



UNICA

# CHATBOT ERASMUS

## REPORT TECNICO



MARZO-APRILE  
2025

Realizzato da

**ENRICO CASTORI  
CRISTIAN CERNICCHIARO**

Per l'insegnamento di

**WEB ANALYTICS E ANALISI  
TESTUALE**

# TABLE OF CONTENTS

INTRODUZIONE AL PROGETTO	1
RACCOLTA E PREPROCESSING DEI DATI	2
RICONOSCIMENTO DELLE ENTITA'	3
SENTIMENT ANALYSIS	4
TOPIC MODELING	5
RAG	6
CONCLUSIONI	7

# INTRODUZIONE AL PROGETTO

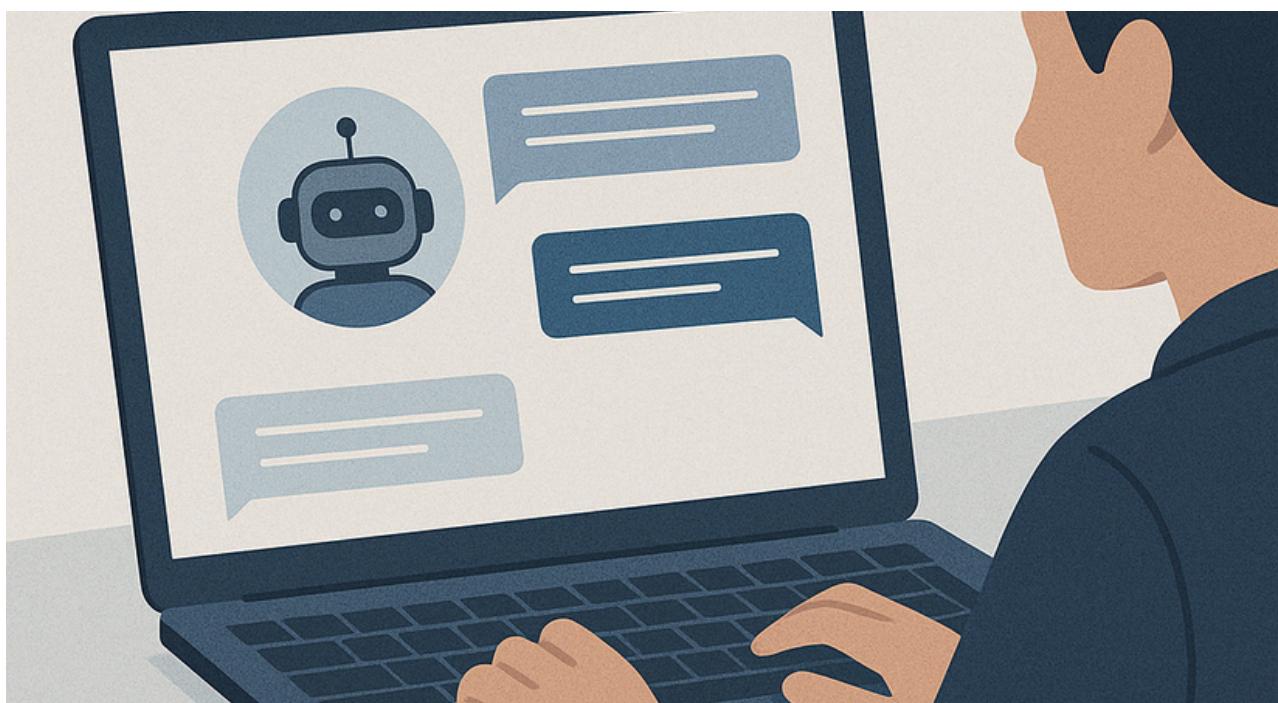
L'esperienza Erasmus rappresenta per molti studenti un'opportunità straordinaria, ma anche un momento di forte incertezza.

Tra dubbi burocratici, difficoltà linguistiche e scambi di informazioni poco strutturate, orientarsi può diventare complicato.

Il nostro progetto nasce con l'obiettivo di facilitare questo percorso, attraverso la costruzione di un chatbot intelligente in grado di fornire risposte concrete e affidabili a domande frequenti, unendo la dimensione informativa a quella esperenziale.

Per raggiungere questo risultato, abbiamo combinato tecniche di analisi del linguaggio naturale (NLP) con modelli avanzati di generazione del linguaggio, utilizzando una base di conoscenza composta da conversazioni autentiche tra studenti e documentazione ufficiale.

L'analisi approfondita delle testimonianze pratiche degli studenti ci ha permesso di progettare un assistente intelligente e capace di affiancare gli studenti con linguaggio chiaro, contesto reale e risposte personalizzate.

