

Corso di Laurea in Ingegneria e Scienze Informatiche

Estrazione di Informazioni Chiave da Documenti Fiscali Italiani: Studio Comparativo tra Approcci Discriminativi e Generativi

Relatrice

Antonella Carbonaro

Candidato

Riccardo Albertini

Sommario

La digitalizzazione dei documenti fiscali è un'attività onerosa per gli studi commercialisti, che trascrivono manualmente i dati di scontrini, ricevute e fatture ai fini contabili e tributari. Questa tesi nasce all'interno di TaxWiz, piattaforma sviluppata in collaborazione con uno studio commercialista per automatizzare tale processo, e affronta la Key Information Extraction da documenti fiscali italiani semi-strutturati acquisiti tramite fotografia, con l'obiettivo di estrarne automaticamente campi rilevanti per la gestione contabile. Dalla casistica reale della piattaforma è stato costruito un dataset proprietario di 1.919 pagine annotate. Sullo stesso dataset e sulla stessa suddivisione vengono confrontati due paradigmi opposti: LayoutLMv3, modello discriminativo OCR-based che formula il task come classificazione token-level in schema BIO, e Qwen2.5-VL-7B-Instruct, Vision-Language Model generativo addestrato tramite QLoRA a produrre direttamente un oggetto JSON dall'immagine. Gli output eterogenei dei due modelli sono ricondotti a una rappresentazione comune campo-valore, che abilita una valutazione field-level uniforme. I due modelli raggiungono prestazioni aggregate comparabili, con micro-F1 pari a 0,5816 contro 0,5761, ma con profili di errore opposti: LayoutLMv3 privilegia la copertura, con recall più alta e più falsi positivi, mentre Qwen2.5-VL è più conservativo, con precision superiore e più omissioni. Il contributo principale è di natura metodologica e consiste nella definizione di un protocollo riproducibile per confrontare in modo equo un approccio discriminativo e uno generativo sullo stesso compito di estrazione, a parità di dati, suddivisione e criteri di misura.

A mia madre, che teneva la luce accesa senza chiedere mai fino a quando.

Indice

Sommario	iii
1 Introduzione	1
1.1 Contesto e motivazione	1
1.2 Obiettivi del progetto	2
1.3 Contributi principali	3
1.3.1 Costruzione di un dataset proprietario annotato	3
1.3.2 Preprocessing differenziato per i due modelli	3
1.3.3 Fine-tuning controllato e comparabile	4
1.3.4 Valutazione comparativa su una rappresentazione comune	4
1.3.5 Analisi sperimentale della robustezza e delle componenti del modello	5
1.3.6 Sintesi del flusso complessivo della pipeline	5
1.4 Struttura della tesi	6
2 Stato dell'arte	7
2.1 Document AI, Document Understanding e KIE	7
2.2 Documenti semi-strutturati e sfide del KIE	9
2.3 Approcci tradizionali OCR-based e pipeline rule-based	11
2.4 Transformer e passaggio ai modelli multimodali	12
2.5 Modelli multimodali: LayoutLM	13
2.6 Vision-Language Models generativi e Qwen2.5-VL	15
2.7 Confronto tra approcci OCR-based e OCR-free	16
3 Dataset	19
3.1 Raccolta e origine dei dati	19
3.2 Acquisizione OCR e annotazione manuale	20
3.2.1 Esportazione e struttura del dataset	21
3.3 Analisi esplorativa	22
3.3.1 Distribuzione delle etichette a livello di token	23
3.3.2 Distribuzione a livello di entità e copertura documentale	24
3.3.3 Analisi spaziale delle entità	26
3.3.4 Osservazioni e implicazioni per il training	27
3.4 Classi sbilanciate BIO e strategie di mitigazione	28
3.4.1 Classi critiche: anno, importo e titolo edilizio	28

3.4.2	Scelta di non adottare una data augmentation estesa	29
3.4.3	Implicazioni sulle metriche di valutazione	29
3.5	Split del dataset	30
4	Architettura dei modelli e implementazione	33
4.1	LayoutLMv3: architettura nel dettaglio	34
4.1.1	Panoramica e posizionamento	34
4.1.2	Le tre modalità di input	35
4.1.3	Fusione multimodale e ruolo del Transformer encoder	37
4.1.4	Pipeline concreta utilizzata nella tesi	38
4.1.5	Obiettivi di pre-training: MLM, MIM e WPA	39
4.1.6	Testa di classificazione per il task BIO	39
4.1.7	Ruolo di LayoutLMv3 nella tesi	41
4.2	Qwen2.5-VL: architettura e KIE generativo	41
4.2.1	Panoramica e posizionamento	41
4.2.2	Architettura generale	42
4.2.3	Pipeline multimodale per documenti	44
4.2.4	Ruolo di Qwen2.5-VL nella tesi	44
4.3	Fine-tuning di LayoutLMv3	44
4.3.1	Obiettivo, modello e dati	44
4.3.2	Validazione e gestione dello sbilanciamento	46
4.3.3	Ottimizzazione e trainer	48
4.3.4	Ricerca iperparametrica e modello finale	50
4.3.5	Considerazioni finali	51
4.4	Fine-tuning di Qwen2.5-VL	51
4.4.1	Obiettivo e modello	51
4.4.2	Preparazione dei dati per la generazione	52
4.4.3	Adattamento parameter-efficient	53
4.4.4	Procedura di addestramento	55
4.4.5	Considerazioni finali	56
4.5	Differenze operative tra LayoutLMv3 e Qwen2.5-VL	57
4.5.1	Formulazione del task e supervisione	57
4.5.2	Input, OCR e rappresentazione del documento	57
4.5.3	Output, verificabilità ed errori	58
4.5.4	Costo computazionale	58
4.5.5	Quadro comparativo	58
5	Setup sperimentale e addestramento dei modelli	61
5.1	Ambiente software e hardware	61
5.2	Preparazione sperimentale dei dati	63
5.2.1	Dataset e split sperimentale	63
5.2.2	Preprocessing per LayoutLMv3	64
5.2.3	Preprocessing per Qwen2.5-VL	66
5.3	Configurazione del training di LayoutLMv3	68
5.3.1	Script di training e dataset tokenizzato	68

5.3.2	Modello e spazio delle etichette	68
5.3.3	Metriche di validazione	69
5.3.4	Focal Loss pesata e pesi di classe	70
5.3.5	Ottimizzatore e Layer-wise Learning Rate Decay	71
5.3.6	Estensione di TrainingArguments e configurazione di base	72
5.3.7	Costruzione del trainer e collegamento con la ricerca iperparametrica	73
5.4	Configurazione del training di Qwen2.5-VL	74
5.4.1	Dataset, campi target e conversione BIO-to-JSON	74
5.4.2	Costruzione del prompt e formato conversazionale	74
5.4.3	Modello di base, processor e quantizzazione QLoRA	75
5.4.4	Configurazione LoRA	76
5.4.5	Argomenti di training con SFTConfig	76
5.4.6	Costruzione del trainer, grid search e salvataggio	78
5.5	Iperparametri e modelli finali	78
5.5.1	Ricerca iperparametrica di LayoutLMv3	79
5.5.2	Ricerca iperparametrica di Qwen2.5-VL	82
5.5.3	Andamento del training dei modelli selezionati	84
6	Valutazione e analisi	89
6.1	Predizioni finali e dataset di valutazione	89
6.1.1	Predizione con LayoutLMv3	89
6.1.2	Predizione con Qwen2.5-VL	91
6.1.3	Output eterogenei e necessità di una rappresentazione comune	92
6.1.4	Costruzione del dataset di valutazione field-level	93
6.2	Metriche e protocollo di valutazione	94
6.2.1	Esiti field-level e metriche di base	94
6.2.2	Metriche aggregate	96
6.2.3	Metriche a livello di documento e sintesi	98
6.3	Analisi di ablation e robustezza su LayoutLMv3	99
6.3.1	Impostazione metodologica: ablation in fase di inferenza	100
6.3.2	Risultati dello studio di ablation	100
6.3.3	Studio di robustezza: degradazioni controllate	101
6.3.4	Sintesi e limiti dell'analisi	102
7	Esperimenti e risultati	105
7.1	Risultati complessivi field-level	105
7.2	Confronto field-level per campo	107
7.3	Exact match a livello di campione	109
7.4	Errori: missing, hallucination, wrong value	110
7.5	Analisi qualitativa	113

INDICE

8 Conclusioni	117
8.1 Riepilogo del progetto	117
8.2 Discussione LayoutLMv3 e Qwen2.5-VL	118
8.3 Limiti sperimentali	121
8.4 Sviluppi futuri	123
Bibliografia	127

Capitolo 1

Introduzione

1.1 Contesto e motivazione

La gestione dei documenti fiscali comporta ancora oggi un considerevole lavoro manuale per contribuenti, professionisti e studi commercialisti. Ricevute, fatture e scontrini vengono raccolti, verificati e categorizzati a fini dichiarativi; le informazioni rilevanti richiedono poi di essere inserite in gestionali o piattaforme dedicate.

Si tratta di un'attività ripetitiva e dispendiosa in termini di tempo, esposta a errori, soprattutto quando i documenti provengono da fotografie acquisite in ambienti non controllati. Gli scontrini presentano spesso una qualità dell'immagine variabile: rotazioni, sfocature, pieghe, inclinazioni irregolari e lo sbiadimento tipico della carta termica.

Il lavoro nasce da un'esigenza concreta emersa durante lo sviluppo di TaxWiz, una piattaforma realizzata in collaborazione con uno studio commercialista per semplificare la gestione documentale legata alla dichiarazione dei redditi. Nel flusso applicativo considerato, il sistema deve supportare l'acquisizione dei documenti, l'individuazione dei campi utili e la loro organizzazione in forma strutturata. L'obiettivo non è eliminare il controllo umano, ma alleggerire le operazioni meccaniche e indirizzare la revisione solamente verso i casi incerti.

Nel dominio fiscale questa distinzione è particolarmente rilevante. Un valore estratto in modo errato, mancante o associato al campo sbagliato compromette la qualità del dato impiegato nelle fasi successive del processo. Per questo motivo, un sistema automatico non deve limitarsi a produrre una predizione, ma deve essere valutato anche rispetto alla sua affidabilità, alla verificabilità dell'output e alla possibilità di integrarsi in un flusso con supervisione umana.

La tesi si inquadra nella *Document AI*, che studia metodi per l'analisi e la comprensione automatica di documenti digitali o digitalizzati, e affronta in particolare il task di *Key Information Extraction (KIE)*: l'estrazione di un insieme predefinito di campi informativi da un documento.

Per affrontare questo problema, la tesi analizza e confronta due modelli che incarnano paradigmi distinti della comprensione automatica dei documenti: **LayoutLMv3-base** e **Qwen2.5-VL-7B-Instruct**. Entrambi sono multimodali, poiché utilizzano

informazioni testuali, visive e spaziali. Il confronto tra questi approcci costituisce il nucleo del lavoro. Da una parte un modello specializzato per la *Document AI*, strettamente legato a OCR, bounding box e annotazioni token-level; dall'altra un *Vision-Language Model* generativo, più flessibile nella formulazione del task ma più oneroso sul piano computazionale e più difficile da controllare nell'output. La valutazione non si esaurisce nelle sole prestazioni di estrazione, ma considera anche la robustezza della pipeline, la complessità del preprocessing, la verificabilità delle predizioni e l'idoneità dei due paradigmi in un reale contesto fiscale.

1.2 Obiettivi del progetto

L'obiettivo principale della tesi è sviluppare e valutare una pipeline di *Key Information Extraction* per documenti fiscali italiani, con particolare attenzione a scontrini, fatture e ricevute acquisiti tramite fotografia. Partendo dall'immagine del documento, il sistema deve individuare un insieme predefinito di campi informativi, restituendoli in una rappresentazione strutturata e idonea alle successive fasi di classificazione, verifica e gestione fiscale.

I campi considerati sono nove: anno, importo, totale del documento, pagamento tracciato, data del documento, codice fiscale, nome e cognome, totale detraibile e titolo edilizio. La loro presenza e la loro forma variano in base alla tipologia del documento e alla qualità dell'acquisizione. Il compito non viene quindi trattato come semplice trascrizione di testo, bensì come estrazione selettiva di informazioni rilevanti all'interno di documenti semi-strutturati.

Il progetto mette a confronto i due approcci già introdotti nella sezione precedente. L'analisi architetturale delle due soluzioni è rimandata al Capitolo 4, mentre in questa sezione l'attenzione si concentra sui criteri fondamentali del confronto. Le due impostazioni richiedono pipeline diverse. LayoutLMv3 dipende da una fase preliminare di OCR, dalla disponibilità delle coordinate spaziali delle parole e dalla conversione delle annotazioni in etichette BIO. Qwen2.5-VL permette invece di descrivere il task tramite prompt e di addestrare il modello a generare direttamente una struttura *field-level*.

Per rendere il confronto il più possibile equo, entrambi i modelli vengono addestrati e valutati sullo stesso dataset, con identica suddivisione in train, validation e test. Iperparametri e strategie di fine-tuning sono invece adattati alle caratteristiche di ciascun modello e ai vincoli hardware disponibili.

Per evitare che il confronto dipenda dalle metriche interne dei singoli modelli, le predizioni vengono ricondotte a una rappresentazione comune campo-valore. La scelta è necessaria perché LayoutLMv3 restituisce sequenze di etichette BIO mentre Qwen2.5-VL genera testo strutturato in JSON; la rappresentazione campo-valore consente quindi di applicare la stessa valutazione a entrambi. Su questa struttura vengono applicate procedure di normalizzazione dedicate alla gestione degli spazi, del *case* e alla standardizzazione degli importi e delle date. Il confronto quantitativo si basa quindi su metriche field-level e su un'analisi qualitativa delle principali tipologie di errore.

Un secondo obiettivo del presente lavoro riguarda la robustezza in condizioni operative reali. I documenti fiscali acquisiti tramite fotografia possono presentare artefatti visivi come rotazioni, sfocature, rumore, bassa risoluzione, testi degradati o layout deformati. Gli esperimenti di ablation e robustezza sono condotti su LayoutLMv3 per misurare quanto le prestazioni dipendano dalle diverse modalità di input e quanto il modello sia sensibile a perturbazioni dell'immagine o del layout.

Il confronto tra i due paradigmi si articola su qualità delle predizioni, complessità del preprocessing, costo computazionale, latenza potenziale, facilità di manutenzione della pipeline, controllabilità dell'output e integrazione in un flusso con supervisione umana. In uno scenario applicativo reale, infatti, il modello più adatto non è necessariamente quello con il punteggio migliore su una singola metrica, ma quello che offre il compromesso più sostenibile tra accuratezza, affidabilità e costo operativo.

È proprio questa differenza a motivare il confronto, poiché i due modelli non sono solo alternative tecniche, ma due modalità distinte di formulare il problema della *Key Information Extraction*.

1.3 Contributi principali

Il lavoro copre l'intero ciclo sperimentale di un sistema di *Key Information Extraction* per documenti fiscali attraverso l'acquisizione dei dati, la costruzione del dataset, il preprocessing, il fine-tuning dei modelli, la generazione delle predizioni e la valutazione comparativa.¹

I contributi principali si possono riassumere nelle sottosezioni seguenti.

1.3.1 Costruzione di un dataset proprietario annotato

Il primo contributo è la costruzione di un dataset originale composto da circa 2.000 record fiscali provenienti dal contesto applicativo di TaxWiz. I dati sono stati estratti dal database della piattaforma tramite uno script in TypeScript, basato su query PostgreSQL su Supabase, e organizzati in una struttura locale contenente le immagini dei documenti, le annotazioni dei campi rilevanti e gli output OCR.

I dati esportati sono stati poi caricati, puliti e trasformati usando script Python, per ottenere una rappresentazione utilizzabile nelle fasi successive. A partire dagli stessi dati grezzi sono state derivate due versioni del dataset: una per LayoutLMv3, basata su annotazioni *token-level* in formato BIO, e una per Qwen2.5-VL, basata su coppie di prompt e target JSON.

1.3.2 Preprocessing differenziato per i due modelli

Il secondo contributo riguarda la definizione di due pipeline di preprocessing distinte, costruite a partire dallo stesso dataset. Per LayoutLMv3, le informazioni testuali

¹codice sorgente e notebook sono disponibili pubblicamente al repository GitHub: <https://github.com/RichiAlbertini/italian-fiscal-kie>.

e spaziali prodotte dall'OCR Amazon Textract vengono convertite nel formato nativo richiesto da `LayoutLMv3Processor`: parole OCR, bounding box, immagine del documento e `ner_tags` (etichette BIO). Per Qwen2.5-VL il documento è rappresentato come input multimodale, composto da immagine e prompt, con il target strutturato come oggetto JSON contenente i campi da estrarre.

Tale approccio evidenzia il contrasto tra i due paradigmi: una formulazione discriminativa del task, centrata sull'etichettatura dei token, e una formulazione generativa, orientata alla produzione diretta di una struttura field-level.

1.3.3 Fine-tuning controllato e comparabile

Il terzo contributo è il fine-tuning dei due modelli sullo stesso compito di estrazione, mantenendo uniforme la suddivisione del dataset in train, validation e test. Tale scelta garantisce la comparabilità dei risultati sul test set, riducendo il rischio che le differenze osservate dipendano dall'utilizzo di insiemi di valutazione differenti. Le strategie di addestramento sono state adattate alle caratteristiche dei modelli:

- **LayoutLMv3** è stato addestrato per un compito di token classification, con particolare attenzione allo sbilanciamento delle classi dovuto alla forte prevalenza della classe *O*.
- **Qwen2.5-VL** è stato addestrato tramite supervised fine-tuning in modalità parameter-efficient, ricorrendo a QLoRA per contenere l'impatto computazionale dell'addestramento.

1.3.4 Valutazione comparativa su una rappresentazione comune

Al fine di gestire l'eterogeneità degli output prodotti dai due modelli, è stata progettata una pipeline di conversione e normalizzazione comune. Le predizioni BIO di LayoutLMv3 vengono ricondotte a campi testuali, mentre le risposte generate da Qwen2.5-VL vengono pulite, parsate e convertite in struttura JSON; entrambi gli output confluiscono così nella stessa rappresentazione campo-valore.

Su questa base condivisa si calcolano metriche field-level quali precision, recall e F_1 -score, affiancate da document exact match e da un'analisi degli errori articolata nelle tre categorie: missing, wrong value e hallucination. Il confronto non si riduce quindi a un punteggio aggregato, ma mette in luce differenze qualitative nel comportamento dei due modelli.

La valutazione complessiva permette quindi non solo di determinare quale modello ottenga risultati migliori sul test set, ma anche di individuare le principali criticità operative.

1.3.5 Analisi sperimentale della robustezza e delle componenti del modello

La tesi include esperimenti di ablation e robustezza su LayoutLMv3, mirati a valutare l'impatto delle diverse modalità di input e la sensibilità del modello rispetto a condizioni realistiche di acquisizione, quali degradazioni dell'immagine, rumore, sfocatura o variazioni del layout. L'analisi risultante consente di esaminare non solo le prestazioni nelle condizioni standard, ma anche la resilienza della pipeline quando i documenti si discostano dalla qualità ideale.

1.3.6 Sintesi del flusso complessivo della pipeline

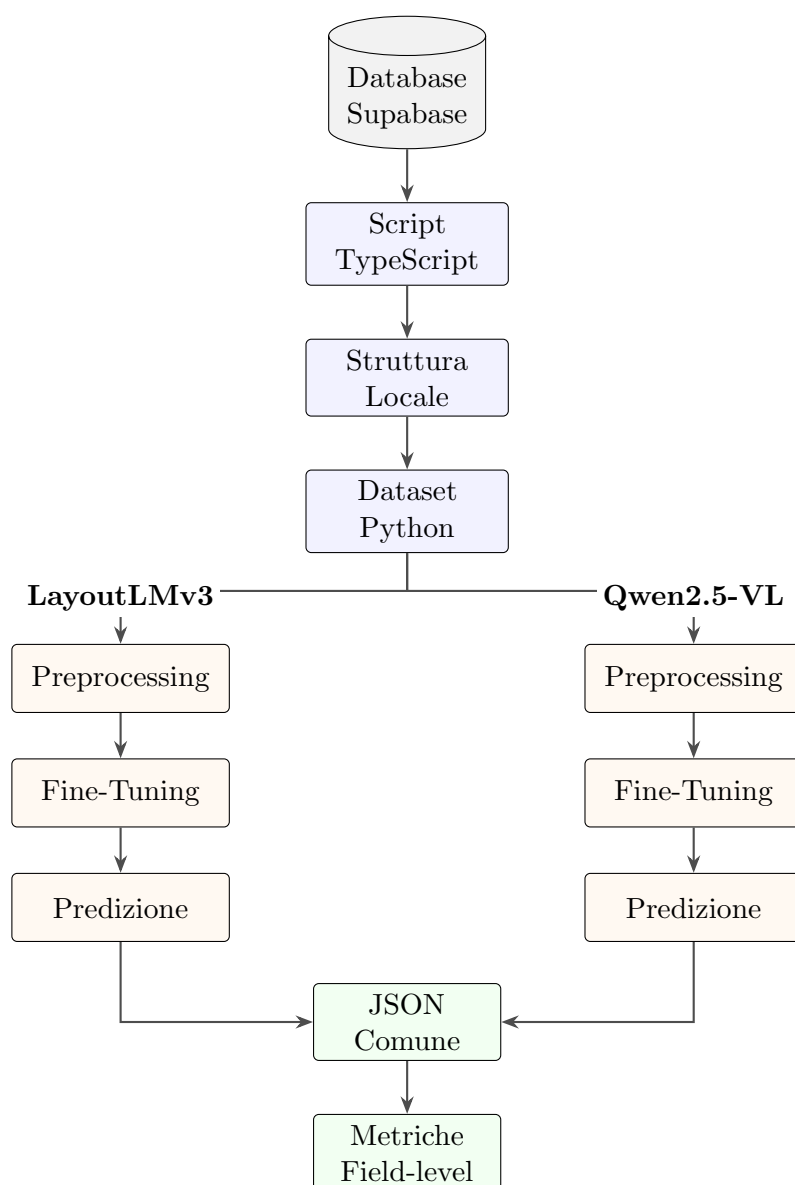


Figura 1.1: Pipeline complessiva del progetto.

La parte iniziale riguarda la costruzione del dataset a partire dai dati disponibili in TaxWiz; la parte centrale mostra la biforcazione tra la pipeline OCR-based di LayoutLMv3 e quella generativa image-to-JSON di Qwen2.5-VL; la parte finale riconduce le predizioni dei due modelli a una rappresentazione comune, necessaria per il confronto sperimentale.

1.4 Struttura della tesi

La tesi si articola in otto capitoli. Dopo questa introduzione, il Capitolo 2 inquadra il contesto teorico e lo stato dell'arte su Document AI, Key Information Extraction e principali approcci alla comprensione automatica di documenti semi-strutturati, introducendo i modelli multimodali per il document understanding e il confronto concettuale tra approcci OCR-based e generativi basati su Vision-Language Models.

Il dataset del progetto è descritto nel Capitolo 3, che ne illustra l'origine, la struttura dei documenti fiscali considerati, le annotazioni disponibili, la trasformazione dei campi target e lo schema BIO, insieme allo sbilanciamento delle classi e alla suddivisione in train, validation e test.

Il Capitolo 4 analizza l'architettura dei due modelli e le soluzioni implementative del fine-tuning, chiarendo le differenze operative tra le due pipeline, dalla rappresentazione dei dati alla preparazione degli input.

Il Capitolo 5 descrive il setup sperimentale, l'ambiente software e hardware, le configurazioni di training, la ricerca degli iperparametri e la selezione dei modelli finali.

Il Capitolo 6 definisce la generazione delle predizioni sul test set, la loro riconduzione a una rappresentazione comune campo-valore e il protocollo di valutazione field-level, e presenta gli esperimenti di ablation e robustezza condotti su LayoutLMv3.

I risultati sperimentali sono raccolti e discussi nel Capitolo 7, dove le prestazioni di LayoutLMv3 e Qwen2.5-VL vengono confrontate tramite metriche field-level, document exact match e un'analisi degli errori per missing, wrong value e hallucination, con riscontri qualitativi ed esempi.

Il Capitolo 8 conclude il lavoro riassumendo risultati, limiti sperimentali e direzioni future, con attenzione ai compromessi tra accuratezza, complessità della pipeline, costo computazionale e integrazione in un sistema di produzione reale.

Capitolo 2

Stato dell'arte

Questo capitolo inquadra i concetti teorici e tecnologici alla base del problema affrontato nella tesi. L'obiettivo è ricostruire l'evoluzione dei sistemi di Document AI e Key Information Extraction, mostrando come il trattamento automatico dei documenti sia passato da pipeline fondate su OCR, regole manuali e modelli puramente testuali a soluzioni multimodali e generative.

L'esposizione procede per gradi. Vengono prima definiti il Document Understanding e il task di Key Information Extraction, con attenzione ai documenti fiscalmente rilevanti e semi-strutturati. Si passa poi alle criticità proprie del dominio, quali la variabilità del layout, il rumore visivo, l'ambiguità contestuale e lo sbilanciamento delle classi. Tale analisi introduce l'evoluzione degli approcci nell'ambito della Document AI, dai sistemi rule-based ai Transformer multimodali fino ai Vision-Language Models generativi.

Il quadro teorico così delineato costituisce il fondamento concettuale necessario per leggere il confronto sperimentale tra LayoutLMv3 e Qwen2.5-VL-7B-Instruct sviluppato nei capitoli successivi.

2.1 Document AI, Document Understanding e Key Information Extraction

Document AI indica in questo contesto l'insieme di metodi impiegati per trasformare documenti digitali o digitalizzati in informazioni strutturate. L'obiettivo di tale disciplina non si limita al riconoscimento testuale di una pagina, ma mira a interpretare il documento come oggetto informativo complesso, in cui contenuto linguistico, disposizione spaziale ed elementi visivi concorrono alla definizione del significato. In uno scontrino, ad esempio, un numero collocato nella parte inferiore della pagina, vicino a parole come “totale” o “importo”, va interpretato diversamente da un numero analogo posto nell'intestazione, in una tabella di dettaglio o all'interno di un codice identificativo. Il valore testuale, da solo, non basta.

Tale transizione concettuale delinea il confine tra il semplice riconoscimento ottico dei caratteri e la Document Understanding: un sistema OCR tradizionale

trascrive le parole presenti nell'immagine, mentre un sistema di Document Understanding attribuisce un ruolo semantico agli elementi riconosciuti.

Questa prospettiva è alla base del Visually Rich Document Understanding, che studia documenti la cui interpretazione richiede di considerare congiuntamente il contenuto testuale e l'organizzazione della pagina. In questo contesto, i Visually Rich Documents sono documenti in cui l'informazione non è veicolata solo dalla sequenza delle parole, ma anche dalla struttura grafica e geometrica della pagina; fatture, ricevute, scontrini, moduli amministrativi, report finanziari e dichiarazioni fiscali rientrano in questa categoria. Elementi come tabelle, sezioni, loghi, separatori, font, allineamenti e posizione relativa degli elementi non sono dettagli puramente estetici, bensì elementi strutturali che determinano il ruolo informativo e il significato semantico dei dati estratti.

Rispetto a un testo lineare, come un articolo o un libro, i documenti di natura fiscale hanno natura bidimensionale. La relazione tra una chiave e il suo valore può dipendere dalla vicinanza spaziale, dall'allineamento geometrico, dall'appartenenza alla stessa sezione o dalla collocazione in una specifica area della pagina. Inoltre, valore e chiave testuale non sono sempre adiacenti, in quanto possono essere separati da colonne, linee, sezioni o altri elementi grafici strutturali; l'associazione corretta richiede pertanto la modellazione di relazioni spaziali anche non locali.

Due elementi semanticamente collegati possono risultare distanti nella sequenza testuale prodotta dall'OCR ma vicini nello spazio visivo del documento; al contrario, token contigui nella trascrizione linearizzata possono appartenere a colonne o sezioni diverse.

Per questo le moderne architetture di Document AI tendono a integrare tre livelli informativi complementari: la componente testuale, costituita dai termini riconosciuti o generati dal sistema; il layout, che descrive la posizione geometrica degli elementi tramite coordinate spaziali; e l'informazione visiva, costituita dall'immagine del documento e dalle caratteristiche grafiche della pagina. La combinazione di questi livelli è il fondamento dei modelli multimodali per la comprensione documentale.

Il *task* di *Key Information Extraction* si inserisce in questo scenario e costituisce l'oggetto del presente lavoro di tesi, configurandosi come un sottoinsieme specialistico rispetto al più generale ambito del *Document Understanding*. Nel caso della KIE, l'attenzione si concentra sull'estrazione di campi informativi predefiniti e sulla loro conversione in una rappresentazione strutturata. Di conseguenza, tale processo richiede la capacità di riconoscimento testuale, interpretazione del layout, robustezza rispetto al rumore visivo e comprensione del ruolo semantico dei valori rilevati. Proprio per questa natura multimodale, questo task si presta al confronto tra approcci specializzati basati su OCR e modelli generativi Vision-Language.

Nel dominio dei documenti fiscali, questi valori possono comprendere campi come data del documento, importo, totale documento, codice fiscale, totale detraibile o tracciabilità del pagamento, oltre a ulteriori attributi richiesti dalla gestione documentale successiva. Il sistema non deve leggere tutto ciò che compare nella pagina, ma localizzare e discriminare i campi corretti, distinguerli da valori simili e

restituirli in una forma coerente e verificabile.

2.2 Documenti semi-strutturati e sfide del KIE

Nella gestione documentale automatica i documenti si distinguono in tre categorie: strutturati, semi-strutturati e a testo libero. Tale distinzione dipende dal grado di regolarità dello schema informativo e dall'importanza che il layout assume nell'interpretazione del contenuto.

I documenti strutturati seguono uno schema rigido e predefinito. I campi occupano posizioni note e ricorrenti, e la loro interpretazione dipende in larga parte dalla collocazione all'interno del modello. È il caso del modello 730, in cui le informazioni risiedono in sezioni definite da una struttura stabile. In questo scenario, l'estrazione può spesso appoggiarsi a template, coordinate fisse o zone di interesse, perché la posizione dei campi è nota a priori e rimane invariata.

All'estremo opposto si trovano i documenti non strutturati o liberi, quali testi narrativi, articoli, relazioni o documenti discorsivi in cui il significato si sviluppa lungo la sequenza linguistica. Il layout può favorire la leggibilità, ma non costituisce la principale fonte di informazione, e l'estrazione viene pertanto affrontata come problema di elaborazione del linguaggio naturale.

I documenti semi-strutturati occupano una posizione intermedia. Presentano una struttura logica riconoscibile e un insieme di informazioni attese, ma non una disposizione spaziale fissa. Fatture, ricevute, scontrini fiscali e ordini d'acquisto rientrano in questa categoria: contengono campi ricorrenti (data, totale, importo, codice fiscale, dati dell'emittente) la cui posizione, formato grafico e formulazione testuale variano tra i diversi documenti. Questa classe costituisce la categoria di riferimento di questa tesi, e da essa derivano le problematiche discusse di seguito.

La prima criticità riguarda la variabilità del layout. Due documenti possono contenere le stesse informazioni fiscali ma organizzarle secondo griglie, colonne, sezioni o allineamenti differenti. Una persona individua il totale di uno scontrino anche quando ne cambia la posizione, perché legge simultaneamente testo, dimensione del carattere, allineamento, vicinanza tra elementi e struttura complessiva della pagina. Per un sistema automatico l'assenza di una topologia stabile rende insufficienti sia un approccio basato solo su coordinate fisse, sia uno fondato unicamente sul testo linearizzato prodotto dall'OCR.

Una seconda criticità deriva dalla modalità di acquisizione. Si distinguono i documenti nativamente digitali, come i PDF generati da software gestionali, da quelli cartacei digitalizzati tramite scanner o fotocamera. I primi tendono ad avere testo più pulito e layout più regolare; i secondi introducono artefatti fisici e visivi che complicano l'estrazione. Nel contesto applicativo di TaxWiz la quasi totalità dei documenti proviene da fotografie acquisite in condizioni non controllate, con qualità variabile. Gli scontrini cartacei sono un caso particolarmente problematico. La carta termica sbiadisce rapidamente, riducendo il contrasto tra testo e sfondo. Inoltre, pieghe, strappi, abrasioni, inclinazioni, distorsioni prospettiche, ombre, riflessi, sfocatura e sfondi rumorosi possono comprometterne la leggibilità.

Non si tratta di un effetto puramente estetico perché questi difetti modificano la qualità dell'informazione disponibile per il sistema di estrazione.

Negli approcci OCR-based tali difetti generano errori di segmentazione, riconoscimento o ordinamento del testo. Caratteri simili, come “0” e “O” oppure “1” e “l”, vengono confusi: una cifra di un importo può essere alterata, una parola può non essere rilevata e una bounding box può risultare imprecisa. In un documento fiscale, un errore minimo può compromettere la validità del valore estratto. Se l'estrazione produce una rappresentazione incompleta o corrotta, il modello a valle lavora su un input già degradato e l'errore si propaga lungo la pipeline.

A questo si aggiunge il problema della generalizzazione. Un modello addestrato su documenti puliti, ben allineati o riconducibili a pochi formati ricorrenti può ottenere buone prestazioni in addestramento, per poi subire un peggioramento quando incontra documenti mai visti, rumorosi o caratterizzati da layout differenti. Per questo motivo, la valutazione di un sistema di Key Information Extraction deve includere variazioni di qualità visiva, rotazioni, rumore, layout eterogenei e formati non osservati durante il training.

Oltre alle difficoltà visive e geometriche, la KIE pone problemi statistici. Nei task formulati come token classification, la maggior parte dei token non appartiene ad alcun campo target: descrizioni di articoli, intestazioni, indirizzi, testi legali, codici interni e informazioni accessorie riempiono gran parte della pagina, mentre i valori rilevanti ne costituiscono una frazione ridotta. Con uno schema BIO questo si traduce in una forte prevalenza della classe *O*, associata ai token esterni alle entità di interesse.

Uno sbilanciamento simile rende l'accuracy poco informativa se presa isolatamente: un modello potrebbe raggiungere un valore elevato predicendo sistematicamente la classe dominante, fallendo nell'estrazione dei campi fiscali rilevanti.

La quantificazione del fenomeno sul dataset proprietario e le strategie adottate per affrontarlo sono discusse nel Capitolo 3 e nei capitoli sperimentali.

Un'ulteriore difficoltà è costituita dall'ambiguità contestuale, poiché lo stesso tipo di valore può comparire più volte nel documento con ruoli semantici diversi. Uno scontrino può riportare prezzi unitari, subtotali, totale pagato e importi detraibili; una fattura può contenere imponibile, IVA, totale del documento, sconti e importi parziali. Il modello deve pertanto distinguere non solo il contenuto del testo, ma quale funzione svolge quel valore nel documento. Questa distinzione dipende dalla relazione spaziale.

Questo insieme di caratteristiche rende i documenti semi-strutturati un dominio particolarmente complesso per la Document AI. Un sistema efficace deve combinare robustezza visiva, comprensione testuale, modellazione del layout e capacità di generalizzazione. La KIE su documenti fiscali si conferma un banco di prova adatto a confrontare approcci OCR-based, fondati sulla classificazione dei token, e approcci generativi Vision-Language, orientati a produrre direttamente una rappresentazione strutturata.

2.3 Approcci tradizionali OCR-based e pipeline rule-based

I primi sistemi di estrazione automatica da documenti si basavano su pipeline modulari e fortemente ingegnerizzate. Il documento veniva acquisito come immagine, convertito in testo tramite OCR e infine analizzato da un insieme di regole manuali, template o modelli testuali. L'estrazione era dunque il prodotto di più componenti indipendenti, disposte in sequenza.

In scenari più controllati questi sistemi garantivano prestazioni soddisfacenti. In presenza di un layout fisso e noto a priori, l'estrazione si basava su tecniche di Zonal OCR, cioè sull'analisi di regioni predefinite della pagina: un campo come la data, il codice fiscale o il totale veniva cercato all'interno della stessa area del documento. In alternativa si ricorreva a keyword, espressioni regolari o pattern testuali per individuare valori dalla forma riconoscibile, come date, importi monetari o codici alfanumerici.

Questa logica risultava efficace soprattutto con documenti strutturati come moduli standardizzati o template amministrativi stabili. Il sistema non doveva realmente generalizzare, poiché sfruttava la prevedibilità della posizione o del formato delle informazioni. Il limite emergeva appena il layout cambiava. Una scansione ruotata, una tabella leggermente diversa, una sezione spostata o una qualità visiva inferiore erano sufficienti a rendere inefficaci regole costruite per una configurazione specifica.

Una seconda fase dell'evoluzione degli approcci OCR-based ha introdotto maggiore flessibilità trattando l'estrazione come problema di Natural Language Processing. In questo paradigma il motore OCR produceva una trascrizione del documento, spesso linearizzata secondo un ordine di lettura standard, che veniva successivamente analizzata da un modello testuale. Il task poteva essere formulato come Named Entity Recognition, con modelli come Conditional Random Fields, reti ricorrenti, LSTM o, più di recente, Transformer testuali.

Rispetto ai sistemi puramente rule-based, questi metodi erano più adattabili, perché apprendevano regolarità dai dati invece di dipendere solo da regole definite manualmente. Mantenevano tuttavia il limite comune per cui il documento veniva trattato prevalentemente come testo lineare. L'immagine originaria e la struttura bidimensionale della pagina venivano semplificate o perse durante la conversione OCR.

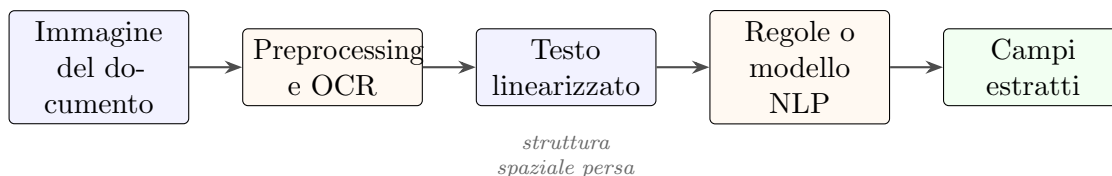


Figura 2.1: Pipeline tradizionale di estrazione basata su OCR.

Il limite principale di questa impostazione è la propagazione dell'errore. Ogni

modulo dipende dall'output del precedente; se l'OCR produce una trascrizione incompleta, coordinate imprecise o un ordine di lettura incoerente, il sistema di estrazione a valle riceve un input già compromesso. Nei documenti fiscali acquisiti in condizioni reali, anche un errore locale su una cifra, una data o un codice fiscale può quindi compromettere il valore finale estratto.

Un secondo limite è la perdita del layout. Molte pipeline tradizionali riducono la pagina a una sequenza monodimensionale di token, rappresentazione problematica nei documenti semi-strutturati, dove il significato di un valore dipende spesso dalla relazione spaziale con intestazioni, colonne, sezioni e chiavi vicine. In presenza di tabelle, layout multi-colonna o allineamenti irregolari, la linearizzazione può quindi separare elementi vicini nello spazio o accostare artificialmente parole appartenenti a sezioni diverse.

Resta il problema della scarsa generalizzazione. Regole fondate su coordinate, keyword o pattern testuali possono funzionare su documenti simili, ma diventano fragili al variare del formato, della qualità dell'immagine o della disposizione grafica. Ciò non rende obsoleti gli approcci tradizionali, che in domini controllati restano semplici, efficienti e interpretabili; li rende però meno adatti a scenari come quello di TaxWiz, caratterizzato da documenti fiscali eterogenei, immagini acquisite tramite fotografia, layout variabili e campi semanticamente ambigui da estrarre.

Queste criticità motivano il passaggio, discusso nelle sezioni successive, verso modelli in grado di integrare contenuto linguistico, posizione spaziale e informazione visiva.

2.4 Transformer e passaggio ai modelli multimodali

L'evoluzione recente della Document AI è strettamente legata allo sviluppo dell'architettura Transformer. Introdotta da Vaswani et al. [VSP⁺17] nel lavoro *Attention Is All You Need*, questa architettura ha innovato la modellazione delle sequenze testuali, sostituendo la ricorrenza tipica di RNN, GRU e LSTM con un meccanismo fondato sull'attenzione.

