ottenuti dai nomi delle componenti e delle feature dei modelli.

Il secondo parametro fondamentale per l'algoritmo LDA riguarda il numero di parole da associare a ciascun topic. Come nel caso di TF-IDF, anche in questo caso è stata adottata una soglia per identificare le parole da associare ai topic e scartare quelle meno rilevanti. La soglia di probabilità, impostata a 2, rappresenta il valore minimo che una parola deve avere rispetto a un dato topic per poter essere assegnata a quest'ultimo. In altre parole, solo le parole che superano questa soglia di probabilità vengono considerate significative e quindi associate al topic. Inoltre, per evitare che, in alcuni casi, nessuna parola superi la soglia per un determinato topic, si è deciso di impostare un numero minimo di parole pari a 5 per ogni topic. Questo valore è stato scelto considerando che, generalmente, i topic latenti individuati da LDA sono descritti da un insieme di parole. Limitare questo numero a un valore troppo basso potrebbe ridurre la capacità dell'algoritmo di rappresentare adeguatamente i temi latenti, compromettendo l'efficacia del modello. La scelta di associare almeno 5 parole per topic permette di garantire che ogni topic contenga abbastanza informazioni per rappresentare correttamente i concetti emersi durante l'applicazione dell'algoritmo.

Utilizzando i parametri descritti in precedenza, l'algoritmo LDA è stato applicato per individuare i topic e assegnare le parole chiave ai cluster. Per ciascun cluster, sono stati generati set di topic basati sia sul contenuto dei nomi dei file che sulle combinazioni di componenti e feature dei modelli AADL. Il numero di topic associato a ciascun cluster è mostrato in Figura 4.20. Dal grafico emerge chiaramente che il numero di topic identificati utilizzando le combinazioni di componenti e feature è significativamente superiore rispetto a quello ottenuto utilizzando i soli nomi dei file. Questo risultato è principalmente dovuto al volume di informazioni contenute nelle due tipologie di dati analizzate. In particolare, i dati derivanti dalle combinazioni di componenti e feature contengono un numero molto maggiore di termini rispetto ai nomi dei file. Questa differenza comporta un impatto sul calcolo del numero ottimale di topic tramite l'uso della perplexity. Per i dati relativi alle combinazioni di componenti e feature, è necessario un numero maggiore di topic per rappresentare adeguatamente la varietà di informazioni, mentre per i nomi dei file, essendo il numero di termini più contenuto, è sufficiente un numero ridotto di topic per rappresentare in modo efficace le informazioni contenute nel testo.

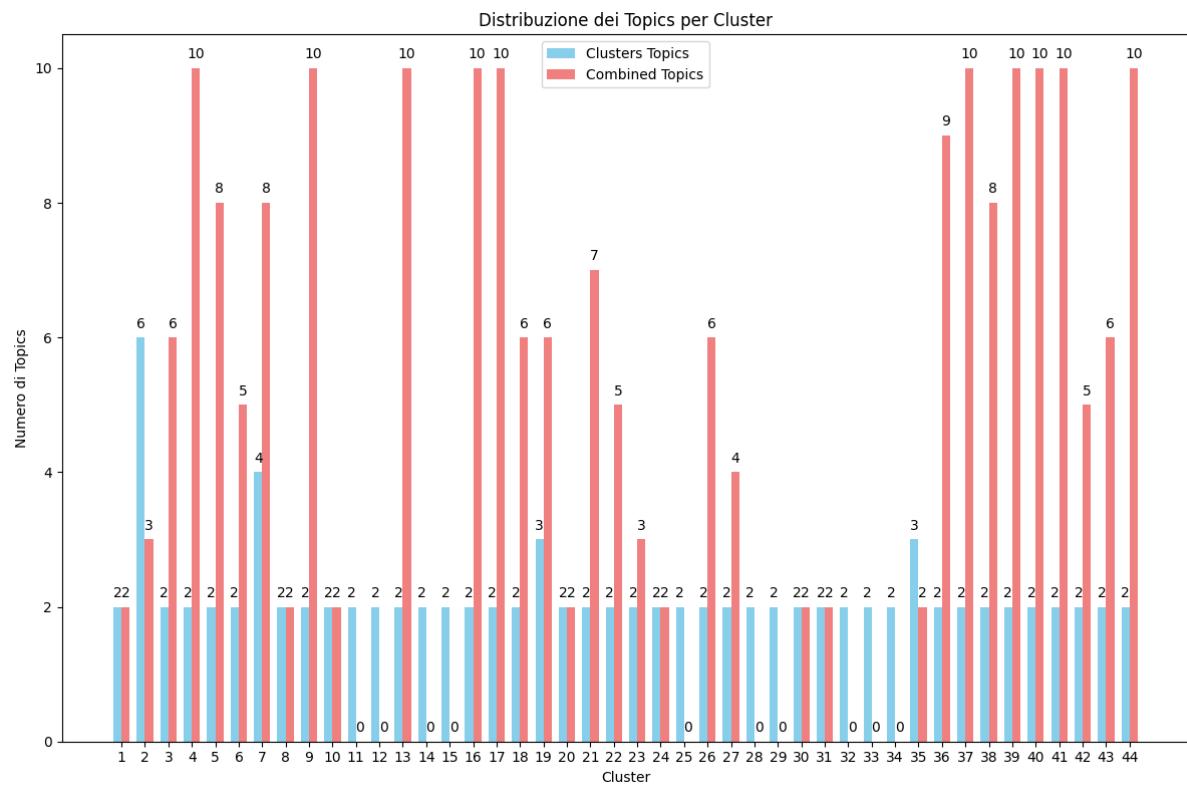


Figura 4.20: Distribuzione dei topic per ogni cluster

A conclusione del processo di applicazione dell'algoritmo di labelling, le etichette ottenute utilizzando i topic identificati da LDA vengono assegnate ai cluster. I set di etichette ottenute sia dai nomi dei file che dai nomi delle componenti e feature vengono poi salvate in un file di cui viene mostrata una porzione in Tabella 4.11.