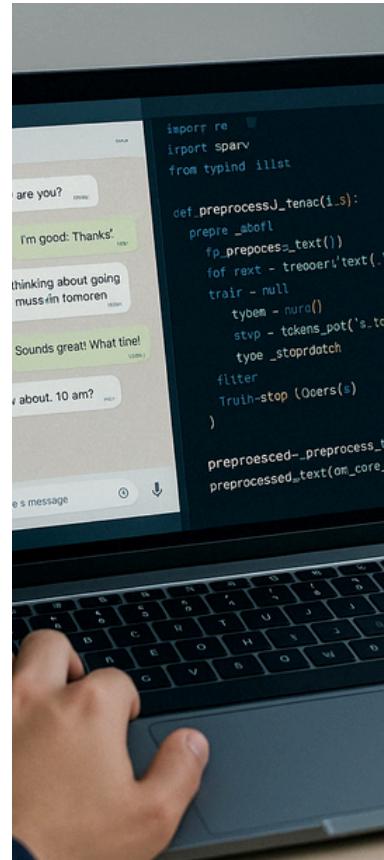


# RACCOLTA E PREPROCESSING DEI DATI

La fase iniziale del progetto ha previsto la raccolta e la pulizia di una conversazione WhatsApp proveniente dal gruppo “**Aiuto Erasmus+ partenze A.A. 2024/25**”, che rappresenta una fonte informale ma ricca di contenuti pratici relativi all’esperienza Erasmus. Il file grezzo conteneva messaggi datati tra marzo 2024 e marzo 2025, inclusi testi, notifiche di sistema (es. “ha usato il link d’invito”, “messaggio eliminato”, etc.), immagini omesse e informazioni meta non rilevanti ai fini dell’analisi.

Per l’estrazione e la normalizzazione dei dati, è stato implementato uno script Python dedicato, `parse_whatsapp_chat()` dal file “clean\_txt.py”, che si è occupato di:

- **Parsing strutturato** dei messaggi tramite espressioni regolari, estraendo data, orario, mittente e contenuto testuale.
- **Rimozione delle notifiche di sistema**, come la crittografia dei messaggi o l’ingresso di nuovi utenti nel gruppo.
- **Pulizia dei caratteri speciali Unicode** indesiderati (\u200e, \u202f) e normalizzazione degli spazi.
- **Filtraggio dei messaggi vuoti**, ripetuti o di sistema, per mantenere nel dataset solo le interazioni effettive tra gli utenti.



Un passaggio fondamentale, che ha accompagnato tutte le fasi successive del progetto, è stato il **merging** dei messaggi. Nel contesto della NLP (Natural Language Processing), il merging è un’operazione che unisce più unità testuali brevi in un’unica unità più estesa e coerente.

Nell’approcciarsi alle diverse fasi di questo progetto, abbiamo quindi unito i messaggi ravvicinati nel tempo per ricostruire blocchi informativi coerenti e il più significativi possibile. Questo ci ha permesso di catturare meglio le relazioni tra domande e risposte, migliorando la qualità della Named Entity Recognition, l’analisi dei sentimenti e l’efficacia complessiva del chatbot.

Il merging si è quindi rivelato una fondamenta trasversale su cui si sono poggiate tutte le fasi successive dell’analisi.

```

def parse_whatsapp_chat(file_path):
    if not os.path.exists(file_path):
        raise FileNotFoundError(f"File non trovato: {file_path}")

    messages = []
    skipped = 0
    pattern = r"\[(\d{2}/\d{2}/\d{2}), (\d{2}:\d{2}:\d{2})\] (.*) (.*)"

    with open(file_path, "r", encoding="utf-8") as file:
        for line in file:
            match = re.match(pattern, line)
            if match:
                date, time, sender, message = match.groups()
                # Pulizia caratteri invisibili
                message = message.replace("\u200e", "").replace("\u202f", "").strip()
                sender = sender.replace("\u200e", "").replace("\u202f", "").strip()
                # Filtra messaggi di sistema
                if any(phrase in message for phrase in [
                    "ha usato il link d'invito per entrare nel gruppo",
                    "ha creato questo gruppo",
                    "I messaggi e le chiamate sono crittografati",
                    "Hai usato il link d'invito per entrare nel gruppo.",
                    "Questo messaggio è stato eliminato.",
                    "Lorenzo Dulcis ha rimosso"
                ]):
                    skipped += 1
                    continue
                messages.append({
                    "date": date,
                    "time": time,
                    "sender": sender,
                    "message": message
    
```

Il risultato di questa elaborazione è un file JSON strutturato (erasmus\_WA.json), composto da un elenco di dizionari, ciascuno rappresentante un singolo messaggio con i seguenti campi:

- **date** (formato gg/mm/aa),
- **time** (formato hh:mm:ss),
- **sender** (nome del mittente o numero telefonico),
- **message** (contenuto testuale del messaggio).

```
{
  "date": "03/03/24",
  "time": "14:09:04",
  "sender": "~condi",
  "message": "ciao ragazzi, qualcuno (soprattutto di scienze della comunicazione) ha già fatto l'Erasmus a Vilnius? o sa dirmi già se ci sono esami che combaciano con quelli di sdc?"
},
```

Questa versione pulita e strutturata ha costituito la base dati operativa su cui sono state poi applicate le fasi successive del progetto, come l'estrazione di entità, l'analisi del sentiment, il topic modeling e la costruzione del sistema RAG (Retrieval-Augmented Generation).



# RICONOSCIMENTO DELLE ENTITÀ (NER)

La **Named Entity Recognition (NER)** è una tecnica di elaborazione del linguaggio naturale (NLP) che consente di identificare automaticamente entità specifiche all'interno di un testo.