Nei modelli ricorrenti i token vengono elaborati sequenzialmente. L'impostazione è naturale per le sequenze, ma limita la parallelizzazione e complica la modellazione delle dipendenze a lungo raggio. Il Transformer procede diversamente: ogni token interagisce direttamente con tutti gli altri attraverso il meccanismo di self-attention; in questo modo, la rappresentazione di una parola dipende non solo dalla parola stessa, ma dal contesto globale in cui compare.

Il meccanismo di attenzione scalata si esprime come:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

dove Q , K e V rappresentano rispettivamente query, key e value, mentre d_k è la dimensione dei vettori key. La formulazione consente al modello di stimare quali

elementi della sequenza siano più rilevanti per costruire la rappresentazione di ciascun token.

Il successo dei Transformer è riconducibile anche al paradigma pre-training/fine-tuning, consolidato da modelli come BERT [DCLT19]. Un modello viene prima addestrato su grandi quantità di testo non annotato tramite obiettivi auto-supervisionati, come il Masked Language Modeling, e successivamente adattato a un task specifico con fine-tuning supervisionato. In questo modo rappresentazioni linguistiche generali si trasferiscono a compiti più circoscritti come classificazione, Named Entity Recognition e question answering anche in presenza di dataset annotati di dimensioni limitate.

Nella loro formulazione originale, tuttavia, i Transformer testuali ereditano i limiti già discussi dei sistemi che trattano il documento come sequenza monodimensionale, poiché la linearizzazione necessaria a elaborare il testo estratto dall'OCR scarta la geometria della pagina e gli indizi visivi che concorrono a definire il ruolo semantico dei valori. Il meccanismo di self-attention, per quanto efficace nel collegare token distanti nella sequenza, non è in grado di recuperare l'informazione spaziale eliminata a monte. Per questo motivo si effettua il passaggio verso architetture multimodali per documenti, capaci di estendere il Transformer oltre la sola sequenza testuale. La famiglia LayoutLM rappresenta una delle evoluzioni principali di questo approccio, poiché integra informazioni linguistiche, spaziali e visive all'interno di una rappresentazione comune.

Una direzione complementare è rappresentata dai Vision-Language Models generativi, che mantengono la centralità del Transformer ma la declinano in un paradigma autoregressivo, discusso nella sezione dedicata.

Nel quadro di questa tesi il Transformer costituisce la base architetturale comune dei due modelli analizzati. Le sezioni successive descrivono come questa base venga declinata, da un lato, nei modelli specializzati per documenti della famiglia LayoutLM e, dall'altro, nei Vision-Language Models generativi come Qwen2.5-VL.

2.5 Modelli multimodali specializzati per documenti: la famiglia LayoutLM

Il superamento dei modelli puramente testuali ha portato la Document AI verso architetture specializzate nella comprensione dei documenti. Tra queste, la famiglia LayoutLM è uno dei riferimenti principali per i task di Document Understanding e Key Information Extraction.

LayoutLMv1 [XLC⁺20] introduce l'uso esplicito delle coordinate bidimensionali dei token all'interno di un'architettura Transformer. A partire dal testo estratto tramite OCR, il modello associa a ciascuna parola la posizione della corrispondente bounding box nella pagina. Rispetto a un modello puramente testuale quale BERT, tale estensione permette di rappresentare sia il contenuto linguistico, sia la collocazione geometrica degli elementi nello spazio del documento. Questa informazione risulta fondamentale nei task di estrazione, poiché consente di distinguere valori

sintatticamente simili in base alla loro posizione e alla relazione spaziale con gli altri elementi della pagina.

LayoutLMv2 [XXL⁺21] consolida questa impostazione attraverso una più stretta integrazione tra testo, layout e immagine. La rappresentazione del documento include caratteristiche visive estratte direttamente dalla pagina; in questo modo, l'immagine concorre al processo di comprensione, e la struttura grafica, i loghi, i font e i separatori diventano segnali informativi per interpretare il ruolo dei token. Tale evoluzione incrementa l'efficacia del modello sui Visually Rich Documents, in cui il significato è definito sia dalle componenti testuali, sia dall'organizzazione visiva della pagina.

LayoutLMv3 [HLC⁺22] compie un ulteriore passo verso una rappresentazione unificata delle modalità. Il modello integra testo, layout e immagine in un'unica architettura Transformer attraverso una strategia di pre-training multimodale pensata per apprendere relazioni tra parole, coordinate e contenuto visivo. Rispetto alle versioni precedenti riduce la separazione tra componente testuale e componente visiva, trattando il documento come combinazione di token testuali, informazioni spaziali e regioni dell'immagine elaborate congiuntamente.

Per la Key Information Extraction questa impostazione si presta bene a una formulazione discriminativa del task. Il documento viene elaborato a partire da parole OCR, bounding box e immagine, e il modello produce una predizione associata ai token, tipicamente mediante schema di etichettatura BIO. L'output così ottenuto è localizzabile: ogni campo estratto è riconducibile a uno span testuale e, di conseguenza, a una regione precisa della pagina. Tale caratteristica rappresenta un vantaggio significativo nei domini applicativi in cui la verificabilità del risultato è vincolante, come nel settore fiscale.

I punti di forza della famiglia LayoutLM derivano proprio dalla loro specializzazione. Rispetto ai modelli puramente testuali, conservano l'informazione sulla struttura bidimensionale del documento; rispetto ai sistemi rule-based, apprendono regolarità dai dati e si adattano meglio a layout variabili; rispetto ai grandi modelli generativi, sono in genere più leggeri, più efficienti in inferenza e più controllabili nella forma dell'output. Per un task con campi target definiti, come quello di questa tesi, l'impostazione consente di modellare direttamente l'estrazione come classificazione dei token.

Resta però la limitazione strutturale della dipendenza dall'OCR. Anche se LayoutLMv3 sfrutta informazione visiva e spaziale, la formulazione token-level richiede che testo e bounding box siano disponibili a monte. Se l'OCR non riconosce una parola, la trascrive male o assegna coordinate imprecise, il modello riceve una rappresentazione incompleta o corrotta del documento.

Un ulteriore limite concerne i documenti lunghi, densi o multipagina. Operando su una sequenza limitata di token e su una rappresentazione visiva della pagina, il modello incontra vincoli pratici relativi al volume di informazioni elaborabili in un singolo passaggio. In presenza di documenti con un numero elevato di parole o strutturati su più pagine, può rendersi necessario il troncamento, la suddivisione o l'analisi pagina per pagina. Questo aspetto assume rilevanza nel dominio fiscale,

dove i documenti variano molto per lunghezza, densità informativa e struttura.

Per tale ragione, nel confronto sperimentale della tesi, LayoutLMv3 rappresenta il paradigma dei modelli multimodali specializzati per documenti: una pipeline OCR-aware, discriminativa e controllabile.

2.6 Vision-Language Models generativi e Qwen2.5-VL

Negli ultimi anni la Document AI è stata influenzata dall'evoluzione dei Vision-Language Models e dei Multimodal Large Language Models. A differenza dei modelli specializzati OCR-based, che ricevono in input testo, bounding box e immagine del documento, questi modelli trattano il documento prevalentemente come input visivo, eventualmente accompagnato da un'istruzione testuale. L'output non è una sequenza di etichette associate ai token, ma una risposta generata in linguaggio naturale o in un formato strutturato.

Per il task di Key Information Extraction questa impostazione consente di formulare il problema in modo diretto: il modello riceve in input l'immagine del documento e un prompt descrittivo che specifica quali campi estrarre, producendo una risposta già nella forma attesa dal sistema applicativo. Nel contesto di questo lavoro di tesi, tale risposta è un oggetto JSON con un insieme predefinito di campi fiscali.

Questi modelli combinano una componente visiva, che trasforma l'immagine in rappresentazioni numeriche, con un modello linguistico autoregressivo capace di generare testo condizionato sia dall'immagine sia dal prompt. Il documento non viene quindi interpretato come una sequenza OCR, ma come input multimodale. Centrale in questo paradigma è il meccanismo di instruction tuning. Il modello viene addestrato a seguire istruzioni in linguaggio naturale, ad esempio “estrai la data e il totale dal documento” oppure “restituisce i campi richiesti in formato JSON”. Il prompting consente così di specificare il task direttamente nell'input, permettendo di modificare i campi richiesti, il formato della risposta o le regole di generazione senza necessariamente intervenire sull'architettura. In contesti specifici il comportamento può essere guidato ulteriormente tramite esempi inseriti nel prompt, secondo la logica dell'in-context learning. Questa flessibilità è utile nella KIE, ma la natura generativa introduce criticità intrinseche: il modello può omettere campi presenti, generare valori assenti dal documento, alterare cifre o date, oppure produrre output non valido sul piano sintattico. In un dominio fiscale, errori di questo tipo richiedono attenzione particolare.

Il modello generativo adottato nella tesi è Qwen2.5-VL-7B-Instruct. La scelta nasce dalla sua natura multimodale, dalla disponibilità open-source, dalla capacità di seguire istruzioni testuali e dalla possibilità di produrre risposte strutturate a partire da immagini. Nel confronto sperimentale rappresenta il paradigma generativo image-to-JSON, contrapposto all'approccio token-level e OCR-based di LayoutLMv3.

Una caratteristica rilevante di Qwen2.5-VL è la gestione dinamica della risoluzione dell'immagine. Nei documenti fiscali il testo può essere molto denso e di dimensioni ridotte, soprattutto in scontrini, ricevute e fatture fotografate; ridimensionamenti troppo aggressivi rischiano di comprometterne la leggibilità su importi, date o codici. L'elaborazione a risoluzione dinamica permette al modello di adattarsi a immagini di dimensioni e densità differenti, preservando meglio il contenuto informativo. Nel dominio documentale risulta rilevante anche la possibilità di fornire input visivi multipli. I documenti fiscali non sono tutti composti da una singola pagina: diverse fatture si estendono su più pagine o vengono acquisite tramite fotografie separate. La capacità di elaborare input visivi multipli, eventualmente intercalati con testo, avvicina il paradigma Vision-Language agli scenari reali, in cui il documento non coincide sempre con una singola immagine isolata.

A sostegno della scelta interviene anche la valutazione su benchmark di document parsing. OmniDocBench [OQZ⁺25] misura modelli e pipeline su documenti complessi, considerando riconoscimento del testo, formule, tabelle e ordine di lettura. Su questo benchmark la famiglia Qwen2.5-VL ottiene risultati competitivi tra i Vision-Language Models generici: sulla leaderboard OmniDocBench v1.5 la variante da 72B raggiunge un punteggio Overall dell'87,02%, indice della maturità della famiglia sui compiti di parsing documentale. La variante da 7 miliardi di parametri adottata in questa tesi ne condivide l'impostazione, con capacità inferiori a fronte di un costo computazionale sostenibile.

Nel confronto sperimentale, questo paradigma viene quindi valutato per la sua capacità di offrire una formulazione più flessibile del task, bilanciata però da costi computazionali più elevati e da un controllo meno immediato sull'output generato.

2.7 Confronto tra approcci OCR-based e OCR-free

Il percorso descritto nelle sezioni precedenti evidenzia due paradigmi principali per affrontare la Key Information Extraction da documenti semi-strutturati. Da un lato gli approcci OCR-based, in cui il documento viene prima convertito in testo e coordinate spaziali da un sistema OCR, poi elaborato da un modello specializzato. Dall'altro si posizionano gli approcci generativi Vision-Language, in cui il documento è fornito direttamente come immagine e il modello produce una risposta strutturata a partire da un prompt.

Nel primo caso, rappresentato in questo lavoro di tesi da LayoutLMv3, la pipeline è esplicita e modulare: Amazon Textract fornisce testo e bounding box, il modello assegna etichette ai token e le sequenze BIO vengono convertite nei campi da estrarre. Il vantaggio principale è la verificabilità dello span estratto; il limite è la dipendenza dalla qualità della rappresentazione prodotta dall'OCR.

Nel secondo caso, rappresentato da Qwen2.5-VL, il task è formulato come generazione condizionata dall'immagine e da un'istruzione testuale, con produzione diretta di un oggetto JSON. Il vantaggio principale è la flessibilità della formulazione;

il limite è il controllo più complesso dell'output, che può contenere omissioni, valori alterati, allucinazioni o JSON non valido.

Il contrasto tra i due paradigmi non si riduce quindi alle metriche finali, ma riguarda il trade-off tra controllo e flessibilità. Un approccio OCR-based ancora ogni predizione a token e coordinate, e offre localizzazione e verificabilità a prezzo di una pipeline più rigida e dipendente dall'OCR. Un approccio generativo Vision-Language produce direttamente la struttura desiderata e si adatta con maggiore flessibilità al task, a prezzo di un costo computazionale maggiore e di un controllo meno immediato sull'output. Il confronto operativo dettagliato tra le due architetture è sviluppato nel Capitolo 4, mentre il confronto sperimentale sui risultati è presentato nel Capitolo 7.

La scelta tra i due approcci non ammette una risposta assoluta. In un contesto applicativo reale il modello più adatto dipende dal tipo di documenti, dalla qualità dell'acquisizione, dal volume di dati da elaborare, dai vincoli hardware, dalla necessità di spiegare o verificare le predizioni e dal tipo di errore ritenuto più accettabile. Un sistema OCR-based è preferibile quando contano soprattutto la localizzazione dello span e l'efficienza; un modello generativo diventa più interessante quando pesano la flessibilità del task e la gestione di layout eterogenei.

Capitolo 3

Dataset

Questo capitolo descrive la costruzione del dataset impiegato per l'addestramento e la valutazione dei modelli. Nei task di Key Information Extraction la qualità dei dati incide in misura determinante sui risultati: la trattazione non si esaurisce nel conteggio dei record disponibili, ma ricostruisce il percorso che conduce dai documenti fiscali grezzi a un formato conforme alle pipeline sperimentali, attraverso le fasi di raccolta, esportazione, annotazione e trasformazione.

Il capitolo prosegue con l'analisi esplorativa del dataset, la quantificazione dello sbilanciamento tra le classi e i criteri di preprocessing adottati per preparare i dati alle pipeline basate su LayoutLMv3 e Qwen2.5-VL.

3.1 Raccolta e origine dei dati

Il dataset utilizzato in questa tesi è stato costruito a partire da documenti fiscali reali raccolti nel contesto applicativo di TaxWiz, una piattaforma per la gestione e l'elaborazione automatica di documentazione fiscale. I dati derivano da una collaborazione con uno studio commercialista e riflettono uno scenario operativo concreto, distinto da quello dei dataset pubblici comunemente impiegati nella letteratura sul document understanding.

Dataset di benchmark come CORD [PSL⁺19] o SROIE [HCH⁺19] costituiscono raccolte standardizzate per la ricerca sull'estrazione da ricevute e scontrini. Il dataset analizzato in questo studio raccoglie invece documenti fiscali italiani provenienti da un flusso operativo reale, in prevalenza scontrini, ricevute e fatture acquisiti sotto forma di immagini o documenti digitalizzati. Questa provenienza introduce variabilità proprie dell'uso pratico: qualità visiva disomogenea, condizioni di illuminazione sfavorevoli, scontrini termici sbiaditi, documenti alterati da piegature o deterioramenti e layout geometrici eterogenei.

Lavorare su dati di origine applicativa consente di valutare i modelli in condizioni prossime a quelle che un sistema di Key Information Extraction affronta in produzione. Il task non si riduce al riconoscimento corretto del testo, ma richiede di localizzare pochi campi fiscalmente significativi all'interno di documenti caratterizzati da rumore visivo e variabilità strutturale.

I documenti sono stati gestiti tramite una piattaforma dedicata, che permette di visualizzare i file acquisiti e di annotare manualmente i valori di interesse. L'annotazione ha riguardato i principali campi rilevanti per il dominio fiscale considerato: data del documento, codice fiscale, totale del documento, totale detraibile, nome e cognome, anno, importo, titolo edilizio e tracciabilità del pagamento. Questi campi costituiscono le entità target delle fasi successive di costruzione del dataset e di addestramento dei modelli.

Trattandosi di documenti di origine applicativa, contenenti dati potenzialmente sensibili, il dataset non è pubblico ed è stato utilizzato esclusivamente in ambiente controllato. Gli esempi riportati nella tesi sono anonimizzati o selezionati in modo da non esporre dati personali riconducibili a soggetti reali.

La raccolta privilegia quindi la rappresentatività rispetto alla pulizia artificiale del dato. Questa scelta comporta criticità legate alla qualità dell'immagine, alla variabilità del layout e alla correttezza dei campi annotati, ma consente di valutare i modelli in condizioni coerenti con l'obiettivo applicativo della tesi.

3.2 Acquisizione OCR e annotazione manuale

Dopo la raccolta delle immagini, il testo e le informazioni spaziali dei documenti sono stati estratti tramite Amazon Textract, integrato come servizio OCR nella pipeline di backend di TaxWiz. L'elaborazione avviene in modalità asincrona: i file caricati nella piattaforma vengono archiviati su Amazon S3, analizzati da Textract e infine salvati con i risultati OCR nel database. La modalità asincrona viene impiegata principalmente per i documenti multi-pagina, la cui analisi OCR può richiedere tempi prolungati.

L'output di Textract è una rappresentazione strutturata del documento: per ogni parola riconosciuta restituisce il testo, un identificatore univoco e la bounding box corrispondente. Gli identificatori dei blocchi OCR consentono di collegare le annotazioni manuali agli elementi testuali riconosciuti dal sistema.

L'annotazione delle entità è stata eseguita manualmente da uno studio commercialista attraverso un'interfaccia dedicata integrata nella piattaforma. Il coinvolgimento di professionisti del dominio fiscale garantisce annotazioni coerenti con i criteri applicativi adottati. Per ciascun documento, l'operatore individuava visivamente i valori di interesse e li associava ai blocchi Textract corrispondenti tramite il rispettivo identificatore univoco. L'annotazione non si limita quindi a registrare il valore atteso di un campo, ma cerca di collegarlo direttamente agli elementi OCR che lo rappresentano, qualora l'informazione sia disponibile.

Questo collegamento costituisce un passaggio centrale nella costruzione del dataset, perché trasforma un'annotazione a livello di campo, ad esempio "totale documento = 23,50", in una supervisione a livello di token.

La rappresentazione token-level è necessaria per formulare il task come token classification e per costruire le etichette BIO usate da LayoutLMv3, consentendo di assegnare a ogni parola OCR un'etichetta semantica coerente con il campo di appartenenza, oppure il tag *O* quando il token non è rilevante.

Il campo pagamento tracciato costituisce un'eccezione, in quanto nel database non risultava associato ad alcun blocco Textract. Poiché la tracciabilità del pagamento si può inferire dalla presenza di espressioni specifiche nel documento, è stato adottato un approccio basato su regole lessicali. Nello specifico, un modulo dedicato analizza il testo OCR e associa il campo ai token che corrispondono a termini indicativi di pagamento elettronico o tracciato, come “Bancomat” o “PagoPA”.

La soluzione ha permesso di includere anche il campo pagamento tracciato nella rappresentazione BIO. Allo stesso tempo, mette in evidenza una criticità ricorrente nei dati di origine applicativa, poiché non tutte le informazioni applicative risultano annotate in allineamento con gli elementi OCR. L'intervento rule-based non altera il significato del campo, ma lo rende utilizzabile nella formulazione token-level del task.

3.2.1 Esportazione e struttura del dataset

I dati sono stati estratti dal database con query PostgreSQL e organizzati in una struttura di cartelle locale tramite codice TypeScript dedicato.

Lo script interroga il database per identificare i record da elaborare, ne recupera dati e annotazioni associate, preleva i rispettivi file e output Textract da Amazon S3, e infine scarica tutto il materiale in cartelle locali. Tale configurazione delle cartelle è stata adottata per ottimizzare la preparazione dei dati alle successive elaborazioni in Python, agevolando la lettura dei record, la verifica manuale dei casi anomali e la costruzione di un dataset uniforme per i modelli.

Per ciascun record viene generata una cartella identificata dall'ID univoco del documento. La directory include la cartella `preview`, in cui sono memorizzati i file fotografici originali; la cartella `textextract`, che ospita il JSON grezzo prodotto dall'OCR; la cartella `fields`, contenente il JSON dei campi attesi in cui le annotazioni delle entità risultano già associate ai corrispondenti identificatori dei blocchi.

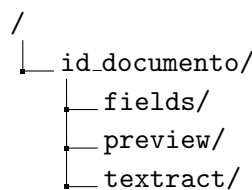


Figura 3.1: Schema della struttura delle cartelle.

A partire da questa struttura, uno script Python esegue le fasi di caricamento, validazione, normalizzazione, associazione e mappatura dei dati nel formato finale. Lo script itera su tutti i documenti e genera un record per ciascuna pagina, strutturato come segue:

```

1 {
2     'id':          str,                # identificatore univoco del documento
3     'image':       str,                # percorso al file immagine
4     'words':       list[str],          # sequenza di token testuali estratti da Textract
5     'bboxes':      list[list[int]],    # coordinate normalizzate [x0, y0, x1, y1] per
                                     token

```

```
6 |     'labels': list[str],          # etichette BIO in formato stringa (es. 'B-  
7 |         TotaleDocumento', '0')  
7 |     'ner_tags': list[int],        # etichette BIO codificate come indici interi  
8 | }
```

Il dataset così strutturato costituisce il punto di partenza comune alle due pipeline sperimentali, compatibile con i requisiti di preprocessing di entrambi i modelli.

Per LayoutLMv3 i dati alimentano il task di token classification tramite le etichette BIO associate ai token.

Per Qwen2.5-VL le medesime informazioni sono riformulate in coppie immagine-risposta, dove l'immagine della pagina funge da input visivo e il JSON dei campi attesi rappresenta il target generativo.

Questa duplice rappresentazione consente di confrontare i due modelli sullo stesso dataset, mantenendo invariato il problema di estrazione e adattandolo alle rispettive architetture.

3.3 Analisi esplorativa

L'analisi esplorativa è stata condotta in seguito alla costruzione del dataset e prima del preprocessing specifico per i modelli, con l'obiettivo di verificare la correttezza e la coerenza strutturale dei dati e di comprendere la distribuzione delle entità annotate, al fine di individuare le criticità rilevanti per l'addestramento.

L'analisi ha interessato il dataset finale, composto da 1.919 record validi, in cui ciascun record corrisponde a una singola pagina documentale. La granularità a livello di pagina deriva dall'impostazione adottata per LayoutLMv3, che elabora sequenze fino a un limite di 512 token. Questo vincolo dimensionale esclude il trattamento congiunto di interi documenti multipagina e impone di gestire ogni pagina come record indipendente, per cui anche i documenti strutturati su più pagine risultano suddivisi in record separati.

Un primo controllo ha riguardato l'integrità strutturale del dataset. La funzione `check_integrity` verifica la validità delle coordinate spaziali e la coerenza tra le tre sequenze parallele di ogni record: parole, bounding box ed etichette associate. Nei controlli effettuati non sono state rilevate anomalie o disallineamenti sui record analizzati. L'esito positivo della verifica assicura che ogni token sia correttamente associato alla propria coordinata spaziale e alla relativa etichetta, validando i dati per le successive fasi di preprocessing e addestramento.

L'approccio qualitativo iniziale è consistito nella visualizzazione manuale di alcuni campioni annotati. La sovrapposizione delle entità riconosciute all'immagine del documento permette di convalidare a vista la corrispondenza geometrica tra testo, bounding box e annotazioni, verificando così la qualità del dato.

3.3. ANALISI ESPLORATIVA



Figura 3.2: Esempio di fattura.

3.3.1 Distribuzione delle etichette a livello di token

L'analisi quantitativa evidenzia un forte sbilanciamento a livello di token, un fenomeno ricorrente nei task di NER applicati a documenti fiscali. In questo dominio la maggior parte del testo non veicola informazioni utili all'estrazione, e il dataset in esame conferma questa tendenza.

Tabella 3.1: Distribuzione dei token per etichetta.

ID	Etichetta	Token	% sul totale
16	O	308.778	97,87
5	B-PagamentoTracciato	1.715	0,54
2	B-DataDocumento	1.464	0,46
10	I-DataDocumento	1.136	0,36
1	B-CodiceFiscale	1.109	0,35
8	B-TotaleDocumento	509	0,16
12	I-PagamentoTracciato	346	0,11
7	B-TotaleDetraibile	121	0,04
9	I-CodiceFiscale	115	0,04
11	I-NomeCognome	73	0,02
4	B-NomeCognome	60	0,02
13	I-TitoloEdilizio	28	0,01
15	I-TotaleDocumento	18	0,01
3	B-Importo	17	0,01
0	B-Anno	7	< 0,01
14	I-TotaleDetraibile	4	< 0,01
6	B-TitoloEdilizio	3	< 0,01
Totale		315.503	100,00

L'asimmetria non costituisce un artefatto della fase di raccolta, bensì riflette la struttura intrinseca dei documenti fiscali italiani: un tipico documento d'acquisto include una quantità elevata di token di testo non rilevante ai fini del task, a fronte

di un sottoinsieme ridotto di token portatori delle informazioni target. Un modello addestrato senza accorgimenti metodologici specifici presenta quindi il rischio di privilegiare la classe maggioritaria O a scapito delle entità di interesse.

3.3.2 Distribuzione a livello di entità e copertura documentale

L'analisi a livello di entità rileva le occorrenze complete di ciascuna categoria, indipendentemente dal numero di token che la compongono, approfondendo il quadro emerso a livello di token. Questo permette di osservare la frequenza di ogni etichetta, la rappresentazione di ciascuna categoria sul piano semantico e la relativa distribuzione tra i documenti.

Tabella 3.2: Distribuzione delle entità per campo.

Campo	Entità	%	Token medi	Mediana	Max
PagamentoTracciato	1.715	34,27	1,20	1,0	3
DataDocumento	1.464	29,25	1,78	2,0	4
CodiceFiscale	1.109	22,16	1,10	1,0	4
TotaleDocumento	509	10,17	1,04	1,0	4
TotaleDetraibile	121	2,42	1,03	1,0	3
NomeCognome	60	1,20	2,22	2,0	7
Importo	17	0,34	1,00	1,0	1
Anno	7	0,14	1,00	1,0	1
TitoloEdilizio	3	0,06	10,33	8,0	16
Totale	5.005	100,00			

Anche a livello semantico la distribuzione è fortemente sbilanciata. L'entità maggiormente rappresentata è il pagamento tracciato, con 1.715 occorrenze (pari al 34,27% delle entità annotate), seguito dalla data del documento con 1.464 occorrenze (29,25%) e dal codice fiscale con 1.109 occorrenze (22,16%). Il totale documento compare 509 volte (10,17%). Le restanti classi presentano frequenze ridotte: totale detraibile 121 volte, nome e cognome 60, mentre importo (17), anno (7) e titolo edilizio (3) mostrano frequenze marginali, ciascuna inferiore al 1% del totale.

Questi conteggi si riferiscono alle entità nello schema BIO, ottenute allineando le annotazioni ai token OCR riconosciuti. Il supporto field-level riportato nel Capitolo 7 deriva invece dalla ground truth ricostruita dalle annotazioni originali e non sempre coincide con tali conteggi. Un campo annotato ma non allineabile ad alcun token OCR non genera un'etichetta B-, pur restando presente nella valutazione field-level. Per le entità più rare il supporto nel test set può così superare il numero di entità BIO sull'intero dataset, come si verifica per il campo anno, senza che ciò costituisca un'incoerenza strutturale.

La posizione di pagamento tracciato in cima alla distribuzione va letta con cautela. Il campo è generato da regole lessicali che marcano ogni token corrispondente a

un'espressione di pagamento tracciato, per cui un singolo documento può generare più occorrenze. Le 1.715 entità si distribuiscono infatti su 727 documenti, con una media superiore a due occorrenze per singolo record. Il conteggio riflette pertanto il criterio di annotazione rule-based piuttosto che una superiore variabilità semantica del campo.

Il marcato sbilanciamento implica che alcune entità dispongano di un supporto statistico limitato. Le classi più rare risultano quindi difficili da apprendere in modo robusto, soprattutto in un'impostazione token-level dominata dalla classe *O*. Un caso specifico è il titolo edilizio che, pur con sole 3 occorrenze, presenta una lunghezza media di 10,33 token per entità e un massimo di 16. Tale entità descrive verosimilmente la denominazione completa di un intervento edilizio, quindi di un'entità descrittiva multi-token, difficile da apprendere sia per rarità sia per estensione. È una classe critica non solo sul piano quantitativo ma anche strutturale, poiché coinvolge span più lunghi e più variabili rispetto agli altri campi.

L'analisi della lunghezza media degli span conferma che la maggior parte delle entità è composta da un numero ridotto di token. Come si osserva nella tabella 3.2, la quasi totalità delle categorie presenta un'estensione estremamente ridotta, spesso vicina all'unità. In questo scenario, le classi relative a codici e importi si esauriscono generalmente in un singolo token, mentre l'ampiezza aumenta per le date e per le generalità anagrafiche. Il titolo edilizio costituisce un'eccezione isolata, caratterizzata da un'estensione testuale superiore a fronte di una spiccata rarità nel dataset.

Tabella 3.3: Copertura dei campi a livello di documento.

Campo	Documenti	Copertura %
DataDocumento	1.464	76,29
CodiceFiscale	1.109	57,79
PagamentoTracciato	727	37,88
TotaleDocumento	509	26,52
TotaleDetraibile	121	6,31
NomeCognome	60	3,13
Importo	17	0,89
Anno	7	0,36
TitoloEdilizio	3	0,16

La copertura documentale misura la percentuale di record in cui ciascun campo compare almeno una volta. La data del documento è presente nel 76,29% dei documenti e il codice fiscale nel 57,79%, confermando come tali campi siano tra i più ricorrenti nei documenti fiscali analizzati. Il pagamento tracciato copre il 37,88% dei record, mentre il totale documento compare nel 26,52%. La copertura contenuta del totale documento, inferiore rispetto alle attese per un campo primario nei documenti fiscali, si spiega con la presenza di documenti multipagina, nei quali l'importo totale figura sull'ultima pagina. Poiché nella pipeline le singole pagine diventano record separati, le pagine iniziali o intermedie del medesimo documento

non contengono tale campo, riducendone la copertura a livello di record.

Le classi restanti sono progressivamente più rare: il totale detraibile compare nel 6,31% dei record, il nome e cognome nel 3,13%, l'importo nello 0,89% e l'anno nello 0,36%. Il titolo edilizio è presente in soli 3 documenti su 1.919 (pari allo 0,16%), configurandosi come il campo più critico per la fase di addestramento.

Il confronto tra la frequenza assoluta delle entità e la copertura documentale evidenzia un andamento interessante: il pagamento tracciato guida la distribuzione delle entità ma scende al terzo posto per copertura, in quanto concentra più occorrenze negli stessi documenti anziché distribuirsi su un numero ampio di record.

Tali valori indicano che le entità non sono universalmente presenti, poiché alcuni campi risultano ricorrenti mentre altri dipendono dalla specifica tipologia documentale. Le prestazioni del modello non devono pertanto essere valutate esclusivamente tramite metriche aggregate, ma analizzate alla luce della frequenza e della copertura delle singole classi.

A livello di documento, l'analisi conferma una bassa densità informativa. In media, ciascun record contiene 3,50 token non appartenenti alla classe *O*, con una mediana di 3,0 e un valore massimo di 21. Il documento tipico include pertanto pochi token estrattivi rispetto alla quantità complessiva di testo. Questa sparsità rende il segnale di apprendimento debole e amplifica l'effetto dello sbilanciamento, soprattutto per le entità rare.

Nel complesso, questi elementi motivano le scelte metodologiche descritte nelle fasi successive, in particolare la gestione dello sbilanciamento e la valutazione per singola classe.

3.3.3 Analisi spaziale delle entità

Oltre alla distribuzione statistica delle classi, è stata esaminata la componente spaziale del dataset tramite heatmap delle posizioni delle entità. L'analisi risulta rilevante in quanto modelli multimodali, come quelli utilizzati, sfruttano non solo il contenuto testuale dei token ma anche la loro posizione sulla pagina.

L'obiettivo è valutare se le entità di interesse presentino distribuzioni spaziali ricorrenti, che il modello possa apprendere come segnale ausiliario durante l'estrazione. Per costruire le heatmap, per ciascuna entità è stato calcolato il centro del bounding box unificato dello span, così da rappresentare la posizione complessiva dell'informazione annotata nella pagina.

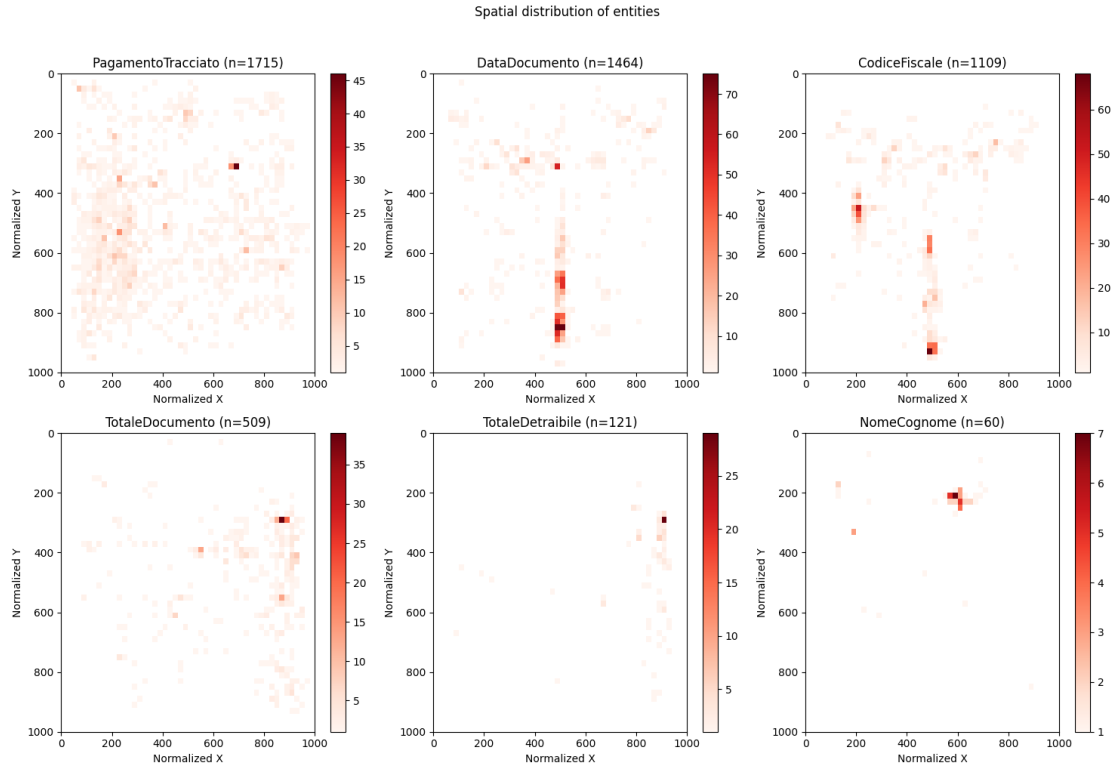


Figura 3.3: Distribuzione spaziale delle sei entità più frequenti nel layout dei documenti.

Le heatmap rivelano pattern spaziali ricorrenti per alcune entità e confermano che, nei documenti semi-strutturati, la posizione sulla pagina costituisce un segnale discriminativo non trascurabile.

Sebbene i documenti fiscali analizzati presentino layout eterogenei e le entità non occupino coordinate fisse, determinati campi, come date, totali o codici fiscali, tendono a concentrarsi in aree relativamente ricorrenti del documento.

Tale dato empirico conferma l'utilità di modelli capaci di integrare contenuto testuale e informazione spaziale, motivando la scelta di architetture multimodali rispetto a modelli puramente testuali, che ignorerebbero questa dimensione.

3.3.4 Osservazioni e implicazioni per il training

L'analisi esplorativa evidenzia alcune caratteristiche chiave del dataset:

- la classe *O* rappresenta il 97,87% dei token (tabella 3.1), confermando un forte sbilanciamento;
- ogni record contiene in media 3,50 token informativi;
- alcune entità hanno copertura documentale molto limitata (tabella 3.3), fino a poche unità su 1.919 documenti;

- la distribuzione spaziale mostra regolarità utili ma non rigide (fig. 3.3).

Tali evidenze definiscono i vincoli intrinseci del problema e motivano le scelte metodologiche discusse nella sezione successiva e nel Capitolo 4, in particolare la gestione dello sbilanciamento e l'adozione di metriche field-level.

3.4 Classi sbilanciate BIO e strategie di mitigazione

Nel dataset strutturato per LayoutLMv3 il task di Key Information Extraction è formulato come token classification mediante schema BIO, già introdotto nel Capitolo 2 e applicato nella fase di costruzione del dataset. Le entità corrispondono ai nove campi applicativi della piattaforma.

La formulazione token-level rende l'estrazione un problema supervisionato adatto a LayoutLMv3, ma eredita lo sbilanciamento quantificato nell'analisi esplorativa. Il problema riguarda sia il rapporto tra background e campi informativi, sia la distribuzione interna delle entità, con classi rare che dispongono di un supporto statistico troppo esiguo per apprendere pattern robusti e generalizzabili.

Queste caratteristiche incidono direttamente sul protocollo di valutazione, approfondito nella sottosezione dedicata.

La gestione dello sbilanciamento parte da due scelte operate già in fase di costruzione del dataset. La prima è la rimozione dei record composti esclusivamente da token *O*, che elimina gli esempi privi di segnale informativo senza però annullare lo squilibrio, dato che nei record mantenuti i token di background restano predominanti. La seconda è la rinuncia a un class-balanced sampling vero e proprio: in un task di token classification le classi non corrispondono a esempi indipendenti, ma sono distribuite all'interno delle sequenze, e sovracampionare i record con entità rare avrebbe amplificato artificialmente pochi esempi, con rischio di overfitting.

Lo sbilanciamento viene perciò affrontato soprattutto a valle e costituisce un vincolo centrale della pipeline sperimentale, motivando le scelte adottate nei capitoli successivi: funzione di loss dedicata, lettura prudente delle classi rare e valutazione orientata ai valori effettivamente estratti.

3.4.1 Classi critiche: anno, importo e titolo edilizio

Le classi anno e titolo edilizio si collocano in una condizione limite, in cui nessuna tecnica di bilanciamento può compensare la scarsità assoluta di esempi. Con 7 e 3 occorrenze rispettivamente sull'intero dataset, non offrono un supporto sufficiente nemmeno per una stima affidabile delle prestazioni, indipendentemente dalla strategia di loss adottata. La classe importo si colloca in una posizione intermedia. Con 17 occorrenze sull'intero dataset, è meno estrema di anno e titolo edilizio, ma resta troppo rara per apprendere un pattern stabile; nel test set questo campo risulterà non estratto correttamente da entrambi i modelli nel Capitolo 7, con un comportamento analogo a quello di anno. I tre campi vanno quindi letti

insieme come le entità non realmente apprendibili, pur con gradi diversi di rarità. Come osservato nella sezione precedente, questi conteggi si riferiscono alle entità BIO e non coincidono necessariamente con il supporto field-level riportato nel Capitolo 7.

Queste classi vengono mantenute nel dataset e nello schema di annotazione per completezza e per rispecchiare le esigenze applicative reali, ma i risultati ottenuti su tali categorie vanno interpretati soprattutto alla luce della loro scarsità.

Il loro mantenimento ha quindi anche valore diagnostico: eventuali miglioramenti prodotti dalla Focal Loss con pesi di classe indicherebbero la presenza di un segnale utile per le classi minoritarie, mentre un valore di F_1 nullo suggerirebbe la necessità di rivedere la loss o valutare strategie alternative di campionamento.

3.4.2 Scelta di non adottare una data augmentation estesa

Una strategia alternativa per mitigare lo sbilanciamento delle classi, ampiamente discussa in letteratura, è la data augmentation, operazione mirata a generare varianti sintetiche dei documenti rari per aumentarne la rappresentazione nel training set. In questa tesi non è stata applicata una data augmentation estesa per due ragioni.

La prima è di natura metodologica. L'introduzione di campioni sintetici o alterati avrebbe potuto compromettere la rappresentatività del dataset rispetto alla distribuzione dei documenti da elaborare in produzione, orientando il modello verso condizioni artificiali.