<i>Cluster</i>	<i>Top Topics per nome file (LDA)</i>	<i>Top Topics per componenti e feature (LDA)</i>
1	isolette thermostat ksu single sensor, isolette sensor single ksu thermostat	temp alarm current heater manage, heat sensor thermostat temperature loss
2	communication aps architecture apscommunication cloudcomponent, qa connectioninheritance top nameresolution se, isolette sensor dual ksu panel, sender authinaadl check resolve dataflow, cwe aaspe security array success, temperature agree aviationrequirementmodeling cooler heater	pin pout sys tin nestpfg, nan datain dataout apsstakeholder tin, dataport trafficmodule aiscommunication internetaccess ship
3	gca aps singleton flying pilot, tank aviate control level isolette	temp status desired alarm upper, mph autospd refspd cruiseactive throttlepos, process first second dataout datain, datain eth gnss dataport dataout, process first second dataout datain, latitude distance longitude outimpact inppc
4	aps architecture communication isolette smartparking, temperature aviationrequirementmodeling agree smartparking aviation	req info paymt loca personal, envout sensorflowpath varspeed sensorscontrol envin, data receive engine state action, asc maindataport dataport backup backupdataport, op ctrl status inf vloc, altitude aoa word discrete baro, park space occupancy update req, temperature alarm monitor regulator failure, timetofailure timetorecovery gca tcas geof, selected weapon radar target body
5	cruisecontrol block refuel motor pump, aviationrequirementmodeling tcas singleton aviation single	fpadata rawdata compressed access bus, cc ev gear door retracted, timetofailure timetorecovery link access bus, compressed processing raw output ctrlpart, temperature alarm monitor regulator failure, pc part bustype pp compressed, mph refspd cruiseactive nearestintruder altitude autospd, th true heading acc set
6	dronesystem hybridsynchaadl master rendezvous thesis, hybridsynchaadl environment dronecontrol thesis rendezvous	accx accy currvx currvy curry, d2 d1 door warn info, doty dotx vely velx dotdoty, cy cx vy vx ox, currx curry vely velx inx

Tabella 4.11: Porzione di etichette ottenute applicando LDA ai cluster di modelli AADL.

5. Analisi dei risultati

In questo capitolo finale vengono illustrati i ragionamenti e le metodologie adottate per eseguire una validazione delle etichette generate nel processo di labelling dei cluster di modelli architetturali. La validazione è un passaggio fondamentale per assicurarsi che le etichette assegnate ai cluster siano effettivamente rappresentative e utili per la consultazione dei modelli. Senza una verifica indipendente dei risultati, non sarebbe possibile confermare che le etichette generate rappresentino effettivamente caratteristiche significative e comuni dei modelli presenti nei singoli cluster.

L'elemento indispensabile per effettuare queste valutazioni in modo puntuale è la **ground truth**. Essa rappresenta un insieme di dati di riferimento che consente una valutazione oggettiva dell'accuratezza e della coerenza delle etichette generate. A seguito della generazione di questa componente, si effettuano diverse misure di confronto tra i risultati ottenuti dal processo di etichettatura automatica e i dati della ground truth. In prima battuta, si misura la **cosine similarity**, o similarità coseno, per determinare quanto le etichette generate rappresentino correttamente il contenuto dei cluster. Infine, si procede con il calcolo di alcune metriche di classificazione, come la **precision**, **recall** e **F1**, per valutare la qualità complessiva delle etichette assegnate.

5.1 Ground truth

Nel contesto della validazione delle etichette assegnate ai cluster di modelli architetturali AADL, la ground truth rappresenta un elemento fondamentale. L'obiettivo di creare una ground truth è disporre di un sistema di riferimento che permetta di verificare l'accuratezza e la qualità delle etichette assegnate dai modelli di labelling, come quelli ottenuti tramite le tecniche di TF-IDF e LDA. Poiché la validazione delle etichette è strettamente legata alla qualità e rappresentatività di queste, si decide di procedere con un'analisi approfondita dei modelli contenuti in ogni cluster, assegnando ad essi delle etichette che ne riassumano le caratteristiche principali. Nel caso specifico dei cluster di modelli architetturali AADL, la ground truth viene costruita associando ad ogni cluster una o più etichette rappresentative, ottenute attraverso un'analisi approfondita dei nomi dei modelli, delle loro componenti e feature. Questo processo è cruciale, in quanto

consente di validare la correttezza delle etichette assegnate ai cluster dai metodi di etichettatura automatica, confrontandole con le etichette generate manualmente più rappresentative dei gruppi di modelli.

Dato che le etichette generate automaticamente sono attribuite ai cluster, si opta per creare una **ground truth per cluster**, anziché per singolo modello, principalmente per semplificare il processo di etichettatura. Questo approccio consente di associare almeno un'etichetta a ogni cluster, dato che il numero di cluster è significativamente inferiore rispetto al numero di modelli AADL contenuti al loro interno. In questo modo, si riduce la complessità del lavoro manuale, evitando di etichettare ogni singolo modello.

Per la generazione delle etichette, si effettua un'analisi accurata dei nomi dei modelli e delle relative componenti e feature. In particolare, per evitare di introdurre rumore, si è scelto di escludere quei modelli la cui categoria era presente solo una volta all'interno del cluster. Questo garantisce che le etichette siano rappresentative della maggior parte dei modelli presenti nei cluster, senza includere categorie marginali che potrebbero influenzare negativamente la qualità delle etichette. In situazioni in cui modelli appartenenti a categorie diverse coesistono all'interno dello stesso cluster, si creano etichette distinte per ciascuna delle categorie. Ad esempio, se un cluster contiene sia modelli di "*sensori di temperatura*" sia modelli di "*smart parking*", si generano due etichette separate per riflettere entrambe le categorie di modelli. Se, invece, un cluster contiene solo modelli appartenenti a una singola categoria, si assegna una sola etichetta. Se un modello appartenente a una categoria compare solo una volta all'interno del cluster, non si genera alcuna etichetta per quel gruppo.

Nel caso in cui le informazioni contenute nei nomi dei modelli e nelle relative componenti e feature non siano sufficienti per determinare un'etichetta chiara e significativa, si decide di non assegnarne alcuna. Tuttavia, quando questa situazione si verifica per tutti i modelli di un cluster, si genera un'etichetta che include le parole più ricorrenti tra i modelli, al fine di averne almeno una per cluster.

La ground truth, generata secondo le regole precedentemente descritte, è parzialmente mostrata nella Tabella 5.2. Le etichette riportate in essa vengono utilizzate per misurare l’accuratezza delle etichette generate automaticamente, attraverso il confronto tra le due.

<i>Cluster</i>	<i>Etichette</i>
1	isolette single sensor thermostat temperature alarm
2	security failure datain dataout, isolette dual sensor temp alarm, communication aps architecture dataport
3	isolette temp monitoring system with alarm, communication architecture for gnss security, aviate control altitude latitude longitude distance speed, tank pressurized valve switch
4	smartparking system for vehicle considering space occupancy and capacity, aviation temperature controller sensor, isolette temperature sensor monitor heat with thermostat and alarm, aps communication architecture
5	aviation supervisor for altitude gcas tcas latitude longitude, car cruisecontrol for speed velocity and throttle, blocks memory management using bus

Tabella 5.2: Porzione di etichette generate per la ground truth.