Le entità possono essere nomi di persone, luoghi, organizzazioni, date, documenti o concetti rilevanti. Nel nostro caso, la NER è servita a individuare le informazioni più importanti all'interno delle conversazioni tra studenti.

## OBIETTIVO DELLA FASE

L'obiettivo prioritario di questa fase è riconoscere in automatico entità chiave come:

- Nomi di università
- Città e destinazioni Erasmus
- Organizzazioni e sigle accademiche (es. CLA, UniCa)
- Sigle e documenti (es. OLA, LA, ISEE, Learning Agreement)

## STRUMENTI UTILIZZATI

Il Primo tentativo è stato fatto sul modello **dbmdz/bert-base-italian-xxl-cased** (basato su transformers), testato in quanto si tratta di un modello avanzato con un'ampia copertura lessicale.

Sono state, però, riscontrate delle criticità legate alle prestazioni che offriva su testo informale (chat brevi, emoji, termini dialettali); inoltre restituiva, molto spesso, falsi positivi in parole come “ok”, “ciao”, “io”, le quali venivano identificate erroneamente come PER o MISC.

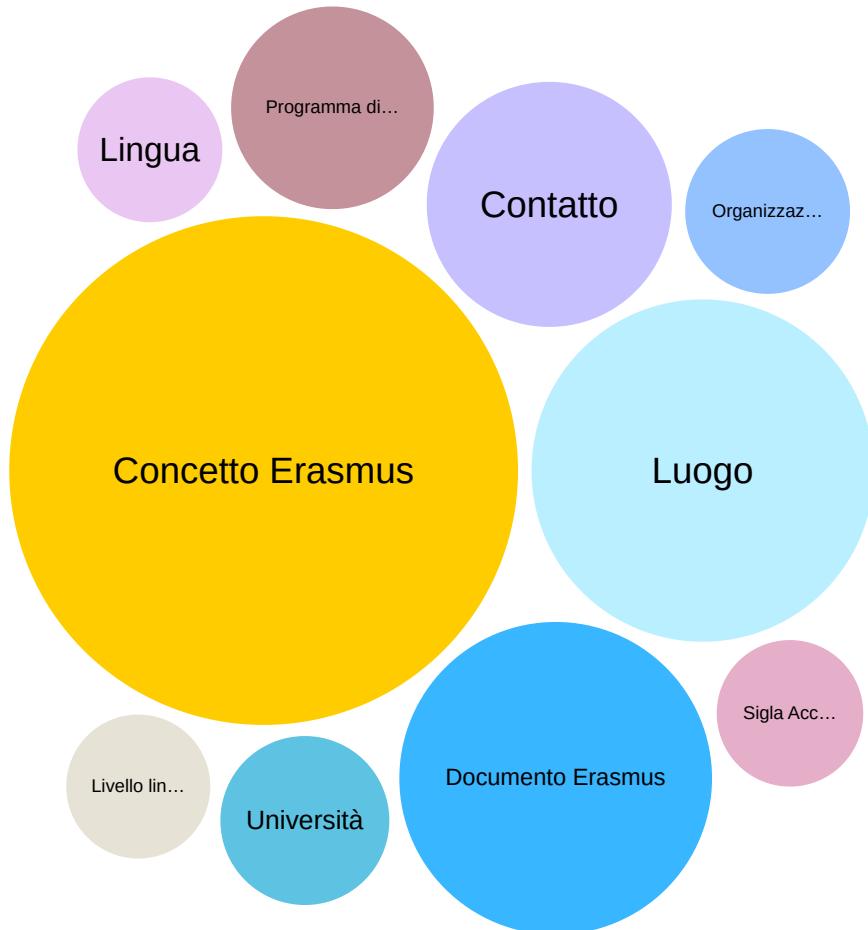
L'approccio che abbiamo scelto di perseguire per la Named Entity Recognition, quindi, si è basato sull'utilizzo del modello **DeepMount00/universal\_ner\_ita**, integrato tramite la libreria **GLiNER** di HuggingFace e pensato per compiti NER multiclasse in lingua italiana che consente di effettuare NER personalizzata, accettando in input un set di etichette definite dall'utente, senza necessità di fine-tuning.

Lo script eseguito per effettuare il riconoscimento delle entità definisce una lista pensata di entità rilevanti per il contesto Erasmus: università, sigle accademiche, documenti, concetti chiave, livelli linguistici e persino valute o contatti, consentendo in questo modo al modello di focalizzarsi su categorie semantiche rilevanti, garantendo un'estrazione mirata e significativa delle informazioni.

```
etichette_personalizzate = ["università", "sigla accademica", "livello linguistico",  
    "lingua", "luogo", "organizzazione", "concetto Erasmus", "documento Erasmus",  
    "programma di studio", "valuta", "data", "contatto"]
```