La seconda ragione è di carattere tecnico. L'addestramento di LayoutLMv3 richiede coerenza tra immagine, testo OCR, bounding box e annotazioni token-level. Qualsiasi trasformazione che alteri in modo significativo il contenuto testuale, la geometria dell'immagine o la posizione dei token avrebbe imposto una nuova esecuzione dell'OCR con Amazon Textract sull'intero corpus modificato: applicare augmentation alle immagini senza aggiornare l'output Textract avrebbe prodotto disallineamenti tra immagine, coordinate e testo. Una riesecuzione sistematica dell'OCR avrebbe comportato costi aggiuntivi, non giustificabili in assenza di un beneficio atteso dimostrabile.

Le perturbazioni visive, come sfocatura, riduzione di risoluzione o aggiunta di rumore, non vengono quindi impiegate come strategia principale di aumento del training set. Sono invece riservate allo studio di robustezza, dove vengono applicate in modo controllato per simulare condizioni di acquisizione degradate e misurare il decadimento delle prestazioni. In tale contesto, l'obiettivo non è aumentare la varietà del training set, ma valutare la stabilità del sistema rispetto a input visivamente degradati.

3.4.3 Implicazioni sulle metriche di valutazione

Lo sbilanciamento ha condizionato anche la scelta delle metriche di valutazione. Con una classe O dominante, le metriche globali come l'accuratezza risultano fuorvianti, per le ragioni già richiamate: un modello può etichettare correttamente

la quasi totalità dei token senza estrarre alcun campo utile. La valutazione si basa perciò su precision, recall e F_1 -score calcolati a livello di entità. L'obiettivo della misurazione non è stabilire quanti token siano correttamente etichettati, bensì verificare se il sistema recupera i valori rilevanti per l'applicazione.

A tali misure si affiancano metriche field-level, che confrontano direttamente i valori estratti con quelli attesi, consentendo un confronto omogeneo tra le due pipeline. La necessità nasce dalla diversa natura degli output: LayoutLMv3 produce una sequenza di etichette token-level, mentre Qwen2.5-VL genera nativamente valori strutturati in formato JSON. Riconducendo entrambi gli output a una rappresentazione comune campo-valore, il confronto misura la capacità effettiva di estrazione.

3.5 Split del dataset

Prima del preprocessing specifico per LayoutLMv3 e Qwen2.5-VL, il dataset è stato suddiviso una sola volta in training, validation e test set. Il risultato viene salvato su disco e utilizzato da entrambe le pipeline, così che i due modelli siano addestrati e valutati sugli stessi dati. Questa scelta è necessaria per un confronto sperimentale corretto: eventuali differenze osservate devono dipendere dai modelli e dalle rispettive pipeline, non da partizioni diverse del dataset.

Il dataset è ripartito secondo una proporzione approssimativa 80/10/10, che produce 1.538 record nel training set, 196 nel validation set e 185 nel test set. Il training set serve all'addestramento dei modelli; il validation set monitora le prestazioni durante il fine-tuning e supporta le scelte sulla configurazione sperimentale; il test set resta completamente separato e viene impiegato solo nella valutazione comparativa finale. In questo modo, le prestazioni vengono misurate su dati non utilizzati né durante l'addestramento né durante l'ottimizzazione degli iperparametri.

Un aspetto rilevante riguarda la prevenzione del data leakage. Ogni record corrisponde a una singola pagina documentale, ma alcuni documenti si compongono di più pagine. Una suddivisione casuale a livello di pagina avrebbe potuto assegnare pagine dello stesso documento a split diversi: il modello avrebbe potuto osservare in training una pagina di un documento presente poi in validation o in test, producendo una contaminazione tra insiemi capace di gonfiare le metriche, perché la valutazione non sarebbe avvenuta su documenti realmente indipendenti.

Per evitarlo è stata adottata la strategia `GroupShuffleSplit` di scikit-learn, che raggruppa i record appartenenti allo stesso documento. Il gruppo è definito a partire dall'identificativo del documento originario, ricavato dall'id della pagina; poiché gli id delle pagine contengono un suffisso del tipo `_page_N`, l'identificativo del documento si ottiene rimuovendo tale suffisso. Tutte le pagine di uno stesso documento finiscono così interamente nello stesso split.

```
1 groups = [get_document_id(sample_id) for sample_id in dataset["id"]]
2
3 splitter = GroupShuffleSplit(
4     n_splits=1,
```

3.5. SPLIT DEL DATASET

```
5     test_size=0.2,
6     random_state=42,
7 )
8
9 train_idx, temp_idx = next(splitter.split(range(len(dataset)), groups=groups))
10
11 train_dataset = dataset.select(train_idx)
12 temp_dataset = dataset.select(temp_idx)
13
14 validation_dataset = temp_dataset.select(valid_rel_idx)
15 test_dataset = temp_dataset.select(test_rel_idx)
16
17 dataset_splitted = DatasetDict(
18     {
19         "train": train_dataset,
20         "validation": validation_dataset,
21     }
22 )
```

Lo split procede in due passaggi. Il primo separa il training set da un insieme temporaneo pari al 20% del dataset. Il secondo ripartisce l'insieme temporaneo in validation e test set di dimensioni analoghe. La procedura mantiene la proporzione approssimativa 80/10/10 e rispetta il vincolo di non separare pagine dello stesso documento, da cui la lieve differenza tra le dimensioni di validation 196 e test 185.

Dopo la suddivisione, un controllo esplicito ha confermato l'assenza di sovrapposizioni tra gli identificativi dei documenti nei tre insiemi.

```
1 def document_ids(dataset):
2     return {get_document_id(sample_id) for sample_id in dataset["id"]}
3
4 train_docs = document_ids(dataset_splitted["train"])
5 valid_docs = document_ids(dataset_splitted["validation"])
6 test_docs = document_ids(dataset_splitted["test"])
7
8 assert train_docs.isdisjoint(valid_docs)
9 assert train_docs.isdisjoint(test_docs)
10 assert valid_docs.isdisjoint(test_docs)
```

Capitolo 4

Architettura dei modelli e implementazione

Questo capitolo descrive le architetture dei due modelli impiegati nella tesi e le principali soluzioni implementative adottate per il fine-tuning. Definita la struttura del dataset e la rappresentazione dei dati, il passo successivo è analizzare il funzionamento interno di LayoutLMv3 e Qwen2.5-VL, con particolare attenzione alle modalità di rappresentazione del documento e di risoluzione del task di Key Information Extraction.

I due modelli operano entrambi in un contesto multimodale, ma con strategie profondamente diverse. LayoutLMv3 è un encoder Transformer specializzato nella comprensione dei documenti, che integra il testo OCR, le coordinate spaziali e la componente visiva della pagina; il problema viene formulato come un processo di token classification tramite schema BIO.

Qwen2.5-VL è un Vision-Language Model autoregressivo progettato per compiti generativi; questo modello elabora congiuntamente l'immagine del documento e un prompt testuale per restituire un output strutturato, contenente i campi estratti.

Il capitolo presenta inizialmente le due architetture sul piano teorico e operativo, approfondendo i meccanismi di integrazione dell'informazione testuale, spaziale e visiva. Successivamente vengono esposte le procedure di fine-tuning, con attenzione ai vincoli hardware, alla gestione della memoria GPU, alla definizione della funzione di loss, all'adozione di tecniche parameter-efficient e alla costruzione degli input di addestramento. Il capitolo si conclude con un confronto operativo tra le due pipeline, che mette a fuoco le differenze principali in termini di dati richiesti, complessità implementativa, costo computazionale, verificabilità delle predizioni e adattabilità a scenari applicativi reali.

4.1 LayoutLMv3: architettura nel dettaglio

4.1.1 Panoramica e posizionamento

LayoutLMv3 è un modello Transformer multimodale pre-addestrato, sviluppato da Microsoft Research per task di Document AI. Il modello è concepito per integrare tre fonti informative centrali nella comprensione dei documenti: il contenuto testuale, la posizione spaziale del testo nella pagina e l'informazione visiva ricavata dall'immagine del documento concorrono alla rappresentazione vettoriale finale.

A differenza di un'architettura puramente testuale, LayoutLMv3 sfrutta questa integrazione per costruire una rappresentazione multimodale del documento. Tale distinzione risulta determinante nei documenti semi-strutturati, in cui il significato di un campo è correlato sia al contenuto testuale, sia alla sua posizione nella pagina e al contesto visivo circostante. LayoutLMv3 introduce una semplificazione architetturale rilevante, eliminando la dipendenza da reti convoluzionali come Faster R-CNN per l'estrazione delle feature visive. Questa componente viene sostituita da un approccio basato su patch, analogo a quello dei Vision Transformer, dove l'immagine del documento viene suddivisa in segmenti poi proiettati in uno spazio vettoriale. In questo modo, l'informazione testuale e visiva si integrano in sequenze di elementi elaborate linearmente dal medesimo Transformer.

Sul piano architetturale, LayoutLMv3 adotta una struttura encoder-only, analoga a BERT e RoBERTa. Il modello non genera testo in modo autoregressivo, ma produce rappresentazioni contestualizzate dell'input, adatte a classificazione, labeling e comprensione documentale. In questa tesi tali rappresentazioni alimentano un task di token classification formulato tramite schema BIO, finalizzato all'estrazione dei campi informativi da documenti fiscali.

La variante impiegata è `microsoft/layoutlmv3-base`: 12 layer Transformer encoder, 12 teste di attenzione, hidden size pari a 768 e circa 133 milioni di parametri. La lunghezza massima di sequenza in ingresso è 512 token, limite rilevante nel progetto perché alcuni documenti superano tale soglia e richiedono truncation durante il preprocessing.

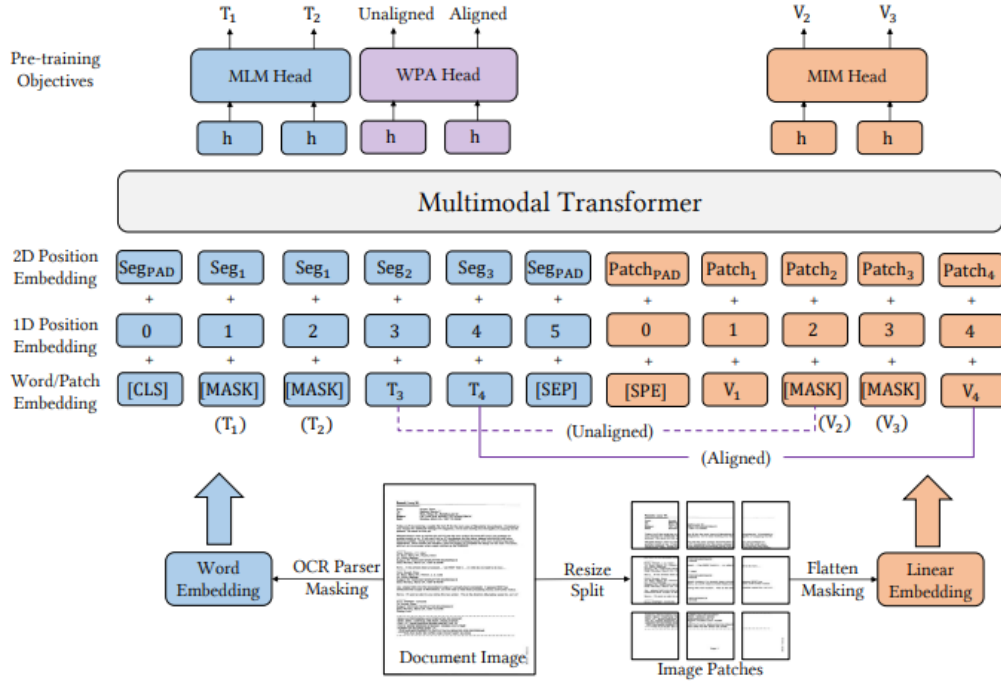


Figura 4.1: Architettura generale di LayoutLMv3. Tratto da [HLC⁺22].

4.1.2 Le tre modalità di input

La sezione seguente distingue le tre componenti dell'input di LayoutLMv3: modalità testuale, modalità spaziale e modalità visiva.

Tabella 4.1: Modalità di input di LayoutLMv3.

Modalità	Informazione rappresentata	Origine nel progetto	Ruolo nel modello
Testo	Token	Textextract	Comprensione semantica del contenuto
Layout	BBox normalizzate dei token	Textextract	Posizione spaziale e relazioni geometriche
Visione	Patch dell'immagine	Immagine RGB della pagina	Contesto visivo, struttura grafica e layout globale

Nel progetto, la modalità testuale e la componente spaziale sono generate da Amazon Textextract, mentre la modalità visiva deriva direttamente dall'immagine della pagina.

Modalità testuale: token embeddings

Nel preprocessing adottato in questa tesi, il documento viene prima elaborato tramite Amazon Textract, utilizzato come motore OCR esterno per ottenere le parole presenti nella pagina. Questa scelta è coerente con i requisiti di LayoutLMv3, che richiede in input sia il contenuto testuale sia l'informazione posizionale associata al testo riconosciuto.

Le parole estratte dall'OCR vengono poi tokenizzate tramite il tokenizer di LayoutLMv3, basato su Byte-Pair Encoding. Nella variante LayoutLMv3-base, i token vengono rappresentati in uno spazio nascosto di dimensione $d = 768$. Gli embedding testuali sono inizializzati a partire dalla matrice di embedding linguistica di RoBERTa e vengono arricchiti con embedding di posizione monodimensionale, che rappresentano l'ordine dei token nella sequenza testuale.

L'embedding testuale può essere espresso in forma semplificata come:

$$\mathbf{e}_i^{\text{text}} = \text{Embedding}(x_i) + \mathbf{e}_i^{\text{1D-pos}} + \mathbf{e}_i^{\text{layout}}$$

dove x_i indica l' i -esimo token testuale, $\mathbf{e}_i^{\text{1D-pos}}$ rappresenta la posizione del token nella sequenza e $\mathbf{e}_i^{\text{layout}}$ codifica la posizione bidimensionale del token nella pagina.

Modalità spaziale: layout embeddings

L'architettura usa embedding di posizione 2D per rappresentare la geometria del documento. La rappresentazione bidimensionale indica la collocazione geometrica di ciascun token all'interno della pagina.

Nel preprocessing adottato le bounding box vengono convertite nel formato richiesto dal modello e normalizzate in una scala discreta, compresa nell'intervallo $[0, 1000]$. Per ogni token, la bounding box è rappresentata dalle coordinate:

$$b_i = [x_{\min}, y_{\min}, x_{\max}, y_{\max}]$$

da cui si derivano larghezza e altezza:

$$w = x_{\max} - x_{\min}, \quad h = y_{\max} - y_{\min}$$

LayoutLMv3 utilizza embedding spaziali separati per rappresentare coordinate orizzontali, coordinate verticali e dimensioni della bounding box, combinandoli in un layout embedding associato al token:

$$\mathbf{e}_i^{\text{layout}} = \text{Emb}(x_{\min}) + \text{Emb}(y_{\min}) + \text{Emb}(x_{\max}) + \text{Emb}(y_{\max}) + \text{Emb}(w) + \text{Emb}(h)$$

Il meccanismo permette al modello di apprendere relazioni geometriche di prossimità, allineamento e gerarchia spaziale. Token con coordinate verticali simili possono appartenere alla stessa riga, mentre elementi allineati lungo un asse verticale possono indicare strutture tabellari o relazioni di tipo chiave-valore.

Modalità visiva: patch embeddings

La componente visiva di LayoutLMv3 segue una configurazione ispirata ai Vision Transformer. L'immagine del documento viene ridimensionata e suddivisa in una griglia di patch quadrate uniformi. Nello specifico, un'immagine di dimensione 224×224 pixel viene divisa in patch di lato 16 pixel, producendo 196 patch visive, trattate come elementi di una sequenza al pari dei token testuali.

Ogni patch viene appiattita e proiettata linearmente in uno spazio di dimensione $d = 768$, corrispondente allo spazio di rappresentazione del modello, generando un patch embedding. A ciascuna patch viene successivamente aggiunto un embedding di posizione monodimensionale.

Il patch embedding, in forma semplificata, è:

$$\mathbf{e}_k^{\text{patch}} = \text{LinearProj}(\text{patch}_k) + \mathbf{e}_k^{\text{1D-pos}}$$

dove patch_k indica la k -esima patch dell'immagine e $\mathbf{e}_k^{\text{1D-pos}}$ rappresenta la sua posizione nella sequenza delle patch.

Il passaggio alle patch costituisce una delle principali differenze rispetto alle versioni precedenti della famiglia LayoutLM e rende più uniforme il trattamento delle modalità testuale e visiva, riducendo la complessità del preprocessing visivo.

4.1.3 Fusione multimodale e ruolo del Transformer encoder

LayoutLMv3 combina le informazioni in una sequenza multimodale elaborata da un Transformer encoder. Gli embedding testuali, arricchiti con informazioni di posizione monodimensionale e bidimensionale, vengono concatenati con la sequenza delle patch visive. L'intera sequenza viene quindi processata dallo stesso encoder, dove token testuali e patch visive interagiscono tramite self-attention.

Il flusso concettuale si riassume così:

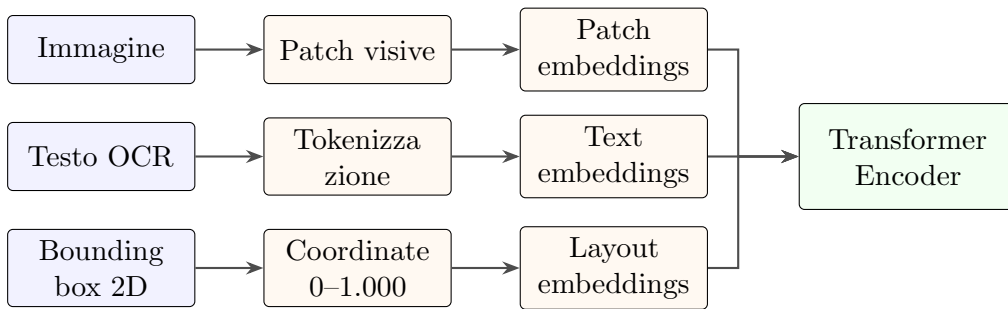


Figura 4.2: Fusione multimodale: i tre rami confluiscono nel Transformer encoder.

Attraverso questo meccanismo di integrazione precoce delle modalità, ciascun token testuale può calcolare pesi di attenzione sia rispetto agli altri elementi testuali, sia in relazione alle patch dell'immagine, e viceversa. Ciò consente al modello di apprendere relazioni multimodali profonde, come l'associazione tra una parola e la sua regione visiva. Questa architettura genera rappresentazioni contestualizzate

che superano i limiti dei modelli puramente testuali, poiché combinano il contenuto verbale riconosciuto dall'OCR, le coordinate spaziali e l'informazione visiva.

Nel contesto specifico dei documenti fiscali, tale sinergia permette di sfruttare congiuntamente il testo, la disposizione degli elementi nella pagina e il contesto grafico per mappare relazioni cruciali tra chiavi e valori vicini, migliorando la capacità di distinguere entità semanticamente diverse ma numericamente simili, come un importo parziale, un totale o una quota detraibile.

4.1.4 Pipeline concreta utilizzata nella tesi

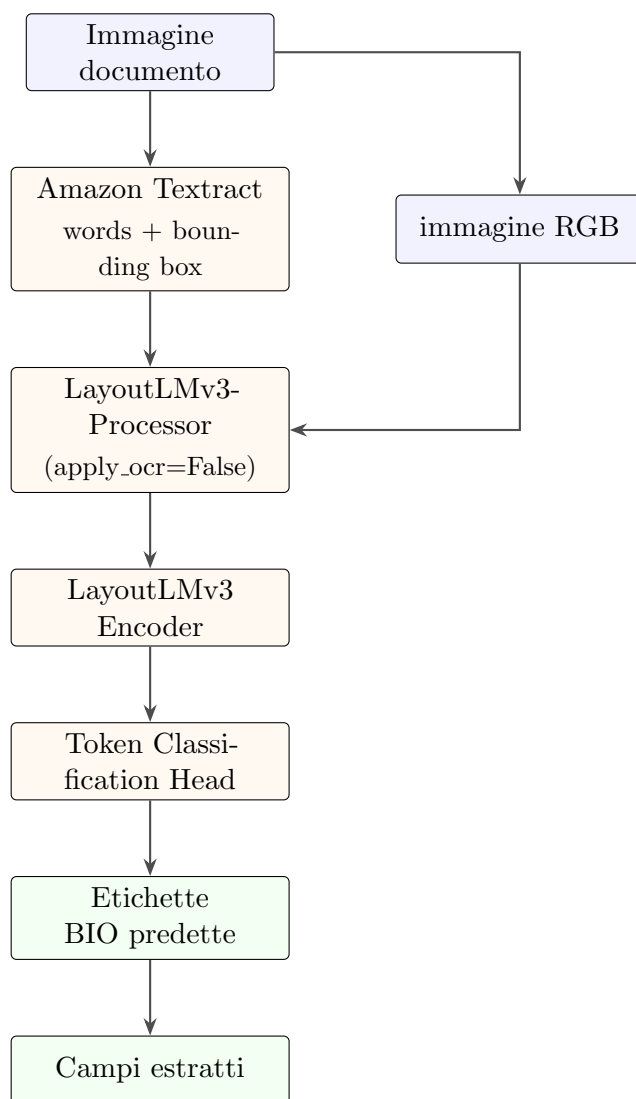


Figura 4.3: Pipeline di input di LayoutLMv3 nella tesi.

Durante il preprocessing, il processor riceve immagini, parole OCR, bounding box ed etichette token-level, gestendo il limite di 512 token del modello.

Il padding non viene applicato staticamente in questa fase, ma demandato al data

collator durante il training, così da effettuare padding dinamico a livello di batch ed evitare di estendere tutti gli esempi alla lunghezza massima quando non serve.

4.1.5 Obiettivi di pre-training: MLM, MIM e WPA

Le prestazioni di LayoutLMv3 dipendono sia dall'architettura multimodale sia dalla strategia di pre-training. Il modello è pre-addestrato con tre obiettivi auto-supervisionati complementari, che allineano congiuntamente testo, layout e immagine: Masked Language Modeling, Masked Image Modeling e Word-Patch Alignment [HLC⁺22].

Obiettivo	Modalità	Descrizione
Masked Language Modeling	Testo, layout, immagine	Predizione dei token testuali mascherati usando contesto linguistico, spaziale e visivo
Masked Image Modeling	Immagine, con contesto testo-layout	Predizione dei token visivi discreti associati a patch mascherate
Word-Patch Alignment	Testo, immagine	Predizione dell'allineamento fine-grained tra parole testuali e patch visive corrispondenti

Tabella 4.2: Obiettivi di pre-training di LayoutLMv3.

La formulazione simmetrica dei tre obiettivi rende il pre-training di LayoutLMv3 più uniforme rispetto ai modelli precedenti: contenuto linguistico, disposizione spaziale e informazione visiva interagiscono già in questa fase, anziché essere appresi separatamente. In questo lavoro il pre-training non viene rieseguito. Si parte dal checkpoint `microsoft/layoutlmv3-base`, le cui rappresentazioni multimodali costituiscono il punto di partenza del fine-tuning supervisionato descritto nelle sezioni successive.

4.1.6 Testa di classificazione per il task BIO

Per adattare LayoutLMv3 al task di Key Information Extraction, il modello pre-addestrato viene esteso con una testa di classificazione token-level applicata direttamente alle rappresentazioni contestualizzate generate dal Transformer encoder. Questo modulo assegna a ciascun token una distribuzione di probabilità sulle classi definite dallo schema BIO.

Per i task di form e receipt understanding, il paper formula il problema come sequence labeling, predicendo l'etichetta associata all'ultimo hidden state di ciascun token testuale. Nell'implementazione della libreria Hugging Face impiegata in questo lavoro, la classe `LayoutLMv3ForTokenClassification` estende il backbone

`LayoutLMv3Model` aggiungendo una testa di classificazione lineare sopra le rappresentazioni finali.

Durante la fase di forward, il modello seleziona la sola porzione testuale dell'output multimodale, applica dropout e calcola i logits tramite il classificatore per assegnare a ciascun elemento una classe specifica dello schema BIO, come `B-DataDocumento`, `I-DataDocumento`, `B-CodiceFiscale`, o la classe `O` per il testo non rilevante. Sebbene le patch visive contribuiscano alla generazione delle rappresentazioni contestualizzate attraverso i meccanismi di attenzione, le etichette vengono predette ed estratte esclusivamente dai token testuali.

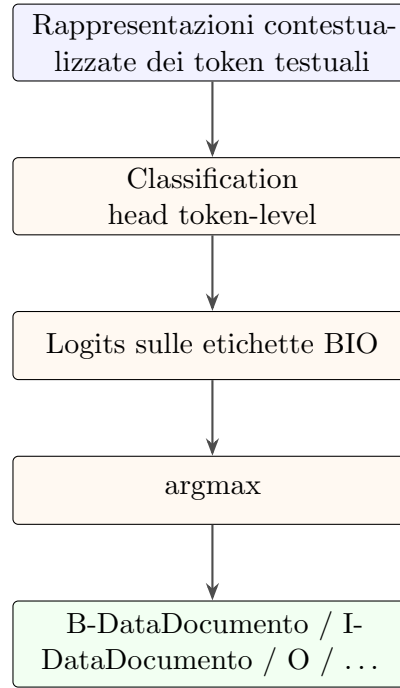


Figura 4.4: Testa di classificazione token-level di LayoutLMv3: dal token alle etichette BIO predette.

La testa proietta l'hidden state di ciascun token nello spazio delle etichette. In forma semplificata:

$$\mathbf{z}_i = \mathbf{W}\mathbf{h}_i + \mathbf{b}$$

$$\hat{\mathbf{y}}_i = \text{softmax}(\mathbf{z}_i)$$

dove $\mathbf{h}_i \in \mathbb{R}^{768}$ è l'hidden state dell' i -esimo token in output dall'encoder, $\mathbf{z}_i$ è il vettore dei logits prodotto dalla testa di classificazione e $\hat{\mathbf{y}}_i$ è la distribuzione di probabilità sulle classi BIO del progetto.

Durante la fase di fine-tuning, il modello apprende a mappare le rappresentazioni multimodali generate dall'encoder nelle corrispondenti etichette dello schema BIO.

4.1.7 Ruolo di LayoutLMv3 nella tesi

Nel quadro della tesi, LayoutLMv3 rappresenta il paradigma discriminativo OCR-based: produce predizioni token-level localizzabili, poi convertite nei campi strutturati richiesti dal task. Gli aspetti implementativi legati a dati, loss, iperparametri e valutazione sono trattati nelle sezioni successive.

4.2 Qwen2.5-VL: architettura nel dettaglio e formulazione generativa del KIE

4.2.1 Panoramica e posizionamento

Qwen2.5-VL [BCL⁺25] appartiene alla famiglia dei Large Vision-Language Models e integra comprensione visiva e generazione testuale all'interno di un'unica pipeline multimodale. Nel contesto dell'analisi comparativa sperimentale condotta in questa tesi, tale modello incarna il paradigma generativo, contrapponendosi direttamente all'approccio di LayoutLMv3. Qwen2.5-VL affronta il task di Key Information Extraction secondo una prospettiva puramente autoregressiva.

Il modello riceve in input l'immagine del documento insieme a un'istruzione testuale e genera in output una risposta strutturata, sotto forma di oggetto JSON contenente i campi informativi richiesti. Il task viene formulato come un compito di tipo image-to-structure, escludendo la necessità di pipeline esplicite di etichettatura BIO o di classificazione discreta token-level.

Il sistema non opera una selezione puntuale dei token associati a una specifica entità, bensì apprende una mappatura diretta tra il documento visivo, l'istruzione in linguaggio naturale e la struttura semantica finale.

La variante utilizzata per la sperimentazione è **Qwen2.5-VL-7B-Instruct**, composta da circa 7 miliardi di parametri. Tale configurazione è stata scelta come compromesso tra capacità visivo-linguistica e sostenibilità computazionale rispetto alle risorse hardware disponibili. La famiglia comprende anche modelli di dimensioni maggiori, come la variante da 72B, destinati a scenari con requisiti computazionali più elevati. Qwen2.5-VL è progettato per compiti di comprensione visiva, localizzazione, parsing documentale, estrazione strutturata di dati da fatture, moduli e tabelle, e generazione di risposte testuali condizionate da input multimodali. Questa caratteristica architetturale lo rende adatto al confronto con LayoutLMv3 nel dominio della Document AI: entrambi i modelli sono applicabili alla risoluzione del problema della Key Information Extraction, formalizzando il meccanismo di inferenza secondo principi architetturali radicalmente divergenti.

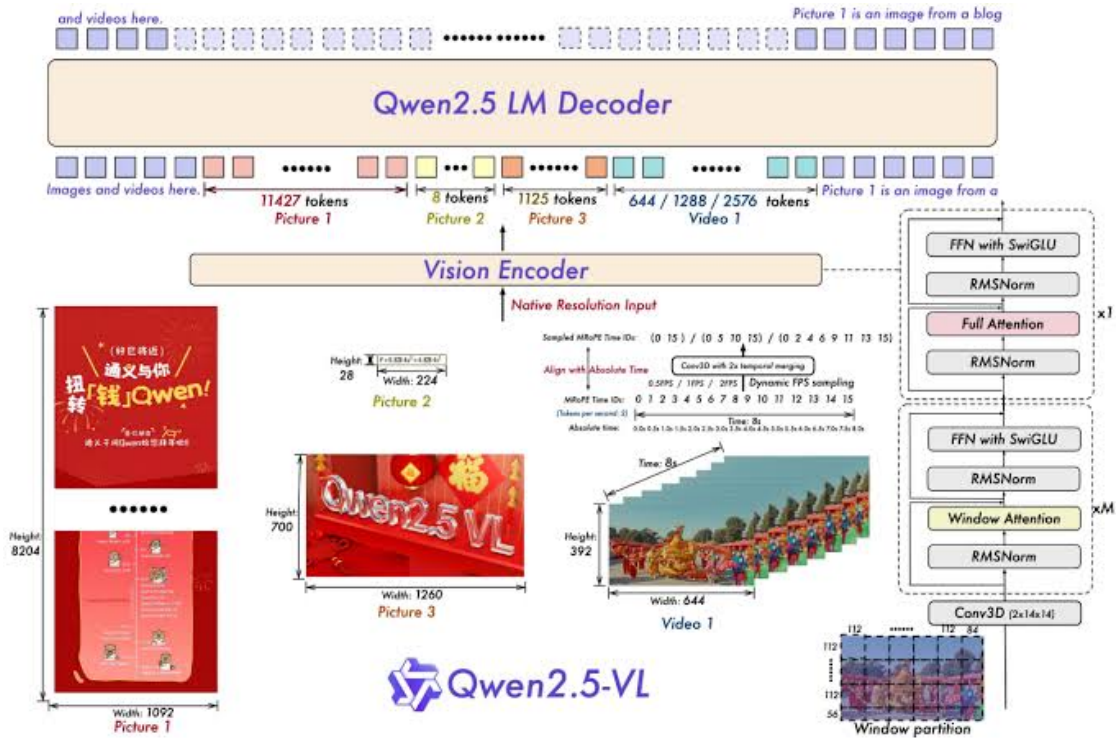


Figura 4.5: Architettura di Qwen2.5-VL. Tratto da [BCL⁺25].

4.2.2 Architettura generale

Qwen2.5-VL si compone di tre moduli principali: un Vision Encoder, un MLP-based Vision-Language Merger e un Large Language Model Decoder. Questa scomposizione è preferibile alla descrizione del modello come semplice architettura encoder-decoder, perché distingue con chiarezza il ruolo del modulo visivo, del modulo di fusione multimodale e del decoder linguistico autoregressivo. I tre moduli definiscono una pipeline in cui l'immagine del documento viene dapprima convertita in rappresentazioni visive, poi compressa e allineata allo spazio del linguaggio, infine interpretata dal decoder per produrre l'output richiesto.

Vision Encoder

Il primo componente analizza l'immagine. Qwen2.5-VL utilizza un Vision Transformer riprogettato, addestrato per gestire immagini di dimensioni variabili e documenti ad alta densità informativa. Una caratteristica distintiva è il supporto alla native dynamic resolution: anziché ricondurre ogni immagine a una risoluzione fissa, come nei Vision Transformer tradizionali, il modello mappa immagini di dimensioni variabili in sequenze di token visivi di lunghezza proporzionata alla dimensione e alla complessità dell'input.

Nel contesto documentale questo accorgimento aiuta a preservare la leggibilità di regioni dense e testo di piccole dimensioni. Nel caso dei documenti fiscali analizzati in questa tesi la proprietà è rilevante, perché scontrini, fatture e ricevute presentano

spesso testo minuto, righe ravvicinate, importi, date e codici collocati in aree visivamente complesse.

Per trattare in modo efficiente immagini ad alta risoluzione, il Vision Encoder ricorre prevalentemente ad attenzione locale a finestre, riservando l'attenzione globale a pochi blocchi. Poiché un'attenzione globale su tutte le patch avrebbe complessità quadratica rispetto al numero di token visivi, la suddivisione in finestre contiene il costo computazionale quando il numero di patch cresce con la risoluzione. Il componente produce infine una sequenza di visual token embeddings che codificano struttura spaziale, layout, elementi grafici e regioni testuali del documento.

Vision-Language Merger

Il secondo componente, indicato nel technical report come modulo MLP-based, trasforma le feature visuali prodotte dal Vision Encoder in rappresentazioni compatibili con lo spazio semantico del modello linguistico. Anziché trasmettere al decoder tutte le feature delle patch, Qwen2.5-VL raggruppa blocchi spazialmente adiacenti di 4 patch, ne concatena le feature e le proietta tramite una MLP a 2 layer. Tale passaggio assolve a una duplice funzione: riduce il numero di token visivi che il language model deve elaborare, contenendo l'onere computazionale, e allinea le rappresentazioni visuali allo spazio del linguaggio, abilitando la successiva generazione autoregressiva.

Nella pipeline adottata nel presente lavoro di tesi, questo passaggio consente di usare direttamente l'immagine e il prompt, senza la necessità di fornire al modello altri dati esterni come input.

Large Language Model Decoder

Il terzo componente è inizializzato a partire dai pesi del modello linguistico Qwen2.5. Si tratta di un Transformer autoregressivo decoder-only, basato su un meccanismo di attenzione causale, che riceve in input il prompt testuale e le rappresentazioni visive prodotte dal Vision-Language Merger, generando la risposta token dopo token. Dal punto di vista concettuale, il processo di generazione può essere descritto tramite la convenzionale fattorizzazione autoregressiva. Formalmente, nel caso di un output testuale $Y = (y_1, \dots, y_T)$, la generazione si descrive come:

$$P(y_1, \dots, y_T \mid V, X) = \prod_{t=1}^T P(y_t \mid y_{<t}, V, X)$$

dove V è la sequenza di token visivi dopo la proiezione multimodale, X è il prompt testuale e y_t è il token generato al passo t . L'output è una stringa di testo generata liberamente; tramite prompt, nel caso della tesi, tale stringa viene richiesta in forma di oggetto JSON strutturato.

Il technical report descrive inoltre una variante multimodale del Rotary Position Embedding (Multimodal RoPE aligned to absolute time), pensata soprattutto per gli input video e non centrale per i documenti statici trattati in questa tesi.

4.2.3 Pipeline multimodale per documenti

Nel dominio documentale, questa architettura consente di trattare la pagina come input visivo globale e di generare direttamente la risposta richiesta dal prompt. La Key Information Extraction viene quindi interpretata come generazione condizionata dall'immagine e dall'istruzione, anziché come assegnazione di etichette a parole già segmentate.

Formulazione generativa del KIE

Con Qwen2.5-VL il task di Key Information Extraction assume forma generativa. L'input è costituito dall'immagine del documento e da un prompt in linguaggio naturale che specifica il compito; in scenari di prompting few-shot, il prompt può includere anche uno o più esempi dimostrativi. L'output consiste nella struttura testuale finale richiesta dal task, costruita secondo i vincoli definiti dal prompt. Per il fine-tuning generativo, la supervisione non assume quindi la forma di etichette BIO token-level, ma di coppie immagine-istruzione e risposta strutturata. Il prompt engineering diventa così una componente operativa della pipeline, poiché definisce i vincoli formali e semantici che guidano la generazione dell'output.

4.2.4 Ruolo di Qwen2.5-VL nella tesi

Nel quadro comparativo della tesi, Qwen2.5-VL rappresenta il paradigma generativo: riduce la dipendenza da input OCR strutturati e consente di definire il task tramite prompt, ma introduce criticità legate al costo computazionale, al controllo del formato dell'output e alla possibile produzione di valori non supportati dal documento. Gli aspetti implementativi legati a dati, prompt, adattamento parameter-efficient e controllo della generazione sono trattati nelle sezioni successive.

4.3 Fine-tuning di LayoutLMv3

4.3.1 Obiettivo, modello e dati

Obiettivo del fine-tuning

Dopo la fase di preprocessing, LayoutLMv3 viene adattato al task di Key Information Extraction tramite fine-tuning supervisionato. L'obiettivo è specializzare il checkpoint pre-addestrato `microsoft/layoutlmv3-base` sul dominio specifico dei documenti fiscali, formulando il problema come token classification in schema BIO.

Nel paradigma di pre-training e fine-tuning, il modello parte da rappresentazioni multimodali già apprese su ampie collezioni documentali. Il backbone di LayoutLMv3 possiede quindi una conoscenza generale relativa a testo, layout e caratteristiche visive; tuttavia, esso non risulta ancora specializzato nel riconoscimento dei campi fiscali definiti nel dataset di questo lavoro di tesi. Il fine-tuning consente di adattare tali rappresentazioni generali a un insieme specifico di etichette, associando a ciascun token valido una classe dello schema BIO.

Caricamento del modello e ruolo della testa di classificazione

Il fine-tuning utilizza la classe `LayoutLMv3ForTokenClassification`, configurata sul numero di etichette BIO definite nel progetto.

Poiché `microsoft/layoutlmv3-base` è un modello pre-addestrato generale, non contiene una testa di classificazione specificamente configurata per le classi del dataset TaxWiz. Durante il caricamento, il backbone multimodale viene quindi inizializzato con pesi pre-addestrati, mentre la testa `classifier` viene adattata al numero di etichette del progetto e inizializzata per il nuovo task.

La distinzione assume un preciso rilievo metodologico. La valutazione del modello base prima del fine-tuning sullo schema BIO risulterebbe infatti priva di valore sperimentale: sebbene le rappresentazioni del backbone siano già informative, la testa di classificazione non ha ancora appreso alcuna associazione tra gli hidden states e le etichette dei documenti fiscali, determinando predizioni iniziali del tutto casuali.

L'insieme delle etichette non comprende tutte le possibili coppie teoriche B- e I- per ciascun campo, bensì riflette esclusivamente le etichette effettivamente presenti nel dataset annotato, oltre alla classe *O*. Alcuni campi, infatti, compaiono soltanto come entità a singolo token e non generano span multi-token. Di conseguenza, lo schema operativo del training comprende 17 classi BIO complessive.

Dataset tokenizzato e validazione degli input

Il training non viene eseguito sui dati grezzi, ma sul dataset già generato durante il preprocessing specifico per LayoutLMv3. Tale dataset contiene le colonne necessarie al modello: identificativi dei token, maschera di attenzione, bounding box normalizzate, valori visivi dell'immagine ed etichette BIO.

La separazione tra le fasi di preprocessing e di training offre un duplice vantaggio. Da un lato, rende la pipeline maggiormente ordinata e riproducibile, poiché il dataset tokenizzato può essere serializzato e riutilizzato senza dover ripetere la preparazione degli input multimodali. Dall'altro, consente di verificare esplicitamente la coerenza del formato dei dati rispetto alle specifiche del modello, prima di avviare sessioni di addestramento computazionalmente onerose.

A questo scopo la pipeline prevede un controllo tramite la funzione `validate_dataset`, che verifica la presenza degli split necessari e delle colonne richieste. Questa validazione è particolarmente importante nei task multimodali. Un singolo errore in una componente dell'input, come bounding box mancanti, immagini convertite in modo scorretto o etichette non allineate, può compromettere l'intero addestramento.