Per preparare le etichette ottenute da TF-IDF e LDA al confronto con la ground truth, i dati devono essere opportunamente elaborati. Questo processo garantisce che tutte le etichette generate automaticamente siano unificate e possano essere confrontate in modo coerente, senza incorrere in difficoltà derivanti da etichette troppo frammentate o incomplete.

Per entrambe le tecniche di labelling, il primo passo consiste nell’unire le etichette generate tramite i nomi dei file con quelle derivate dalle componenti e feature dei modelli AADL. La combinazione delle etichette è particolarmente importante poiché non tutti i cluster possiedono delle etichette generate dai nomi delle componenti e delle feature. Tuttavia, ogni cluster ha sempre almeno un’etichetta generata dai nomi dei file dei modelli. Unificando le due tipologie di etichette, si ottiene un unico elenco di etichette associate a ogni cluster, garantendo così

la presenza di almeno un’etichetta per ciascun cluster. Questo approccio assicura che nessun cluster venga privato di un’etichetta e che ogni etichetta generata possa essere confrontata con la ground truth.

Per quanto riguarda il trattamento specifico delle etichette ottenute da TF-IDF, viene eseguita un’ulteriore operazione: tutte le singole parole identificate nel processo di etichettatura vengono concatenate in un’unica etichetta. Questo passaggio è essenziale, in quanto confrontare le etichette della ground truth con parole singole avrebbe poco senso e produrrebbe dei risultati non soddisfacenti nel calcolo delle metriche di confronto. La concatenazione di più parole in una singola etichetta garantisce che il confronto con la ground truth, che generalmente include descrizioni più complete, sia significativo e coerente. Tale unione delle etichette non è necessaria nel caso di LDA, in quanto i topic individuati dall’algoritmo sono composti da diverse parole.

5.2 Cosine Similarity

La prima misura calcolata per confrontare ground truth ed etichette generate automaticamente è la similarità coseno; una misura di somiglianza tra due vettori in uno spazio n-dimensionale, ampiamente utilizzata nell’ambito del **Natural Language Processing** (NLP) per misurare la somiglianza tra due documenti o, come in questo caso specifico, due etichette. Tanto più la rappresentazione delle etichette generate attraverso LDA e TF-IDF è simile ai vettori ottenuti dalla ground truth, tanto più sarà alto il valore di similarità coseno.

La cosine similarity calcola il coseno dell’angolo tra due vettori ed è definita dalla formula:

$$\cos \theta = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i \cdot B_i}{\sqrt{\sum_{i=1}^n A_i^2} \cdot \sqrt{\sum_{i=1}^n B_i^2}}$$

dove A e B sono i vettori da confrontare, mentre n è il numero di componenti di ciascun vettore. Il prodotto scalare dei due vettori A e B , al numeratore, è ottenuto dalla somma dei prodotti delle componenti dei vettori che va da i a n , mentre a denominatore si calcola il prodotto delle norme dei singoli vettori.

Nel calcolo della cosine similarity nel codice, il processo si svolge in vari passaggi determinati da scelte operative per il caso d’uso specifico. L’approccio di base consiste nell’ottenere una rappresentazione vettoriale delle etichette per ogni cluster, sia della ground truth che delle etichette generate automaticamente, e nel calcolare la somiglianza coseno tra queste rappresen-

tazioni. I vettori numerici che rappresentano le etichette sono ottenuti attraverso l'utilizzo di **Word2Vec**, un modello pre-addestrato di *word embedding* che rappresenta ogni parola in uno spazio vettoriale continuo.

Tuttavia, esiste una criticità importante nel rappresentare vettorialmente le etichette generate manualmente nella ground truth e quelle generate automaticamente tramite TF-IDF e LDA. Poiché le etichette manuali potrebbero essere espresse in modo diverso rispetto a quelle automatiche, ad esempio attraverso l'uso di sinonimi o terminologie differenti, è altamente improbabile che queste etichette siano rappresentate in modo identico nei vettori. Ciò potrebbe comportare misurazioni inaccurate nel calcolo della cosine similarity, con risultati che non riflettono realmente la somiglianza tra le etichette.

Per risolvere questo problema, si è scelto di utilizzare **WordNet**, una libreria linguistica che mappa le parole a *synset*, ovvero gruppi di parole che condividono lo stesso significato, permettendo così di normalizzare le parole e di mappare termini con significati identici su una forma comune. Utilizzando WordNet, le parole simili vengono convertite in lemmi o sinonimi, che sono rappresentati nello stesso modo, creando così una rappresentazione coerente e identica dello stesso concetto. Questo approccio consente di ottenere una rappresentazione più precisa e veritiera delle etichette, in quanto le parole, pur essendo espresse in modo diverso, vengono normalizzate nel loro significato. Di conseguenza, i vettori risultanti dalle etichette manuali e da quelle automatiche saranno più simili tra loro, e le misure di somiglianza, come la cosine similarity, restituiranno risultati più accurati e significativi.

Una volta ottenuti i vettori di ground truth e delle etichette generate, il calcolo della cosine similarity tra di essi viene effettuato tramite una funzione che calcola la somiglianza coseno tra il vettore rappresentante l'etichetta della ground truth e il vettore corrispondente alle etichette generate automaticamente. I risultati ottenuti per TF-IDF e LDA sono riassunti graficamente nelle figure 5.1 e 5.2.

Entrambe le tecniche mostrano una capacità di produrre etichette che sono simili a quelle della ground truth, ma con performance variabili tra i cluster. In generale, LDA ottiene risultati più coerenti, con valori di cosine similarity relativamente stabili tra i cluster, mentre TF-IDF mostra una maggiore variabilità nei risultati, con alcuni cluster che ottengono valori molto elevati di similarità, mentre altri presentano somiglianze più basse.