Per ogni messaggio, viene effettuata la predizione delle entità tramite **model.predict\_entities**, con restituzione di un dizionario contenente testo e label. Viene rimosso un piccolo set di falsi positivi (es. "cioè") contenuti in un dizionario di parole inutili; Alcune entità, inoltre, vengono riclassificate manualmente per migliorarne la coerenza: se il testo riconosciuto, ad esempio, è un livello linguistico (es. "B2"), viene etichettato come "livelli linguistici".

```
falsi_positivi = {"informazioni", "rischi", "scegliere", "bisogno", "tranquilla", "cioè", "ahhh"}  
  
documenti_erasmus = {"application form", "LA", "transcript of records", "grant agreement"}  
  
lingue = {"inglese", "spagnolo", "francese", "tedesco", "italiano", "portoghese", "olandese", "svedese"}  
livelli_linguistici = {"B1", "B2", "C1", "C2"}  
  
with open(file_input, "r", encoding="utf-8") as file:  
    messaggi = json.load(file)  
  
risultati = []  
  
print("\n Analizzando TUTTI i messaggi con il modello Universal NER ITA...")  
for msg in tqdm(messaggi, desc="Processando messaggi", unit="msg"):  
    entita = model.predict_entities(msg["message"], etichette_personalizzate)  
  
    refined_entities = []  
    for e in entita:  
        testo = e["text"]  
        tipo = e["label"]
```



L'analisi delle entità estratte mostra chiaramente la prevalenza di alcune tipologie, come “**programma di studio**”, “**luogo**” e “**concetto Erasmus**”, che insieme costituiscono il cuore semantico delle conversazioni analizzate. Questa distribuzione riflette l'orientamento informativo dei messaggi: richieste di chiarimenti sul bando Erasmus, domande sui requisiti linguistici, dubbi logistici e organizzativi.

L'implementazione della Named Entity Recognition ha permesso di strutturare semanticamente un corpus caotico e informale come quello delle chat studentesche, restituendone le informazioni chiave in una forma più ordinata, leggibile e interrogabile.

Il valore aggiunto di questa fase non risiede unicamente nell'estrazione automatica delle entità, ma soprattutto nella loro personalizzazione: di fatti, definendo etichette specifiche per il contesto Erasmus (come programma di studio, documento Erasmus o concetto Erasmus), si è potuto valorizzare ciò che realmente conta per gli studenti nel momento in cui affrontano un'esperienza di mobilità internazionale.

Però, dopo aver sperimentato la Named Entity Recognition sui messaggi singoli, abbiamo deciso di rieseguirla in seguito **all'unificazione del dataset**, per cogliere tutte quelle sfumature che richiedono un minimo di contesto per essere comprese e classificate correttamente.

Per farlo, abbiamo costruito un **sistema di aggregazione temporale** che raggruppa i messaggi in intervalli di 60 minuti. Questo ha permesso di ottenere “documenti” più ricchi, su cui applicare nuovamente la NER con maggiore efficacia.

Questa struttura, infatti, è risultata molto più efficace per l'estrazione entitativa: il modello non si è trovato a lavorare su messaggi isolati o ambigui, ma su contenuti estesi e ricchi di significato.

L'intervento non ha solo migliorato la quantità di entità estratte ma, soprattutto, la loro qualità dal punto di vista semantico

Così come nella NER effettuata prima del merging, il modello GLiNER viene inizializzato con la lista personalizzata di etichette, al quale si è aggiunto il medesimo sistema di pulizia intelligente, effettuata creando un dizionario di falsi positivi e dei dizionari di riclassificazione per entità che possono risultare ambigue o ricorrenti.

Successivamente, dopo aver applicato la Named Entity Recognition ai documenti aggregati, ottenendo una lista di entità strutturata nel formato **{text, label}**, si è resa necessaria una fase di pulizia e normalizzazione. Questa è stata gestita dalla funzione **step5\_clean\_ner()**, successiva alla NER effettuata in chatbot\_pipeline.py, la quale ha avuto il compito di:

- Semplificare la struttura della colonna entities, estrapolando solo le etichette semantiche (label)
- Rimuovere duplicati, mantenendo l'ordine originale
- Preparare i dati per analisi esplorative, grafici e raggruppamenti tematici

```
9
10 def extract_texts(ner_list):
11     if not isinstance(ner_list, list):
12         return []
13     texts = [ent.get("label") for ent in ner_list if isinstance(ent, dict) and "label" in ent]
14     return list(dict.fromkeys(texts))
15
16
```

Questa funzione effettua un check sull'effettivo inserimento dell'oggetto nella lista dizionari; dopodiché estrae i valori del campo label, mantenendo l'ordine e rimuovendo le ripetizioni e permettendo di ottenere una nuova colonna, più essenziale, ma comunque rappresentativa del contenuto semantico di ciascun documento.

# SENTIMENT ANALYSIS

La **Sentiment Analysis** è una tecnica di NLP che consente di determinare il tono emotivo di un testo, classificandolo generalmente come positivo, negativo o neutro.

Viene spesso utilizzata per analizzare opinioni, recensioni o, nel nostro caso, conversazioni tra studenti, per cogliere il clima emotivo associato a determinate tematiche, esperienze, dubbi e procedure specifiche.