Il processor impiegato è `LayoutLMv3Processor`, configurato con l'argomento `apply_ocr=False`. Tale scelta deriva dal fatto che l'operazione di OCR non viene eseguita internamente dal processor, bensì effettuata a monte tramite il servizio Amazon Textract. Il processor riceve in input i token testuali, le bounding box e l'immagine già reciprocamente allineati, in perfetta coerenza con la formulazione OCR-based di LayoutLMv3 adottata nel presente lavoro di tesi.

Gestione delle etichette ignorate

Nei task di token classification, è necessario distinguere tra i token che devono contribuire alla supervisione e le posizioni che devono essere ignorate nel calcolo della loss e delle metriche di valutazione. Nel dataset impiegato per il fine-tuning, queste ultime posizioni sono contrassegnate esplicitamente con il valore -100 .

Il valore è una convenzione comune nelle pipeline basate su Hugging Face per segnalare posizioni da escludere dalla funzione di loss. Rientrano in questa categoria i token speciali, i token di padding e le posizioni che non devono essere considerate nella valutazione. Inoltre, quando una parola viene suddivisa in più subtoken dal tokenizer, lo stesso accorgimento evita che la parola originale venga valutata in modo scorretto più volte, indipendentemente dalla specifica strategia di allineamento adottata.

L'impiego del valore -100 mantiene quindi l'allineamento corretto tra le parole originali, i token elaborati dal modello e le etichette dello schema BIO. In assenza di tale accorgimento, la funzione di loss e le metriche di valutazione includerebbero posizioni strutturali o prive di rilevanza semantica, determinando una stima distorta delle prestazioni reali dell'architettura.

4.3.2 Validazione e gestione dello sbilanciamento

Metriche di validazione

Durante il fine-tuning vengono monitorate precision, recall, F_1 -score e accuracy tramite la libreria `segeval`, adatta alla valutazione di task di sequence labeling in schema BIO. A differenza di una semplice accuracy token-level, le metriche computate su `segeval` valutano le entità come unità strutturate, considerando la correttezza degli span predetti.

La differenza è sostanziale, poiché nei compiti di Key Information Extraction la predizione corretta dei singoli token isolati non è sufficiente; il modello deve riconoscere l'inizio, la continuazione e il confine dell'entità, e un errore su una parte dello span può compromettere l'intera estrazione del campo.

La funzione `make_compute_metrics` costruisce il wrapper usato dal Trainer per calcolare le metriche durante la validazione. Riceve i logits prodotti dal modello, seleziona la classe più probabile per ogni token tramite `argmax`, ricostruisce le sequenze di etichette testuali e filtra tutte le posizioni con etichetta -100 . Solo dopo il filtraggio le predizioni vengono confrontate con le etichette reali tramite `segeval`.

Le metriche di validazione svolgono tre funzioni: monitorare l'andamento del fine-tuning, guidare la ricerca iperparametrica e selezionare il miglior checkpoint. Non rappresentano però la valutazione finale della tesi, ma una misura di validazione interna alla pipeline di LayoutLMv3.

Sbilanciamento delle classi BIO

Una delle principali difficoltà del fine-tuning di LayoutLMv3 è lo sbilanciamento delle classi. Come emerso nell'analisi del dataset, la maggior parte dei token appartiene alla classe *O*, mentre le entità di interesse rappresentano una frazione ridotta del totale.

Lo squilibrio ha conseguenze dirette sul training. Con una cross-entropy standard priva di correttivi, il modello potrebbe ottenere buoni valori di accuracy predicendo correttamente molti token di background, fallendo però nel riconoscimento dei campi realmente importanti. L'accuracy token-level potrebbe quindi risultare elevata anche a fronte di prestazioni applicative insoddisfacenti.

Per mitigare tale effetto, la pipeline introduce le strategie complementari dei pesi di classe smussati e della Focal Loss pesata. L'obiettivo non è solo migliorare le metriche aggregate, ma aumentare la capacità del modello di riconoscere le entità minoritarie senza destabilizzare l'ottimizzazione.

Pesi di classe smussati

Il primo accorgimento calcola pesi di classe basati sulla frequenza delle etichette nel training set, assegnando un peso maggiore alle classi rare e minore a quelle frequenti. Un errore su una classe poco rappresentata contribuisce così alla loss più di un errore associato a una classe dominante.

Una pesatura inversamente proporzionale alla frequenza diventa però problematica quando alcune classi sono estremamente rare, poiché il peso associato a poche occorrenze può crescere molto, causando instabilità nei gradienti o sovra-adattamento a un numero limitato di esempi. Per questo viene adottata una variante smussata dell'inverse-frequency weighting.

Indicando con f_c la frequenza della classe c , il peso associato alla classe si esprime come:

$$w_c = \frac{1}{f_c^\alpha}$$

dove α è un fattore di smoothing compreso tra 0 e 1. Con $\alpha = 1$ si ottiene una pesatura pienamente inversa rispetto alla frequenza; con $\alpha = 0$ tutte le classi ricevono lo stesso peso. I valori intermedi aumentano l'importanza delle classi rare senza generare pesi estremi.

Nel progetto la logica è separata in due funzioni: `compute_label_frequencies`, che calcola la distribuzione delle etichette nel training set ignorando le posizioni con valore `-100`, e `compute_smoothed_class_weights`, che trasforma tali frequenze in pesi smussati. I pesi vengono poi normalizzati rispetto alla media dei pesi validi, al fine di mantenere controllata la scala complessiva della loss. Il fattore di smoothing non è trattato come costante assoluta, ma come iperparametro del training, in modo da valutare sperimentalmente quanto rendere aggressiva la compensazione dello sbilanciamento.

Focal Loss pesata

La seconda strategia è la Focal Loss [LGG⁺17], introdotta per ridurre il contributo degli esempi facili e concentrare l'ottimizzazione su quelli difficili. La proprietà risulta utile in un dataset dominato dalla classe O , perché i token di background diventano rapidamente esempi facili durante il training.

Nel caso multiclasse, dato un token la cui classe corretta è y , si considera la probabilità p_t assegnata dal modello alla classe corretta. La cross-entropy standard si scrive come:

$$CE = -\log(p_t)$$

La Focal Loss introduce un fattore modulante:

$$FL = (1 - p_t)^\gamma \cdot CE$$

dove γ controlla quanto la loss debba concentrarsi sugli esempi difficili. Quando p_t è alto, cioè quando il modello classifica correttamente e con elevata confidenza, il termine $(1 - p_t)^\gamma$ diventa piccolo e riduce il contributo di quell'esempio. Quando p_t è basso, cioè quando il modello è incerto o sbaglia, il contributo resta più elevato. Nel progetto viene utilizzata una versione pesata della Focal Loss:

$$FL = w_y (1 - p_t)^\gamma \cdot CE$$

dove w_y è il peso associato alla classe corretta, calcolato tramite la pesatura smussata descritta sopra. La loss tiene così conto contemporaneamente della difficoltà dell'esempio e della rarità della classe.

La formulazione affronta contemporaneamente la predominanza della classe O e la tendenza del modello a concentrarsi sugli esempi più facili, risolvendo così due problemi distinti ma strettamente collegati. La combinazione di class weighting e Focal Loss rende l'ottimizzazione più sensibile alle entità rare e agli errori più significativi sul piano applicativo.

4.3.3 Ottimizzazione e trainer

Layer-wise Learning Rate Decay

Nel fine-tuning di un'architettura Transformer pre-addestrata, risulta metodologicamente opportuno differenziare l'aggiornamento dei pesi tra i diversi livelli della rete. Gli strati più vicini all'input tendono infatti a codificare rappresentazioni generali e ampiamente trasferibili, mentre quelli più profondi, insieme alla testa di classificazione, sono maggiormente legati alla specificità del task finale.

L'adozione di una strategia di Layer-wise Learning Rate Decay, che assegna tassi di apprendimento ridotti ai livelli iniziali e valori progressivamente maggiori alle componenti più specializzate del modello, consente quindi di preservare la conoscenza generica appresa durante il pre-addestramento, concentrando l'adattamento sui moduli più rilevanti per il compito target.

Indicando con lr_{base} il learning rate massimo, con L il numero totale di layer encoder, con i l'indice del layer e con λ il fattore di decadimento, il learning rate del layer i si esprime come:

$$lr_i = lr_{\text{base}} \cdot \lambda^{(L-1-i)}$$

I layer più vicini all'output ricevono learning rate maggiori, quelli più vicini all'input aggiornamenti più conservativi. La testa di classificazione riceve il learning rate massimo, perché è inizializzata per il task specifico e deve apprendere da zero l'associazione tra rappresentazioni contestualizzate ed etichette BIO.

In linea con le buone pratiche del fine-tuning dei modelli Transformer, che sconsigliano l'applicazione del weight decay a bias, parametri di normalizzazione e layer normalization, la funzione `create_layerwise_optimizer` differenzia il learning rate per gruppi di parametri e isola le componenti esenti da regolarizzazione durante la costruzione dell'ottimizzatore.

Il Layer-wise Learning Rate Decay non corregge direttamente lo sbilanciamento delle classi, ma contribuisce alla stabilità del fine-tuning. Riduce il rischio di alterare eccessivamente il backbone pre-addestrato e permette alla testa di classificazione e agli ultimi layer di adattarsi più rapidamente al dominio fiscale.

Trainer personalizzato

La scelta di un **Trainer** personalizzato consente di raccogliere in un'unica struttura la logica sperimentale specifica del modello, includendo la Focal Loss, la gestione dei pesi di classe, l'ottimizzatore con Layer-wise Learning Rate Decay e la compatibilità con le API di Hugging Face.

A questo scopo è stato definito **FocalLossTrainer**, sviluppato estendendo la classe **Trainer**. Il metodo `compute_loss` viene ridefinito per calcolare la Focal Loss pesata sui soli token validi, escludendo le posizioni con etichetta `-100`. Il trainer gestisce inoltre i pesi di classe, ricalcolandoli in funzione del parametro di smoothing e mantenendoli coerenti con il dispositivo e il tipo numerico utilizzati durante il training. In questo modo, la pesatura delle classi non viene trattata come un passaggio esterno, ma diventa parte integrante della procedura di addestramento.

La costruzione dell'ottimizzatore viene invece ridefinita tramite il metodo `create_optimizer`, così da integrare direttamente il Layer-wise Learning Rate Decay. Questa scelta permette di assegnare learning rate differenziati ai diversi livelli del modello senza separare la strategia di ottimizzazione dalla logica del trainer.

Un ulteriore vantaggio riguarda la compatibilità con `hyperparameter_search`: gli iperparametri specifici della pipeline, come il valore di γ della Focal Loss, il fattore di smoothing dei pesi e il decadimento layer-wise del learning rate, possono così essere trattati come parte della configurazione esplorata durante la ricerca iperparametrica.

Ottimizzazione e gestione della memoria

Il fine-tuning multimodale di LayoutLMv3 è più oneroso di quello di un modello puramente testuale, perché elabora contemporaneamente testo, coordinate spaziali e immagine. La pipeline adotta quindi alcune scelte implementative volte a contenere il consumo di memoria GPU senza modificare la formulazione del task.

L'ottimizzatore è AdamW in versione a 8 bit, fornito dalla libreria `bitsandbytes`, che comprime gli stati interni dell'ottimizzatore, i quali nei Transformer occupano una memoria comparabile a quella dei parametri stessi. Il training usa mixed precision, selezionando `bfloat16` quando l'hardware lo supporta e `float16` altrimenti, e gradient accumulation, che accumula i gradienti su più mini-batch per ottenere un batch effettivo più ampio senza aumentare la memoria del singolo passo. La costruzione dei batch è affidata a `DataCollatorForTokenClassification`, che applica padding dinamico a livello di batch ed evita di estendere ogni sequenza alla lunghezza massima del modello.

Gli iperparametri specifici della pipeline sono raccolti in una classe dedicata, `LayoutTrainingArguments`, che estende `TrainingArguments` di Hugging Face con `focal_gamma`, `smooth_factor` e `llrd_decay_factor`. Trattarli come parte della configurazione li rende compatibili con la riproducibilità degli esperimenti e con la ricerca iperparametrica.

4.3.4 Ricerca iperparametrica e modello finale

Ricerca iperparametrica con Optuna

La scelta degli iperparametri ha un impatto rilevante sul fine-tuning, soprattutto in presenza di class imbalance, loss personalizzata e ottimizzazione layer-wise. Per questo viene utilizzata una ricerca iperparametrica tramite Optuna, integrata con il metodo `hyperparameter_search` del `Trainer`.

La ricerca permette di esplorare in modo sistematico diverse configurazioni del training. La metrica obiettivo è la F_1 sul validation set, più informativa dell'accuracy in presenza di classi sbilanciate. Lo spazio di ricerca comprende sia iperparametri standard, come learning rate, weight decay, numero di epoche e scheduler, sia iperparametri specifici della pipeline, come `focal_gamma`, `smooth_factor` e `llrd_decay_factor`.

L'uso di Optuna [ASY⁺19] è coerente con l'obiettivo metodologico della tesi, volto ad automatizzare la scelta della configurazione finale ed escludere modifiche manuali o valutazioni isolate. I dettagli esecutivi della ricerca, il numero di trial, i valori testati e i risultati quantitativi saranno discussi nel capitolo successivo.

Addestramento finale del modello migliore

Dopo la ricerca iperparametrica, il modello finale viene addestrato con la configurazione migliore individuata sul validation set. La fase è gestita dalla funzione `train_final_model`, che ricostruisce il modello e avvia il training con gli iperparametri selezionati.

Questa scelta separa la fase di esplorazione da quella di addestramento finale: la ricerca iperparametrica individua una configurazione promettente, mentre il training finale produce il modello effettivamente utilizzato nelle successive fasi di predizione e valutazione. In questa seconda fase, il miglior checkpoint viene selezionato sulla base della F_1 di validazione.

I risultati quantitativi del modello finale non vengono riportati in questa sezione, perché appartengono alla parte sperimentale della tesi. L'obiettivo qui è descrivere la logica metodologica del fine-tuning; la valutazione numerica è presentata in seguito.

4.3.5 Considerazioni finali

Il fine-tuning di LayoutLMv3 è strutturato come una pipeline supervisionata e controllata, calibrata sulle caratteristiche del dataset e sui vincoli computazionali disponibili. La formulazione token-level favorisce l'interpretabilità e la localizzazione spaziale delle predizioni, ma introduce difficoltà specifiche, in particolare lo sbilanciamento verso la classe O .

Per affrontare tali criticità, la pipeline combina strategie orientate sia alla stabilità dell'ottimizzazione sia alla gestione delle classi rare: pesi di classe smussati, Focal Loss pesata e Layer-wise Learning Rate Decay. A queste si affiancano scelte implementative, come AdamW 8-bit, mixed precision, gradient accumulation e padding dinamico, finalizzate a rendere sostenibile il training senza modificare la formulazione del task.

Il risultato della procedura è un modello LayoutLMv3 specializzato per token classification su documenti fiscali. Questa impostazione rappresenta il paradigma discriminativo del confronto sperimentale e costituisce il riferimento rispetto al quale viene valutato l'approccio generativo basato su Qwen2.5-VL.

4.4 Fine-tuning di Qwen2.5-VL

4.4.1 Obiettivo e modello

Obiettivo del fine-tuning

Il fine-tuning di Qwen2.5-VL è impostato in modo strutturalmente diverso rispetto a quello di LayoutLMv3, poiché adatta il modello a una formulazione generativa multimodale del task.

La supervisione assume una configurazione input-output: a ciascun documento viene associata la risposta strutturata attesa.

Il modello deve quindi apprendere a interpretare il documento visivo, seguire l'istruzione e produrre una sequenza testuale formalmente valida e semanticamente coerente con il contenuto del documento.

Modello e variante utilizzata

La variante implementata nella pipeline è **Qwen/Qwen2.5-VL-7B-Instruct**, scelta per sfruttare una maggiore capacità visivo-linguistica rispetto alle varianti più compatte. La scelta comporta un incremento del costo computazionale, ma mette a disposizione un modello con maggiore capacità generativa, più idoneo a un task di estrazione strutturata da documenti complessi.

Il comportamento desiderato viene appreso mediante esempi supervisionati in cui la risposta attesa corrisponde direttamente al JSON finale. L'impostazione consente di specificare nel prompt sia l'insieme dei campi richiesti sia il formato dell'output.

Le criticità specifiche di questa formulazione, relative al controllo dell'output generato, vengono affrontate nelle sezioni dedicate al prompt, alla loss e alla post-elaborazione delle predizioni.

4.4.2 Preparazione dei dati per la generazione

Conversione delle annotazioni BIO in target JSON

Il dataset di partenza è lo stesso usato per il confronto tra i modelli; tuttavia, per Qwen2.5-VL, le annotazioni devono assumere una struttura differente rispetto a quella predisposta per LayoutLMv3.

Qwen2.5-VL richiede un target testuale da generare. Le annotazioni BIO vengono perciò convertite in un dizionario JSON field-level: gli span di entità identificati dallo schema BIO vengono ricostruiti e associati al campo finale corrispondente.

La funzione `extract_json_from_bio_dynamic` svolge concettualmente questa trasformazione: a partire dalle parole del documento, dalle etichette BIO e dall'elenco dei campi target, individua gli span annotati, concatena i token appartenenti alla stessa entità e produce un dizionario in cui ogni chiave rappresenta un campo da estrarre. I campi assenti nel documento vengono impostati a `null` nella rappresentazione JSON finale. La funzione `add_json_dict` applica la conversione agli esempi del dataset.

I campi target utilizzati nella pipeline sono:

Tabella 4.3: Campi target dello schema di estrazione.

Campo target	Significato generale
<code>anno</code>	Anno di riferimento
<code>importo</code>	Importo specifico o parziale
<code>totaledocumento</code>	Totale del documento
<code>pagamentotracciato</code>	Indicazione relativa al pagamento tracciato
<code>datadocumento</code>	Data del documento
<code>codicefiscale</code>	Codice fiscale rilevante
<code>nomecognome</code>	Nome e cognome del soggetto
<code>totaledetraibile</code>	Totale detraibile
<code>titoloedilizio</code>	Titolo o riferimento edilizio

La trasformazione mantiene una base annotativa comune tra i due approcci. Le stesse annotazioni originarie vengono tuttavia usate in modo diverso. LayoutLMv3 le impiega come supervisione token-level, mentre Qwen2.5-VL le utilizza per costruire il JSON target da generare.

Prompt di estrazione

La funzione `build_extraction_prompt` costruisce il prompt istruzionale utilizzato nella pipeline. Sul piano concettuale, il prompt definisce il compito di estrazione, specifica lo schema di output sotto forma di oggetto JSON e stabilisce le regole che la risposta deve rispettare. In particolare, il modello è vincolato a restituire esclusivamente JSON valido, usare `null` per i campi assenti, preservare il formato originale delle date e non generare valori non supportati dal documento.

L'inclusione esplicita dello schema JSON, con i campi attesi inizializzati a `null`, orienta il modello verso una struttura stabile e limita la libertà generativa.

Il prompt da solo, tuttavia, non garantisce sempre la correttezza formale dell'output; per questo motivo, le predizioni generate vengono successivamente sottoposte a parsing, pulizia e normalizzazione prima della valutazione finale.

Formattazione multimodale del dataset

Dopo la conversione delle annotazioni BIO in JSON, ciascun esempio viene mappato in una rappresentazione compatibile con il fine-tuning multimodale del modello. La funzione `format_dataset_to_vlm` organizza ogni record secondo una struttura conversazionale composta da prompt e completion.

Il prompt corrisponde al messaggio dell'utente e contiene due elementi: un placeholder per l'immagine e il testo istruzionale. La completion corrisponde alla risposta attesa dell'assistente e contiene il JSON target. La separazione riflette il comportamento desiderato in fase di inferenza, richiedendo al modello di generare esclusivamente la risposta strutturata a partire da un documento e dalla relativa richiesta di input.

4.4.3 Adattamento parameter-efficient

Quantizzazione a 4 bit e QLoRA

Il fine-tuning completo di un modello Vision-Language da 7 miliardi di parametri richiederebbe risorse computazionali molto elevate. Per rendere sostenibile l'adattamento, la pipeline adotta un approccio QLoRA, cioè la combinazione di quantizzazione del modello di base e Low-Rank Adaptation.

La quantizzazione riduce l'occupazione di memoria necessaria per rappresentare i pesi del modello. Nella configurazione il modello viene caricato a 4 bit con formato NF4 e double quantization. I parametri del modello pre-addestrato restano congelati, mentre durante il training vengono aggiornati soltanto gli adattatori LoRA integrati in moduli specifici dell'architettura.

LoRA [HSW⁺22], acronimo di Low-Rank Adaptation, è una tecnica di fine-tuning efficiente in termini di parametri. Invece di aggiornare direttamente l'intera matrice di pesi W del modello pre-addestrato, LoRA apprende una variazione ΔW rappresentata come prodotto di due matrici a basso rango:

$$W' = W + \Delta W, \quad \Delta W = BA$$

dove A e B sono matrici molto più piccole della matrice originale. Il rango r regola il bilanciamento tra capacità di adattamento e numero di parametri addestrabili; un rango più alto estende la capacità di rappresentazione ma aumenta il costo di training, un rango più basso riduce i parametri ma può limitare l'adattamento al dominio dei documenti fiscali.

QLoRA [DPHZ23] combina in questo modo due idee complementari: il modello base viene mantenuto in una rappresentazione quantizzata a bassa precisione, mentre l'adattamento è affidato a un numero ridotto di parametri addestrabili. Questo permette di specializzare un modello di grandi dimensioni senza un full fine-tuning.

Moduli adattati con LoRA

La configurazione LoRA è costruita tramite la funzione `build_lora_config`. Gli adattatori non vengono applicati all'intero modello, ma a un insieme selezionato di moduli particolarmente rilevanti per il comportamento generativo.

LoRA viene applicato alle proiezioni dell'attenzione:

Tabella 4.4: Moduli adattati con LoRA.

Gruppo	Modulo	Ruolo generale
Attenzione	<code>q_proj</code>	Proiezione delle query
	<code>k_proj</code>	Proiezione delle key
	<code>v_proj</code>	Proiezione dei value
	<code>o_proj</code>	Proiezione di output dell'attenzione
Feed-forward	<code>gate_proj</code>	Componente di gating del feed-forward network
	<code>up_proj</code>	Proiezione di espansione del feed-forward network
	<code>down_proj</code>	Proiezione di riduzione del feed-forward network

La configurazione estende l'adattamento non solo alle proiezioni di attenzione, ma anche ad alcuni moduli MLP del decoder linguistico. L'intervento consente di coinvolgere una parte più ampia del comportamento generativo del modello, mantenendo comunque un numero di parametri addestrabili molto inferiore rispetto all'aggiornamento completo.

Risoluzione visiva vincolata

La gestione della risoluzione visiva è un aspetto rilevante del fine-tuning di Qwen2.5-VL, poiché il numero di token visivi prodotti dal Vision Encoder cresce con la

risoluzione dell'immagine e incide direttamente sul consumo di memoria e sul costo di addestramento. Vincolare la risoluzione significa bilanciare due esigenze opposte: preservare il dettaglio necessario a leggere testo minuto, importi e codici, e contenere la lunghezza della sequenza visiva entro i limiti hardware disponibili.

Il controllo è possibile perché il numero di token visivi è legato all'area dell'immagine da una relazione diretta. Poiché ogni token visivo corrisponde a una regione di 28×28 pixel, vale

$$\text{pixel} = N_{\text{token visivi}} \cdot 28 \cdot 28.$$

Fissare un intervallo ammissibile per il numero di token visivi equivale quindi a fissare un intervallo in pixel, imposto in fase di preprocessing. I valori concreti adottati e la loro configurazione nel processor sono riportati nel Capitolo 5.

4.4.4 Procedura di addestramento

Supervised Fine-Tuning con SFTTrainer

Il fine-tuning è strutturato come Supervised Fine-Tuning generativo, addestrando il modello su esempi input-output coerenti con la formulazione multimodale del task. La pipeline utilizza `SFTTrainer` e `SFTConfig`, strumenti pensati per l'addestramento supervisionato di modelli generativi. Il loro ruolo è gestire l'ottimizzazione del modello multimodale adattato tramite QLoRA, mantenendo la struttura prompt-completion degli esempi.

Un parametro particolarmente rilevante è `completion_only_loss`. Tramite questa impostazione la loss viene calcolata soltanto sulla completion, cioè sulla risposta JSON attesa, e non sul prompt dell'utente. La scelta è coerente con l'obiettivo del task, poiché concentra l'apprendimento sulla risposta attesa anziché sulla riproduzione dell'istruzione di input.

Strategie per la gestione della memoria

Il fine-tuning di Qwen2.5-VL condivide con LayoutLMv3 alcune tecniche di contenimento della memoria già descritte in Sezione 4.3.3: la mixed precision, con selezione di `bfloat16` o `float16` in base all'hardware, e la gradient accumulation, che accumula i gradienti su più mini-batch per ottenere un batch effettivo più ampio senza aumentare la memoria del singolo passo.

A queste si aggiungono accorgimenti resi necessari dalla scala del modello generativo da 7 miliardi di parametri. Il gradient checkpointing riduce la memoria del backward pass evitando di conservare tutte le attivazioni intermedie, che vengono in parte ricalcolate durante la backpropagation, a fronte di un maggior tempo di calcolo per passo. L'ottimizzatore è `paged_adamw_8bit`, che riduce l'occupazione degli stati dell'ottimizzatore e sfrutta meccanismi pensati per contenere i problemi di memoria nel training di modelli di grandi dimensioni. Insieme a QLoRA e alla quantizzazione a 4 bit, queste strategie non intervengono sulla formulazione del

task, ma rendono concretamente praticabile il fine-tuning entro i vincoli hardware disponibili.

Grid search manuale

Per Qwen2.5-VL non viene usata una ricerca iperparametrica automatizzata con Optuna. La ragione principale è il costo computazionale più elevato di ogni trial: il modello è più grande, generativo e multimodale, e ogni configurazione richiede un addestramento sensibilmente più oneroso.

Viene quindi adottata una grid search manuale su un insieme ristretto di configurazioni. La griglia esplora combinazioni di learning rate, rango LoRA e parametro `lora_alpha`, tre elementi che incidono direttamente sulla capacità di adattamento del modello e sulla stabilità del training.

La funzione `run_trial` rappresenta concettualmente l'unità sperimentale della grid search: per ogni configurazione vengono costruiti modello, processor, configurazione LoRA e trainer, viene eseguito il fine-tuning e la configurazione viene valutata sul validation set. La selezione della configurazione migliore avviene in base alla validation loss, cioè alla capacità del modello di generare la sequenza target sul set di validazione.

I risultati quantitativi dei trial non vengono riportati qui, perché appartengono al capitolo sperimentale. Una griglia manuale ristretta è un compromesso tra esplorazione degli iperparametri e costo computazionale.

4.4.5 Considerazioni finali

Il fine-tuning di Qwen2.5-VL è stato configurato come un adattamento generativo multimodale tramite QLoRA. Il modello base, nella variante `Qwen/Qwen2.5-VL-7B-Instruct`, resta quantizzato e congelato, mentre l'aggiornamento è affidato ad adattatori LoRA applicati a moduli selezionati dell'architettura. Questa impostazione consente di specializzare un Vision-Language Model di grandi dimensioni sul task di Key Information Extraction senza ricorrere a un full fine-tuning.

La pipeline riformula le annotazioni del dataset comune in una rappresentazione compatibile con il paradigma generativo: dagli span BIO si ricava un target JSON, mentre il prompt definisce il compito e vincola la struttura dell'output. Il training viene quindi orientato alla generazione della risposta strutturata, con scelte implementative pensate per contenere il costo computazionale del modello e rendere sostenibile l'addestramento entro i vincoli hardware disponibili.

Il risultato metodologico è una pipeline image-to-JSON, in cui il documento fiscale viene trasformato direttamente in una struttura contenente i campi di interesse. La complessità principale di questo adattamento non riguarda solo il costo del modello, ma anche la stabilità della generazione, la correttezza del formato JSON e il controllo delle hallucination.

4.5 Differenze operative tra LayoutLMv3 e Qwen2.5-VL

Il confronto tra LayoutLMv3 e Qwen2.5-VL non riguarda soltanto le prestazioni finali, bensì le modalità con cui i due modelli vengono preparati, addestrati e integrati in una pipeline di Key Information Extraction.

I due approcci riflettono i paradigmi descritti nelle sezioni precedenti: discriminativo OCR-based per LayoutLMv3 e generativo image-to-JSON per Qwen2.5-VL. La differenza riguarda l'intero ciclo di vita del sistema: preprocessing, tipo di supervisione, funzione di training, gestione degli errori, costo computazionale e verificabilità dell'output. Il confronto sperimentale non si lascia quindi interpretare solo attraverso le metriche finali, ma richiede di considerare i vincoli pratici delle due pipeline.

4.5.1 Formulazione del task e supervisione

Con LayoutLMv3 la Key Information Extraction è formulata come token classification: il documento viene rappresentato tramite immagine, parole OCR e bounding box, e ogni token valido riceve un'etichetta BIO. La supervisione è quindi distribuita sull'intera sequenza di input, inclusi i numerosi token appartenenti alla classe *O*.

Qwen2.5-VL affronta invece il task come generazione strutturata. L'input è composto dall'immagine del documento e da un prompt istruzionale, mentre l'output atteso è un JSON con i campi richiesti. La supervisione non riguarda le parole dell'input, ma la sequenza generata, ottimizzata tramite `completion_only_loss`.

Lo sbilanciamento non scompare, ma cambia forma: nel paradigma generativo non riguarda più la predominanza della classe *O*, bensì la diversa frequenza e difficoltà dei campi da generare nel JSON finale.

4.5.2 Input, OCR e rappresentazione del documento

La differenza tra i due modelli emerge anche nel modo in cui il documento viene rappresentato prima dell'inferenza. LayoutLMv3 richiede una rappresentazione esplicita del testo e del layout: le parole e le bounding box prodotte da Amazon Textract diventano parte integrante dell'input, insieme all'immagine della pagina. Questa impostazione rende la pipeline controllabile e ispezionabile, ma la espone agli errori generati a monte dall'OCR.

Qwen2.5-VL, nella pipeline adottata, riceve invece direttamente l'immagine del documento e demanda al modello l'interpretazione visiva del contenuto testuale e della struttura della pagina. L'input risulta quindi meno vincolato dal preprocessing OCR, ma il controllo si sposta sulla generazione dell'output e sulla successiva validazione della risposta prodotta.

Nel primo caso il layout è rappresentato in modo esplicito tramite coordinate; nel secondo è appreso implicitamente attraverso la comprensione visiva del documento.

4.5.3 Output, verificabilità ed errori

LayoutLMv3 produce una sequenza di etichette BIO. Ogni valore estratto deriva da uno o più token OCR classificati dal modello e può quindi essere ricondotto a una posizione precisa nella pagina, rendendo l'output localizzabile e verificabile.

Qwen2.5-VL produce invece direttamente una risposta strutturata generata. Questo formato è più vicino all'output applicativo desiderato, ma richiede controlli successivi sulla validità formale, sulla completezza dei campi e sulla coerenza dei valori estratti. Nel dominio fiscale questo aspetto è critico, perché una sola cifra errata in un importo, in una data o in un codice fiscale può compromettere l'affidabilità del sistema. Il rischio è inoltre asimmetrico rispetto a LayoutLMv3: un modello discriminativo OCR-based può classificare soltanto token già presenti nella sequenza riconosciuta dall'OCR, mentre un modello generativo può alterare cifre o introdurre valori non attestati nella pagina.

La diversa natura dell'output e degli errori rende quindi necessario ricondurre entrambe le predizioni a una rappresentazione comune campo-valore, su cui applicare lo stesso protocollo di valutazione.

4.5.4 Costo computazionale

LayoutLMv3 è più leggero e produce le predizioni tramite un singolo passaggio dell'encoder. Nella variante base utilizzata nella tesi, il modello presenta circa 133 milioni di parametri e risulta più sostenibile sia in training sia in inferenza. Richiede tuttavia una pipeline OCR e una fase di post-processing per convertire le etichette BIO in campi strutturati.

Qwen2.5-VL, nella variante da 7 miliardi di parametri utilizzata nella tesi, presenta un costo computazionale nettamente superiore. La generazione è autoregressiva, token dopo token, e il fine-tuning richiede tecniche specifiche per contenere l'occupazione di memoria: QLoRA, quantizzazione a 4 bit, gradient checkpointing e ottimizzatori efficienti. In compenso, garantisce una maggiore flessibilità nella formulazione del task, poiché lo schema di output può essere modificato tramite il prompt senza ridefinire una testa di classificazione. Tuttavia, dopo il fine-tuning su uno schema JSON specifico, è necessario verificare sperimentalmente se il modello mantiene la stessa stabilità anche quando lo schema richiesto viene modificato.

4.5.5 Quadro comparativo

La tabella 4.5 riassume le principali differenze operative tra i due approcci.

4.5. DIFFERENZE OPERATIVE TRA LAYOUTLMV3 E QWEN2.5-VL

Aspetto	LayoutLMv3	Qwen2.5-VL
Tipo di modello	Modello specializzato per Document AI	Vision-Language Model generalista
Paradigma	Discriminativo	Generativo
Architettura	Transformer multimodale encoder-only	Vision Encoder, Vision-Language Merger e Language Model Decoder
Dimensione indicativa	Circa 133M parametri nella variante base	7B parametri nella variante utilizzata
Formulazione del task	Token classification BIO	Generazione strutturata image-to-JSON
Tipo di supervisione	Token-level	Sequence-level / completion-level
Input principali	Immagine, parole OCR, bounding box	Immagine e prompt testuale
Dipendenza da OCR esterno	Alta	Non richiede OCR esterno esplicito
Layout awareness	Esplicita tramite bounding box 2D	Implicita tramite comprensione visiva
Output	Etichetta BIO per token	JSON generato
Gestione dello sbilanciamento	Centrale, soprattutto per la classe O	Non presente come sbilanciamento BIO; possibile squilibrio tra campi
Complessità del preprocessing	Elevata, per allineamento OCR-box-etichetta	Più semplice lato input, ma dipendente dal prompt e dal target JSON
Complessità del training	Loss personalizzata, trainer custom, LLRD	SFTTrainer, LoRA/QLoRA, quantizzazione
Verificabilità dell'output	Alta, perché ogni predizione è collegata a token OCR	Più indiretta, perché l'output è generato
Rischi principali	Errori OCR, truncation, predizione errata degli span	JSON non valido, hallucination, wrong value
Costo computazionale	Inferiore	Superiore
Flessibilità	Minore, richiede schema BIO e training specifico	Maggiore, grazie a prompt e generazione
Generalizzazione	Più dipendente dal dataset annotato	Potenzialmente più flessibile grazie al pre-training generalista
Deployment	Pipeline modulare OCR → BIO → campi	Pipeline image-to-JSON con controlli sul parsing

Tabella 4.5: Confronto operativo tra LayoutLMv3 e Qwen2.5-VL.

Le differenze operative mostrano come LayoutLMv3 e Qwen2.5-VL traducano in pipeline distinte due paradigmi della Key Information Extraction. Il primo mantiene una struttura token-level localizzabile, mentre il secondo produce direttamente una rappresentazione strutturata tramite generazione condizionata da immagine e prompt. Le implicazioni applicative di tali differenze, in termini di controllo, interpretabilità, costi e tipologie di errore, sono discusse nei capitoli dedicati ai risultati e alle conclusioni.

Capitolo 5

Setup sperimentale e addestramento dei modelli

Questo capitolo descrive l'ambiente sperimentale e le procedure con cui LayoutLMv3 e Qwen2.5-VL sono stati addestrati e selezionati.

Dopo aver definito il dataset e le procedure di preprocessing nei capitoli precedenti, occorre specificare il contesto hardware e software in cui gli esperimenti sono stati eseguiti. L'infrastruttura tecnologica vincola direttamente le scelte implementative, determinando sia i tempi di addestramento sia le dimensioni massime dei modelli adottati.

Il capitolo tratta successivamente la preparazione sperimentale dei dati che, a partire dallo stesso dataset, deriva due rappresentazioni distinte: una token-level per LayoutLMv3 e una generativa image-to-JSON per Qwen2.5-VL.

Vengono infine descritte le configurazioni di training dei due modelli sul piano implementativo, la ricerca degli iperparametri e la selezione dei modelli finali. Le predizioni dei modelli selezionati sono valutate nel capitolo successivo.

5.1 Ambiente software e hardware

Lo sviluppo del progetto è stato condotto principalmente in ambiente Python, utilizzando un ambiente virtuale gestito tramite Conda per controllare le dipendenze e ridurre il rischio di incompatibilità tra librerie, versioni di CUDA e framework di deep learning. Visual Studio Code è stato impiegato come ambiente di sviluppo, sia per la gestione dei notebook sia per gli script Python di supporto.

La versione di Python utilizzata è la 3.10.20. Le librerie principali sono raccolte nel file `requirements.txt`, per facilitare la ricostruzione dell'ambiente di lavoro; un secondo file, `requirements_completo.txt`, documenta la configurazione effettivamente installata durante gli esperimenti, con le versioni esatte dei pacchetti.

Tra le librerie più rilevanti rientrano PyTorch, Transformers, Datasets, Evaluate, segeval, scikit-learn, pandas, NumPy e Matplotlib. Il fine-tuning efficiente dei modelli multimodali si appoggia a PEFT, TRL e bitsandbytes, impiegate rispettivamente per LoRA/QLoRA, per il supervised fine-tuning di Qwen2.5-VL e per

l'uso di ottimizzatori e quantizzazione a ridotto consumo di memoria. Optuna gestisce la ricerca iperparametrica di LayoutLMv3.

La macchina locale è stata impiegata per lo sviluppo della pipeline, la preparazione e il controllo del formato dei dati, l'esecuzione di test su campioni ridotti, la correzione di errori di preprocessing o formattazione e la verifica del corretto funzionamento dei notebook. La configurazione hardware utilizzata è riportata di seguito.

Tabella 5.1: Specifiche hardware dell'ambiente di sviluppo.

Componente	Specifica
CPU	Intel Core i7-5820K @ 3.30 GHz
GPU	NVIDIA GeForce GTX 1660 Ti
RAM	16 GB

Le fasi computazionalmente più onerose, tra cui il fine-tuning, la grid search per la ricerca iperparametrica e la generazione dei checkpoint finali, sono state eseguite sul cluster GPU del DISI, poiché la memoria video disponibile localmente non era sufficiente a sostenere l'addestramento completo dei modelli multimodali considerati.

L'accesso è avvenuto tramite SSH, con gli esperimenti lanciati come job batch gestiti dallo scheduler SLURM. Il progetto è stato collocato nella directory `/scratch.hpc/`, adatta alla gestione di dataset, checkpoint, log e modelli salvati, dato che il training produce file di dimensioni non trascurabili, tra cui checkpoint intermedi, adattatori LoRA e risultati delle ricerche iperparametriche.

Gli esperimenti principali sono stati eseguiti sulle GPU del cluster riportate di seguito.

Tabella 5.2: Configurazione hardware utilizzata per il training sul cluster HPC.

Modello	Partizione	GPU	VRAM
LayoutLMv3	rtx2080	NVIDIA GeForce RTX 2080 Ti	11 GB
Qwen2.5-VL	140	NVIDIA L40	48 GB

Tale distinzione riflette il diverso costo computazionale dei modelli. LayoutLMv3, pur multimodale, è un modello documentale specializzato e più leggero, addestrato sulla partizione `rtx2080`; Qwen2.5-VL è un Vision-Language Model generativo molto più grande e richiede più memoria, motivo per cui è stato addestrato sulla partizione `140`.

L'esecuzione sul cluster non è avvenuta in modalità interattiva tramite notebook, ma tramite script Python autonomi, `train_layout.py` e `train_qwen.py`. I notebook sono stati usati per esplorazione, preprocessing, analisi e valutazione, mentre gli script hanno gestito le fasi computazionalmente più costose. La separazione ha reso la procedura sperimentale più ordinata e riproducibile.