Per comprendere le motivazioni che hanno portato a ottenere dei valori di similarità coseno nettamente differenti tra le due tecniche, è necessario approfondire l'analisi, osservando le

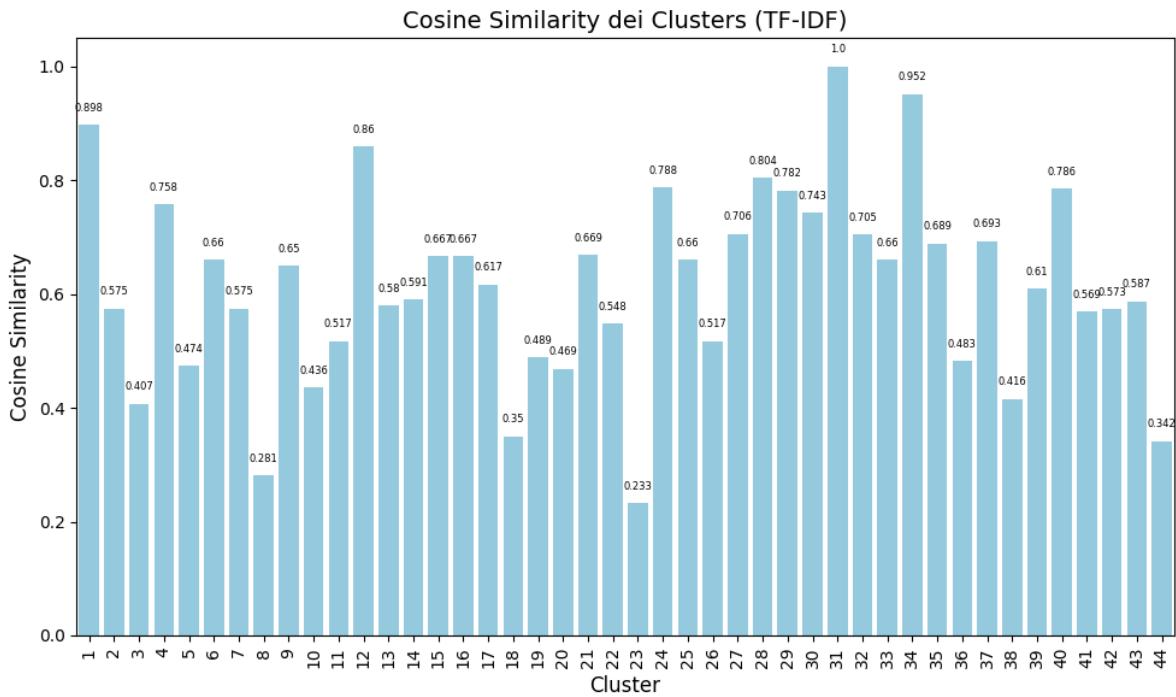


Figura 5.1: Valore delle similarità coseno per cluster ottenuta confrontando ground truth e etichette di TF-IDF.

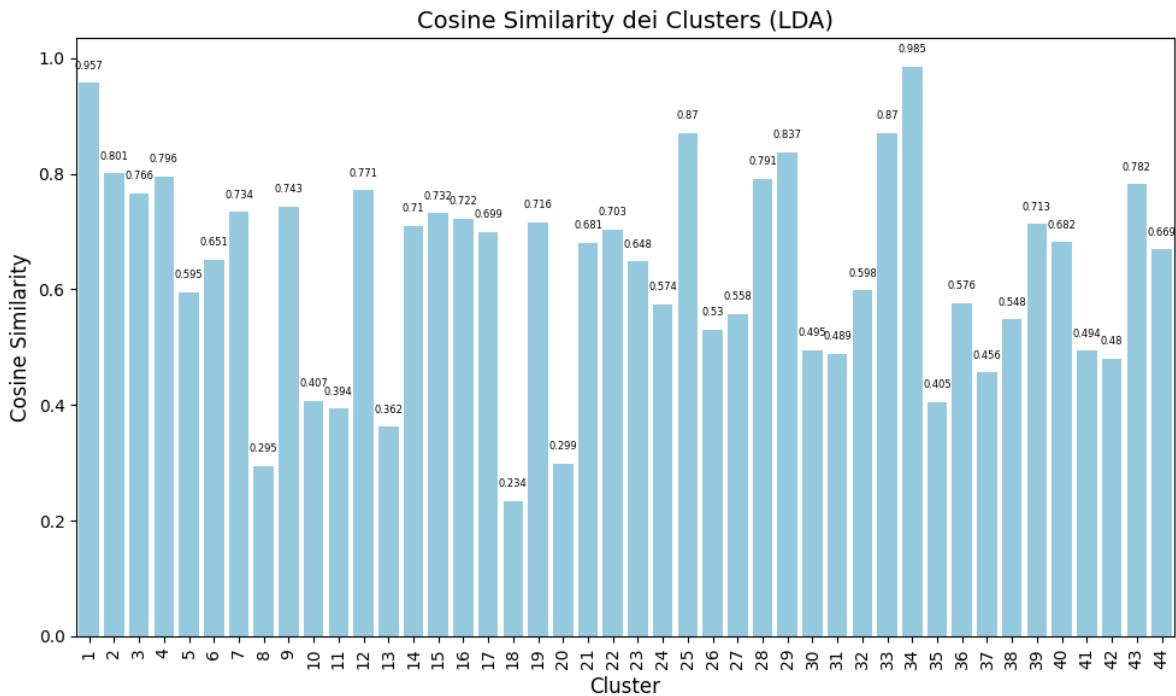


Figura 5.2: Valore delle similarità coseno per cluster ottenuta confrontando ground truth e etichette di LDA.

etichette generate dai metodi automatici e, eventualmente, la struttura generale dei modelli nei cluster.

Ad esempio, nel **cluster 31**, il punteggio di cosine similarity per TF-IDF è pari a **1.0**, mentre per LDA equivale a **0.489**. Come mostrato nella Tabella 5.4, l'utilizzo di TF-IDF genera un'etichetta identica a quella della ground truth. LDA, invece, genera etichette che comprendono tutte le parole presenti nella ground truth, ma introduce anche altre parole prive di significato che non contribuiscono a rappresentare correttamente il contenuto del cluster. Questo fenomeno è causato dalla struttura dei modelli presenti nel cluster, dove il nome del file è sempre **"iplprojects mapa map"**, mentre il contenuto delle componenti e delle feature dei modelli è una combinazione dei termini **"edge"** e **"l1"**, **"l2"**, **"l3"**, ecc. La ripetitività delle parole all'interno dei modelli ha determinato la corretta estrazione delle parole chiave da parte del modello TF-IDF, mentre la presenza di numerose componenti e feature dal basso contenuto informativo ha causato la loro inclusione nelle etichette generate da LDA.