## OBIETTIVO DELLA FASE

Abbiamo deciso di includere questa fase per:

- Rilevare lo stato d'animo degli studenti in relazione a problemi ricorrenti (es. attese, ritardi, confusione)
- Analizzare il livello di frustrazione o soddisfazione legato a enti, documenti e momenti critici del percorso Erasmus

## STRUMENTI UTILIZZATI

Dopo una fase di valutazione iniziale, per eseguire l'analisi del sentiment abbiamo utilizzato un modello pre-addestrato specifico per l'italiano, denominato **feel-it-italian-sentiment**, basato su architettura BERT.

Questo modello, integrato all'interno di una pipeline HuggingFace, ci ha restituito per ogni messaggio una classificazione in positivo o negativo, accompagnata da un punteggio di confidenza.

In questa fase ci siamo concentrati esclusivamente sull'identificazione della polarità dei messaggi, senza considerare la dimensione emozionale. Tuttavia, l'analisi del sentiment rappresenta solo una parte del quadro emotivo generale: sapere se un messaggio è positivo o negativo è utile, ma non sufficiente per comprendere a fondo lo stato d'animo degli studenti.

```

model_name = "MilaNLProc/feel-it-italian-sentiment"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSequenceClassification.from_pretrained(model_name)

# Creazione pipeline
sentiment_pipeline = pipeline(task: "sentiment-analysis",
                               model=model,
                               tokenizer=tokenizer,
                               batch_size=25,
                               truncation=True,
                               max_length=512)

df["sentiment"] = df["documento"].progress_apply(lambda x: sentiment_pipeline(x)[0]["label"])
df["sentiment_score"] = df["documento"].progress_apply(lambda x: sentiment_pipeline(x)[0]["score"])

```

Quindi, in parallelo e per non limitarci ad una analisi del sentimento basata esclusivamente sulla contrapposizione positivo-negativo, abbiamo affiancato un secondo modello, **feel-it-italian-emotion**, per rilevare l'emozione prevalente associata al testo, selezionando tra le categorie joy, sadness, anger e fear e distinguere meglio le diverse sfumature emotive presenti nei messaggi. Questa combinazione ci ha permesso di interpretare con maggiore precisione il contenuto ed il contesto delle conversazioni, superando la semplice dicotomia positivo/negativo.

```

emotion_pipeline = pipeline(
    task: "text-classification",
    model="MilaNLProc/feel-it-italian-emotion",
    top_k=1 # restituisce solo l'emozione dominante
)

```

Entrambi i modelli sono stati applicati in modalità batch tramite pipeline HuggingFace, con troncamento automatico a 512 caratteri per rispettare i vincoli dell'architettura Transformer. I risultati ottenuti sono stati salvati in un dataset strutturato, utilizzato come base per le successive analisi descrittive e grafiche.

Dall'analisi dei messaggi è emersa una netta prevalenza di contenuti con valenza negativa, che costituiscono l'82,6% del totale, a fronte di un 17,4% di messaggi positivi. Questo dato evidenzia una forte concentrazione di problematiche, dubbi e richieste di supporto da parte degli utenti.

La componente negativa è riconducibile principalmente a questioni pratiche (es. gestione delle scadenze, modulistica, difficoltà nella comprensione delle procedure Erasmus) e a una percezione di incertezza generalizzata.

A livello emozionale:

- La **tristezza** è l'emozione prevalente, riscontrata nel 38,9% dei messaggi.
- La **rabbia** rappresenta il 29,7% del totale, spesso legata a disservizi o incomprensioni.
- La **paura** si attesta al 18,2%, riflettendo timori legati a errori procedurali o perdite di opportunità.
- Solo il 13,2% dei messaggi esprime **gioia**, generalmente connessa a esperienze positive concluse o suggerimenti entusiasti.

Considerando, nel dettaglio, esempi tipici di sentimento negativo, il quale spesso è legato a dubbi pratici, difficoltà burocratiche o mancanza di risposte chiare. In molti casi, la frustrazione è associata a emozioni come la tristezza, la rabbia o la paura, confermando quanto l'incertezza possa pesare sull'esperienza degli studenti.

Messaggio	Sentiment	Score	Emozione	Emotion Score
A me devono ancora firmare il learning agreement.	negativo	0.999879	sadness	0.997172
Aspe.. in che senso borsa di 10 mesi?	negativo	0.997704	anger	0.706929
Comunque spero vivamente che rispondano presto...	negativo	0.999697	sadness	0.996947
Questa cosa dei 6 mesi chi ve l'ha detta? Perché non l'ho trovata scritta da nessuna parte.	negativo	0.999790	anger	0.998425
Scusatemi, io non riesco a trovare l'OLA. Qualcuno sa dove sta?	negativo	0.998450	fear	0.978392

I messaggi classificati come positivi, invece, pur rappresentando una minoranza, evidenziano momenti di chiarezza, aiuto ricevuto o esperienze concluse con successo.

Messaggio	Sentiment	Score	Emozione	Emotion Score
Ragazzi qualcuno per caso sa la mail a cui mandare l'OLA? Mi hanno risposto subito.	positivo	0.999684	joy	0.999288
Nella mail hanno scritto che se vogliamo acquistare il biglietto possiamo farlo da adesso.	positivo	0.535565	joy	0.976506
Io l'ho inoltrato alla referente che poi ha provveduto ad approvarlo. Tutto ok.	positivo	0.999268	joy	0.995000

L'analisi del sentiment ha, quindi, evidenziato una netta predominanza di sentiment negativo e di emozioni come tristezza, rabbia e paura, a conferma di un diffuso senso di incertezza e frustrazione tra gli studenti che intendono fare un'esperienza Erasmus. I messaggi analizzati rivelano difficoltà legate a procedure complesse, mancanza di informazioni chiare e bisogno costante di supporto pratico.