Entrambi gli script prevedono una modalità di debug controllata dalla variabile d'ambiente `DEBUG_RUN`. Quando è attiva, il training viene eseguito su un numero molto ridotto di esempi e di step, così da verificare rapidamente il caricamento del dataset, la costruzione del modello, il forward pass, il backward pass e il salvataggio dei checkpoint prima di avviare gli esperimenti completi sul cluster.

5.2 Preparazione sperimentale dei dati

5.2.1 Dataset e split sperimentale

Gli esperimenti descritti in questo capitolo si basano sul dataset finale costruito nelle fasi precedenti della pipeline. Come descritto nel Capitolo 3, gli esperimenti utilizzano un dataset comune di documenti fiscali annotati, successivamente adattato alle diverse formulazioni del task richieste dai due modelli. Lo stesso insieme di dati alimenta quindi sia LayoutLMv3 sia Qwen2.5-VL, ma viene trasformato in rappresentazioni differenti a seconda della pipeline considerata.

Lo split sperimentale è stato eseguito prima del preprocessing specifico dei singoli modelli, per evitare che trasformazioni successive introducessero disallineamenti tra training, validation e test set. La suddivisione segue una logica basata su gruppi per ridurre il rischio di data leakage e rendere più affidabile la stima della capacità di generalizzazione. La costruzione del dataset e la distribuzione delle classi sono trattate nel Capitolo 3; questa sezione si concentra sulle trasformazioni che rendono i dati compatibili con LayoutLMv3 e Qwen2.5-VL.

Tabella 5.3: Suddivisione del dataset negli split di addestramento, validazione e test.

Split	Record	Utilizzo
Train	1.538	Addestramento dei modelli
Validation	196	Selezione checkpoint e iperparametri
Test	185	Valutazione finale comparativa
Totale	1.919	

L'unità di elaborazione è la singola pagina, ovvero il singolo record; anche quando più pagine appartengono allo stesso documento sorgente, i modelli operano su una pagina alla volta. Questa scelta mantiene il confronto coerente tra le due pipeline e le sue implicazioni vengono riprese nel preprocessing di Qwen2.5-VL.

Normalizzazione delle immagini

Prima del preprocessing specifico dei modelli, le immagini vengono convertite in formato RGB. Alcuni file possono infatti presentare codifiche diverse, ad esempio RGBA, mentre i processor richiedono input RGB. La conversione non altera il contenuto informativo del documento, ma garantisce una rappresentazione uniforme

e previene errori nella costruzione degli input multimodali durante il training in modalità batch sul cluster.

5.2.2 Preprocessing per LayoutLMv3

Per LayoutLMv3 il dataset sperimentale viene trasformato in una rappresentazione token-level. Il modello riceve in input l'immagine della pagina, le parole riconosciute dall'OCR, le relative bounding box e le etichette BIO associate ai token. Poiché testo OCR e coordinate spaziali sono già disponibili dalla fase precedente basata su Amazon Textract, il processor viene configurato in modo da non rieseguire l'OCR sull'immagine.

Le informazioni fornite al processor sono le seguenti.

Tabella 5.4: Campi di un record del dataset preprocessato.

Campo	Contenuto
<code>image</code>	Immagine della pagina
<code>words</code>	Parole estratte tramite OCR
<code>bboxes</code>	Bounding box normalizzate
<code>ner_tags</code>	Etichette BIO numeriche

Durante la tokenizzazione il processor riceve le parole OCR da `words` e le relative etichette BIO da `ner_tags`. Poiché il tokenizer può suddividere una parola in più sub-token, viene mantenuto l'allineamento tra parole originali, token prodotti ed etichette; le posizioni non valutabili, come token speciali e posizioni non allineate, sono marcate con il valore `-100` e ignorate nel calcolo della loss e delle metriche.

Listing 5.1: Tokenizzazione multimodale degli esempi tramite il processor di LayoutLMv3.

```
1 def prepare_examples(examples):
2     # images: list of PIL objects
3     # text: list[list[str]]
4     # boxes: list[list[[x1,y1,x2,y2]]]
5     # word_labels: list[int]
6     encoding = processor(
7         images=examples["image"],
8         text=examples["words"],
9         boxes=examples["bboxes"],
10        word_labels=examples["ner_tags"],
11        truncation=True,
12        padding=False,
13        max_length=512
14    )
15     return encoding
```

Il preprocessing usa `truncation=True` e `max_length=512`, in linea con il limite massimo di sequenza supportato da LayoutLMv3.

Al termine della tokenizzazione vengono conservate soltanto le colonne necessarie al training.

Tabella 5.5: Colonne del dataset tokenizzato fornite in input a LayoutLMv3.

Colonna	Ruolo
<code>pixel_values</code>	Rappresentazione visiva dell'immagine
<code>input_ids</code>	Token testuali
<code>attention_mask</code>	Maschera di attenzione
<code>bbox</code>	Coordinate spaziali dei token
<code>labels</code>	Etichette BIO allineate ai token

Analisi delle sequenze lunghe

Un aspetto specifico del preprocessing di LayoutLMv3 riguarda le sequenze che superano il limite massimo del modello. Poiché LayoutLMv3 supporta al massimo 512 token, i record oltre questa soglia vengono troncati. Per valutare l'impatto della scelta, prima della tokenizzazione finale è stata analizzata la distribuzione delle lunghezze tokenizzate, così da verificare quanti record superassero il limite consentito.

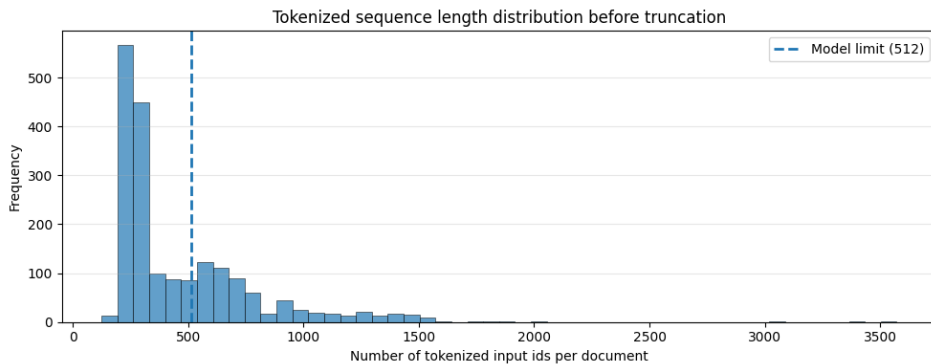


Figura 5.1: Distribuzione delle lunghezze tokenizzate dei documenti.

Sul dataset completo la lunghezza media tokenizzata è di 477,9 token, con mediana 310 e massimo 3.572. I record che superano i 512 token sono 643, pari al 33,51% del totale.

Il superamento dei 512 token non implica necessariamente la perdita di informazioni rilevanti per il task. Nei documenti fiscali, molti token appartengono alla classe *O*, cioè a contenuto di background non direttamente associato ai campi target. È stata perciò condotta anche un'analisi dell'impatto della truncation sui token informativi, quelli con etichetta diversa da *O*, sul training set.

La maggior parte dei record non perde entità informative dopo la truncation. Su 1.538 record, 1.368 non subiscono perdita di token rilevanti, mentre 170 perdono almeno un token associato a un'entità.

La truncation interessa quindi circa l'11%, ma il numero assoluto di token informativi persi non è trascurabile. La perdita resta concentrata in pochi documenti, dove però può sottrarre uno o più campi target alla ground truth disponibile per

LayoutLMv3, imponendo un limite superiore implicito al recall del modello su quei documenti. Per questa versione della pipeline si è scelto di mantenere la truncation standard a 512 token, accettandola come compromesso tra semplicità implementativa, compatibilità con il modello pre-addestrato e costo computazionale.

Sono state considerate anche strategie alternative, come l'uso di finestre sovrapposte con stride o la rimozione controllata di token della classe *O*. Una strategia a finestre avrebbe coperto l'intero documento, ma avrebbe aumentato il numero di esempi e reso più complessa la ricostruzione delle predizioni finali. La rimozione selettiva dei token della classe *O* avrebbe richiesto una logica più sofisticata per preservare le aree semanticamente rilevanti senza alterare troppo il contesto. Entrambe restano possibili sviluppi futuri, mentre negli esperimenti principali è stata mantenuta la procedura standard.

5.2.3 Preprocessing per Qwen2.5-VL

Nel caso di Qwen2.5-VL, la preparazione dei dati richiede la trasformazione delle annotazioni BIO in target field-level. Per mantenere il confronto coerente con LayoutLMv3, i target JSON di Qwen2.5-VL vengono ricostruiti a partire dalle stesse annotazioni BIO usate dal modello discriminativo. Gli span vengono individuati tramite `get_entities` di `sequeval`, aggregati per campo dalla funzione `extract_json_from_bio_dynamic` e convertiti in un dizionario in cui ogni chiave corrisponde a un'entità target. I campi non presenti nel documento vengono impostati a `None`. La funzione `add_json_dict` applica poi la conversione a ogni esempio tramite `map`, aggiungendo il campo `json_dict`.

Listing 5.2: Conversione delle annotazioni BIO in dizionario JSON field-level.

```

1 def extract_json_from_bio_dynamic(words, ner_tags, target_entities):
2     temp_result = {ent: [] for ent in target_entities}
3     for ent_type, start, end in get_entities(ner_tags):
4         ent_type = ent_type.lower()
5         if ent_type in target_entities:
6             extracted_str = " ".join(words[start : end + 1])
7             temp_result[ent_type].append(extracted_str)
8     return {k: (" ".join(v) if v else None) for k, v in temp_result.items()}
9
10
11 def add_json_dict(example):
12     example["json_dict"] = extract_json_from_bio_dynamic(
13         example["words"], example["labels"], TARGET_ENTITIES
14     )
15     return example

```

La scelta mantiene una ground truth comune tra i due modelli. LayoutLMv3 apprende dagli stessi dati in forma token-level, mentre Qwen2.5-VL apprende a generare la corrispondente rappresentazione strutturata. Il target JSON assume una forma simile alla seguente, dove i valori sensibili sono oscurati a scopo illustrativo.

Listing 5.3: Esempio di output JSON generato da Qwen2.5-VL.

```

1 {
2     "anno": null,
3     "importo": null,
4     "totaledocumento": null,

```

```

5     "pagamentotracciato": null,
6     "datadocumento": "01-01-2025 12:00",
7     "codicefiscale": "XXXXXX00X00X000X",
8     "nomecognome": null,
9     "totaledetraibile": "1,50",
10    "titoloedilizio": null
11 }

```

Nel dataset originario, l'etichetta pagamento tracciato non risultava associata a specifici blocchi OCR. Al fine di renderla utilizzabile nella formulazione BIO, essa è stata integrata in fase di generazione delle annotazioni tramite regole basate sulla presenza di termini coerenti con la modalità di pagamento. La ricostruzione del target JSON a partire dalle annotazioni BIO consente pertanto di mantenere allineati i target dei due modelli rispetto alla stessa rappresentazione finale.

Costruzione del prompt e formato prompt/completion

Ogni esempio per Qwen2.5-VL viene convertito in un formato supervisionato multimodale. Il prompt contiene l'immagine del documento e un'istruzione testuale che specifica i campi da estrarre e il formato dell'output; la completion contiene il JSON target atteso. Le regole testuali che vincolano la generazione sono definite nella funzione `build_extraction_prompt` e descritte nella Sezione 5.4.2.

Listing 5.4: Formattazione di un esempio nel formato conversazionale per il fine-tuning multimodale.

```

1  def format_dataset_to_vlm(example: dict) -> dict:
2      prompt = build_extraction_prompt()
3      target = json.dumps(example["json_dict"], ensure_ascii=False)
4
5      return {
6          "prompt": [
7              {
8                  "role": "user",
9                  "content": [
10                     {"type": "image"},
11                     {"type": "text", "text": prompt},
12                 ],
13             },
14         ],
15         "completion": [
16             {
17                 "role": "assistant",
18                 "content": [
19                     {"type": "text", "text": target},
20                 ],
21             },
22         ],
23     }

```

Qwen2.5-VL supporta input visivi multipli, caratteristica utile negli scenari reali in cui un documento può comporsi di più pagine. In questa tesi ogni esempio è costruito con una sola immagine alla volta, coerentemente con la granularità a singola pagina adottata anche per LayoutLMv3, così che entrambi i modelli vengano valutati sulla stessa unità sperimentale.

5.3 Configurazione del training di LayoutLMv3

Il training di LayoutLMv3 è implementato nello script `train_layout.py`, progettato per l'addestramento non interattivo sul cluster HPC. Lo script contiene l'intera procedura sperimentale, dal caricamento del dataset tokenizzato all'inizializzazione del modello, fino alla definizione delle metriche, alla costruzione del trainer personalizzato, alla ricerca iperparametrica e al salvataggio del modello finale.

Gli aspetti concettuali della pipeline, come la formulazione token-level, lo schema BIO, la Focal Loss pesata, i pesi di classe smussati e il Layer-wise Learning Rate Decay, sono stati discussi nel Capitolo 4. Qui gli stessi elementi vengono ripresi sul piano implementativo, mostrando come sono stati tradotti in codice e quali parametri sono stati effettivamente usati negli esperimenti.

5.3.1 Script di training e dataset tokenizzato

LayoutLMv3 viene addestrato a partire dal dataset già preprocessato e salvato su disco nella directory `data/tokenized_dataset`.

Prima di avviare il training, la funzione `validate_dataset` verifica la presenza degli split necessari, in particolare train e validation, e di tutte le colonne richieste dalla formulazione multimodale descritta nel Capitolo 4.

Listing 5.5: Validazione degli split e delle colonne del dataset tokenizzato.

```

1 def validate_dataset(dataset) -> None:
2     required_splits = {"train", "validation"}
3     missing_splits = required_splits - set(dataset.keys())
4
5     if missing_splits:
6         raise ValueError(f"Missing dataset splits: {missing_splits}")
7
8     required_columns = {
9         "input_ids",
10        "attention_mask",
11        "bbox",
12        "pixel_values",
13        "labels",
14    }
15
16    for split_name in required_splits:
17        split_columns = set(dataset[split_name].column_names)
18        missing_columns = required_columns - split_columns
19
20        if missing_columns:
21            raise ValueError(
22                f"Split {split_name!r} is missing columns: {missing_columns}"
23            )

```

Per costruire i batch viene usato `DataCollatorForTokenClassification`, che applica il padding dinamico e assegna con `-100` le posizioni da ignorare.

5.3.2 Modello e spazio delle etichette

Il modello di base è `microsoft/layoutlmv3-base`, caricato tramite `LayoutLMv3ForTokenClassification`, a cui viene aggiunta una testa di classificazione

token-level configurata per il nostro target. Lo spazio delle etichette comprende le 17 classi BIO, costruite esplicitamente tramite le mappature `label2id` e `id2label`. Le mappe `label2id` e `id2label` vengono passate al costruttore del modello, così da mantenere un mapping stabile tra identificatori numerici ed etichette testuali, usato sia durante il training sia nella conversione delle predizioni in etichette BIO leggibili.

Il modello viene inizializzato con `ignore_mismatched_sizes=True`, parametro necessario perché il checkpoint pre-addestrato non contiene una testa di classificazione compatibile con il numero di classi del task. I pesi del backbone multimodale vengono caricati dal checkpoint pre-addestrato, mentre la testa di classificazione viene reinizializzata e appresa durante il fine-tuning sul dataset TaxWiz. Il report di caricamento segnala infatti che alcuni parametri della testa non provengono dal checkpoint, in particolare `classifier.dense.weight`, `classifier.out_proj.weight`, `classifier.dense.bias` e `classifier.out_proj.bias`.

Listing 5.6: Caricamento del modello pre-addestrato e configurazione della testa di classificazione.

```
1 def build_model():
2     return LayoutLMv3ForTokenClassification.from_pretrained(
3         MODEL_ID,
4         num_labels=len(id2label),
5         id2label=id2label,
6         label2id=label2id,
7         ignore_mismatched_sizes=True,
8     )
```

5.3.3 Metriche di validazione

Durante il training le prestazioni vengono valutate sul validation set tramite la libreria `segeval`, adatta ai task di sequence labeling. La funzione di metriche applica l'argmax sui logits per ottenere la classe più probabile di ciascun token e filtra tutte le posizioni con etichetta `-100`, escludendo padding, token speciali e posizioni non rilevanti.

Listing 5.7: Costruzione della funzione di calcolo delle metriche con `segeval` per la validazione.

```
1 def make_compute_metrics(metric, zero_division=0, return_entity_level_metrics=
2     False):
3     def compute_metrics(p):
4         predictions, labels = p
5         predictions = np.argmax(predictions, axis=2)
6
7         true_predictions = [
8             id2label[int(pred_id)]
9             for pred_id, label_id in zip(prediction, label)
10             if label_id != -100
11         ]
12         for prediction, label in zip(predictions, labels)
13     ]
14
15     true_labels = [
16         id2label[int(pred_id)]
17         for pred_id, label_id in zip(prediction, label)
18         if label_id != -100
19     ]
```

```

17         id2label[int(label_id)]
18         for pred_id, label_id in zip(prediction, label)
19             if label_id != -100
20     ]
21     for prediction, label in zip(predictions, labels)
22 ]
23
24 if not true_labels or not true_predictions:
25     return {
26         "precision": 0.0,
27         "recall": 0.0,
28         "f1": 0.0,
29         "accuracy": 0.0,
30     }
31
32 results = metric.compute(
33     predictions=true_predictions,
34     references=true_labels,
35     zero_division=zero_division,
36 )
37
38 if return_entity_level_metrics:
39     final_results = {}
40     for key, value in results.items():
41         if isinstance(value, dict):
42             for n, v in value.items():
43                 final_results[f"{key}_{n}"] = v
44         else:
45             final_results[key] = value
46     return final_results
47
48 return {
49     "precision": results["overall_precision"],
50     "recall": results["overall_recall"],
51     "f1": results["overall_f1"],
52     "accuracy": results["overall_accuracy"],
53 }
54
55 return compute_metrics

```

Le metriche calcolate sono precision, recall, F_1 e accuracy a livello di sequenza BIO. La F_1 sul validation set è il criterio di selezione del modello, per le ragioni legate allo sbilanciamento verso la classe *O* già discusse nel Capitolo 4. Per le classi rare prive di predizioni in alcune epoche, il calcolo usa `zero_division=0`.

5.3.4 Focal Loss pesata e pesi di classe

Per gestire lo sbilanciamento delle classi BIO, lo script usa una sottoclasse personalizzata di `Trainer`, `FocalLossTrainer`, che ridefinisce il metodo `compute_loss`. La loss standard viene sostituita dalla Focal Loss pesata, la cui formulazione è descritta nel Capitolo 4.

Operativamente, logits ed etichette vengono appiattiti e filtrati tramite una maschera booleana, così da escludere padding, token speciali e posizioni non valutabili. Il parametro `focal_gamma` controlla l'intensità del termine focale, mentre i pesi di classe modulano il contributo in funzione della rarità dell'etichetta corretta; la loss del batch è la media sui token validi. Il calcolo corrisponde al codice seguente.

Listing 5.8: Estratto del metodo `compute_loss`: calcolo della Focal Loss pesata sui token validi.

```

1 ce_loss = F.cross_entropy(logits, labels, reduction="none")
2
3 pt = torch.exp(-ce_loss)
4 gamma = float(getattr(self.args, "focal_gamma", 2.0))
5
6 focal_loss = ((1.0 - pt) ** gamma) * ce_loss
7
8 class_weights = self._get_class_weights(
9     device=labels.device,
10    dtype=focal_loss.dtype,
11 )
12
13 focal_loss = class_weights[labels] * focal_loss
14 loss = focal_loss.mean()

```

I pesi di classe vengono calcolati dalle frequenze relative delle etichette nel training set.

La funzione `compute_label_frequencies` calcola tali frequenze ignorando le posizioni con etichetta `-100`, mentre `compute_smoothed_class_weights` costruisce un vettore di pesi smussati, normalizzati rispetto alla loro media per non alterare troppo la scala complessiva della loss. Il parametro `smooth_factor` regola l'intensità della correzione; valori più bassi producono pesi più conservativi, valori più alti aumentano il peso relativo delle classi rare. L'implementazione corrisponde a quanto segue.

Listing 5.9: Calcolo delle frequenze delle etichette e dei pesi di classe smussati.

```

1 def compute_label_frequencies(train_dataset, num_labels, ignore_index=-100):
2     labels = train_dataset["labels"]
3     flat = [l for seq in labels for l in seq if l != ignore_index]
4
5     counts = torch.bincount(torch.tensor(flat), minlength=num_labels).float()
6     return counts / counts.sum()
7
8
9 def compute_smoothed_class_weights(frequencies, smooth_factor, device, dtype):
10    frequencies = frequencies.to(device=device, dtype=dtype)
11
12    weights = torch.zeros_like(frequencies)
13    valid = frequencies > 0
14
15    weights[valid] = 1.0 / (frequencies[valid] ** smooth_factor)
16    weights[valid] = weights[valid] / weights[valid].mean()
17    return weights

```

Le frequenze vengono calcolate una sola volta all'inizio del training. I pesi di classe vengono ricalcolati solo quando cambia la combinazione di `smooth_factor`, dispositivo e tipo numerico; a questo scopo `FocalLossTrainer` mantiene una cache interna dei pesi, evitando ricalcoli inutili durante l'addestramento.

5.3.5 Ottimizzatore e Layer-wise Learning Rate Decay

Il secondo elemento personalizzato di `FocalLossTrainer` riguarda la costruzione dell'ottimizzatore. Il metodo `create_optimizer` viene ridefinito per applicare il Layer-wise Learning Rate Decay, descritto nel Capitolo 4 e implementato tramite

`create_layerwise_optimizer`.

La funzione assegna learning rate diversi ai parametri in base alla loro posizione nell'architettura, controllati dal parametro `llrd_decay_factor`. La testa di classificazione usa il learning rate di base, i layer inferiori dell'encoder ricevono valori progressivamente più piccoli e i parametri che non appartengono né alla testa né ai layer dell'encoder, come gli embedding, vengono aggiornati in modo più conservativo. L'assegnazione ai tre gruppi corrisponde al codice seguente.

Listing 5.10: Costruzione dell'ottimizzatore con Layer-wise Learning Rate Decay.

```

1 def create_layerwise_optimizer(model, base_lr, weight_decay, decay_factor):
2     no_decay = ["bias", "LayerNorm.weight"]
3     num_layers = model.config.num_hidden_layers
4     groups = {}
5
6     for name, param in model.named_parameters():
7         if not param.requires_grad:
8             continue
9
10        if "classifier" in name:
11            lr = base_lr
12        elif "encoder.layer." in name:
13            layer_idx = int(name.split(".")[name.split(".").index("layer") + 1])
14            lr = base_lr * (decay_factor ** (num_layers - 1 - layer_idx))
15        else:
16            lr = base_lr * (decay_factor ** num_layers)
17
18        use_decay = not any(nd in name for nd in no_decay)
19        key = (lr, use_decay)
20
21        groups.setdefault(key, {
22            "params": [],
23            "lr": lr,
24            "weight_decay": weight_decay if use_decay else 0.0,
25        })
26        groups[key]["params"].append(param)
27
28    return AdamW8bit(list(groups.values()), lr=base_lr, betas=(0.9, 0.999), eps=1e-8)

```

Nella costruzione dei gruppi, bias e pesi di LayerNorm vengono esclusi dal weight decay. I gruppi così ottenuti vengono passati ad `AdamW8bit`, fornito da `bitsandbytes`, con $\beta = (0, 9; 0, 999)$ ed $\epsilon = 10^{-8}$. L'ottimizzatore a 8 bit riduce il consumo di memoria degli stati dell'optimizer, mantenendo la configurazione compatibile con i vincoli della GPU usata negli esperimenti.

5.3.6 Estensione di `TrainingArguments` e configurazione di base

Poiché `focal_gamma`, `smooth_factor` e `llrd_decay_factor` non fanno parte dell'API standard di Hugging Face, è stata definita una dataclass personalizzata, `LayoutTrainingArguments`, che estende `TrainingArguments` introducendo questi tre campi.

I tre parametri controllano componenti non presenti nel Trainer standard.

Tabella 5.6: Iperparametri specifici della pipeline di training di LayoutLMv3.

Parametro	Ruolo
<code>focal_gamma</code>	Controlla l'intensità della Focal Loss
<code>smooth_factor</code>	Controlla il calcolo dei pesi di classe
<code>llrd_decay_factor</code>	Controlla il decadimento del learning rate tra i layer

L'estensione propaga tali valori attraverso `self.args` all'interno del trainer, mantenendo compatibilità con l'infrastruttura nativa di Hugging Face, incluso il meccanismo di `hyperparameter_search` usato nella Sezione 5.5.1.

La configurazione iniziale è definita nella funzione `build_base_training_args`, che imposta i principali parametri di addestramento, logging, valutazione e salvataggio dei checkpoint.

Il batch size per dispositivo pari a 1 è dettato dai vincoli di memoria GPU imposti dal processing multimodale di LayoutLMv3. Per compensarlo si usa la gradient accumulation con `gradient_accumulation_steps=16`, che accumula i gradienti su più mini-batch prima dell'aggiornamento dei pesi, ottenendo un batch effettivo pari a 16. La precisione numerica è selezionata dinamicamente, `bf16` se la GPU lo supporta, altrimenti `fp16`, come descritto nel Capitolo 4.

La valutazione viene eseguita alla fine di ogni epoca, con checkpoint salvati allo stesso intervallo e un limite di due checkpoint su disco. Al termine del training viene ricaricato il checkpoint con la migliore F_1 di validazione, tramite `load_best_model_at_end=True` e `metric_for_best_model="f1"`. I seed `seed=42` e `data_seed=42` sono fissati per la riproducibilità.

5.3.7 Costruzione del trainer e collegamento con la ricerca iperparametrica

Il trainer viene istanziato tramite la classe personalizzata `FocalLossTrainer`, a cui vengono passati il dataset di training, il validation set, il data collator, la funzione di metriche, il processor e le frequenze delle etichette calcolate sul training set.

L'uso di `model_init` permette di inizializzare un nuovo modello per ciascun trial della ricerca iperparametrica, evitando che i pesi di un'esecuzione influenzino le successive. Lo script integra infatti la selezione della configurazione in una ricerca tramite `trainer.hyperparameter_search`, i cui dettagli, spazio di ricerca e risultati sono discussi nella Sezione 5.5.1.

Listing 5.11: Avvio della ricerca iperparametrica con Optuna tramite l'API del Trainer.

```
1 best_run = trainer.hyperparameter_search(  
2     hp_space=optuna_hp_space,  
3     compute_objective=compute_objective,  
4     n_trials=N_TRIALS,  
5     direction="maximize",  
6     backend="optuna",  
7 )
```

Nel complesso, il training di LayoutLMv3 estende il **Trainer** standard con Focal Loss pesata, pesi di classe smussati, Layer-wise Learning Rate Decay e ottimizzatore AdamW a 8 bit, selezionando il modello sulla F_1 di validazione. L'implementazione riflette i vincoli del problema, dallo sbilanciamento verso la classe O ai limiti di memoria GPU.

5.4 Configurazione del training di Qwen2.5-VL

Il training di Qwen2.5-VL è implementato nello script `train_qwen.py`, lanciato tramite script `sbatch` sullo stesso server HPC usato per LayoutLMv3.

Questa sezione descrive le scelte operative dello script, dal caricamento del dataset alla conversione BIO-to-JSON, dalla costruzione del prompt alla quantizzazione, fino alla configurazione LoRA e agli argomenti di training. Il dettaglio delle configurazioni esplorate tramite grid search è rimandato alla Sezione 5.5.2.

5.4.1 Dataset, campi target e conversione BIO-to-JSON

Lo script carica il dataset già suddiviso in train, validation e test dalla `data/dataset_splitted`. I nove campi da estrarre sono definiti nello script tramite la lista `TARGET_ENTITIES`. La conversione delle annotazioni BIO in target JSON segue la procedura già descritta; in questa sezione viene richiamata solo come passaggio operativo dello script di training.

5.4.2 Costruzione del prompt e formato conversazionale

Il prompt usato nel training è quello già introdotto nella fase di preprocessing e viene generato dalla funzione `build_extraction_prompt`. In questa sezione se ne richiama il ruolo all'interno dello script di training, dove ogni esempio viene rappresentato come coppia prompt/completion.

Listing 5.12: Struttura di un esempio formattato per il fine-tuning di Qwen2.5-VL (prompt e completion).

```

1 dict_keys(['images', 'messages'])
2 <class 'PIL.PngImagePlugin.PngImageFile'>
3
4 # messaggio utente (prompt)
5 [{ 'type': 'image' },
6  { 'type': 'text', 'text': 'Analizza attentamente il documento e
7    restituisci esclusivamente un oggetto JSON valido con i seguenti campi:
8    { "anno": null, "importo": null, ... "titoloedilizio": null }
9    Regole: restituisci solo JSON valido; non aggiungere spiegazioni;
10   usa null se il valore non e\' presente; mantieni il formato originale
11   delle date; non inventare valori' }]
12
13 # messaggio assistente (completion / JSON target)
14 [{ 'type': 'text', 'text': '{"anno": null, "importo": null,
15   "totaledocumento": null, "pagamentotracciato": null,
16   "datadocumento": "XX-XX-XXXX XX:XX",
17   "codicefiscale": "XXXXXXXXXXXXXXXX",
18   "nomecognome": null, "totaledetraibile": "1,50",
19   "titoloedilizio": null}' }]

```

Le regole vincolano l'output di un modello generativo, che altrimenti potrebbe produrre testo aggiuntivo o campi non previsti. Il vincolo di non inventare valori è particolarmente rilevante nella Key Information Extraction, dove le predizioni possono introdurre dati assenti dal documento; la tassonomia degli errori della Sezione 6.2.1 distingue infatti tra correct extraction, missing, wrong value e hallucination.

Durante la formattazione lo script conserva la colonna `image` e rimuove le colonne non necessarie al training generativo.

5.4.3 Modello di base, processor e quantizzazione QLoRA

Il modello di base è `Qwen/Qwen2.5-VL-7B-Instruct`, caricato tramite `Qwen2_5_VLForConditionalGeneration`, con il processor caricato tramite `AutoProcessor`.

Listing 5.13: Configurazione della quantizzazione a 4 bit (QLoRA) e caricamento del modello.

```

1 bnb_config = BitsAndBytesConfig(
2     load_in_4bit=True,
3     bnb_4bit_quant_type="nf4",
4     bnb_4bit_use_double_quant=True,
5     bnb_4bit_compute_dtype=compute_dtype,
6 )
7
8 def build_model_and_processor():
9     model = Qwen2_5_VLForConditionalGeneration.from_pretrained(
10         model_id,
11         device_map="auto",
12         torch_dtype=compute_dtype,
13         quantization_config=bnb_config,
14     )
15     model.config.use_cache = False
16
17     processor = AutoProcessor.from_pretrained(
18         model_id,
19         min_pixels=MIN_PIXELS,
20         max_pixels=MAX_PIXELS,
21     )
22     model = prepare_model_for_kbit_training(model)
23     return model, processor

```

Il tipo di calcolo è selezionato dinamicamente: `torch.bfloat16` se la GPU lo supporta, altrimenti `torch.float16`. Il modello quantizzato viene poi preparato per il training a basso numero di bit tramite `prepare_model_for_kbit_training` della libreria `peft`.

La risoluzione visiva è vincolata tra 256 e 1.280 token, secondo il compromesso tra dettaglio e costo computazionale descritto nel Capitolo 4. I limiti in pixel si ricavano moltiplicando il numero di token visivi per 28×28 e vengono passati al processor tramite `min_pixels` e `max_pixels`.

L'impostazione `device_map="auto"` colloca automaticamente il modello sui dispositivi disponibili. Durante il training `use_cache` viene disattivato, poiché la cache di generazione non è necessaria in addestramento e aumenterebbe il consumo di memoria.

5.4.4 Configurazione LoRA

Il fine-tuning non aggiorna tutti i parametri del modello, ma usa adattatori LoRA configurati tramite `LoraConfig` della libreria `peft`, coerentemente con l'approccio QLoRA descritto nel Capitolo 4, in cui il modello base resta quantizzato e vengono addestrati soltanto gli adattatori.

Listing 5.14: Configurazione di LoRA con i moduli target dell'adattamento.

```
1 def build_lora_config(config: dict) -> LoraConfig:
2     return LoraConfig(
3         r=config["lora_r"],
4         lora_alpha=config["lora_alpha"],
5         lora_dropout=0.05,
6         bias="none",
7         task_type="CAUSAL_LM",
8         target_modules=[
9             "q_proj",
10            "k_proj",
11            "v_proj",
12            "o_proj",
13            "gate_proj",
14            "up_proj",
15            "down_proj",
16        ],
17     )
```

Alcuni valori restano fissi in tutte le configurazioni, `lora_dropout=0.05`, `bias="none"` e `task_type="CAUSAL_LM"`, mentre `lora_r` e `lora_alpha` variano tra i trial della grid search e vengono discussi nella Sezione 5.5.2. Gli adattatori sono applicati alle proiezioni di attenzione (`q_proj`, `k_proj`, `v_proj`, `o_proj`) e ad alcune componenti del blocco MLP (`gate_proj`, `up_proj`, `down_proj`), come descritto nel Capitolo 4, mantenendo un numero di parametri addestrabili molto inferiore rispetto al full fine-tuning.

5.4.5 Argomenti di training con SFTConfig

Il supervised fine-tuning è configurato tramite `SFTConfig`, usato da `SFTTrainer` della libreria `trl`. La configurazione definisce i parametri di training, valutazione, logging e salvataggio dei checkpoint.

La configurazione principale si riassume come segue.

5.4. CONFIGURAZIONE DEL TRAINING DI QWEN2.5-VL

Parametro	Valore	Ruolo
Modello	Qwen2.5-VL-7B-Instruct	Modello base multimodale generativo
Formulazione	Image-to-JSON	Generazione diretta dell'oggetto JSON
Fine-tuning	QLoRA	Modello quantizzato con adattatori LoRA
Quantizzazione	4-bit NF4 + double quant	Riduzione della memoria richiesta
Token visivi min	256	Risoluzione visiva minima
Token visivi max	1.280	Risoluzione visiva massima
Batch size per device	1	Memoria ridotta per step
Gradient accumulation	16	Batch effettivo pari a 16
Scheduler	Linear	Decadimento progressivo del learning rate
Warmup ratio	0,05	Stabilizzazione iniziale
Weight decay	0,0	Nessuna regolarizzazione L2 esplicita
Max grad norm	1,0	Gradient clipping
Optimizer	paged_adamw_8bit	Ottimizzatore efficiente in memoria
Gradient checkpointing	Attivo	Riduzione memoria delle attivazioni
use_reentrant	False	Configurazione del gradient checkpointing
Packing	Disattivo	Nessuna concatenazione di esempi
completion_only_loss	True	Loss solo sulla risposta JSON
EOS token	<code>< im_end ></code>	Token di fine sequenza
Precisione	bf16 / fp16	Mixed precision dinamica
Evaluation	A fine epoca	Valutazione sul validation set
Save strategy	A fine epoca	Salvataggio dei checkpoint
save_total_limit	2	Checkpoint massimi mantenuti
Best model	Minima eval_loss	Selezione del checkpoint migliore
Logging	TensorBoard	Monitoraggio del training
logging_steps	10	Frequenza del logging
Seed	42	Riproducibilità
remove_unused_columns	False	Conserva le colonne multimodali

Tabella 5.7: Configurazione completa del fine-tuning di Qwen2.5-VL.

Il batch size per device è pari a 1, per il costo computazionale del modello e la memoria richiesta dagli input multimodali, e viene compensato dalla gradient accumulation pari a 16. L'ottimizzatore **paged_adamw_8bit**, il gradient checkpointing e la mixed precision dinamica contengono il consumo di memoria, secondo le strategie descritte nel Capitolo 4.

Valutazione e salvataggio dei checkpoint avvengono a fine di ogni epoca, con selezione del miglior checkpoint sulla **eval_loss** da minimizzare, tramite **metric_for_best_model="eval_loss"** e **greater_is_better=False**. Durante il training non viene calcolata una F_1 token-level con sequeval, poiché l'output supervisionato è una completion testuale e non una sequenza BIO.

Completion-only loss

Un parametro centrale è **completion_only_loss=True**, che limita il calcolo della loss alla sola completion, ovvero alla risposta JSON attesa, escludendo il prompt dell'utente, come descritto nel Capitolo 4.

La scelta è coerente con l'obiettivo del task, poiché il modello non deve riprodurre l'istruzione del prompt ma generare il JSON corretto a partire dall'immagine e dall'istruzione. La validation loss permette di selezionare il trial migliore, mentre la

qualità applicativa dell'estrazione viene valutata successivamente tramite parsing, normalizzazione e confronto field-level.

5.4.6 Costruzione del trainer, grid search e salvataggio

Il trainer viene costruito nella funzione `run_trial`. Per ogni configurazione della griglia lo script inizializza modello, processor, configurazione LoRA e argomenti di training, quindi crea un `SFTTrainer`.

Il parametro `peft_config` applica la configurazione LoRA al modello, mentre `processing_class` indica il processor per la preparazione degli input multimodali. Dopo l'addestramento, il modello viene valutato sul validation set e il trial viene salvato.

Per ogni trial vengono registrati il nome del run, la directory finale, il tempo di training, la training loss, gli iperparametri e le metriche di valutazione, scritti progressivamente nel file `grid_results.jsonl` per mantenere traccia di tutti gli esperimenti anche in caso di interruzione.

Il dettaglio della grid search, del criterio di selezione e della configurazione finale è discusso nella Sezione 5.5.2.

A differenza di LayoutLMv3, non viene eseguito un retraining finale. Ogni trial è già un'esecuzione completa che salva il proprio adapter LoRA, e il modello finale corrisponde all'adapter del trial con minore `eval_loss`, riutilizzato insieme al processor nelle fasi di predizione. La scelta riflette la maggiore onerosità del fine-tuning di Qwen2.5-VL.

Nel complesso, il training di Qwen2.5-VL traduce il task KIE in un supervised fine-tuning generativo multimodale, in cui il modello apprende a produrre un JSON strutturato a partire dall'immagine del documento e dal prompt. Le principali criticità operative riguardano la stabilità dell'output JSON, il rispetto dello schema e il costo computazionale del modello.

5.5 Ricerca degli iperparametri e selezione dei modelli finali

Definita la configurazione di training dei due modelli, è stato necessario selezionare le impostazioni finali da usare per la generazione delle predizioni e per il confronto sul test set. La scelta degli iperparametri incide in modo significativo sulla stabilità del training e sulle prestazioni di validazione, attraverso valori come learning rate, numero di epoche, strategia di ottimizzazione o configurazione degli adattatori.

Per entrambi i modelli la selezione è stata trattata come fase sperimentale distinta, successiva alla configurazione di base descritta nelle Sezioni 5.3 e 5.4. Le due strategie adottate riflettono la diversa natura dei modelli e dei rispettivi processi di training. LayoutLMv3, più leggero, ha permesso una ricerca iperparametrica automatizzata tramite Optuna, con uno spazio di ricerca esteso a parametri di ottimizzazione e di gestione dello sbilanciamento e con la F_1 sul validation set come obiettivo da massimizzare. Per Qwen2.5-VL è stata invece

adottata una grid search manuale su un insieme ristretto di configurazioni QLoRA. La scelta è motivata dal costo computazionale più elevato di ogni trial, poiché **Qwen/Qwen2.5-VL-7B-Instruct** è un Vision-Language Model multimodale più oneroso da addestrare. In questo caso il criterio di selezione è la validation loss, coerente con la formulazione del training come language modeling supervisionato sulla completion JSON.

Le sottosezioni successive descrivono separatamente le due procedure, iniziando dalla ricerca iperparametrica di LayoutLMv3 e passando poi alla grid search di Qwen2.5-VL. Per ciascun modello vengono riportati lo spazio di ricerca, i risultati principali dei trial, il criterio di selezione e la configurazione finale.

5.5.1 Ricerca iperparametrica di LayoutLMv3

Per LayoutLMv3 è stata effettuata una ricerca iperparametrica automatizzata, con l'obiettivo di individuare una configurazione di training adatta al task. Dato il forte sbilanciamento verso la classe O , oltre agli iperparametri classici sono stati inclusi nella ricerca anche parametri legati alla Focal Loss, al calcolo dei pesi di classe e al Layer-wise Learning Rate Decay.