Un caso opposto si riscontra nel **cluster 23**, dove LDA ottiene un punteggio di **0.684**, mentre TF-IDF ha un valore di similarità coseno pari a **0.233**. In questo caso, la massiccia presenza di nomi di componenti e feature come **"port"** e le sue varianti **"outport"** e **"inport"** ha causato l'estrazione di parole chiave dal basso contenuto informativo da parte di TF-IDF che, come mostrato in Tabella 5.4, ha comportato la generazione di etichette poco pertinenti. Al contrario, i topic individuati da LDA hanno identificato correttamente i temi latenti presenti nel contenuto dei modelli, generando delle etichette più coerenti rispetto alla ground truth.

Vi sono, tuttavia, dei casi in cui entrambi i modelli non ottengono dei valori di similarità coseno soddisfacenti. Per il **cluster 8**, infatti, il valore di cosine similarity ottenuto da TF-IDF è pari a **0.281** e quello ottenuto da LDA è **0.295**. I modelli appartenenti a questo cluster, a seguito del preprocessing, presentano un contenuto limitato a sigle che identificano le componenti e le feature dei modelli, ma il loro contenuto informativo non è rilevante nella generazione delle etichette. La presenza di questi elementi all'interno delle etichette generate sia da TF-IDF che da LDA ha comportato il calo del valore della similarità coseno per entrambe le tecniche.

5.3 Precision, Recall e F1 score

L'utilizzo di diverse metriche nel contesto della validazione è fondamentale per valutare la qualità delle etichette in modo più dettagliato, considerando vari aspetti delle performance del

<i>Cluster</i>	<i>Etichette della Ground truth</i>	<i>Etichette generate da TF-IDF</i>	<i>Etichette generate da LDA</i>
8	aaspe security timing requirements	aaspe security incorrect r5 cwe f1 f2 pr d1 d2	r4 correct aaspe security ftatest, security aaspe incorrect r5 cwe pe pd nan d2 d1, pr f2 f1 d1 d2
23	structural deformation inspection using drones	issue qa errormodelv fgs app output port import fromapport subthread	issue qa errormodelv fgs app, drone inspect network structural deformation port process subthread application software, output import fromapport ap ac, mph sub refspd method throttlepos
31	iplprojects edge map	map iplprojects edge	mapa iplprojects map mapb, map iplprojects mapb mapa edge l4 l1 l2 l3, l8 l6 l5 l7 l3

Tabella 5.4: Confronto tra etichette della ground truth e etichette generate automaticamente.

modello. Le misure di **precision**, **recall** e **F1** sono particolarmente utili in questo contesto, poiché permettono di ottenere una panoramica più completa sulle prestazioni dei modelli di etichettatura automatica. Prima di calcolare queste misure, è necessario calcolare alcune misure preliminari:

- **True Positive (TP)**: rappresentano le parole comuni tra l’etichetta della ground truth e quella generata automaticamente, ovvero le parole correttamente predette dai modelli di etichettatura automatica.
- **False Positive (FP)**: sono le parole che appaiono nell’etichetta generata dal modello, ma che non si trovano nell’etichetta della ground truth, rappresentando le parole erroneamente predette.
- **False Negative (FN)**: sono le parole che appaiono nell’etichetta della ground truth, ma che non si trovano nell’etichetta generata dal modello. Rappresentano le parole che i modelli non sono riusciti a prevedere correttamente.

La prima misura calcolata è la **precision**, che determina la percentuale di parole corrette tra tutte quelle che sono state predette dal modello. In altre parole, essa indica quante delle parole assegnate automaticamente sono effettivamente corrette, rispetto a quelle erroneamente attribuite dal modello. Il calcolo della precision è dato dalla formula:

$$Precision = \frac{TP}{TP + FP}$$

Il **recall** misura la percentuale di parole corrette che sono state effettivamente rilevate dal modello. In pratica, il recall indica quante delle parole che avrebbero dovuto essere assegnate sono state effettivamente rilevate dal modello. La formula per il calcolo del recall è la seguente:

$$Recall = \frac{TP}{TP + FN}$$

Infine, il **F1 score** rappresenta una metrica di sintesi che bilancia precision e recall, restituendo un singolo valore che tiene conto sia della correttezza delle parole assegnate che della loro completezza. Il calcolo dell'F1 score si ottiene mediante la media armonica tra precision e recall:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Questa misura è particolarmente utile quando c'è un trade-off tra precisione e completezza, ovvero quando si cerca di ottenere un equilibrio tra l'affidabilità delle etichette assegnate e la capacità di recuperare tutte le etichette corrette.

Nel codice implementato, le etichette della ground truth e quelle generate automaticamente vengono confrontate direttamente, senza l'ausilio di tecniche di embedding e normalizzazione come WordNet e Word2Vec. Questo approccio è possibile perché le misure di precision, recall e F1 si basano su un confronto diretto tra le parole presenti nelle etichette della ground truth e quelle predette dal modello.

A differenza della cosine similarity, che richiede una rappresentazione semantica dei termini tramite tecniche di embedding come Word2Vec o WordNet, in precision e recall si considera una corrispondenza esatta tra le parole. Non è quindi necessario fare uso di tecniche di embedding semantico, poiché l'obiettivo di queste misure è semplicemente determinare quante parole corrette sono state assegnate dal modello e quanto il modello è riuscito a identificare tutte le etichette pertinenti.

I risultati ottenuti a seguito del calcolo delle misure di precision e recall per ogni cluster sono mostrati nelle Figure 5.3 e 5.4.