Le interazioni positive, pur rappresentando una minoranza, mostrano quanto sia apprezzato un supporto efficace e tempestivo, spesso associato a emozioni di gioia e sollievo.

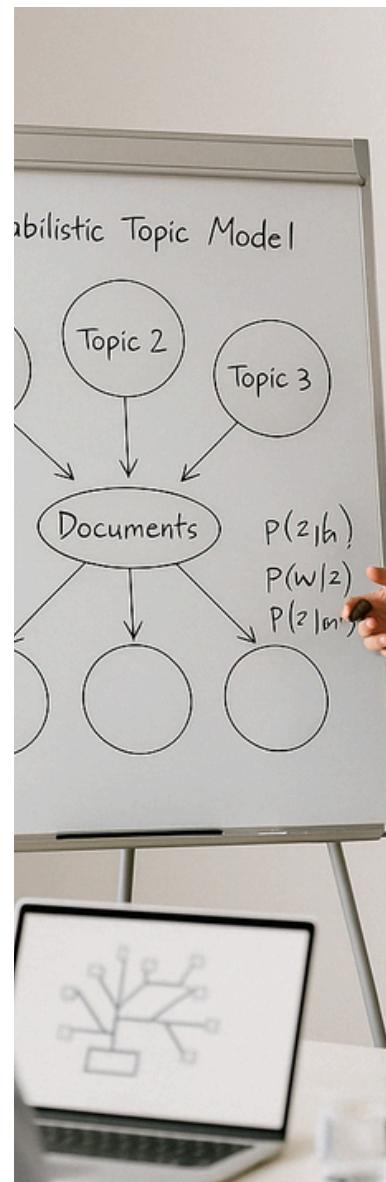
Questi risultati confermano la necessità di strumenti che possano offrire risposte rapide, affidabili e accessibili. Il chatbot Erasmus, infatti, nasce esattamente in risposta a questa esigenza, con l'obiettivo di migliorare l'esperienza informativa degli studenti e ridurre il carico emotivo associato alla gestione autonoma delle procedure.

# TOPIC MODELING

Nel nostro progetto, uno degli obiettivi principali era organizzare il contenuto delle conversazioni Erasmus in modo da riconoscere automaticamente i temi principali affrontati dagli studenti. A tale scopo, abbiamo adottato una tecnica di NLP chiamata **Topic Modeling**, utile per esplorare strutture semantiche latenti all'interno di grandi collezioni testuali. Questa metodologia, non supervisionata, permette di individuare insiemi di parole statisticamente correlate che rappresentano concetti o argomenti ricorrenti, senza bisogno di categorizzazioni predefinite.

I modelli principali di topic modeling utilizzanti in ambito NLP sono LDA e BERTopic. Il primo è un modello probabilistico che rappresenta ogni documento come una combinazione di topic ed ogni topic come una distribuzione di parole; tuttavia, si basa su co-occorenze di parole e richiede testi relativamente lunghi e strutturati per funzionare in modo efficiente. **BERTopic**, invece, è un modello più recente che sfrutta embedding semantici generati da modelli pre-addestrati, combinandoli con tecniche di clustering; è una tecnica che, almeno sulla carta, consente di individuare topic coerenti anche in presenza di dati testuali brevi, informali e frammentati. È proprio per la sua capacità di gestire testi brevi e poco strutturata che abbiamo deciso di utilizzare quest'ultimo modello.

Applicato al nostro corpus, il topic modeling avrebbe dovuto restituire una rappresentazione sintetica e strutturata delle principali aree tematiche emerse spontaneamente dai messaggi.



## OBIETTIVO DELLA FASE

L'obiettivo principale di questa fase è stato quello di identificare automaticamente i temi ricorrenti nei messaggi raccolti, con l'intento di migliorare la struttura informativa del dataset e renderlo più funzionale al recupero delle risposte da parte del chatbot. In particolare, il topic modeling è stato introdotto per supportare la fase di retrieval all'interno dell'architettura RAG: organizzando i contenuti in base alla loro coerenza tematica, abbiamo reso più semplice e mirato il recupero dei documenti rilevanti rispetto a una domanda dell'utente.

Oltre alla funzione esplorativa, che ci avrebbe permesso di ottenere una panoramica dei macro-temi presenti nei messaggi (come test linguistici, OLA, burocrazia, alloggi), la topic modeling è stata pensata per guidare il chatbot verso documenti semanticamente coerenti con la richiesta, riducendo la dispersione e migliorando, potenzialmente, la qualità delle risposte generate.