La ricerca è stata implementata nello script `train_layout.py` tramite il metodo `hyperparameter_search` del `Trainer` di Hugging Face, con backend Optuna. L'obiettivo ottimizzato è la F_1 sul validation set, calcolata con `segeval`. Il criterio resta quindi coerente con la configurazione di validazione descritta precedentemente. La ricerca è stata configurata per eseguire otto trial nell'esperimento completo. Ogni trial inizializza un nuovo modello a partire dal checkpoint `microsoft/layoutlmv3-base`, evitando che i pesi appresi in un'esecuzione influenzino le successive. Lo spazio di ricerca comprendeva iperparametri standard del training e parametri introdotti dalla pipeline personalizzata.

5.5. IPERPARAMETRI E MODELLI FINALI

Iperparametro	Valori / intervallo	Significato
<code>learning_rate</code>	$[2 \cdot 10^{-5}, 7 \cdot 10^{-5}]$, log	Learning rate di base
<code>weight_decay</code>	0,0 / 0,01	Regolarizzazione dei pesi
<code>num_train_epochs</code>	8 / 10	Numero di epoche
<code>focal_gamma</code>	1,0 / 2,0	Intensità della Focal Loss
<code>smooth_factor</code>	0,3 / 0,5 / 0,7	Smussamento dei pesi di classe
<code>llrd_decay_factor</code>	0,90 / 0,95 / 0,98	Decadimento layer-wise del learning rate
<code>lr_scheduler_type</code>	linear / reduce_lr_on_plateau	Scheduler del learning rate
<i>Solo con scheduler linear:</i>		
<code>warmup_ratio</code>	0,05 / 0,1	Frazione di step di warmup
<i>Solo con scheduler reduce_lr_on_plateau:</i>		
<code>factor</code>	0,5	Fattore di riduzione del learning rate
<code>patience</code>	2	Epoche senza miglioramento prima della riduzione
<code>threshold</code>	10^{-4}	Soglia minima di miglioramento
<code>min_lr</code>	10^{-7}	Learning rate minimo

Tabella 5.8: Spazio di ricerca degli iperparametri esplorato con Optuna.

Con lo scheduler `linear`, Optuna ha selezionato anche il valore di `warmup_ratio`. Con `reduce_lr_on_plateau` il warmup è stato impostato a zero e sono stati usati parametri dedicati per ridurre il learning rate quando la metrica di validazione non migliorava, ovvero `factor=0.5`, `patience=2`, `threshold=1e-4` e `min_lr=1e-7`.

La ricerca ha prodotto otto trial complessivi. La tabella riporta i risultati disponibili dai log dell'esecuzione.

Trial	Stato	F1 val.	Scheduler	LR	W.d.	Ep.	γ	Smooth	LLRD	Warmup
0	completato	0,8156	linear	$5,28 \cdot 10^{-5}$	0,01	10	1,0	0,3	0,90	0,10
1	completato	0,7156	reduce_ on_plateau	$4,74 \cdot 10^{-5}$	0,00	10	2,0	0,5	0,90	—
2	completato	0,4820	reduce_ on_plateau	$2,14 \cdot 10^{-5}$	0,00	8	1,0	0,7	0,90	—
3	completato	0,5296	linear	$5,46 \cdot 10^{-5}$	0,00	10	2,0	0,7	0,90	0,10
4	completato	0,6998	linear	$2,70 \cdot 10^{-5}$	0,01	10	1,0	0,5	0,95	0,10
5	pruned	—	—	—	—	—	—	—	—	—
6	completato	0,7766	linear	$6,97 \cdot 10^{-5}$	0,00	8	2,0	0,3	0,95	0,10
7	pruned	—	—	—	—	—	—	—	—	—

Tabella 5.9: Esiti dei trial della ricerca iperparametrica con Optuna per LayoutLMv3.

I trial 5 e 7 sono stati interrotti dal meccanismo di pruning di Optuna, che ha evitato di completare configurazioni poco promettenti rispetto ai risultati intermedi disponibili, contenendo il costo complessivo della ricerca.

Il miglior trial è risultato il trial 0, con una F_1 di validazione pari a circa 0,8156. La configurazione selezionata è la seguente.

Parametro	Valore selezionato
<code>learning_rate</code>	$5,28 \cdot 10^{-5}$
<code>lr_scheduler_type</code>	linear
<code>weight_decay</code>	0,01
<code>num_train_epochs</code>	10
<code>focal_gamma</code>	1,0
<code>smooth_factor</code>	0,3
<code>llrd_decay_factor</code>	0,90
<code>warmup_ratio</code>	0,1

Tabella 5.10: Configurazione finale di LayoutLMv3 selezionata dalla ricerca iperparametrica.

La configurazione migliore presenta alcuni aspetti rilevanti. Il numero di epoche selezionato è 10, il valore più alto tra quelli considerati. Il learning rate scelto è relativamente alto all'interno dell'intervallo di ricerca, ma resta coerente con un fine-tuning controllato di un Transformer pre-addestrato. Lo scheduler selezionato è quello lineare, con warmup pari a 0,1. Per la gestione dello sbilanciamento, la configurazione migliore non usa valori aggressivi: `focal_gamma=1.0` avvicina il comportamento della Focal Loss a quello della cross-entropy standard, mentre `smooth_factor=0.3` applica una correzione più conservativa rispetto a 0.5 e 0.7. Il risultato suggerisce che, nel dataset considerato, una compensazione troppo forte delle classi rare può non essere ottimale e rischia di rendere il training meno stabile. Il valore `llrd_decay_factor=0.9` introduce invece un decadimento più marcato del learning rate tra i layer, aggiornando in modo più conservativo gli strati inferiori del modello.

Si osserva anche che i trial con scheduler `reduce_lr_on_plateau` e quelli con combinazioni più aggressive di Focal Loss e smoothing dei pesi hanno ottenuto prestazioni inferiori. Pur non bastando a trarre conclusioni generali, il comportamento indica che, in questa configurazione sperimentale, uno scheduler più semplice e una correzione moderata dello sbilanciamento hanno favorito una convergenza migliore.

Dal punto di vista computazionale, dai log il job di LayoutLMv3, comprendente ricerca iperparametrica e retraining finale, ha richiesto circa 10 ore e 34 minuti. I trial completati sono durati nell'ordine di un'ora e mezza ciascuno, mentre quelli interrotti anticipatamente hanno richiesto meno tempo, a conferma dell'utilità del pruning di Optuna nel ridurre il costo complessivo della ricerca.

Dopo la ricerca iperparametrica lo script ha eseguito un nuovo training finale con gli iperparametri selezionati, anziché riutilizzare il checkpoint del trial migliore. La scelta rende più ordinata la procedura sperimentale, perché permette di salvare

un modello finale dedicato, separato dai checkpoint intermedi della ricerca, e di usarlo in modo chiaro nelle fasi successive di predizione.

Il modello finale riaddestrato è stato valutato sul validation set, con i risultati seguenti.

Metrica	Valore
Loss	0,0049
Precision	0,7300
Recall	0,9109
F1	0,8104
Accuracy	0,9903
Epoche	10

Tabella 5.11: Metriche di validazione di LayoutLMv3 con la configurazione finale.

La F_1 finale risulta leggermente inferiore all’objective del miglior trial Optuna, pari a circa 0,8156. La differenza è contenuta e non modifica la scelta della configurazione, poiché il modello finale viene riaddestrato in una nuova esecuzione a partire dagli stessi iperparametri e non coincide necessariamente in modo esatto con il checkpoint prodotto durante il trial. Piccole variazioni possono derivare dalla dinamica del training, dalla selezione dei checkpoint e dalla gestione numerica su GPU.

Nel complesso, la ricerca iperparametrica ha permesso di selezionare una configurazione equilibrata per LayoutLMv3, che combina una Focal Loss non eccessivamente aggressiva, pesi di classe smussati, Layer-wise Learning Rate Decay marcato e scheduler lineare con warmup. Questa configurazione è stata usata per generare le predizioni finali sul test set.

5.5.2 Ricerca iperparametrica di Qwen2.5-VL

Per Qwen2.5-VL la selezione degli iperparametri è stata impostata tramite una grid search manuale su un numero limitato di configurazioni, anziché una ricerca automatica più ampia.

L’obiettivo non era esplorare in modo esaustivo l’intero spazio degli iperparametri, ma confrontare alcune configurazioni significative mantenendo fisso il resto della procedura. Sono stati variati il learning rate e la configurazione LoRA, mentre numero di epoche, batch size, gradient accumulation, scheduler, quantizzazione e `completion_only_loss` sono stati mantenuti costanti. La griglia è definita direttamente nello script `train_qwen.py`.

La ricerca considera cinque configurazioni, riportate nella tabella seguente.

5.5. IPERPARAMETRI E MODELLI FINALI

Trial	Learning rate	<code>lora_r</code>	<code>lora_alpha</code>	Epoche
0	$5 \cdot 10^{-5}$	4	8	3
1	$1 \cdot 10^{-4}$	4	8	3
2	$2 \cdot 10^{-4}$	4	8	3
3	$1 \cdot 10^{-4}$	8	16	3
4	$5 \cdot 10^{-5}$	8	16	3

Tabella 5.12: Configurazioni esplorate nella grid search di Qwen2.5-VL.

Tutti i trial sono stati eseguiti mantenendo fissa la configurazione di training descritta nella Sezione 5.4.6, variando soltanto learning rate e parametri LoRA. Per ogni configurazione, lo script esegue il training, valuta il modello sul validation set e salva i risultati nel file `grid_results.jsonl`.

I risultati ottenuti dai cinque trial sono riportati nella tabella seguente.

Trial	Learning rate	LoRA r	LoRA α	Train loss	Eval loss	Token acc.	Tempo
0	$5 \cdot 10^{-5}$	4	8	0,0749	0,0581	0,9822	2h 35m
1	$1 \cdot 10^{-4}$	4	8	0,0598	0,0438	0,9862	2h 35m
2	$2 \cdot 10^{-4}$	4	8	0,0485	0,0354	0,9887	2h 35m
3	$1 \cdot 10^{-4}$	8	16	0,0533	0,0408	0,9873	2h 35m
4	$5 \cdot 10^{-5}$	8	16	0,0667	0,0512	0,9843	2h 35m

Tabella 5.13: Risultati della grid search di Qwen2.5-VL. In grassetto la configurazione selezionata, corrispondente alla minima eval loss.

Il trial 2 viene selezionato perché raggiunge la eval loss minima; la configurazione finale è riportata nella tabella seguente.

Parametro	Valore selezionato
Trial	2
Learning rate	$2 \cdot 10^{-4}$
<code>lora_r</code>	4
<code>lora_alpha</code>	8
Epoche	3
Train loss	0,0485
Eval loss	0,0354
Token accuracy	0,9887
Eval entropy	0,0312

Tabella 5.14: Configurazione finale di Qwen2.5-VL selezionata dalla grid search.

All'interno della griglia il learning rate emerge come il fattore dominante. Nella famiglia con `lora_r`=4 e `lora_alpha`=8, l'aumento da $5 \cdot 10^{-5}$ a $1 \cdot 10^{-4}$ e poi a

$2 \cdot 10^{-4}$ produce un miglioramento progressivo della validation loss, da 0,058100 a 0,043829 e infine a 0,035439. A parità di learning rate, l'aumento del rango da 4 a 8 porta un miglioramento più contenuto: con $1 \cdot 10^{-4}$ il trial 3, con `lora_r=8`, ottiene 0,040793 contro 0,043829 del trial 1; con $5 \cdot 10^{-5}$ il trial 4 ottiene 0,051231 contro 0,058100 del trial 0. Poiché nessuna configurazione con `lora_r=8` è stata testata a $2 \cdot 10^{-4}$, il miglior risultato complessivo resta il trial 2, che combina adattatori compatti con il learning rate più alto della griglia.

Il successo del trial 2 non implica che un rango maggiore sia inferiore in assoluto, ma soltanto che, tra le combinazioni provate, gli adattatori compatti hanno sfruttato meglio il learning rate più alto. La griglia di cinque punti non consente di isolare l'effetto del rango in corrispondenza del learning rate ottimale, ma il suo scopo non è caratterizzare in modo esaustivo lo spazio degli iperparametri, bensì individuare una configurazione efficace e compatibile con il budget computazionale disponibile.

Il tempo di training è sostanzialmente costante tra le configurazioni, intorno a 2 ore e 35 minuti per trial. Le differenze di rango testate, da 4 a 8, non incidono in modo significativo sul costo complessivo, dominato dal forward e dal backward pass sul modello base quantizzato più che dal numero di parametri addestrabili degli adattatori. Dai log, l'intero job di Qwen2.5-VL ha richiesto circa 13 ore, a conferma del costo elevato del fine-tuning generativo multimodale anche con QLoRA.

Dopo la grid search non viene eseguito un retraining finale: il modello selezionato corrisponde all'adapter LoRA del trial con minore `eval_loss`, riutilizzato insieme al processor salvato nella stessa directory. La configurazione finale è stata quindi usata per generare le predizioni sul test set e per il confronto sperimentale con LayoutLMv3 discusso nel Capitolo 7.

5.5.3 Andamento del training dei modelli selezionati

Dopo la selezione degli iperparametri è utile osservare l'andamento del training dei modelli finali. L'analisi non sostituisce la valutazione sul test set, ma permette di verificare la stabilità dell'addestramento, l'evoluzione delle metriche di validazione e l'eventuale presenza di oscillazioni o segnali di instabilità.

Le curve si riferiscono ai modelli selezionati nelle sottosezioni precedenti, cioè il modello LayoutLMv3 riaddestrato con i migliori iperparametri Optuna e il miglior trial Qwen2.5-VL della grid search manuale.

Andamento del training di LayoutLMv3

Per LayoutLMv3 è stato analizzato il training finale eseguito con la configurazione selezionata da Optuna. Il modello è stato addestrato per 10 epoche, con la F_1 sul validation set come criterio principale di selezione. Durante l'addestramento sono state monitorate la loss di training e le metriche precision, recall, F_1 e accuracy calcolate tramite `segeval`.

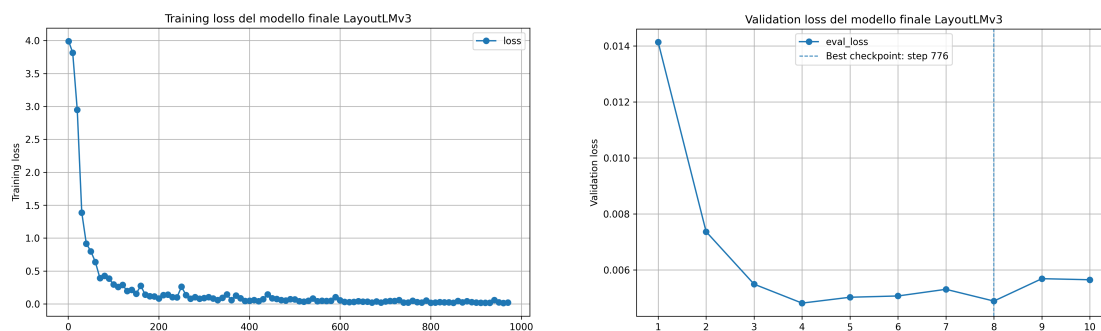


Figura 5.2: Training loss (sinistra) e validation loss (destra) del miglior trial di LayoutLMv3.

La curva della training loss mostra l'evoluzione dell'ottimizzazione sui dati di addestramento e serve a verificare che il processo proceda in modo regolare, senza segnali evidenti di divergenza.

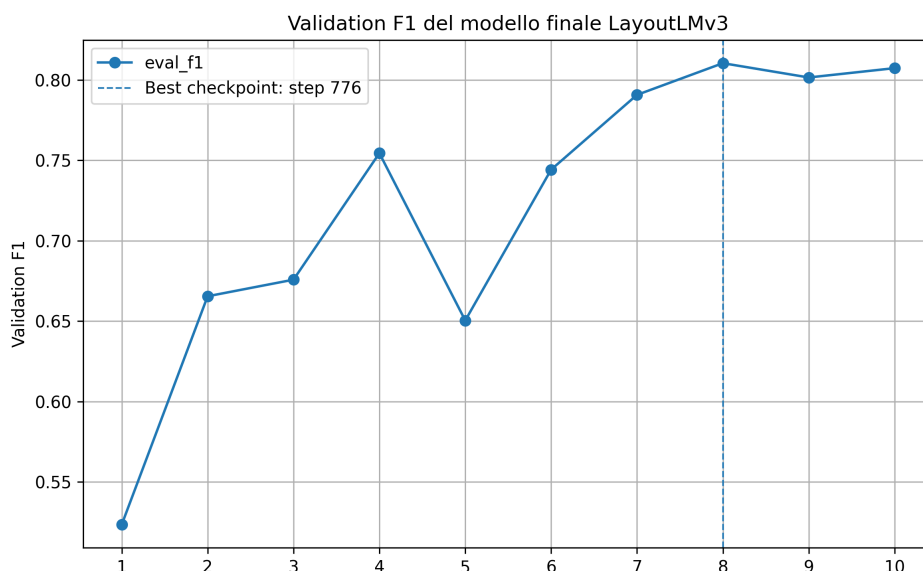


Figura 5.3: F_1 di validazione di LayoutLMv3 per epoca durante l'addestramento finale.

La curva della F_1 di validazione è particolarmente importante, perché questa metrica è stata usata per selezionare il checkpoint migliore. Nel training finale la F_1 ha un andamento non perfettamente monotono: cresce nelle prime epoche, raggiunge il valore migliore durante l'addestramento e poi oscilla leggermente nelle epoche successive senza superare il massimo osservato. Si tratta di un comportamento normale in un fine-tuning su validation set, dove il miglior checkpoint non coincide necessariamente con l'ultima epoca.

Grazie all'impostazione `load_best_model_at_end=True`, al termine dell'addestramento viene ricaricato il checkpoint migliore secondo la F_1 di validazione.

Sebbene il training venga completato per tutte le 10 epoche, il modello finale salvato corrisponde quindi al checkpoint con la prestazione migliore sul validation set, con F_1 di circa 0,8104, precision di circa 0,730 e recall di circa 0,911.

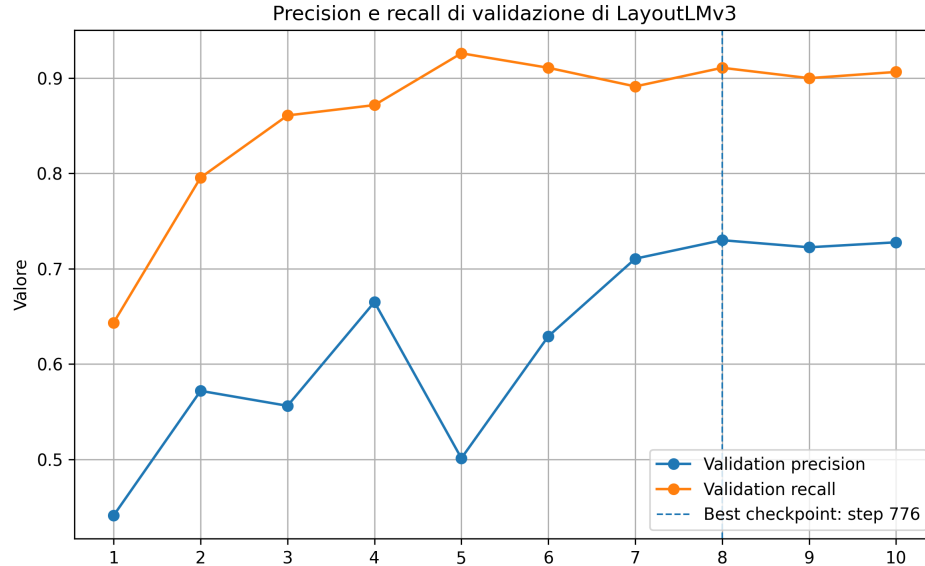


Figura 5.4: Precision e recall di validazione di LayoutLMv3 per epoca durante l’addestramento finale.

L’andamento di precision e recall aiuta a interpretare il comportamento del modello. La recall risulta superiore alla precision, quindi LayoutLMv3 tende a recuperare una quota elevata delle entità presenti, accettando però più falsi positivi. Il comportamento è coerente con l’uso combinato di Focal Loss e pesi di classe, introdotti per contrastare lo sbilanciamento verso la classe *O* e per spingere il modello a non ignorare le classi rare. La tendenza viene ripresa nel Capitolo 7, dove le predizioni sono analizzate a livello di campo e gli errori classificati in categorie più interpretabili, come missing, wrong value e hallucination.

Andamento del training di Qwen2.5-VL

Per Qwen2.5-VL è stato analizzato il miglior trial della grid search manuale, con learning rate pari a $2 \cdot 10^{-4}$, `lora_r=4` e `lora_alpha=8`. Le metriche osservate sono la training loss, la `eval_loss` e la mean token accuracy sulla sequenza target.

5.5. IPERPARAMETRI E MODELLI FINALI

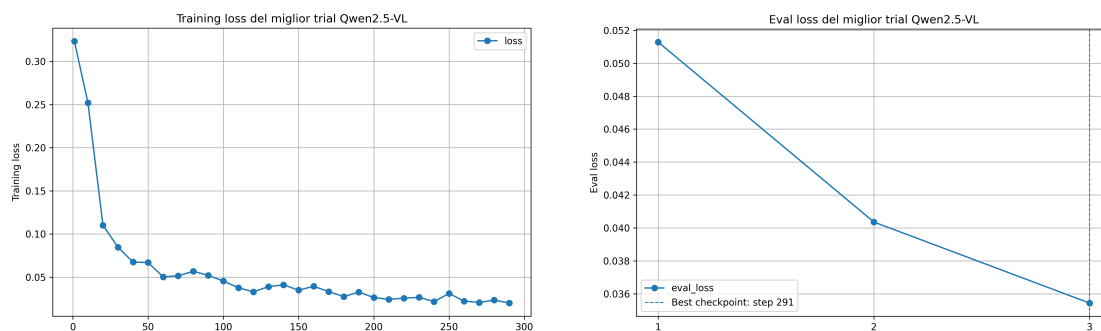


Figura 5.5: Training loss (sinistra) e validation loss (destra) del miglior trial di Qwen2.5-VL.

La curva della training loss mostra l'evoluzione dell'errore nell'ottimizzazione della risposta JSON attesa. Poiché il training è stato eseguito per 3 epoche, le curve risultano meno dense di quelle di LayoutLMv3, ma restano utili a verificare la regolarità del processo.

La `eval_loss` sul validation set è la metrica usata per selezionare il miglior trial. Il trial selezionato raggiunge una `eval_loss` di circa 0,0354, la più bassa tra le configurazioni testate.

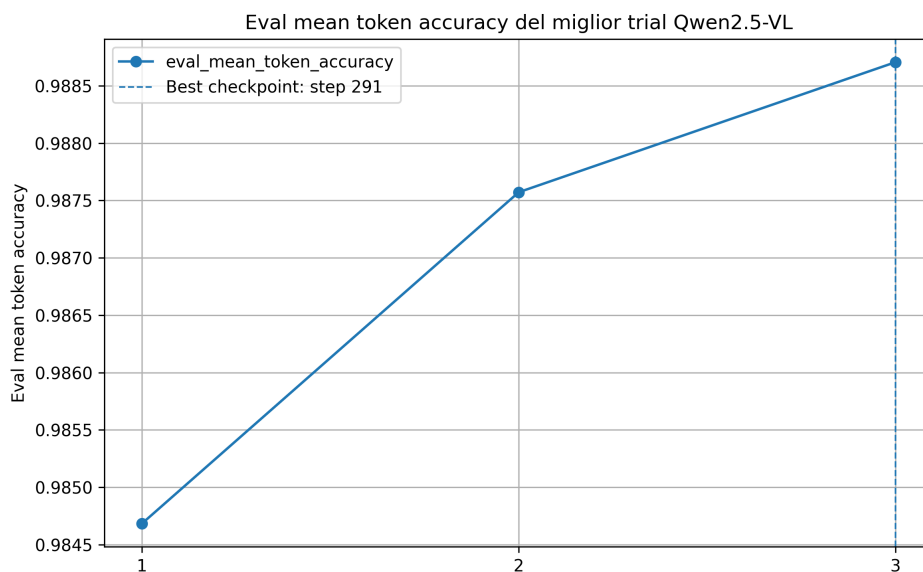


Figura 5.6: Andamento della mean token accuracy di validazione del miglior trial di Qwen2.5-VL.

La mean token accuracy indica la quota di token della risposta target predetti correttamente durante la validazione, e nel trial migliore raggiunge circa il 98,87%. Non va però interpretata come accuracy finale di estrazione dei campi, perché misura la corrispondenza token per token rispetto alla sequenza JSON attesa, includendo anche elementi strutturali come parentesi, virgole, virgolette e nomi dei campi, e non solo i valori informativi.

Confronto qualitativo dell'andamento

Le curve dei due modelli non sono confrontabili in senso stretto, perché descrivono due formulazioni diverse del task. LayoutLMv3, addestrato come classificatore token-level, viene monitorato tramite precision, recall e F_1 sulle sequenze BIO. Qwen2.5-VL, addestrato come modello generativo image-to-JSON, viene monitorato tramite loss e accuratezza token-level sulla completion testuale, che richiede poi parsing, normalizzazione e confronto field-level. Le curve servono quindi a verificare che i due processi di addestramento siano coerenti e stabili, ma non bastano a determinare quale modello sia migliore nel task finale. Il confronto quantitativo e qualitativo sulle predizioni generate è svolto nel Capitolo 7.

Capitolo 6

Valutazione e analisi

Selezionati i modelli finali nel capitolo precedente, questo capitolo descrive come le loro predizioni sul test set vengono generate, ricondotte a una rappresentazione comune e valutate, e presenta un'analisi diagnostica sulla pipeline LayoutLMv3.

Le due pipeline producono output di natura diversa, una sequenza di etichette BIO per LayoutLMv3 e una stringa JSON per Qwen2.5-VL, ricondotti a una stessa rappresentazione field-level su cui calcolare le metriche.

Viene quindi definito il protocollo di valutazione, con la classificazione degli esiti, le metriche aggregate e il document exact match.

Il capitolo si chiude con un'analisi diagnostica di ablation e robustezza su LayoutLMv3, che ne misura la sensibilità rispetto alle modalità informative in input. I risultati del confronto tra i due modelli sono presentati nel capitolo successivo.

6.1 Generazione delle predizioni finali e costruzione del dataset di valutazione

La pipeline procede con la generazione delle predizioni sul test set e con la costruzione di una rappresentazione comune su cui calcolare le metriche.

Le predizioni sono state generate sui 185 record del test set, tenuti separati durante addestramento e validazione. Per ciascun modello è stato caricato il miglior checkpoint individuato nelle sezioni precedenti. L'obiettivo di questa fase non è ancora il calcolo delle metriche, ma la trasformazione degli output eterogenei in una struttura comune a livello di campo, denominata `evaluation_rows`. La generazione è affidata a due script distinti, `predict_layoutlmv3_best.py` e `predict_qwen_best.py`, mentre la normalizzazione, il parsing e la costruzione della rappresentazione comune avvengono nel notebook di confronto finale.

6.1.1 Predizione con LayoutLMv3

La predizione con LayoutLMv3 carica il modello finale salvato dopo il retraining con i migliori iperparametri Optuna. Lo script carica due versioni del dataset. La prima è il dataset originale splittato, che conserva le parole OCR e gli identificativi dei

record; la seconda è il dataset tokenizzato, cioè la rappresentazione effettivamente consumata dal modello, composta da token, bounding box, `pixel_values` ed etichette allineate.

Prima dell'inferenza, la funzione `check_split_alignment` verifica la corrispondenza tra le due versioni. Il controllo accerta che lo split di test sia presente in entrambe, che il numero di esempi coincida e che gli id compaiano nello stesso ordine.

La verifica è necessaria perché le predizioni vengono prodotte sul dataset tokenizzato, ma il loro significato deve poi essere ricondotto alle parole OCR originali per costruire la valutazione a livello di campo.

Il modello viene istanziato tramite `LayoutLMv3ForTokenClassification`, insieme al processor; se quest'ultimo non è presente nella directory del modello finale, lo script ricade sul processor base `microsoft/layoutlmv3-base`. L'inferenza è affidata a un'istanza di `Trainer` configurata per la sola predizione. Il metodo `predict` restituisce, per ciascun token, i logits associati alle possibili etichette BIO, e la classe predetta si ottiene applicando l'`argmax`.

Listing 6.1: Generazione delle predizioni di LayoutLMv3 e conversione in formato field-level.

```

1 def predict_layout_split(split_name, original_split, tokenized_split, model,
2   processor):
3     trainer = build_trainer(model, processor, tokenized_split)
4     output = trainer.predict(tokenized_split)
5
6     predictions = output.predictions
7     if isinstance(predictions, tuple):
8         predictions = predictions[0]
9
10    rows = convert_predictions_to_rows(
11        original_split=original_split,
12        predictions=predictions,
13        label_ids=output.label_ids,
14        id2label=model.config.id2label,
15        split_name=split_name,
16    )
17    save_jsonl(rows, PREDICTION_DIR / f"layout_raw_predictions_{split_name}.jsonl"
18    )

```

In fase di salvataggio lo script applica una maschera di validità (`true_ids != -100`), conservando soltanto le posizioni confrontabili con le parole del documento. Le predizioni numeriche vengono convertite in etichette testuali tramite la mappatura `id2label`; accanto alle etichette predette, lo script mantiene anche quelle vere presenti nel dataset tokenizzato, utili per i controlli di coerenza sulle posizioni valide.

Poiché LayoutLMv3 opera con una lunghezza massima di input, alcuni documenti contengono più parole OCR di quante ne vengano effettivamente elaborate. Per ciascun record lo script conserva quindi un'informazione sintetica sulla troncatura, ovvero numero di parole originali, numero di parole utilizzate e flag `is_truncated`, utile a interpretare gli errori riconducibili a una porzione di documento rimasta fuori dall'input.

Lo script verifica, infine, che l'ordine degli identificativi nelle predizioni coincida con quello del test set originale, così da garantire che ogni sequenza di etichette

sia associata al documento corretto. Le predizioni grezze di LayoutLMv3 restano espresse a livello di token ed etichette BIO, e vengono convertite in formato campo-valore nella fase successiva della pipeline di valutazione.

6.1.2 Predizione con Qwen2.5-VL

La predizione con Qwen2.5-VL segue una procedura coerente con la natura generativa del modello. Lo script `predict_qwen_best.py` carica il dataset splittato, il modello base `Qwen/Qwen2.5-VL-7B-Instruct` e l'adapter LoRA del miglior trial. La configurazione del trial migliore viene letta dal file `best_grid_result.json`, da cui si recupera la directory dell'adapter selezionato. Il modello base viene caricato nuovamente in quantizzazione a 4 bit NF4, con la stessa impostazione usata in addestramento, e l'adapter viene applicato tramite `PeftModel.from_pretrained` con `is_trainable=False`, ottenendo il modello fine-tuned in sola inferenza.

Rispetto al training, durante il quale `use_cache` era disattivato per compatibilità con il gradient checkpointing, in fase di predizione la cache di generazione viene riattivata. La scelta è coerente con l'inferenza autoregressiva, perché consente di riutilizzare gli stati già calcolati e rende più efficiente la produzione dei token successivi.

Per ogni documento del test set viene costruito un messaggio multimodale in formato conversazionale, composto dall'immagine e dallo stesso prompt istruzionale usato nel fine-tuning, che richiede di restituire esclusivamente un oggetto JSON valido con i campi target. Il messaggio viene convertito nel formato atteso dal modello tramite `apply_chat_template`, mentre gli input visivi vengono preparati con `process_vision_info` e passati al processor insieme al testo.

Listing 6.2: Preparazione di un input multimodale per l'inferenza con Qwen2.5-VL.

```
1 def prepare_qwen_inputs(sample, processor, prompt):
2     messages = build_qwen_messages(sample, prompt)
3
4     text = processor.apply_chat_template(
5         messages, tokenize=False, add_generation_prompt=True
6     )
7     image_inputs, video_inputs = process_vision_info(messages)
8
9     return processor(
10         text=[text], images=image_inputs, videos=video_inputs,
11         padding=True, return_tensors="pt",
12     )
```

La generazione è deterministica. Non viene usato il sampling (`do_sample=False`) e viene impostato un singolo beam (`num_beams=1`), configurazione che corrisponde a una decodifica greedy e rende le predizioni riproducibili, sottraendo la valutazione alla variabilità stocastica della generazione. Il numero massimo di nuovi token è fissato a 192, sufficiente a contenere l'oggetto JSON con i nove campi anche in presenza di valori testuali più lunghi, e al tempo stesso un limite contro generazioni eccessivamente estese in caso di comportamenti anomali.

Listing 6.3: Generazione della predizione JSON per un singolo documento con Qwen2.5-VL.

```

1 def predict_one_qwen(sample, model, processor, prompt, max_new_tokens):
2     inputs = prepare_qwen_inputs(sample, processor, prompt)
3     inputs = inputs.to(get_model_input_device(model))
4
5     tokenizer = processor.tokenizer
6
7     with torch.inference_mode():
8         generated_ids = model.generate(
9             **inputs,
10            max_new_tokens=max_new_tokens,
11            do_sample=False,
12            num_beams=1,
13            eos_token_id=tokenizer.eos_token_id,
14        )
15
16     # rimuove il prompt, mantenendo solo i token generati
17     trimmed = [out[len(inp):] for inp, out in zip(inputs.input_ids, generated_ids)]
18
19     generated_text = processor.batch_decode(trimmed, skip_special_tokens=True)[0]
20
21     return {"id": sample["id"], "raw_prediction": generated_text.strip(), "error":
        None}

```

Conclusa la generazione, la porzione corrispondente al prompt viene rimossa e si conservano soltanto i nuovi token prodotti. Il testo generato viene decodificato e salvato come `raw_prediction`, senza alcun parsing JSON in questa fase: la verifica strutturale, la pulizia e la normalizzazione sono demandate alla pipeline di valutazione. La separazione distingue con nettezza l'inferenza dalla valutazione, perché lo script registra ciò che il modello ha effettivamente generato, mentre il notebook di confronto interpreta quell'output in modo uniforme rispetto alle predizioni di LayoutLMv3. Lo script è progettato anche per riprendere l'esecuzione dopo un'interruzione. Prima di generare una nuova predizione, controlla se per lo stesso identificativo esiste già un risultato salvato e, in tal caso, lo salta, accorgimento utile perché la generazione autoregressiva con un modello da 7 miliardi di parametri può richiedere tempi elevati. Eventuali errori durante la generazione vengono registrati nel record corrispondente, senza interrompere l'elaborazione dell'intero test set. Al termine, lo script riordina le righe secondo l'ordine originale del dataset e verifica che gli identificativi coincidano con quelli del test set.

6.1.3 Output eterogenei e necessità di una rappresentazione comune

Le due pipeline producono output strutturalmente diversi, conseguenza diretta dei paradigmi adottati. LayoutLMv3 restituisce per ciascun documento una sequenza di etichette BIO associate alle parole OCR. L'output è già allineato alla sequenza testuale del documento, ma non corrisponde a un dizionario campo-valore: per ottenere i valori estratti occorre ricostruire gli span predetti, raggruppando le parole che condividono la stessa entità.

Qwen2.5-VL restituisce invece una stringa testuale. Nelle condizioni attese la stringa rappresenta un oggetto JSON con i campi richiesti, ma la generazione autoregressiva

non offre alcuna garanzia strutturale, e l'output può contenere formattazioni indesiderate, testo accessorio, campi non previsti o JSON non perfettamente valido.

Questa eterogeneità impedisce un confronto diretto tra le predizioni raw. Prima di calcolare qualsiasi metrica occorre ricondurre entrambi gli output a una rappresentazione comune a livello di campo, nella quale ogni record è descritto dagli stessi campi target e ogni valore predetto può essere confrontato con il corrispondente valore atteso della ground truth. La trasformazione ha quindi una funzione metodologica precisa, perché permette di valutare un modello token-level come LayoutLMv3 e un modello generativo image-to-JSON come Qwen2.5-VL all'interno dello stesso protocollo.

6.1.4 Costruzione del dataset di valutazione field-level

La costruzione del dataset di valutazione avviene nel notebook di confronto finale. Il primo controllo verifica che entrambi i modelli abbiano prodotto predizioni per gli stessi record. L'esecuzione finale ha restituito 185 identificativi comuni, senza record presenti in una sola delle due pipeline e senza identificativi duplicati. La verifica è necessaria perché le metriche confrontano, per ogni record, il valore atteso, la predizione di LayoutLMv3 e quella di Qwen2.5-VL, e un disallineamento degli identificativi renderebbe il confronto privo di significato.

Le predizioni di LayoutLMv3 vengono convertite da sequenze BIO a dizionari campo-valore. La funzione di conversione scorre la sequenza di parole ed etichette, individua gli span associati a ciascun campo e ricostruisce il valore testuale corrispondente; quando per uno stesso campo emergono più span, viene scelto quello testualmente più lungo dopo una pulizia minima del testo.

Le predizioni di Qwen2.5-VL vengono invece trattate come testo generativo. Prima del parsing, la stringa viene ripulita dagli elementi estranei al JSON, come blocchi Markdown o testo aggiuntivo, isolando quando possibile il contenuto compreso tra la prima parentesi graffa aperta e l'ultima chiusa, che viene poi parsato come JSON. Nel corso di questa fase le chiavi vengono normalizzate, i campi non previsti scartati e i valori vuoti convertiti in `None`, riconducendo anche l'output generativo a un dizionario con gli stessi campi target.

Il calcolo delle metriche richiede che la ground truth sia disponibile nello stesso formato. I valori attesi vengono quindi ricostruiti dai file JSON delle annotazioni originali, recuperando per ciascun record i campi annotati e utilizzando `textOverride` quando presente e, in sua assenza, il campo `text` standard dell'annotazione.

Ottenute le predizioni field-level dei due modelli e la ground truth, si applica una normalizzazione comune dei valori, dipendente dal tipo di campo, con l'obiettivo di ridurre le differenze puramente formali senza rendere la valutazione troppo permissiva. La normalizzazione generale gestisce spazi, maiuscole e minuscole e valori vuoti; per gli importi uniforma simboli di valuta e separatori decimali; per le date tenta la conversione a un formato standard. Il trattamento specifico del campo pagamento tracciato viene descritto nella sezione dedicata alla classificazione degli esiti field-level. La valutazione resta comunque stretta, perché due valori sono considerati equivalenti solo se coincidono dopo la normalizzazione, scelta che riduce

i falsi negativi dovuti a differenze formali ma continua a penalizzare le differenze reali nel contenuto.

L'ultima fase costruisce la struttura `evaluation_rows`, ovvero il dataset di valutazione vero e proprio. Ogni elemento contiene l'identificativo del record, i valori attesi e le predizioni normalizzate dei due modelli, secondo la struttura logica seguente.

Listing 6.4: Esempio di confronto field-level su un documento.

1	campo	expected	qwen	layout
2	-----	-----	-----	-----
3	datadocumento	2023-11-28	2023-11-28	2023-11-28
4	codicefiscale	XXXXX...	XXXXX...	null
5	totaledetraibile	15,50	15,50	15,50
6	pagamentotracciato	False	False	False
7	...			

L'esecuzione finale produce un dataset di valutazione di 185 record. Convertendo `evaluation_rows` in un `DataFrame` si ottiene una tabella con 28 colonne, una per l'identificativo e, per ciascuno dei nove campi target, tre colonne dedicate al valore atteso, alla predizione di Qwen2.5-VL e a quella di LayoutLMv3. Questa struttura viene utilizzata nella fase successiva per applicare il protocollo di valutazione field-level.

6.2 Metriche e protocollo di valutazione

La valutazione finale dei modelli è stata condotta a livello di campo, in coerenza con l'obiettivo applicativo della tesi. Nel task di Key Information Extraction non basta sapere se un singolo token sia stato classificato correttamente; dal punto di vista pratico interessa verificare se il valore finale di ciascun campo risulti estratto in modo corretto.

Per ciascun modello si effettuano 1.665 confronti a livello di campo, pari a 185 record per nove campi target. Ogni confronto riguarda una coppia (d, f) , dove d indica un documento del test set e f uno dei campi da estrarre.