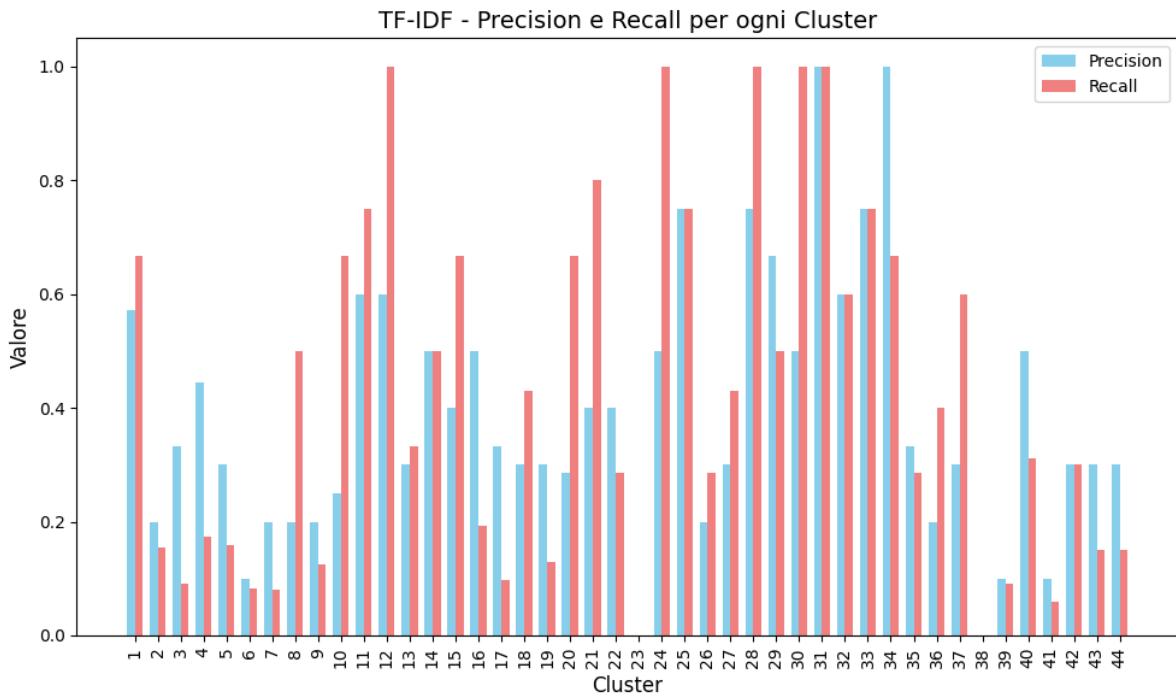


Figura 5.3: Valori di precisione e recall per cluster ottenuti confrontando ground truth e etichette di TF-IDF.

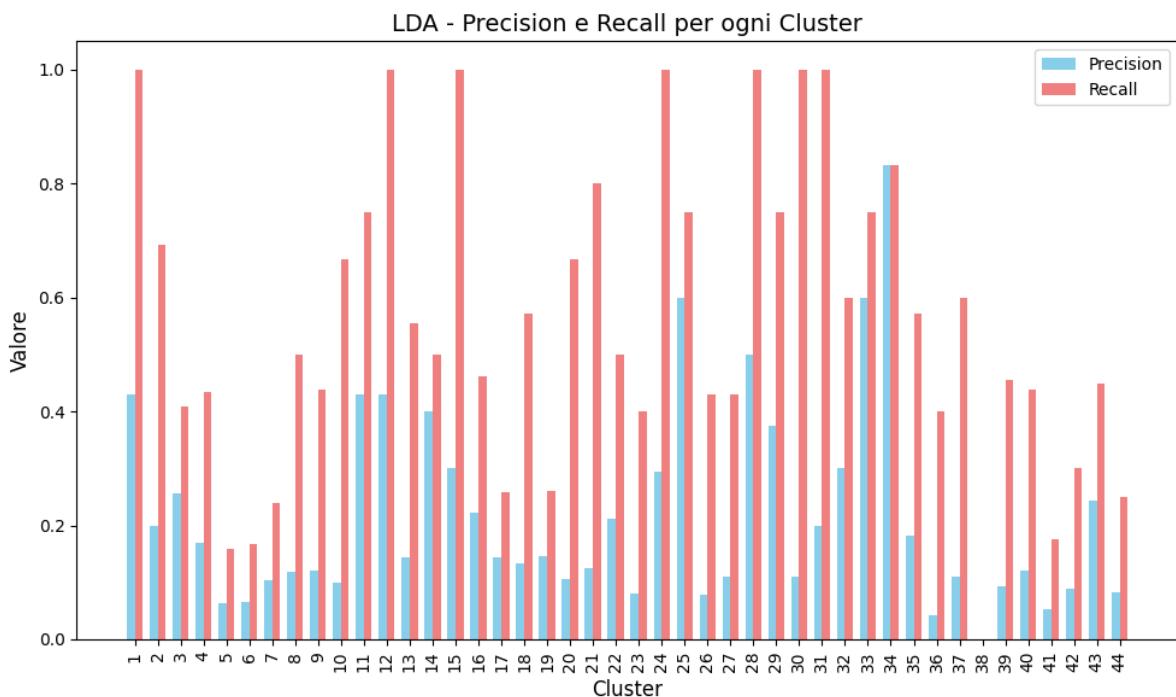


Figura 5.4: Valori di precisione e recall per cluster ottenuti confrontando ground truth e etichette di LDA.

Dai grafici emerge chiaramente un fenomeno ricorrente: la recall tende a essere più alta rispetto alla precision per entrambe le tecniche adottate. Questo comportamento può essere attribuito principalmente alla differenza tra il numero di etichette generate automaticamente e il numero di etichette nella ground truth.

Un valore di recall elevato per le etichette generate da LDA indica che la tecnica è riuscita a recuperare molte delle parole presenti nella ground truth, ma a scapito di una minore precision. Questo è dovuto al fatto che LDA tende ad individuare topic latenti che possono includere parole che non sono direttamente pertinenti alla descrizione dei modelli e la presenza di queste parole si riflette nel basso valore di precision ottenuto.

Anche nel caso di TF-IDF, la recall è generalmente più alta della precision, ma con una variabilità maggiore tra i cluster. TF-IDF è focalizzato sull'estrazione di parole chiave significative e, quindi, in alcuni cluster la precision può essere relativamente più alta rispetto a LDA, che all'interno dei suoi topic può inserire parole poco significative ma rilevanti in termini di co-occidenza con altre parole.

Probabilmente la causa di questo fenomeno per entrambe le tecniche è dovuta al contenuto della ground truth. Infatti, essa è stata costruita con un numero limitato di etichette, che coprono solo i concetti chiave associati ai modelli, mentre TF-IDF e, soprattutto, LDA tendono a generare un numero maggiore di etichette quando i cluster contengono una grande varietà di componenti e feature, anche se queste non sono significative dal punto di vista semantico. Questa differenza nel numero di etichette ha portato ad un maggiore recupero delle parole corrette, aumentando la recall. Tuttavia, la generazione di molte etichette può anche includere parole che non sono direttamente rilevanti per il significato del modello, il che penalizza la precision, in quanto il modello predice parole che non corrispondono alla ground truth.

Riprendendo le informazioni riguardanti le etichette di ground truth e quelle generate automaticamente mostrate in Tabella 5.4, è possibile spiegare questo fenomeno attraverso un esempio. Per il **cluster 31**, i valori di precision e recall per TF-IDF sono massimi, in quanto il modello è riuscito a prevedere esattamente le stesse parole presenti nella ground truth. Nel caso di LDA, invece, nonostante il modello complessivamente sia riuscito a prevedere tutte le parole presenti nella ground truth, la presenza di numerosi altri termini nei topic che non sono presenti nella ground truth ha penalizzato fortemente il valore della precision.