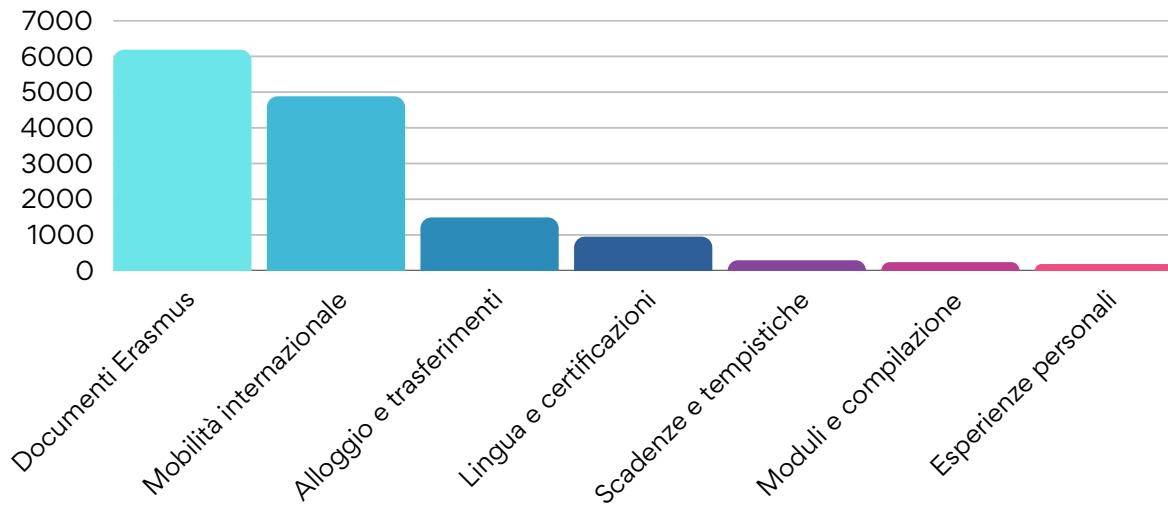
## PRIMO APPROCCIO

Abbiamo inizialmente sperimentato un approccio diretto, applicando il topic modeling sull'intero corpus dei messaggi raccolti, analizzando ciascun testo come unità indipendente.

```
from bertopic import BERTopic
from sentence_transformers import SentenceTransformer

embedding_model = SentenceTransformer("distiluse-base-multilingual-cased-v1")
topic_model = BERTopic(embedding_model=embedding_model)
topics, _ = topic_model.fit_transform(messaggi_puliti)
```

La speranza era quella di ottenere un insieme di cluster tematici coerenti, sufficientemente rappresentativi da poter essere etichettati e utilizzati come riferimento sia nell'analisi, sia nella fase di retrieval. Tuttavia, l'analisi critica dei risultati ha evidenziato limiti significativi. I messaggi presenti nel dataset, per quanto autentici e numerosi, sono spesso estremamente brevi, frammentati o scollegati tra loro. Molti contengono risposte minime ("sì", "ok", "perfetto"), esclamazioni isolate, o richieste vaghe prive di contesto. L'applicazione del topic modeling su testi di questo tipo ha, infatti, prodotto una segmentazione semantica confusa, con cluster dominati da contenuti generici o semanticamente ambigui. Anche i tentativi di etichettatura manuale si sono rivelati complessi, in quanto i messaggi assegnati a uno stesso topic risultavano spesso eterogenei, e le etichette scelte faticavano a sintetizzare in modo univoco il contenuto dei gruppi.



Questa esperienza ci ha portati a riflettere su alcuni punti fondamentali, ossia:

- Applicare il topic modeling su messaggi isolati produce spesso cluster deboli.
- I messaggi brevi o frammentati riducono la coerenza semantica dei risultati.
- Serve un contesto: documenti densi, coerenti e quanto più strutturati.
- La segmentazione deve sempre essere accompagnata da una validazione qualitativa.

Un risultato pienamente soddisfacente del topic modeling, nel nostro caso, dipende in modo diretto dalla qualità e dalla coerenza semantica delle unità testuali su cui opera; analizzare singoli messaggi scollegati tra loro non permette al modello di cogliere una direzione tematica chiara, perché manca la continuità e il contesto necessari per dare un senso chiaro.

Per questo motivo, infatti abbiamo scelto di riprogettare l'intera fase di analisi tematica, rinviando l'applicazione definitiva del **topic modeling dopo il merging dei messaggi**, una fase di consolidamento semantico in grado di restituirci documenti coerenti ed articolati, sulle quali sarà possibile condurre un'analisi tematica davvero significativa.

## ■ SECONDO APPROCCIO

. In virtù delle criticità emerse con il primo approccio, abbiamo eseguito, prima di effettuare il topic modeling, il processo di **merging** semantico e temporale già considerato precedentemente, unificando in un unico documento i messaggi scambiati tra gli stessi interlocutori in un intervallo di tempo ristretto, ossia 60 minuti.