6.2.1 Esiti field-level e metriche di base

Classificazione degli esiti field-level

Per ciascuna coppia documento-campo vengono confrontati il valore atteso e il valore predetto, entrambi già normalizzati. L'esito appartiene a una delle seguenti categorie mutuamente esclusive.

Valore atteso	Valore predetto	Esito	Contributo
Presente	Uguale all'atteso	Correct extraction	TP
Presente	Assente	Missing	FN
Assente	Presente	Hallucination	FP
Presente	Presente, ma diverso	Wrong value	FP + FN
Assente	Assente	True null	escluso

Tabella 6.1: Tassonomia degli esiti del confronto field-level e contributo alle metriche.

L'estrazione corretta si verifica quando il valore è presente nella ground truth e il modello predice esattamente lo stesso valore dopo normalizzazione, e conta come vero positivo (TP). Il caso missing si verifica quando il valore è presente nella ground truth ma il modello non produce alcuna predizione per quel campo, omettendo un'informazione che avrebbe dovuto estrarre, e conta come falso negativo (FN). Il caso hallucination si verifica quando il valore è assente nella ground truth ma il modello produce comunque una predizione, introducendo un valore per un campo non presente nel documento, e conta come falso positivo (FP).

Il caso wrong value si verifica quando il valore è presente nella ground truth e il modello produce una predizione diversa da quella attesa. L'errore viene conteggiato sia come falso positivo sia come falso negativo (FP e FN), poiché il valore predetto è una predizione positiva non corretta, mentre il valore corretto non è stato effettivamente estratto. In questo modo vengono penalizzati in modo coerente i casi in cui il modello individua un campo ma ne restituisce un contenuto sbagliato. Il caso true null si verifica quando il valore è assente nella ground truth e il modello lascia correttamente vuoto il campo. L'esito segnala una corretta non-estrazione, ma non contribuisce a precision, recall e F_1 -score, che misurano la capacità di estrarre correttamente valori presenti, non quella di predire l'assenza di un dato.

Un campo richiede un trattamento distinto. Il campo pagamento tracciato non è associato nella ground truth a uno span testuale da confrontare, ma a un'informazione di presenza o assenza. Per questo non viene valutato con il criterio field-value strict, basato sul confronto tra valore atteso e valore predetto, ma con un criterio di presence detection booleano, in cui valore atteso e valore predetto vengono ridotti a presente o assente. I conteggi che ne derivano rientrano comunque nelle metriche aggregate descritte nelle sezioni seguenti, e il campo viene inoltre riportato separatamente nella valutazione per campo.

Nel caso della presence detection, le categorie si riducono a quattro. L'estrazione corretta si ha quando il campo risulta presente sia nella ground truth sia nella predizione, con contributo come vero positivo. Il missing si ha quando il campo è presente nella ground truth ma assente nella predizione, con contributo come falso negativo. Il caso hallucination si ha quando il campo è assente nella ground truth ma presente nella predizione, con contributo come falso positivo. Il true null si ha quando il campo è assente in entrambe, ed è un esito corretto escluso da precision, recall e F_1 . La categoria wrong value non si applica, perché un confronto booleano distingue soltanto tra presenza e assenza, non tra valori diversi.

La classificazione proposta è più informativa di una semplice distinzione tra corretto ed errato, perché separa tipologie di errore con significato applicativo diverso: omissione di informazioni necessarie, produzione di dati non presenti nella ground truth e restituzione di valori non corretti.

Precision, recall e F1-score

Dai conteggi di veri positivi, falsi positivi e falsi negativi si calcolano precision, recall e F_1 -score. La precision misura la correttezza delle predizioni prodotte dal modello.

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

Un valore alto di precision indica che, quando il modello produce una predizione, questa tende a essere corretta. Nella Key Information Extraction una precision bassa può segnalare valori errati o campi predetti pur non essendo presenti nel documento. La recall misura invece la capacità del modello di recuperare i valori presenti nella ground truth.

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

Un valore alto di recall indica che il modello tende a non omettere i campi rilevanti, mentre una recall bassa segnala che lascia spesso vuoti campi che avrebbe dovuto estrarre. La metrica F_1 combina precision e recall in un'unica misura tramite media armonica.

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

La metrica è utile quando si confrontano modelli con comportamenti diversi. Un modello prudente può raggiungere una precision alta ma una recall più bassa, perché produce poche predizioni errate omettendo però molti campi; un modello permissivo ottiene il profilo opposto, con recall più alta e precision inferiore, perché recupera più valori introducendo anche più errori. Quando il denominatore di una metrica è pari a zero, il valore viene posto a zero, scelta che evita risultati non definiti e rende il calcolo robusto anche per campi rari o assenti.

6.2.2 Metriche aggregate

Field-level micro F1

La metrica principale del confronto è il field-level micro- F_1 . I conteggi di veri positivi, falsi positivi e falsi negativi vengono aggregati su tutti i documenti e su tutti i campi prima di calcolare precision, recall e F_1 -score. Indicando con D l'insieme dei documenti del test set e con F l'insieme dei campi valutati, i conteggi micro sono definiti come segue.

$$TP_{\text{micro}} = \sum_{d \in D} \sum_{f \in F} TP_{d,f} \quad FP_{\text{micro}} = \sum_{d \in D} \sum_{f \in F} FP_{d,f} \quad FN_{\text{micro}} = \sum_{d \in D} \sum_{f \in F} FN_{d,f}$$

Le metriche micro si calcolano quindi come segue.

$$\text{Precision}_{\text{micro}} = \frac{TP_{\text{micro}}}{TP_{\text{micro}} + FP_{\text{micro}}} \quad \text{Recall}_{\text{micro}} = \frac{TP_{\text{micro}}}{TP_{\text{micro}} + FN_{\text{micro}}}$$

$$F_{1,\text{micro}} = \frac{2 \cdot \text{Precision}_{\text{micro}} \cdot \text{Recall}_{\text{micro}}}{\text{Precision}_{\text{micro}} + \text{Recall}_{\text{micro}}}$$

Il micro- F_1 fornisce una misura complessiva della qualità di estrazione sull'intero test set. Poiché i conteggi vengono aggregati prima del calcolo, i campi più frequenti pesano di più sul risultato finale rispetto ai campi rari. Il micro- F_1 descrive quindi bene la prestazione globale del modello, ma resta poco sensibile alle differenze di comportamento sui singoli campi.

Macro F1 per campo

Per compensare il peso maggiore dei campi più frequenti si calcola anche la macro- F_1 per campo. Precision, recall e F_1 vengono prima calcolate separatamente per ciascun campo, aggregando su tutti i documenti. Per ogni campo f si definiscono le tre quantità seguenti.

$$\text{Precision}_f = \frac{TP_f}{TP_f + FP_f} \quad \text{Recall}_f = \frac{TP_f}{TP_f + FN_f}$$

$$F_{1,f} = \frac{2 \cdot \text{Precision}_f \cdot \text{Recall}_f}{\text{Precision}_f + \text{Recall}_f}.$$

La macro- F_1 è poi la media degli F_1 -score dei campi con supporto maggiore di zero.

$$\text{Macro-}F_1 = \frac{1}{|F^+|} \sum_{f \in F^+} F_{1,f}$$

dove F^+ indica l'insieme dei campi che compaiono almeno una volta nella ground truth del test set. La macro- F_1 assegna lo stesso peso a ogni campo, indipendentemente dalla frequenza, ed è quindi utile per verificare se il modello mantiene prestazioni accettabili anche sui campi meno frequenti. Un modello potrebbe ottenere un micro- F_1 discreto grazie ai campi più comuni, ma una macro- F_1 più bassa se fallisce su quelli rari.

Metriche per campo

Oltre alle metriche aggregate, vengono calcolate metriche specifiche per ciascun campo. Per ogni campo f si considerano veri positivi, falsi positivi, falsi negativi,

supporto, numero di predizioni e conteggi delle diverse tipologie di errore. Il supporto di un campo indica quante volte quel campo è effettivamente presente nella ground truth.

$$\text{support}_f = |\{d \in D : \text{expected}_{d,f} \neq \text{null}\}|$$

Il numero di predizioni indica invece quante volte il modello ha prodotto un valore per quel campo.

$$\text{predictions}_f = |\{d \in D : \text{predicted}_{d,f} \neq \text{null}\}|$$

Questi conteggi aiutano a interpretare le prestazioni. Un campo raro può avere un F_1 -score instabile, perché pochi errori bastano a modificarne sensibilmente il valore, mentre un campo molto frequente può influenzare in misura rilevante le metriche aggregate.

Le metriche per campo permettono quindi di individuare quali informazioni risultano più semplici o più difficili da estrarre, aspetto rilevante in un task KIE dove i campi presentano caratteristiche eterogenee. Alcuni valori sono brevi e facilmente riconoscibili, altri dipendono dal layout, dalla qualità dell'OCR, dalla formulazione del documento o dalla capacità del modello di associare il valore al campo richiesto.

6.2.3 Metriche a livello di documento e sintesi

Document exact match

Una metrica ulteriore è il document exact match, calcolato a livello di documento. Un documento viene considerato corretto solo se tutti i campi target risultano corretti contemporaneamente, compresi i campi assenti che il modello deve lasciare vuoti. Per un documento d , l'exact match è definito come segue.

$$EM_d = \begin{cases} 1 & \text{se } \forall f \in F, \text{ predicted}_{d,f} = \text{expected}_{d,f} \\ 0 & \text{altrimenti} \end{cases}$$

Il document exact match complessivo è quindi la media sui documenti del test set.

$$\text{Document Exact Match} = \frac{1}{|D|} \sum_{d \in D} EM_d$$

La metrica è molto severa, perché un singolo errore su un solo campo basta a classificare l'intero documento come non corretto, e tende quindi ad assumere valori più bassi rispetto alle metriche field-level. Dal punto di vista applicativo resta però significativa, perché misura quanti documenti potrebbero essere accettati senza alcuna correzione manuale.

Analisi delle tipologie di errore

Oltre alle metriche numeriche aggregate, vengono mantenuti i conteggi di missing, hallucination e wrong value. Questi conteggi permettono di distinguere modelli che ottengono punteggi aggregati simili ma commettono errori di natura diversa: omissioni, predizioni non supportate dal documento o valori associati al campo sbagliato.

Per il campo pagamento tracciato si applica il criterio di presence detection descritto in precedenza. La distinzione è utile nel confronto tra LayoutLMv3 e Qwen2.5-VL, perché i due modelli possono raggiungere punteggi aggregati simili pur con profili di errore diversi. L'analisi separata delle tipologie di errore consente quindi di andare oltre il singolo valore dell' F_1 -score e di comprendere meglio i limiti pratici dei due approcci.

Sintesi del protocollo di valutazione

Il protocollo adottato produce una valutazione su più livelli. A livello di singola coppia documento-campo, ogni confronto viene classificato in una delle categorie descritte sopra. A livello di campo, si calcolano precision, recall, F_1 , supporto, numero di predizioni e conteggi delle tipologie di errore. A livello di modello, si calcolano le metriche aggregate, tra cui micro- F_1 , macro- F_1 per campo e document exact match.

L'impostazione consente di confrontare LayoutLMv3 e Qwen2.5-VL in modo uniforme, nonostante la diversa natura dei loro output, perché la valutazione si basa sul risultato applicativo finale: la correttezza dei valori estratti per ciascun campo. Le visualizzazioni prodotte nel notebook di confronto finale rappresentano graficamente le metriche principali, l' F_1 per campo e la distribuzione delle tipologie di errore. I risultati quantitativi e qualitativi sono discussi nel Capitolo 7.

6.3 Analisi di ablation e robustezza su LayoutLMv3

A completamento del setup sperimentale è stata condotta un'analisi diagnostica aggiuntiva sulla pipeline LayoutLMv3, composta da uno studio di ablation e da uno studio di robustezza. L'analisi non costituisce il confronto principale tra i modelli, svolto nel Capitolo 7, ma serve a comprendere la sensibilità di LayoutLMv3 rispetto alle modalità informative disponibili in input, ovvero testo OCR, coordinate spaziali e immagine del documento.

Le metriche riportate in questa sezione sono token/entity-level, calcolate con segeval sulle sequenze BIO del test set, e non coincidono con le metriche field-level usate nel confronto finale tra LayoutLMv3 e Qwen2.5-VL. La baseline con F_1 pari a 0,8529 funge da riferimento interno per l'ablation e lo studio di robustezza. Questo valore non va confuso con la F_1 di validazione pari a circa 0,8104 discussa nella

sezione precedente, poiché il primo è calcolato sul test set, il secondo sul validation set durante il training finale.

LayoutLMv3 si presta a questo tipo di analisi perché espone in modo separabile le componenti informative del documento. Il modello riceve token testuali, bounding box normalizzate e rappresentazione visiva dell'immagine, il che consente di rimuovere o perturbare una singola modalità alla volta, mantenendo invariato il modello addestrato e osservando il conseguente cambiamento delle prestazioni.

Gli esperimenti sono stati eseguiti sul test set, composto da 185 record, utilizzando il modello LayoutLMv3 fine-tuned selezionato nella fase precedente.

Un'analisi equivalente per Qwen2.5-VL richiederebbe un disegno sperimentale diverso, basato ad esempio su degradazioni dell'immagine originale, variazioni del prompt o strategie di decoding alternative, e non sarebbe direttamente confrontabile con la rimozione controllata di testo, layout e immagine in LayoutLMv3.

6.3.1 Impostazione metodologica: ablation in fase di inferenza

Lo studio di ablation può seguire due logiche distinte. La prima è l'ablation in fase di training, in cui si addestrano modelli separati, ciascuno privato di una modalità informativa, per misurare quanto ogni modalità contribuisca all'apprendimento. La seconda è l'ablation in fase di inferenza, in cui un unico modello già addestrato viene valutato più volte, rimuovendo o neutralizzando una singola modalità alla volta in fase di inferenza, a pesi invariati.

In questo lavoro è stata adottata la seconda strategia. La scelta deriva dall'obiettivo dell'analisi, che non è quantificare l'utilità di ciascuna modalità durante l'apprendimento (operazione che richiederebbe il riaddestramento completo del modello per ogni configurazione), ma stimare quanto il modello finale dipenda da ciascuna fonte informativa nelle sue predizioni effettive. L'ablation condotta qui va quindi interpretata come una misura di sensibilità del modello già fine-tuned.

Sono state considerate tre varianti di ablation, confrontate con la configurazione baseline.

Nella variante `no_image` i valori dei pixel vengono azzerati, così che il modello non riceva informazione visiva utile pur conservando testo OCR e coordinate.

Nella variante `no_text` i token testuali vengono sostituiti dal token sconosciuto del tokenizer, preservando i token speciali, una perturbazione che riduce drasticamente l'informazione lessicale disponibile.

Nella variante `no_layout` tutte le bounding box vengono sostituite con coordinate nulle, lasciando al modello testo e immagine ma sottraendogli l'informazione spaziale esplicita associata ai token OCR.

In tutti i casi le etichette di riferimento restano inalterate e vengono utilizzate solo per il calcolo delle metriche.

6.3.2 Risultati dello studio di ablation

I risultati dello studio di ablation sono riportati nella tabella seguente.

Variante	Descrizione	F1	Δ F1
<code>full</code>	Input completo (testo, layout, immagine)	0,8529	—
<code>no_image</code>	Componente visiva annullata	0,8436	−1,09%
<code>no_layout</code>	Bounding box azzerate	0,1901	−77,71%
<code>no_text</code>	Token testuali rimossi	0,0370	−95,67%

Tabella 6.2: Studio di ablazione di LayoutLMv3: impatto della rimozione di ciascuna modalità.

I risultati mostrano una gerarchia di dipendenza netta. La rimozione della componente visiva produce un calo marginale della F_1 , pari all'1,09%. Per il task considerato, quindi, il contenuto puramente grafico dell'immagine ha un ruolo secondario rispetto alle altre modalità.

La rimozione del testo ha un impatto drastico. La F_1 passa da 0,8529 a 0,0370, con un calo superiore al 95,67%, segno che il contenuto testuale OCR è la principale sorgente informativa per la predizione delle entità. Senza testo, il modello non riesce quasi più a individuare correttamente gli span rilevanti.

Anche la rimozione del layout causa un forte decadimento. La F_1 scende a 0,1901, con un calo del 77,71%. Il risultato evidenzia il peso delle coordinate spaziali nel task di Key Information Extraction. Pur conservando l'accesso al testo, il modello perde gran parte della capacità di associare i valori ai campi quando viene rimossa la posizione dei token nella pagina.

Nel complesso, l'ablation indica che LayoutLMv3, nel contesto sperimentale considerato, costruisce le predizioni soprattutto sulla combinazione di testo OCR e layout spaziale. La componente immagine contribuisce in misura più limitata, almeno per la tipologia di documenti fiscali analizzati e per i campi target considerati.

6.3.3 Studio di robustezza: degradazioni controllate

A differenza dell'ablation, che rimuove quasi del tutto una modalità, lo studio di robustezza introduce degradazioni graduali e controllate, con intensità crescente. L'obiettivo è simulare condizioni non ideali e misurare quanto le prestazioni del modello restino stabili rispetto a perturbazioni realistiche degli input.

Anche in questo caso le perturbazioni vengono applicate solo in fase di valutazione. La configurazione `clean`, senza perturbazioni, corrisponde alla baseline con F_1 pari a 0,8529. Le degradazioni sono organizzate in tre gruppi.

Le degradazioni visive agiscono sulla componente immagine.

Il rumore sulle bounding box altera la posizione dei token.

Il mascheramento testuale sostituisce casualmente una percentuale crescente di token.

Il rumore sulle coordinate e il mascheramento dei token sono generati in modo deterministico, in modo da rendere gli esperimenti riproducibili.

Modalità	Degradazione	Calo di F1		
		Mild	Medium	Hard
Immagine	Sfocatura gaussiana (3×3 / 5×5 / 7×7)	0,95%	0,75%	0,26%
Immagine	Rumore gaussiano (σ 0,05 / 0,10 / 0,20)	-0,23%	0,10%	0,46%
Immagine	Basso contrasto (0,75 / 0,50 / 0,25)	-0,20%	-0,29%	-0,56%
Layout	Rumore su bbox (σ 3 / 7 / 15 px)	0,43%	5,38%	35,86%
Testo	Mascheramento token (5% / 10% / 20%)	2,56%	3,30%	5,23%

Tabella 6.3: Robustezza di LayoutLMv3 alla degradazione progressiva di ciascuna modalità.

Le tre degradazioni applicate alla componente visiva producono effetti molto contenuti. Anche al livello di severità massima, le variazioni di F_1 restano in genere inferiori all'1%. In alcuni casi si osservano piccoli cali negativi, cioè lievi aumenti della F_1 rispetto alla baseline. Questi valori non vanno letti come un reale miglioramento causato dalla degradazione, ma come oscillazioni trascurabili della metrica.

Il rumore sulle bounding box mostra un comportamento diverso. A livello **mild** il calo è quasi nullo, a livello **medium** la F_1 diminuisce in modo più evidente, e con perturbazione **hard** il calo arriva al 35,86%. Il dato suggerisce l'esistenza di una soglia di tolleranza spaziale. Piccoli spostamenti delle bounding box vengono assorbiti senza conseguenze rilevanti, mentre alterazioni più ampie modificano le relazioni spaziali tra i token e degradano in modo sostanziale la capacità di estrazione.

Il mascheramento dei token testuali produce un degrado monotono e graduale. Il calo passa dal 2,56% con mascheramento al 5%, al 3,30% con mascheramento al 10%, fino al 5,23% con mascheramento al 20%. Il confronto con l'ablation completa del testo, che provocava un collasso quasi totale della F_1 , permette di chiarire meglio il peso della componente testuale nel modello.

Il modello tollera una perdita parziale dell'informazione lessicale, sfruttando probabilmente il contesto residuo e il layout, ma non mantiene buone prestazioni quando il testo viene rimosso quasi del tutto.

6.3.4 Sintesi e limiti dell'analisi

L'ablation e lo studio di robustezza convergono verso una lettura coerente: il modello è molto sensibile alla qualità del testo OCR e dell'informazione posizionale, mentre risulta meno dipendente dalla componente visiva pura.

Il risultato può dipendere dalla natura dei documenti fiscali analizzati. In molti casi l'informazione rilevante è già catturata dal testo OCR e dalle bounding box, perché codici fiscali, date e importi sono valori fortemente testuali, mentre la posizione nella pagina aiuta a disambiguare campi simili o valori numerici ripetuti.

Sul piano applicativo, i risultati suggeriscono che il miglioramento della pipeline non dovrebbe concentrarsi solo sul modello, ma anche sulla qualità degli input strutturati forniti a LayoutLMv3. Un OCR più accurato, bounding box coerenti e strategie di preprocessing robuste possono incidere in modo significativo sull'affidabilità dell'estrazione.

L'analisi presenta anche un limite metodologico. Le perturbazioni descritte agiscono sugli input già processati dal modello, cioè su `pixel_values`, `bbox` e `input_ids`, non sulle immagini originali prima della pipeline di OCR e tokenizzazione. L'esperimento misura quindi la sensibilità di LayoutLMv3 a parità di OCR e preprocessing, ma non cattura l'effetto che una reale degradazione dell'immagine sorgente avrebbe sull'intera pipeline, OCR compreso.

Una possibile estensione futura consisterebbe in un'analisi a livello di pipeline, con le perturbazioni applicate direttamente alle immagini originali prima dell'OCR. In questo modo sarebbe possibile misurare non solo la robustezza del modello, ma anche quella dell'intero processo di estrazione, dal riconoscimento del testo fino alla predizione delle entità.

Capitolo 7

Esperimenti e risultati

In questo capitolo vengono presentati e confrontati i risultati sperimentali di LayoutLMv3 e Qwen2.5-VL nel task di Key Information Extraction, ottenuti sul test set.

La valutazione è field-level, sulla base della correttezza del valore estratto per ciascun campo target. L'utilizzo di una metrica orientata all'esito applicativo consente di confrontare output eterogenei, ricondotti a una rappresentazione comune campo-valore prima del calcolo delle metriche.

Tabella 7.1: Dettagli del test set e configurazione dei confronti per i modelli valutati.

Elemento	Valore
Record del test set	185
Campi target valutati	9
Confronti field-level per modello	1.665
Modelli confrontati	LayoutLMv3, Qwen2.5-VL

Le metriche di valutazione adottate a livello di campo sono precision, recall e F_1 , affiancate dal document exact match come criterio più severo. A queste si aggiunge la lettura degli errori, distinti in missing, hallucination e wrong value. L'analisi estende il confronto oltre i punteggi aggregati, esaminando la distribuzione e la natura degli errori di ciascun approccio.

L'esposizione procede su cinque livelli: metriche aggregate, dettaglio per singolo campo, exact match, analisi delle principali tipologie di errore e lettura qualitativa. I risultati vengono interpretati come confronto tra due paradigmi, e non come valori numerici isolati.

7.1 Risultati complessivi field-level

Il primo livello di analisi riguarda le metriche aggregate sull'intero test set. La base di confronto è identica per i due modelli: 185 record e 9 campi target, da cui derivano 1.665 confronti field-level ciascuno. I valori complessivi sono raccolti nella tabella.

7.1. RISULTATI COMPLESSIVI FIELD-LEVEL

Tabella 7.2: Metriche di performance aggregate e di confronto field-level.

Modello	TP	FP	FN	Precision	Recall	Micro F1	Macro F1 per campo	Document exact match
LayoutLMv3	319	135	324	0,7026	0,4961	0,5816	0,4339	0,0973
Qwen2.5-VL	299	96	344	0,7570	0,4650	0,5761	0,4008	0,0865

I valori di micro- F_1 field-level riportati in questa sezione non sono direttamente confrontabili con le F_1 a livello di entità BIO calcolate con segeval e discusse nella Sezione 6.3, in particolare con la baseline di test pari a 0,8529 impiegata nello studio di ablation. Le due misure rispondono a domande diverse. La F1 BIO valuta la corretta etichettatura degli span token-level, mentre la micro-F1 field-level richiede che il valore finale di ciascun campo coincida esattamente con la ground truth dopo la normalizzazione. Il criterio field-level è più severo, poiché un singolo carattere errato o uno span ricostruito in modo incompleto invalidano l'intero campo, e produce quindi valori più bassi senza che ciò indichi un peggioramento del modello.

La metrica micro- F_1 evidenzia prestazioni aggregate comparabili tra i due modelli. LayoutLMv3 ottiene 0,581586, mentre Qwen2.5-VL ottiene 0,576108. La differenza è ridotta, ma non deriva dallo stesso comportamento interno. I due modelli raggiungono valori simili di F_1 attraverso un diverso bilanciamento tra precision e recall.

La precision è più alta per Qwen2.5-VL, che raggiunge 0,7570 contro 0,7026 di LayoutLMv3. Tale evidenza indica che le predizioni effettuate dal modello generativo risultano corrette con una frequenza maggiore. Il rapporto si inverte sulla recall, dove LayoutLMv3 ottiene 0,4961 contro 0,4650 di Qwen2.5-VL. Il modello OCR-based rileva quindi una porzione leggermente più ampia dei valori annotati, ma con una precision inferiore, segnale di una maggiore incidenza relativa di falsi positivi. Questo diverso bilanciamento descrive due comportamenti distinti dei modelli. LayoutLMv3 tende a intercettare una porzione leggermente più ampia dei valori annotati, mentre Qwen2.5-VL mostra un comportamento più selettivo, generando meno valori ma con una precision più elevata.

Il vantaggio di LayoutLMv3 ricompare anche nella macro- F_1 per campo, con 0,4339 contro 0,4008. Questa metrica assegna lo stesso peso a ogni campo, indipendentemente dalla sua frequenza, e mostra quindi un rendimento medio migliore di LayoutLMv3 sull'insieme delle entità considerate. Sebbene Qwen2.5-VL ottenga una micro- F_1 vicina, presenta una distribuzione delle prestazioni più sbilanciata tra i campi, come emerge dall'analisi entità per entità nella sezione successiva.

Una precisazione riguarda il calcolo di questa media. La macro- F_1 considera solo i campi con support positivo nel test set, di cui calcola la media aritmetica dei valori di F_1 . Il campo titolo edilizio appartiene allo schema di estrazione ma non compare nella ground truth del test set, quindi ha support nullo. Resta perciò marcato come non valutabile nella tabella per campo e non entra nella media, evitando che

un'entità assente alteri il risultato complessivo. Tra i campi valutati, pagamento tracciato segue il criterio di presence detection booleano descritto precedentemente, e i conteggi che ne derivano rientrano negli aggregati riportati qui.

Il document exact match mostra un andamento coerente con le metriche precedenti. Poiché questa metrica richiede che tutti i campi del documento siano corretti simultaneamente, il risultato viene discusso nella sezione dedicata.

In sintesi, il quadro aggregato mostra due modelli vicini sulla micro- F_1 , ma diversi nel profilo di errore. Le sezioni seguenti approfondiscono questa differenza da due prospettive complementari, analizzando prima il comportamento campo per campo e poi la distribuzione delle principali tipologie di errore.

7.2 Confronto field-level per campo

Le metriche aggregate mostrano prestazioni complessivamente vicine tra LayoutLMv3 e Qwen2.5-VL, ma non consentono di individuare quali campi siano estratti correttamente e quali presentino maggiori difficoltà. Per questo motivo, il confronto viene esteso al livello field-level, analizzando separatamente le prestazioni ottenute su ciascuna entità. Questa distinzione è rilevante in un task di Key Information Extraction, perché le entità target differiscono per frequenza, struttura e difficoltà: alcune sono brevi e ricorrenti, come date e codici fiscali, mentre altre sono rare, ambigue o legate al contesto del documento.

Il support indica i documenti del test set in cui il campo è presente nella ground truth field-level e può differire dai conteggi a livello di entità BIO del Capitolo 3.

Tabella 7.3: Dettaglio delle metriche di Precision, Recall e F1-score (P/R/F1) disaggregate per singolo campo target.

Campo	Support	LayoutLMv3 P/R/F1	Qwen2.5-VL P/R/F1
anno	9	0,000 / 0,000 / 0,000	0,000 / 0,000 / 0,000
importo	10	0,000 / 0,000 / 0,000	0,000 / 0,000 / 0,000
titoloedilizio	0	n.d.	n.d.
pagamentotracciato	67	0,704 / 0,851 / 0,770	0,864 / 0,761 / 0,810
codicefiscale	181	0,504 / 0,337 / 0,404	0,509 / 0,309 / 0,385
totaledetraibile	69	0,773 / 0,246 / 0,374	1,000 / 0,101 / 0,184
nomecognome	10	0,833 / 0,500 / 0,625	1,000 / 0,300 / 0,462
totaledocumento	116	0,770 / 0,405 / 0,531	0,912 / 0,448 / 0,601
datadocumento	181	0,810 / 0,729 / 0,767	0,818 / 0,718 / 0,765

L'analisi riga per riga della tabella mostra differenze marcate tra i campi.

Tre entità risultano non valutabili o prive di estrazioni corrette. Il campo titolo edilizio non figura nella ground truth del test set, quindi precision, recall e F_1 non sono definibili. I campi anno e importo, presenti rispettivamente in 9 e 10 documenti, ottengono una F_1 nulla per entrambi i modelli, poiché nessuno dei due produce una predizione corretta. La rarità di queste entità spiega il risultato, dato

che pochi esempi non bastano a fissare un pattern stabile, né per un classificatore token-level OCR-based né per un generatore image-to-JSON.

Una precisazione precede la lettura del campo pagamento tracciato: come definito nella Sezione 6.2.1, è valutato con il criterio di presence detection booleano e non come entità testuale, quindi il confronto avviene tra presenza attesa e presenza predetta.

Il campo pagamento tracciato risulta uno dei campi meglio gestiti. È presente nella ground truth di 67 documenti su 185 e raggiunge valori di F_1 elevati per entrambi i modelli: 0,770 per LayoutLMv3 e 0,810 per Qwen2.5-VL.

I profili restano coerenti con il comportamento generale. LayoutLMv3 privilegia la copertura, con recall pari a 0,851 e precision pari a 0,704. Questa tendenza si riflette anche nella distribuzione degli errori, con 24 hallucination e 10 missing. Qwen2.5-VL risulta invece più selettivo, con precision pari a 0,864 e recall pari a 0,761, accompagnate da 8 hallucination e 16 missing.

Il risultato dipende dal criterio di valutazione adottato, perché verificare la sola presenza di un pagamento tracciato è più robusto del confronto esatto della modalità di pagamento, che può essere espressa con formulazioni diverse ma semanticamente equivalenti.

Il campo più affidabile tra le entità testuali è la data del documento. Entrambi i modelli superano una F_1 di 0,75 e ottengono risultati quasi equivalenti, con 0,767 per LayoutLMv3 e 0,765 per Qwen2.5-VL. Anche precision e recall risultano molto vicine. La data del documento si comporta come un campo regolare, perché ricorre con frequenza, segue un formato abbastanza standardizzato e tende a occupare posizioni stabili nella pagina. Queste caratteristiche agevolano sia un modello fondato su testo e layout sia un generativo capace di riconoscere pattern testuali ricorrenti nell'immagine.

Un confronto rilevante riguarda i due campi monetari, totale documento e totale detraibile, che mostrano comportamenti opposti. Per totale documento, Qwen2.5-VL risulta più efficace di LayoutLMv3, con F_1 pari a 0,601 contro 0,531, precision pari a 0,912 e recall leggermente superiore. Il totale del documento è un'informazione visivamente e semanticamente rilevante, spesso associata a diciture esplicite come "totale".

Sul totale detraibile il rapporto tra i due modelli si inverte. LayoutLMv3 ottiene una F_1 pari a 0,374, mentre Qwen2.5-VL si ferma a 0,184. Il dato più rilevante riguarda Qwen2.5-VL, che raggiunge una precision pari a 1,000 ma una recall di appena 0,101, perché il modello genera solo 7 estrazioni corrette sulle 69 occorrenze attese. LayoutLMv3 mostra invece un comportamento meno conservativo, con precision pari a 0,773 e recall pari a 0,246, tentando l'estrazione con maggiore frequenza.

Questo risultato suggerisce che Qwen2.5-VL incontri maggiori difficoltà nel distinguere il totale del documento dal totale detraibile. Il primo è di norma più evidente e associato a indicazioni esplicite, mentre il secondo può essere marcato in modo meno diretto e dipendere maggiormente dal contesto fiscale del documento.

Lo stesso profilo si ritrova su nome e cognome. Qwen2.5-VL combina una

precision pari a 1,000 e una recall pari a 0,300, quindi poche predizioni ma corrette, mentre LayoutLMv3 estrae con maggiore frequenza e raggiunge una recall di 0,500 e una F_1 di 0,625. Il support del campo, fermo a 10 occorrenze, impone però cautela, perché su numeri così piccoli pochi valori corretti o errati spostano in modo vistoso le metriche.

Il campo codice fiscale mostra un risultato intermedio. I valori di F_1 sono vicini, con 0,404 per LayoutLMv3 e 0,385 per Qwen2.5-VL, ma entrambi restano condizionati da un numero elevato di predizioni non esatte. Questo dipende dalla natura del campo, perché il codice fiscale è una stringa alfanumerica lunga e, nella valutazione *strict*, è sufficiente un solo carattere diverso per rendere errata l'intera predizione. Di conseguenza, un errore di OCR, uno span selezionato in modo incompleto o un singolo carattere generato in modo errato vengono penalizzati come mancata corrispondenza esatta.

La somiglianza aggregata tra i due modelli deriva quindi da comportamenti diversi sulle singole entità. LayoutLMv3 estrae più predizioni e ottiene una recall più alta su campi come totale detraibile e nome e cognome, ma questo comporta anche una maggiore incidenza di falsi positivi o valori errati. Qwen2.5-VL mostra un comportamento più selettivo, raggiunge valori di precision elevati su alcune entità, ma produce più falsi negativi, soprattutto quando il campo è poco rilevante o difficile da distinguere dal contesto del documento.

Per questo motivo, il confronto deve considerare anche le tipologie di errore, analizzate nella sezione successiva.

7.3 Exact match a livello di campione

Alla valutazione field-level è utile affiancare un criterio più esigente, indicato come document exact match e calcolato a livello di campione. A differenza delle metriche per campo, questa misura considera corretto un campione solo quando tutti i campi valutati coincidono con la ground truth. Di conseguenza, un singolo campo errato, mancante o predetto in eccesso è sufficiente a invalidare l'intero documento.

Tabella 7.4: Dettaglio dell'exact match a livello di documento e conteggio dei campioni interamente corretti.

Modello	Exact match	Campioni completamente corretti
LayoutLMv3	0,0973	18 / 185
Qwen2.5-VL	0,0865	16 / 185

I valori di document exact match restano bassi per entrambi i modelli. LayoutLMv3 ottiene 0,097297, con 18 campioni interamente corretti su 185, pari a circa il 9,73% del test set. Qwen2.5-VL ottiene 0,086486, con 16 campioni interamente corretti su 185, pari a circa l'8,65%.

Quando il numero di campi da estrarre aumenta, cresce anche la probabilità che

almeno uno risultati mancante, errato o predetto in eccesso. Per questo motivo, il document exact match è una metrica molto rigida.

Il confronto tra i due modelli conferma, a livello di documento, il quadro osservato nelle metriche aggregate. Il margine di LayoutLMv3 è limitato a due campioni e va quindi interpretato con prudenza: il totale dei documenti perfetti resta esiguo per entrambi e non basta, da solo, a fondare un giudizio netto sulla superiorità di un approccio. Il risultato conferma piuttosto quanto sia severo il criterio, e come prestazioni field-level vicine restino vicine anche quando si richiede la correttezza simultanea di tutti i campi.

L'analisi mantiene una precisa rilevanza applicativa. In uno scenario reale un record con tutti i campi corretti è l'esito ideale, ma sono le metriche field-level a indicare quali parti dell'estrazione funzionano e quali vanno migliorate. Il document exact match va inteso perciò come misura complementare a precision, recall e F_1 , non come loro sostituto.

7.4 Analisi degli errori: missing, hallucination e wrong value

Per comprendere il comportamento dei due modelli, non è sufficiente osservare il numero di predizioni corrette; si rende pertanto necessaria una classificazione delle tipologie di errore prodotte.

Riprendendo la tassonomia definita nel protocollo di valutazione, l'analisi distingue gli errori in missing, hallucination e wrong value, così da separare omissioni, predizioni su campi assenti e valori presenti ma non correttamente estratti.

Il campo pagamento tracciato costituisce un caso particolare. Poiché viene valutato in termini di presenza, e non tramite confronto esatto del valore testuale, esso non genera errori di tipo wrong value; di conseguenza, i suoi errori si distribuiscono solo tra missing e hallucination.

Tabella 7.5: Analisi comparativa delle tipologie di errore.

Categoria	LayoutLMv3	Qwen2.5-VL
correct extraction	319 (49,6% dei positivi)	299 (46,5% dei positivi)
missing	221 (34,4% dei positivi)	256 (39,8% dei positivi)
wrong value	103 (16,0% dei positivi)	88 (13,7% dei positivi)
hallucination	32 (3,1% dei negativi)	8 (0,8% dei negativi)
true null	990 (96,9% dei negativi)	1.014 (99,2% dei negativi)

La distribuzione degli errori conferma che valori simili di micro- F_1 derivano da comportamenti diversi. I campi effettivamente presenti nella ground truth sono 643. Su questi valori attesi, LayoutLMv3 produce 319 estrazioni corrette, pari al 49,6% dei positivi, mentre Qwen2.5-VL ne produce 299, pari al 46,5%. Il divario nelle estrazioni corrette è quindi contenuto, con un leggero margine a favore di LayoutLMv3.

Gli errori di tipo missing chiariscono ulteriormente la differenza tra i due modelli. LayoutLMv3 ne registra 221, pari al 34,4% dei campi presenti nella ground truth, mentre Qwen2.5-VL arriva a 256, pari al 39,8%. Questa differenza è coerente con la recall leggermente più alta del modello OCR-based, che recupera una quota maggiore dei valori annotati. Qwen2.5-VL, al contrario, omette più spesso campi effettivamente presenti nella ground truth.

La tendenza di LayoutLMv3 a produrre più predizioni si riflette anche negli errori di tipo wrong value. Il modello ne registra 103, contro gli 88 di Qwen2.5-VL. In questi casi il campo è presente nella ground truth e il modello restituisce un valore, ma la predizione non coincide con quello atteso dopo la normalizzazione. Il wrong value si distingue dalle altre categorie. Non è un'estrazione corretta, perché il valore predetto è errato, e non è un missing, perché una predizione viene comunque prodotta. Resta quindi una categoria a sé, ma nel calcolo di precision e recall pesa due volte. Il valore atteso non recuperato conta come falso negativo, mentre il valore errato prodotto conta come falso positivo. Per questo un wrong value penalizza il modello più di un missing, che incide solo sulla recall.

La discrepanza diventa più marcata sugli errori di tipo hallucination. LayoutLMv3 ne genera 32, mentre Qwen2.5-VL solo 8. Considerando i 1.022 campi assenti nella ground truth, LayoutLMv3 mantiene correttamente il campo vuoto in 990 casi, pari al 96,9% dei negativi, mentre Qwen2.5-VL lo fa in 1.014 casi, pari al 99,2%. Questo conferma la maggiore selettività del modello generativo, che assegna valori a campi assenti con frequenza molto più bassa. Il quadro è quindi coerente con il comportamento già osservato nelle metriche aggregate: LayoutLMv3 privilegia la copertura, mentre Qwen2.5-VL assume un profilo più conservativo.

La cautela di Qwen2.5-VL si lega anche alla formulazione del prompt nella pipeline generativa. Le istruzioni impongono di restituire solo un JSON valido, di non aggiungere spiegazioni, di usare `null` in assenza del valore e di non inventare valori. Attribuire l'intero profilo di errore a queste istruzioni sarebbe arbitrario, ma è plausibile che il prompt spinga il modello a un comportamento conservativo qualora l'evidenza visiva o testuale presenti ambiguità.

Sulla base di tali evidenze, il prompt engineering diventa una leva concreta per regolare il compromesso tra missing e hallucination nei modelli generativi image-to-JSON.