La misura F1 risulta particolarmente utile in questo scenario, poiché bilancia le due metriche, consentendo una valutazione complessiva delle performance del modello. Nel grafico

boxplot mostrato in Figura 5.5, si mostrano i valori di precision, recall e F1 score. Come indicato precedentemente, la recall tende ad essere più alta rispetto alla precision in entrambi i modelli, principalmente a causa della sovra-generazione di etichette, specialmente per LDA, che potrebbe introdurre molte parole generiche o poco pertinenti. In scenari come questo, dove la recall è più alta e la precision più bassa, l'F1 score fornisce una valutazione equilibrata delle performance del modello.

I valori di F1 mostrati nel grafico evidenziano come, nonostante la sensibile differenza nel punteggio di recall tra TF-IDF e LDA, la tecnica che ottiene complessivamente i risultati migliori sia TF-IDF.

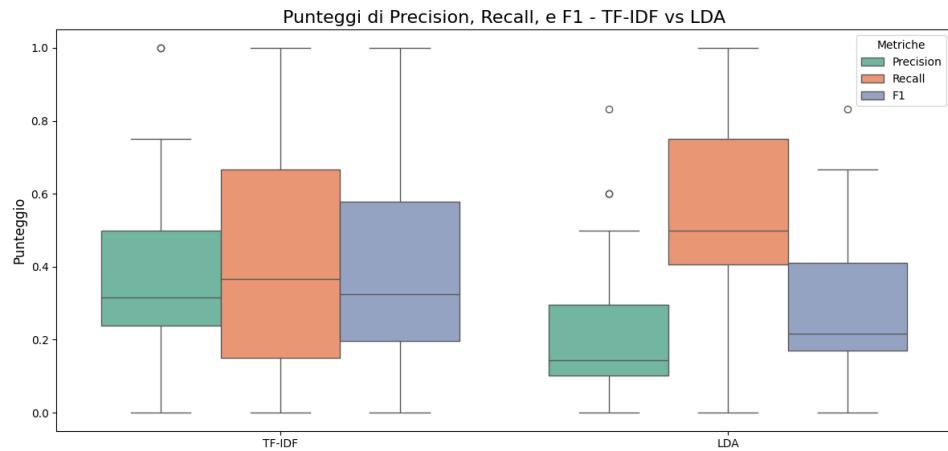


Figura 5.5: Boxplot dei valori di precision, recall e F1 per le etichette di TF-IDF e LDA.

Nel complesso, l'approccio di etichettatura automatica utilizzato ha mostrato buoni risultati, con un buon livello di recall per entrambe le tecniche LDA e TF-IDF. Questo indica che i modelli sono stati in grado di recuperare una parte significativa delle etichette corrette. Tuttavia, la precision ha mostrato margini di miglioramento, a causa della generazione di etichette non sempre perfettamente corrispondenti a quelle della ground truth. La recall elevata, infatti, suggerisce che i modelli sono riusciti a identificare molte parole corrette, ma la precision inferiore indica che sono state incluse anche etichette irrilevanti o generiche. L'F1 score si è rivelato particolarmente utile per bilanciare queste due misure, fornendo una visione equilibrata delle performance complessive dei modelli.

Conclusioni

In questo elaborato è stato proposto un approccio di etichettatura automatica per cluster di modelli architetturali AADL, utilizzando le tecniche TF-IDF e LDA per estrarre parole chiave da associare a ciascun cluster, con l'obiettivo di fornire una rappresentazione concisa ed efficace del contenuto dei modelli. Questo approccio facilita la consultazione e il riutilizzo dei modelli, contribuendo a migliorare l'efficienza nella gestione di grandi volumi di modelli architetturali.

Gli obiettivi raggiunti attraverso le metodologie adottate possono essere sintetizzati come segue:

- **Selezione delle informazioni:** le analisi manuali, supportate dall'uso di strumenti come OSATE, hanno permesso di identificare le caratteristiche principali dei modelli AADL da cui estrarre informazioni per garantire un corretto processo di etichettatura. L'esclusione di informazioni ridondanti o irrilevanti ha semplificato il processo di etichettatura, migliorando l'accuratezza e la rilevanza delle etichette generate.
- **Utilità del preprocessing:** l'utilizzo di tecniche di preprocessing per la normalizzazione del contenuto degli elementi estratti dai modelli si è dimostrato particolarmente utile ai fini dell'etichettatura automatica. Infatti, ricondurre parole dallo stesso significato ad una rappresentazione comune migliora notevolmente la coerenza delle etichette. Inoltre la rimozione di parole prive di contenuto informativo ha ridotto il rumore all'interno delle informazioni estratte migliorando la qualità delle etichette generate.
- **Efficacia delle tecniche di labelling:** le tecniche TF-IDF e LDA si sono rivelate efficaci nel generare etichette per i cluster, riuscendo a riflettere accuratamente il contenuto dei modelli presenti nei cluster, come confermato dalla validazione delle etichette rispetto a una ground truth manuale.

L'approccio adottato ha raggiunto quindi gli obiettivi prefissati ad inizio progetto. Tuttavia, le metodologie adottate hanno portato alla luce diverse criticità, la cui gestione potrebbe migliorare i risultati ottenuti. Le principali aree di miglioramento sono le seguenti:

- **Espansione della base dati:** La base dati utilizzata per il progetto presenta una raccolta di modelli ottenuti da repository pubblici. Espandere il dataset, individuando modelli pro-

venienti da diversi domini applicativi, consentirebbe di verificare se le tecniche proposte possano essere applicate con successo anche in contesti diversi.