7.4. ERRORI: MISSING, HALLUCINATION, WRONG VALUE

Campo	Supp.	LayoutLMv3				Qwen2.5-VL			
		Miss.	Alluc.	V.err.	Tot.	Miss.	Alluc.	V.err.	Tot.
anno	9	9	0	0	9	9	0	0	9
importo	10	10	0	0	10	10	0	0	10
totale documento	116	57	2	12	71	59	0	5	64
pagamento tracciato	67	10	24	0	34	16	8	0	24
data del documento	181	18	0	31	49	22	0	29	51
codice fiscale	181	62	2	58	122	71	0	54	125
nome e cognome	10	4	0	1	5	7	0	0	7
totale detraibile	69	51	4	1	56	62	0	0	62
titolo edilizio	0	0	0	0	0	0	0	0	0
Totale	643	221	32	103	356	256	8	88	352

Tabella 7.6: Errori per campo e modello.

L’analisi per campo individua quali entità causano il maggior numero di errori, senza ripercorrere il confronto field-level della sezione precedente. La lettura per tipologia di errore aggiunge un livello di analisi diverso, perché mostra da cosa derivano le difficoltà. Alcune entità raccolgono soprattutto missing, segno che il modello non riconosce o non localizza un valore atteso. Altre concentrano wrong value, indizio che un valore viene individuato ma non riprodotto correttamente. Le hallucination restano poche, ma pesano sul piano applicativo, perché introducono valori privi di riscontro nella ground truth.

I campi anno e importo producono esclusivamente errori di tipo missing per entrambi i modelli. Pur comparendo rispettivamente 9 e 10 volte nel test set, non danno luogo ad alcuna predizione corretta. Il problema non riguarda quindi la generazione di valori sbagliati, ma il mancato recupero del campo, in linea con la loro rarità e con la difficoltà di apprendere un pattern stabile da un numero così ridotto di esempi.

Il campo pagamento tracciato concentra quasi tutte le hallucination osservate nell’esperimento. LayoutLMv3 produce 24 hallucination su pagamento tracciato e 10 missing, mentre Qwen2.5-VL registra 8 hallucination e 16 missing. Il dato è coerente con il comportamento generale dei due modelli. Tutte le 8 hallucination di Qwen2.5-VL ricadono su questo campo, così come 24 delle 32 hallucination complessive di LayoutLMv3. Riconoscere la presenza di un pagamento tracciato risulta quindi il caso che più spesso porta i due modelli a segnalare un’informazione non supportata dalla ground truth. Va però anticipato che una parte di queste hallucination su pagamento tracciato non riflette un errore dei modelli, ma deriva da modalità di pagamento presenti nel documento e non annotate nella ground truth. Il fenomeno, che incide proprio su questo campo, è esaminato nella Sezione 7.5 e ripreso nella Sezione 8.2.

Nel codice fiscale prevalgono errori di tipo wrong value, con valori simili per i due modelli: 58 casi per LayoutLMv3 e 54 per Qwen2.5-VL, su 181 occorrenze. Come osservato nella Sezione 7.2, la valutazione *strict* penalizza come wrong value

anche una differenza di un solo carattere, riconducendo a questa categoria errori di OCR e span ricostruiti in modo incompleto.

Il confronto non può basarsi esclusivamente sulla micro- F_1 . Sebbene la prestazione aggregata dei due modelli sia vicina, le tipologie di errore evidenziano rischi applicativi diversi. LayoutLMv3 recupera una quota leggermente maggiore di valori presenti nella ground truth, ma produce più errori tra wrong value e hallucination. Qwen2.5-VL riduce invece le hallucination e gli errori di contenuto, ma lascia non predetto un numero maggiore di campi effettivamente presenti.

Questa distinzione è rilevante per la lettura dei risultati. Dove conviene evitare valori inesistenti o potenzialmente fuorvianti, il comportamento più selettivo di Qwen2.5-VL può rappresentare un vantaggio. Quando invece l'obiettivo è recuperare il maggior numero possibile di campi, accettando una revisione manuale più ampia, LayoutLMv3 risulta più adatto. La preferenza tra i due approcci dipende quindi dal tipo di errore considerato più costoso nel caso applicativo, e non dalla sola metrica aggregata.

7.5 Analisi qualitativa

Le metriche quantitative riassumono il confronto tra LayoutLMv3 e Qwen2.5-VL, ma non sono sufficienti a descrivere completamente il comportamento dei due modelli. All'analisi numerica si affianca quindi una breve analisi qualitativa delle predizioni, mirata a osservare esempi concreti di successo e di errore.

In un task di Key Information Extraction, questo approfondimento è utile perché errori diversi possono produrre lo stesso effetto sulle metriche. Un campo mancante, un valore estratto solo parzialmente, una data in formato non riconosciuto o un valore attribuito al campo sbagliato rientrano tutti tra le predizioni errate, ma segnalano problemi distinti a livello di modello e di pipeline. Osservare direttamente alcuni casi permette quindi di collegare le metriche aggregate al comportamento effettivo dei due sistemi.

7.5. ANALISI QUALITATIVA

Tabella 7.7: Casi di studio selezionati per l’analisi qualitativa.

Caso	Descrizione	Obiettivo dell’esempio
Caso 1	Entrambi i modelli <i>correct extraction</i>	Mostrare un esempio in cui la normalizzazione e la valutazione field-level funzionano correttamente
Caso 2	LayoutLMv3 <i>correct extraction</i> , Qwen2.5-VL <i>missing</i>	Evidenziare il comportamento più conservativo di Qwen2.5-VL
Caso 3	Qwen2.5-VL <i>correct extraction</i> , LayoutLMv3 <i>wrong value</i>	Mostrare un caso in cui l’approccio generativo produce un valore più adatto
Caso 4	Entrambi i modelli <i>missing</i> o errati	Evidenziare un campo o un campione difficile per entrambi gli approcci
Caso 5	<i>hallucination</i>	Mostrare un caso in cui un modello predice un valore per un campo assente

Campo	Atteso	LayoutLMv3		Qwen2.5-VL	
		Predizione	Esito	Predizione	Esito
data del documento	2023-11-28	2023-11-28	corretto	2023-11-28	corretto
nome e cognome	[nome redatto]	[nome redatto]	corretto	—	missing
data del documento	2023-05-26	2023-09-26	wrong value	2023-05-26	corretto
importo	3.000,00	—	missing	—	missing
pagamento tracciato	assente	presente	hallucination	assente	corretto

Tabella 7.8: Casi rappresentativi del confronto tra i due modelli a livello di singolo campo.

Il primo gruppo comprende i campioni in cui entrambi i modelli estraggono correttamente gli stessi valori. Quando i valori risultano equivalenti dopo la normalizzazione, le predizioni vengono considerate corrette anche se il testo originale non coincide carattere per carattere.

Il secondo gruppo comprende i casi in cui LayoutLMv3 produce una predizione corretta e Qwen2.5-VL restituisce un valore `null`. Questo comportamento è coerente con l’analisi degli errori, nella quale Qwen2.5-VL presenta un numero maggiore di missing e tende a omettere più spesso valori presenti nella ground truth. Tali campioni evidenziano il limite di un comportamento molto prudente. Il modello riduce il rischio di generare valori non supportati, ma in alcuni casi non estrae informazioni che il documento riporta in modo chiaro.

Il terzo gruppo comprende i casi in cui Qwen2.5-VL fornisce una predizione corretta, mentre LayoutLMv3 produce un errore di tipo *wrong value*. Sono casi che mostrano il possibile vantaggio di un modello generativo multimodale nella comprensione complessiva del documento. La differenza di architettura non garantisce prestazioni migliori in generale, ma può essere vantaggiosa quando la classificazione token-level restituisce uno span incompleto o associato a un valore non corretto.

Il quarto gruppo comprende i casi in cui entrambi i modelli falliscono sullo stesso campo. Questi esempi sono i più informativi, perché segnalano una difficoltà che non dipende da una singola architettura ma dalla natura del dato, dalla qualità del documento, dalla formulazione del campo o dalla sua scarsità nel dataset. Quando un campo risulta sistematicamente difficile per entrambi i modelli, la causa va cercata in fattori più generali, come l'ambiguità semantica, la bassa frequenza nel training set o l'assenza di una posizione ricorrente nel layout.

Il quinto gruppo comprende i casi in cui almeno uno dei due modelli produce una *hallucination*, cioè assegna un valore a un campo assente nella ground truth. Questi esempi sono utili perché mostrano un tipo di errore diverso dal *missing* e dal *wrong value*. In un'applicazione di Key Information Extraction questo errore è particolarmente rilevante, perché corrisponde a valori che la ground truth non contiene. In un sistema di estrazione automatica un errore di questo tipo può risultare più insidioso di un'omissione. Un campo vuoto segnala in modo esplicito l'assenza di una predizione, mentre un valore inventato può essere scambiato per informazione valida.

Il campo pagamento tracciato è uno dei più significativi nell'analisi qualitativa, poiché nelle sezioni precedenti è emerso come caso critico per entrambi i modelli. La difficoltà riguarda soprattutto la gestione della presenza del campo: nei casi di *missing* il pagamento tracciato è presente nella ground truth ma non viene riconosciuto, mentre nei casi di *hallucination* il modello segnala il campo come presente senza riscontro nella ground truth.

Tabella 7.9: Confronto tra valore atteso e predizioni dei due modelli su campioni in cui il campo *pagamento tracciato* è assente nel ground truth.

Campione	Valore atteso	Predizione LayoutLMv3	Predizione Qwen2.5-VL
...b078_page_1	<i>None</i>	bonifico	bonifico bonifico
...9628_page_1	<i>None</i>	Carta di credito	carta di credito
...a0fb_page_1	<i>None</i>	<i>None</i>	pagopa
...53c8_page_1	<i>None</i>	<i>None</i>	pagopa
...89eb_page_1	<i>None</i>	Pagamento elettronico	pagamento elettronico

Le *hallucination* di Qwen2.5-VL e LayoutLMv3 su pagamento tracciato non sono valori privi di senso sul piano linguistico. Le predizioni corrispondono a modalità di pagamento plausibili, come bonifico, carta di credito, PagoPA o pagamento elettronico. L'errore non sta nella produzione di testo ma nell'assegnare un valore

semanticamente coerente a un campo che, secondo la ground truth, dovrebbe restare vuoto.

Una parte di queste predizioni conteggiate come hallucination non è causata dai modelli, ma deriva da discrepanze nella ground truth. In diversi scontrini la modalità di pagamento è presente nel documento ma non è stata annotata e la corretta segnalazione del campo finisce conteggiata come hallucination, sovrastimando gli errori reali su pagamento tracciato. Questo fenomeno costituisce un caso di rumore nelle annotazioni, trattato in dettaglio nella Sezione 8.2.

Il primo esempio mostra anche un artefatto generativo, la duplicazione “bonifico bonifico”. È un errore tipico di un modello autoregressivo. La struttura JSON resta formalmente valida, ma il contenuto del campo rimane comunque errato. Si distingue così dagli errori di LayoutLMv3, che non genera testo libero e si limita ad assegnare etichette BIO a token OCR già presenti nel documento.

Il confronto con LayoutLMv3 chiarisce la differenza. Anche LayoutLMv3 produce hallucination, ma con un meccanismo diverso. Non inventa una stringa da zero, ma assegna un’etichetta come **B-PagamentoTracciato** o **I-PagamentoTracciato** a una sequenza di parole OCR realmente presente nel documento, non annotata come valore del campo nella ground truth. L’esito è simile, perché in entrambi i casi viene proposto un valore dove non dovrebbe essercene uno. La natura dell’errore però cambia. Per LayoutLMv3 è un errore di classificazione token-level; per Qwen2.5-VL, un errore nella generazione del valore strutturato.

Il caso illustra bene la distanza tra i due paradigmi. LayoutLMv3 dipende dalla qualità dell’OCR, dalla segmentazione in token, dalle bounding box e dalla corretta assegnazione delle etichette BIO. Qwen2.5-VL non è vincolato a uno span OCR già estratto e deve decidere direttamente quale valore inserire nel JSON finale. La maggiore flessibilità aiuta quando il valore è chiaramente leggibile nell’immagine, ma diventa un limite quando il campo richiede di distinguere tra valore presente, valore assente e valore soltanto plausibile.

Nel complesso, l’analisi qualitativa rende concreto il profilo di errore già emerso dalle metriche aggregate. A una $\text{micro}F_1$ field-level quasi identica corrispondono modalità di errore diverse, con LayoutLMv3 più propenso a proporre valori e Qwen2.5-VL più conservativo.

Capitolo 8

Conclusioni

8.1 Riepilogo del progetto

Questa tesi affronta il problema della Key Information Extraction da documenti fiscali italiani di natura semi-strutturata, con l'obiettivo di estrarre automaticamente un insieme di campi rilevanti a partire da immagini di documenti. Tale compito appartiene all'ambito della Document AI, in cui il riconoscimento del testo rappresenta solo una fase della pipeline. Le informazioni estratte dal documento devono essere interpretate, associate ai campi corretti e trasformate in dati strutturati, confrontabili e riutilizzabili dalle applicazioni a valle.

Il progetto risponde a un'esigenza interna di TaxWiz, una piattaforma orientata alla digitalizzazione e alla categorizzazione di documenti fiscali per la dichiarazione dei redditi e le detrazioni.

Il dataset comprende 1.919 pagine, costituite in prevalenza da scontrini e fatture, con una maggioranza di documenti a pagina singola e una quota minore di documenti multipagina. La ripartizione in training, validation e test set è stata condotta a livello di documento logico, così che pagine appartenenti allo stesso documento non potessero finire in split diversi. Questa scelta riduce il rischio di data leakage e rende più affidabile la valutazione sperimentale.

I dati presentano un forte sbilanciamento tra le classi. La classe *O* domina la distribuzione, rappresentando il 97,87% dei token, mentre le entità informative risultano molto più rare. Questo squilibrio ha influenzato sia la formulazione del problema sia le strategie di addestramento, rendendo poco significativa una valutazione basata sulla sola accuracy e motivando l'uso di metriche e accorgimenti più adatti alla rarità di alcuni campi target.

La costruzione della pipeline è partita dalla preparazione dei dati. Per ciascuna pagina sono stati utilizzati output OCR, bounding box e annotazioni dei campi target. Le annotazioni, prodotte tramite un'interfaccia dedicata e collegate ai blocchi estratti da Amazon Textract, sono state poi convertite nei formati richiesti dai due modelli.

Nel caso di LayoutLMv3 le informazioni annotate sono state rappresentate mediante uno schema BIO con 17 classi, riferite a 9 tipologie di entità fiscalmente rilevanti,

fra cui data del documento, codice fiscale, importi e indicazione sul pagamento tracciato. Nel caso di Qwen2.5-VL gli stessi campi sono stati trattati come target JSON, prodotti a partire dall'immagine del documento e da un prompt testuale.

Il confronto sperimentale ha testato due paradigmi distinti. LayoutLMv3 rappresenta un approccio specializzato per i documenti, OCR-based e discriminativo, nel quale il task viene formulato come token classification. Il modello elabora il testo OCR, le coordinate spaziali e il segnale visivo della pagina per generare una sequenza di etichette BIO, successivamente ricondotta a coppie campo-valore.

Per l'ottimizzazione di questo modulo sono stati introdotti accorgimenti quali Focal Loss, pesi di classe, Layer-wise Learning Rate Decay e una ricerca degli iperparametri basata su Optuna.

Qwen2.5-VL utilizza un approccio generativo tipico dei Vision-Language Models. Il documento viene passato come immagine e il modello genera direttamente una risposta strutturata in JSON. Il fine-tuning è stato realizzato con QLoRA sulla variante `Qwen2.5-VL-7B-Instruct`, utilizzando un prompt istruzionale statico e una configurazione in cui la loss è calcolata solo sui token della risposta generata. La ricerca degli iperparametri è stata effettuata tramite grid search manuale.

Lo studio ha richiesto lo sviluppo di una pipeline sperimentale end-to-end. La procedura complessiva comprende la preparazione del dataset, il fine-tuning, la generazione delle predizioni sul sottoinsieme di test, la conversione degli output in una rappresentazione unificata, la normalizzazione dei valori e la valutazione finale. Questo passaggio è necessario per confrontare i due approcci, dato che LayoutLMv3 restituisce etichette "token-level" mentre Qwen2.5-VL genera testo strutturato. La conversione in un formato omogeneo ha permesso di applicare le stesse metriche di valutazione.

Il confronto principale è stato affiancato da uno studio di ablation e robustezza, limitato a LayoutLMv3, con l'obiettivo di misurare il contributo delle singole modalità informative e la sensibilità del modello ad alterazioni controllate degli input.

8.2 Sintesi e discussione del confronto tra LayoutLMv3 e Qwen2.5-VL

Il confronto tra LayoutLMv3 e Qwen2.5-VL non ha come obiettivo individuare un modello superiore in assoluto, bensì analizzare due differenti formulazioni della Key Information Extraction, evidenziando per ciascuna vantaggi, limiti e trade-off specifici. La valutazione mira pertanto a caratterizzare due paradigmi metodologicamente distinti applicati allo stesso task.

LayoutLMv3 è un approccio specializzato per la Document AI. La sua architettura integra testo, layout e segnale visivo, e si adatta bene ai documenti in cui la posizione spaziale degli elementi è rilevante. Il modello opera su una rappresentazione esplicita fatta di parole OCR, bounding box e patch d'immagine, e definisce il task come classificazione token-level. La natura discriminativa rende le predizioni

verificabili e interpretabili. Ogni valore estratto si riconduce a una sequenza di token OCR, consentendo di individuare quali parole siano state associate a una determinata entità e ricostruire lo span predetto.

Questa proprietà avvicina LayoutLMv3 alle pipeline tradizionali di estrazione strutturata e consente un'analisi granulare del comportamento del modello, come evidenziato nello studio di ablation e robustezza trattato nella Sezione 6.3.

Questa specializzazione comporta maggiore rigidità. Il modello richiede un motore OCR esterno, bounding box accurate, annotazioni convertite in etichette BIO e una gestione attenta dello sbilanciamento tra la classe *O* e le classi informative. Nel progetto, questo aspetto è stato affrontato mediante l'implementazione di Focal Loss, pesi di classe e strategie di ottimizzazione dedicate. Rispetto a un modello generativo di dimensioni molto maggiori, il suo addestramento richiede comunque risorse computazionali ridotte e presenta una maggiore stabilità.

Qwen2.5-VL ha una formulazione più diretta: riceve l'immagine del documento e un prompt con la descrizione dei campi target, per poi generare un oggetto JSON. Il processo di Key Information Extraction diventa così una trasformazione image-to-JSON. L'output coincide con il formato strutturato richiesto dall'applicazione, riducendo i passaggi intermedi tra la predizione e la rappresentazione finale dei dati.

La stessa flessibilità generativa introduce tuttavia una classe di problemi poco presente nel paradigma discriminativo. La struttura dell'output non è garantita sul piano strutturale e può contenere testo aggiuntivo, sintassi JSON malformata o valori solo parzialmente corretti, il che rende necessaria una fase di pulizia, parsing e validazione delle predizioni.

Sebbene la letteratura associ ai modelli generativi il rischio di hallucination, intesa come generazione di valori per campi assenti nella ground truth, i dati raccolti in questo lavoro ridimensionano tale aspettativa. Qwen2.5-VL registra 8 casi di hallucination a fronte dei 32 di LayoutLMv3; il suo profilo di errore si orienta verso l'omissione, con 256 campi mancanti a fronte dei 221 del modello discriminativo.

L'analisi dei risultati attribuisce tale comportamento alle differenti configurazioni di addestramento. Dopo il fine-tuning con QLoRA e con il prompt istruzionale statico, Qwen2.5-VL tende a restituire un valore `null` quando l'evidenza è incerta. Questa impostazione aumenta la precision a 0,7570 e contiene le hallucination, ma riduce la recall a 0,4650 e aumenta i missing. LayoutLMv3 presenta una dinamica opposta per effetto delle funzioni di loss adottate. La Focal Loss riduce l'impatto degli esempi facili, in larga parte token appartenenti alla classe *O*, mentre i pesi di classe aumentano il contributo delle classi informative più rare. La combinazione di questi fattori orienta il modello a etichettare un maggior numero di entità, con una recall più alta, pari a 0,4961, e una precision più bassa, 0,7026, con un conseguente aumento di hallucination e di errori di tipo wrong value, pari a 103 per LayoutLMv3 contro gli 88 di Qwen2.5-VL.

Il rischio di hallucination, atteso come tratto distintivo del paradigma generativo, non si è quindi manifestato come modalità di errore dominante. Per Qwen2.5-VL la debolezza principale è stata l'omissione, non l'invenzione.

Sul piano aggregato, i due modelli raggiungono un micro- F_1 field-level molto vicino, 0,581586 per LayoutLMv3 e 0,576108 per Qwen2.5-VL, con uno scarto di circa 0,0055. La macro- F_1 , che assegna lo stesso peso a ogni entità indipendentemente dalla frequenza, evidenzia un leggero vantaggio per LayoutLMv3, 0,4339 contro 0,4008, e indica un comportamento medio leggermente più equilibrato tra i campi. Il document exact match resta basso per entrambi, 18 campioni interamente corretti su 185 per LayoutLMv3, pari al 9,73%, e 16 su 185 per Qwen2.5-VL, pari all'8,65%: l'omissione o l'errata identificazione anche di un singolo campo compromette infatti l'accuratezza dell'intero record. La differenza principale tra i due paradigmi non riguarda quindi il livello complessivo delle prestazioni, ma il modo in cui distribuiscono le tipologie di errore.

L'analisi per campo, tabella 7.3, mostra come due metriche aggregate quasi identiche si scompongano in comportamenti molto diversi a livello di singolo campo. I campi più rari, anno e importo, non vengono estratti correttamente da nessuno dei due modelli.

L'entità testuale più affidabile si dimostra essere "Data del documento" con una F_1 superiore a 0,75 per entrambi.

I due campi monetari mostrano andamenti opposti: Qwen2.5-VL è più efficace su "totale documento" mentre LayoutLMv3 ottiene risultati migliori su "totale detraibile".

L'entità "pagamento tracciato" ottiene la F_1 più alta in assoluto, 0,770 per LayoutLMv3 e 0,810 per Qwen2.5-VL, ma richiede un'interpretazione distinta dalle altre poiché la valutazione si basa sulla presenza e non sull'estrazione di un valore testuale.

Il profilo completo degli errori conferma quindi la lettura: le prestazioni aggregate simili nascondono profili di rischio diversi, con LayoutLMv3 più propenso a proporre valori e Qwen2.5-VL più conservativo.

Questi risultati devono essere interpretati considerando i limiti della ground truth, già evidenziati nell'analisi qualitativa (Sezione 7.5). La valutazione field-level confronta ogni predizione normalizzata con un'unica annotazione di riferimento, classificando come errore qualunque discrepanza rispetto a essa. Quando l'annotazione originale risulti incompleta, presenti un formato diverso da quello generato dal modello o si riferisca a uno span alternativo, una predizione corretta viene erroneamente considerata come wrong value o hallucination.

Questo fenomeno non costituisce un'anomalia isolata, bensì un limite sistematico del protocollo di valutazione, determinato dal confronto rigido con un'unica annotazione di riferimento anziché con un insieme di risposte ammissibili. Il campo "pagamento tracciato", analizzato nella Sezione 7.5, rappresenta l'esempio più evidente: la presenza della modalità di pagamento nel documento, se non annotata nella ground truth, comporta la classificazione della corretta rilevazione da parte dei modelli come una hallucination.

Tale effetto comporta tre implicazioni principali per l'interpretazione dei risultati. Essendo entrambi i modelli valutati sulla stessa ground truth, la direzione del confronto comparativo rimane valida. L'impatto si concentra sui conteggi di

hallucination e wrong value, influenzando di riflesso sulle metriche di precision, recall e F_1 , i cui valori devono pertanto essere interpretati come una stima conservativa delle prestazioni reali. Infine, il reale impatto di questo fenomeno risulta complesso da quantificare senza una riannotazione sistematica del test set, un'operazione che esula dagli obiettivi del presente lavoro ma che rappresenterebbe il presupposto fondamentale per una valutazione più accurata. Il confronto tra i due paradigmi mantiene intatta la sua validità metodologica, essendo entrambe le architetture esposte al medesimo rumore di annotazione.

Nel complesso, lo studio descrive quindi due strategie distinte e non intercambiabili. LayoutLMv3 mostra una tendenza a identificare più entità, a fronte di più falsi positivi, inclusi casi di hallucination su campi assenti e wrong value su campi presenti.

Qwen2.5-VL privilegia invece la precision rispetto alla recall, mantenendo un comportamento più conservativo ma con una minore tracciabilità dell'output.

Per il modello generativo restano alcuni limiti che non emergono dal conteggio delle hallucination. Nella formulazione generativa pura adottata in questo lavoro, Qwen2.5-VL offre una minore tracciabilità, poiché non associa direttamente il valore testuale generato nel JSON alla specifica posizione geometrica o al token OCR di origine.

La scelta tra le due architetture dipende dal contesto applicativo e dal costo relativo associato alle diverse tipologie di errore. Un sistema che richiede tracciabilità delle predizioni, controllo sullo span estratto e integrazione con pipeline documentali preesistenti trova in LayoutLMv3 una soluzione idonea; in questo caso, la recall più alta del modello risulta competitiva quando la priorità sia massimizzare il recupero delle informazioni per una successiva revisione manuale.

Un sistema che privilegia flessibilità, formulazione diretta del task e produzione immediata di un output strutturato può orientarsi verso un Vision-Language Model come Qwen2.5-VL, a condizione di presidiare parsing, validazione e controllo delle predizioni generate e di accettare un tasso di missing più elevato. In un contesto come TaxWiz, in cui la revisione dei dati estratti è parte del flusso di lavoro, il criterio di scelta non è la sola metrica aggregata ma il tipo di errore più tollerabile nell'applicazione.

8.3 Limiti sperimentali

Un elemento di criticità riguarda la dimensione e la composizione del dataset. Pur essendo reali e rappresentativi del caso d'uso TaxWiz, i dati hanno un volume limitato e sono circoscritti a un dominio applicativo specifico. Questa caratterizzazione può limitare la capacità dei modelli di generalizzare verso documenti fiscali con tipologie, formati o qualità visiva differenti rispetto a quelli inclusi nel training set. Tale limitazione emerge sui campi a bassa frequenza, in cui le entità compaiono in pochissimi esempi, il che ostacola l'apprendimento di pattern stabili e riduce l'affidabilità delle conclusioni sulle prestazioni per campo. Nel test set i campi anno e importo ricorrono rispettivamente 9 e 10 volte. Di conseguenza, le metriche

calcolate su questi campi risultano numericamente instabili: variazioni minime nel numero di predizioni corrette o errate alterano significativamente precision, recall e F_1 , senza permettere giudizi definitivi sulle reali capacità dei modelli rispetto a queste entità.

Un’ulteriore limitazione deriva dalla differenza metodologica strutturale tra i due paradigmi, i quali generano output di natura differente: LayoutLMv3 fornisce etichette BIO vincolate ai token OCR, mentre Qwen2.5-VL genera testo in formato JSON. Il confronto ha quindi richiesto una fase di conversione verso una rappresentazione comune campo-valore, la quale comporta specifiche scelte metodologiche: la ricostruzione degli span BIO, la selezione degli span predetti, il parsing del JSON e la normalizzazione dei valori. Questo aspetto risulta particolarmente rilevante per Qwen2.5-VL, poiché, a differenza di un classificatore token-level vincolato a un insieme finito di etichette, il modello può produrre risposte non pienamente conformi al formato richiesto, come JSON incompleti o non validi, testo aggiuntivo non previsto, valori solo parzialmente corretti o non correttamente formattati. Tali scelte sono necessarie per rendere confrontabili i modelli, ma incidono sulla valutazione finale e rendono la misura delle prestazioni del modello generativo più sensibile alle logiche di post-processing.

A questo si aggiunge un criterio di valutazione *strict*. Anche quando il contenuto generato è semanticamente vicino al valore corretto, la predizione conta come errata se non coincide con la ground truth dopo la normalizzazione. Tale approccio rende la valutazione più rigorosa e coerente, ma penalizza varianti formali parzialmente informative.

Questo fenomeno influisce in particolare su campi come codice fiscale, date e importi, dove una differenza minima di caratteri o formato determina un errore nel calcolo delle metriche. Questo criterio si combina con il rumore di annotazione analizzato nella Sezione 8.2: nel primo caso, una predizione corretta viene classificata come errore a causa dell’etichettatura della ground truth; nel secondo, una predizione prossima al valore atteso viene respinta dal confronto esatto. La combinazione di questi fattori determina un incremento degli errori rilevati, conducendo a una stima conservativa dei valori di recall e F_1 rispetto al comportamento effettivo dei modelli.

Sul piano computazionale, Qwen2.5-VL, essendo un modello generativo da 7 miliardi di parametri, richiede risorse più elevate sia in fase di fine-tuning sia di inferenza. L’uso della quantizzazione a 4 bit e di tecniche come QLoRA consente l’adattamento del modello su infrastrutture hardware più contenute rispetto al fine-tuning completo, ma non elimina il maggior carico associato alle dimensioni del modello e alla generazione autoregressiva. In fase di predizione, la generazione sequenziale dei token della risposta JSON comporta tempi di inferenza superiori e incrementa la dipendenza da risorse GPU adeguate.

Questo aspetto non incide direttamente sull’accuratezza delle predizioni, ma condiziona la fattibilità operativa di un impiego in produzione. LayoutLMv3, pur richiedendo una pipeline OCR e annotazioni più strutturate, presenta costi computazionali inferiori in fase di addestramento ed esecuzione, risultando quindi più

gestibile in scenari con risorse hardware limitate.

L'analisi di ablation e di robustezza è stata condotta solo su LayoutLMv3. Questa scelta dipende dall'architettura del modello, che permette di isolare testo OCR, bounding box e immagine, consentendo di valutare separatamente il contributo delle singole modalità informative.

Una valutazione equivalente su Qwen2.5-VL richiederebbe un disegno sperimentale diverso, basato ad esempio su variazioni del prompt, degradazioni controllate dell'immagine o aggiunta e rimozione di informazioni testuali in input. In un Vision-Language Model generativo l'informazione visiva e quella testuale risultano meno separabili rispetto a un'architettura discriminativa con input più esplicitamente strutturati.

L'assenza di un'ablation su Qwen2.5-VL costituisce un limite metodologico legato al paradigma generativo, più che una semplice mancanza sperimentale. Un confronto simmetrico tra i due modelli richiederebbe esperimenti progettati specificamente per i Vision-Language Models, capaci di misurare l'impatto del prompt, della qualità dell'immagine, della quantità di contesto visivo e dei vincoli imposti sul formato di output.

8.4 Sviluppi futuri

Dai limiti discussi e dalle osservazioni emerse in fase sperimentale si delineano diverse linee di ricerca future.

Un primo ambito di sviluppo concerne l'ampliamento e la revisione del dataset. L'incremento del numero di pagine e documenti annotati, unitamente all'inclusione di una maggiore varietà di formati fiscali, scontrini e fatture, consentirebbe di ottenere metriche più stabili e una stima affidabile della capacità di generalizzazione dei modelli. Risulta inoltre necessario raccogliere ulteriori istanze per i campi meno rappresentati, così da rendere più robuste le conclusioni statistiche relative alle loro prestazioni. Il dataset esteso dovrà continuare a includere documenti acquisiti in condizioni visive eterogenee, come scansioni degradate, immagini prospetticamente distorte, a bassa risoluzione, affette da rumore, illuminazione irregolare o contrasto ridotto.

Un differente orientamento riguarda la gestione dei documenti multipagina. Nel setup attuale, ogni pagina è trattata come record indipendente sia in fase di predizione sia di valutazione. Negli scenari applicativi, tuttavia, le informazioni rilevanti possono distribuirsi su più pagine dello stesso documento fiscale. Un'estensione naturale della pipeline consisterebbe nell'aggregazione delle predizioni delle pagine appartenenti allo stesso documento logico, combinando i valori estratti da pagine distinte al fine di allineare la valutazione ai contesti operativi.

Per i modelli generativi, un possibile miglioramento riguarda la gestione dell'output strutturato. Nel lavoro svolto, le risposte di Qwen2.5-VL sono state sottoposte a una fase di pulizia, parsing e recupero euristico del JSON; tuttavia, l'analisi dei missing mostra che tali errori non derivano da un fallimento di questa procedura. Tutte le predizioni del test set risultano infatti parsabili come JSON

e, nei casi analizzati, il modello restituisce `null` per campi che nella ground truth sono invece valorizzati. I missing vanno quindi interpretati come falsi negativi a livello field-level, cioè come mancata estrazione del valore da parte del modello, non come perdita di informazione durante la conversione dell'output testuale in campi strutturati.

Resta comunque utile sperimentare strategie più robuste, come la validazione automatica dello schema, vincoli più stringenti sul formato e tecniche di structured generation, nelle quali i vincoli sintattici sono imposti durante la generazione anziché corretti a posteriori. Un approccio di questo tipo ridurrebbe gli errori di formato e aiuterebbe a distinguere le anomalie sintattiche dagli errori effettivi di contenuto.

Uno sviluppo ulteriore riguarda l'estensione dell'analisi sperimentale ai Vision-Language Models, sia in termini di analisi di ablation sia in termini di robustezza. Nel caso di Qwen2.5-VL, questo tipo di analisi permetterebbe di valutare in che misura le prestazioni dipendano dalla componente visiva, dal testo eventualmente fornito nel prompt, dalla formulazione dell'istruzione e dalle conoscenze linguistiche acquisite in pre-training.

In parallelo, sarebbe utile estendere lo studio di robustezza applicando perturbazioni analoghe a entrambi i modelli, come rotazioni, rumore, blur, bassa risoluzione, contrasto ridotto e peggioramento della qualità dell'OCR. Un confronto nelle stesse condizioni di acquisizione consentirebbe di misurare direttamente la sensibilità dei due paradigmi, OCR-based e VLM-based, e di valutare meglio la loro tenuta su documenti raccolti in ambienti non controllati.

In ambito architetturale, l'integrazione di differenti modelli di riferimento consentirebbe di inserire LayoutLMv3 e Qwen2.5-VL in un quadro sperimentale esteso. Le possibili inclusioni comprendono una pipeline basata su OCR e regole euristiche e un modello operante esclusivamente sul testo OCR privo di componente visiva. Un confronto esteso sarebbe volto a verificare se le discrepanze rilevate siano ascrivibili ai singoli modelli selezionati o, in alternativa, alle rispettive impostazioni architetturali e metodologiche.

Sarebbe inoltre interessante sperimentare modelli generativi di dimensioni maggiori. VLM più grandi potrebbero offrire una migliore capacità di interpretare i documenti complessi, gestire i campi ambigui e produrre output strutturati più coerenti. Questa ipotesi andrebbe però valutata insieme al maggiore costo computazionale e alla sua reale utilità applicativa rispetto a modelli più leggeri. Nella stessa prospettiva, il confronto tra versioni diverse dello stesso modello permetterebbe di misurare quanto l'aumento di scala incida sulle metriche field-level e sul profilo degli errori.

Un altro aspetto riguarda l'affinamento del fine-tuning. Per LayoutLMv3 si potrebbe aumentare il numero di trial nella ricerca degli iperparametri, esplorare altre strategie di bilanciamento delle classi e valutare configurazioni di ottimizzazione alternative. Per Qwen2.5-VL si potrebbero testare diverse configurazioni LoRA e QLoRA, variando rango, moduli target e `lora.alpha`. Sarebbe utile sperimentare prompt istruzionali differenti, con vincoli più espliciti sui casi di incertezza, sulla gestione dei valori assenti e sulla distinzione tra campi semanticamente simili. Per

i campi più ambigui come pagamento tracciato, un prompt engineering mirato potrebbe chiedere al modello di restituire un valore solo quando la modalità è chiaramente leggibile o esplicitamente indicata nel documento, riducendo le predizioni non supportate. In termini generali converrebbe affiancare ai modelli controlli di validazione dedicati e criteri di annotazione e valutazione più espliciti.

Si inserisce in questo contesto l'introduzione di una forma esplicita di visual grounding per Qwen2.5-VL. Nel setup sperimentale adottato, il modello è configurato per generare direttamente l'output JSON campo-valore a partire dall'immagine, senza richiedere la localizzazione spaziale dei segmenti estratti. Una formulazione alternativa prevede l'associazione a ciascun campo sia del valore testuale sia della bounding box. Tale approccio esplicita il legame tra l'evidenza visiva e la predizione, riducendo l'ambiguità tra valori omologhi nel medesimo documento, a fronte di un incremento della complessità nella pipeline di annotazione e valutazione.

Infine, nell'ambito della data augmentation, si potrebbero valutare strategie capaci di simulare la variabilità dei documenti fiscali. Per i modelli OCR-based questo permetterebbe di analizzare la sensibilità a errori di riconoscimento, bounding box rumorose e alterazioni del layout. Per i modelli generativi sarebbe interessante verificare se l'esposizione a immagini degradate o a formati più vari migliori la capacità di produrre JSON corretti anche in condizioni non ideali.

In conclusione, l'attività di ricerca dimostra che la Key Information Extraction da documenti fiscali può essere eseguita tramite paradigmi architetturali molto diversi. LayoutLMv3 e Qwen2.5-VL raggiungono prestazioni aggregate comparabili, ma differiscono per caratteristiche operative e profilo di errore.

L'evidenza principale dello studio risiede in tale differenziazione: la selezione di un modello per l'estrazione di informazioni da documenti semi-strutturati richiede di valutare fattori quali l'interpretabilità, l'onere computazionale, la robustezza della pipeline, la complessità di integrazione e la tolleranza a specifiche classi d'errore nel contesto applicativo, superando l'analisi basata sulla sola metrica aggregata.

Bibliografia

- [ASY⁺19] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, pages 2623–2631, 2019.
- [BCL⁺25] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibong Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025.
- [DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.
- [DPHZ23] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*, 2023.
- [HCH⁺19] Zheng Huang, Kai Chen, Jianhua He, Xiang Bai, Dimosthenis Karatzas, Shijian Lu, and C. V. Jawahar. ICDAR2019 competition on scanned receipt OCR and information extraction. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1516–1520, 2019.
- [HLC⁺22] Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. LayoutLMv3: Pre-training for document AI with unified text and image masking. In *Proceedings of the 30th ACM International Conference on Multimedia (MM)*, pages 4083–4091, 2022.
- [HSW⁺22] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank

- adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.
- [LGG⁺17] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, 2017.
- [OQZ⁺25] Linke Ouyang, Yuan Qu, Hongbin Zhou, Jiawei Zhu, Rui Zhang, Qunshu Lin, Bin Wang, Zhiyuan Zhao, Man Jiang, Xiaomeng Zhao, Jin Shi, Fan Wu, Pei Chu, Minghao Liu, Zhenxiang Li, Chao Xu, Bo Zhang, Botian Shi, Zhongying Tu, and Conghui He. OmniDocBench: Benchmarking diverse PDF document parsing with comprehensive annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24838–24848, 2025.
- [PSL⁺19] Seunghyun Park, Seung Shin, Bado Lee, Junyeop Lee, Jaeheung Surh, Minjoon Seo, and Hwalsuk Lee. CORD: A consolidated receipt dataset for post-OCR parsing. In *Workshop on Document Intelligence at NeurIPS 2019*, 2019.
- [VSP⁺17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*, pages 5998–6008, 2017.
- [XLC⁺20] Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. LayoutLM: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, pages 1192–1200, 2020.
- [XXL⁺21] Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, Min Zhang, and Lidong Zhou. LayoutLMv2: Multi-modal pre-training for visually-rich document understanding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 2579–2591, 2021.

Ringraziamenti

Desidero ringraziare la mia relatrice, la Prof.ssa Antonella Carbonaro, per la guida e la disponibilità durante lo svolgimento di questo lavoro.

Alla mia famiglia, che mi ha deriso con la stessa costanza con cui mi ha voluto bene. È anche grazie a quelle risate, e a quel non darmi mai tregua, che sono arrivato fin qui.

Ai miei gatti, il Signor Rossini e Cesare Pompeo, i miei compagni di studio più assidui, per avermi sorvegliato dalla scrivania durante le lunghe ore di studio.

Ai miei amici, a cui voglio bene, per esserci stati comunque.