- **Uniformità nel contenuto dei modelli:** un altro ostacolo emerso durante il processo di etichettatura riguarda la presenza di sigle e abbreviazioni, che hanno reso difficile l'estrazione di etichette significative. Modelli con una denominazione più chiara e coerente delle proprie componenti e feature, in cui le sigle siano esplicitate, migliorerebbero significativamente la qualità delle etichette generate.
- **Affinamento della ground truth:** la ground truth utilizzata in questo lavoro è risultata parziale e non sufficientemente dettagliata, limitando la precisione delle etichette generate. Una ground truth più completa e dettagliata, che includa etichette più varie e rappresentative, permetterebbe di ottenere risultati più precisi nella valutazione della qualità delle etichette generate automaticamente. Sebbene la creazione di una ground truth più ricca richieda un impegno maggiore, l'intervento di esperti del dominio per la sua creazione risulterebbe decisamente vantaggioso.
- **Adozione di tecniche di labelling complesse:** sebbene le tecniche LDA e TF-IDF abbiano fornito risultati soddisfacenti, l'adozione di metodi più avanzati potrebbe migliorare ulteriormente la qualità delle etichette generate. Tecniche che prevedono l'uso di modelli di linguaggio pre-addestrati, come ad esempio GPT-4, oppure tecniche che sfruttano l'apprendimento semi-supervisionato possono offrire una comprensione più profonda del contenuto dei modelli AADL, permettendo una generazione di etichette più accurata e contestualizzata.

In conclusione, questo progetto rappresenta un contributo innovativo per la catalogazione e il riutilizzo dei modelli architetturali AADL, offrendo un approccio automatico per l'assegnazione di etichette ai cluster di modelli. Sebbene si tratti di un primo approccio, i risultati ottenuti confermano l'efficacia delle tecniche applicate, mostrando come l'etichettatura automatica possa facilitare l'organizzazione e l'accesso ai modelli. Il lavoro svolto apre anche la strada a numerosi miglioramenti, come l'adozione di tecniche di etichettatura più avanzate e l'espansione del dataset, che potrebbero ulteriormente ottimizzare i risultati e ampliare le potenzialità applicative. In questo contesto, la tematica affrontata si è rivelata essere non solo utile, ma anche promettente per sviluppi futuri nel campo della gestione e riutilizzo dei modelli architetturali.

Bibliografia

- [1] Ahmed Abd-Allah et al. «On the Definition of Software System Architecture». In: (apr. 1997).
- [2] Len Bass, Paul Clements e Rick Kazman. «Software Architecture in Practice». In: 3rd. Addison-Wesley Professional, 2012. ISBN: 0321815734.
- [3] David M. Blei, Andrew Y. Ng e Michael I. Jordan. «Latent dirichlet allocation». In: *J. Mach. Learn. Res.* 3.null (mar. 2003), pp. 993–1022. ISSN: 1532-4435.
- [4] Paul Clements et al. «Documenting software architectures: views and beyond». In: *Proceedings of the 25th International Conference on Software Engineering*. ICSE '03. Portland, Oregon: IEEE Computer Society, 2003, pp. 740–741. ISBN: 076951877X.
- [5] Edward Colbert et al. «An Overview of the SAE Architecture Analysis and Design Language (AADL) Standard: A Basis for Model-Based Architecture-Driven Embedded Systems Engineering». In: *International Federation for Information Processing Digital Library; Architecture Description Languages*; 176 (gen. 2005). DOI: 10.1007/0-387-24590-1_1.
- [6] Jacob Devlin et al. «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding». In: Minneapolis, Minnesota: Association for Computational Linguistics, 2019. DOI: 10.18653/v1/N19-1423. URL: <https://aclanthology.org/N19-1423/>.
- [7] Peter H. Feiler, David P. Gluch e John J. Hudak. «The Architecture Analysis & Design Language (AADL): An Introduction». In: 2006. URL: <https://api.semanticscholar.org/CorpusID:235953221>.
- [8] Vishal Gupta e Gurpreet Lehal. «A Survey of Text Summarization Extractive Techniques». In: *Journal of Emerging Technologies in Web Intelligence* 2 (ago. 2010). DOI: 10.4304/jetwi.2.3.258-268.
- [9] Amir Jalilifard et al. «Semantic Sensitive TF-IDF to Determine Word Relevance in Documents». In: gen. 2021, pp. 327–337. ISBN: 978-981-33-6986-3. DOI: 10.1007/978-981-33-6987-0_27.

- [10] Rada Mihalcea e Paul Tarau. «TextRank: Bringing Order into Text». In: *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. A cura di Dekang Lin e Dekai Wu. Barcelona, Spain: Association for Computational Linguistics, lug. 2004.
- [11] Zarmeen Nasim e Sajjad Haider. «Automatic Labeling of Clusters for a Low-Resource Urdu Language». In: *ACM Transactions on Asian and Low-Resource Language Information Processing* 21 (set. 2022), pp. 1–22. DOI: 10.1145/3511097.
- [12] Ani Nenkova e Kathleen McKeown. «A Survey of Text Summarization Techniques». In: gen. 2012. ISBN: 978-1-4614-3222-7. DOI: 10.1007/978-1-4614-3223-4_3#citeas.
- [13] Dewayne E. Perry e Alexander L. Wolf. «Foundations for the study of software architecture». In: *SIGSOFT Softw. Eng. Notes* 17.4 (ott. 1992), pp. 40–52. ISSN: 0163-5948. DOI: 10.1145/141874.141884. URL: <https://doi.org/10.1145/141874.141884>.
- [14] Chau Pham et al. «TopicGPT: A Prompt-based Topic Modeling Framework». In: gen. 2024, pp. 2956–2984. DOI: 10.18653/v1/2024.nacl-long.164.
- [15] Douglas Schmidt. «Model-driven engineering». In: *IEEE Comput* 39 (gen. 2006).
- [16] Mary Shaw e David Garlan. *Software architecture: perspectives on an emerging discipline*. USA: Prentice-Hall, Inc., 1996. ISBN: 0131829572.
- [17] Thomas Stahl, Markus Voelter e Krzysztof Czarnecki. *Model-Driven Software Development: Technology, Engineering, Management*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2006. ISBN: 0470025700.
- [18] The OSATE Team. *OSATE: Open Source AADL Tool Environment*. Ott. 2024. URL: <https://osate.org/>.
- [19] Min-Hsien Weng, Shaoqun Wu e Mark Dyer. «Identification and Visualization of Key Topics in Scientific Publications with Transformer-Based Language Models and Document Clustering Methods». In: *Applied Sciences* 12.21 (2022). ISSN: 2076-3417. DOI: 10.3390/app122111220. URL: <https://www.mdpi.com/2076-3417/12/21/11220>.

- [20] Hong Zhu. «4 - Software Architecture». In: *Software Design Methodology*. A cura di Hong Zhu. Oxford: Butterworth-Heinemann, 2005, pp. 73–110. ISBN: 978-0-7506-6075-4. DOI: <https://doi.org/10.1016/B978-075066075-4/50007-8>. URL: <https://www.sciencedirect.com/science/article/pii/B9780750660754500078>.