Questa strategia ci ha permesso di ridurre la frammentazione semantica e costruire unità testuali più strutturate, lunghe e significative, in cui il modello potesse cogliere un'intenzione comunicativa completa, anziché frammenti disgiunti. Ogni documento generato include:

- **un intervallo temporale (start\_time – end\_time)**
- **i partecipanti alla conversazione**
- **il testo concatenato dei messaggi scambiati**
- **il numero di messaggi unificati**

Una volta generato questo corpus contestualizzato, abbiamo eseguito nuovamente la topic modeling, applicando il modello BERTopic, integrato con **SentenceTransformer multilingua**, per generare rappresentazioni semantiche profonde dei documenti contestualizzati.

```
from bertopic import BERTopic
from sentence_transformers import SentenceTransformer

embedding_model = SentenceTransformer("distiluse-base-multilingual-cased-v1")
topic_model = BERTopic(embedding_model=embedding_model, language="multilingual", calculate_probabilities=True)
```

Grazie agli embedding contestuali, ogni documento è stato trasformato in un vettore numerico, successivamente utilizzato per la generazione dei cluster tematici.

I documenti ottenuti dal merging sono stati elaborati dal modello per determinare il topic più probabile per ciascuno:

```
topics, probs = topic_model.fit_transform(documenti)
```

I risultati sono stati decisamente più promettenti. L'output ha mostrato:

- una maggiore coerenza semantica all'interno dei cluster
- una distribuzione più bilanciata dei topic
- una base testuale più adatta per il successivo impiego nel sistema RAG

Per garantire che ogni cluster fosse semanticamente interpretabile, abbiamo previsto un **passaggio di etichettatura manuale**.

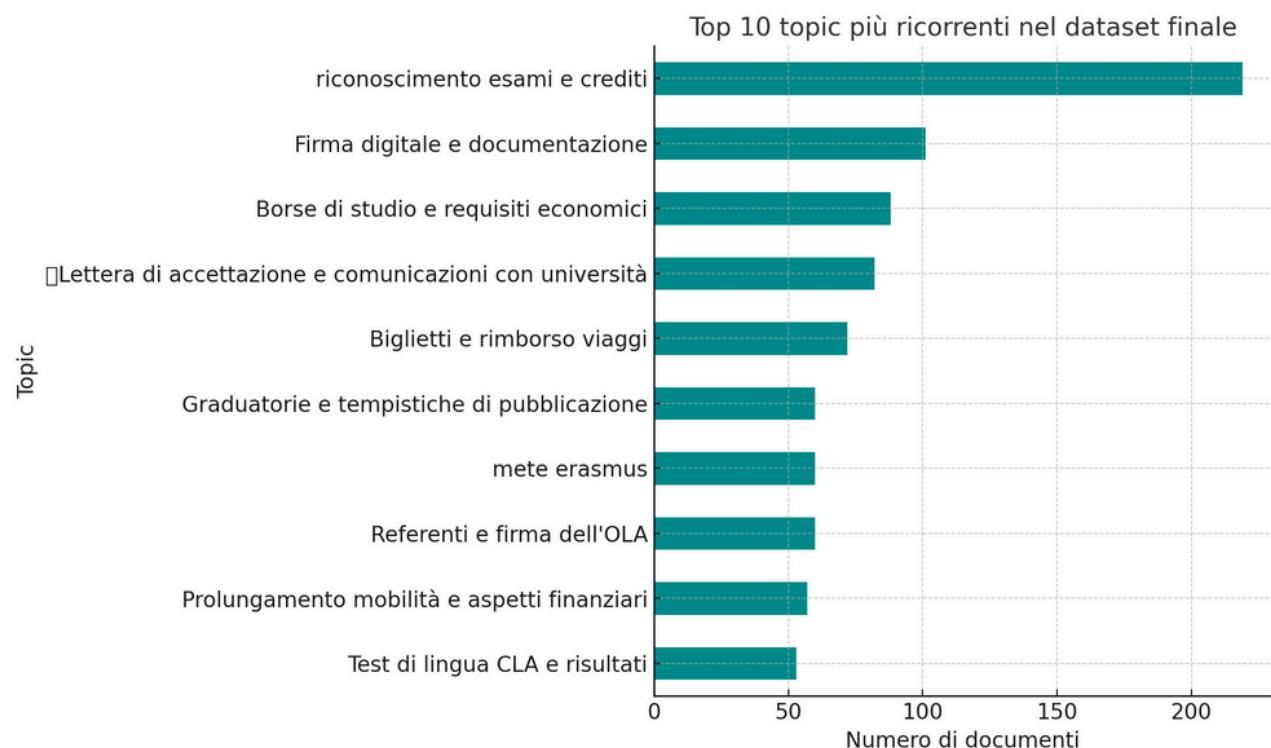
Il sistema proponeva le parole chiave associate a ciascun topic e ci chiedeva di assegnare un'etichetta coerente. Questo processo ha portato alla definizione di 61 etichette tematiche distinte, risultanti da un'analisi qualitativa diretta del contenuto dei documenti.

```
for topic_id in topic_info["Topic"]:
    if topic_id == -1:
        continue
    keywords = ", ".join([word for word, _ in keywords_per_topic[topic_id]])
    print(f"\n[Topic {topic_id}] Parole chiave: {keywords}")
    label = input("Etichetta assegnata (es. 'Documenti Erasmus', 'Non rilevante', ecc.): ")
    etichettatura.append({
        "topic": topic_id,
        "keywords": keywords,
        "topic_label": label
    })
```

Dopo l'operazione di merging e l'assegnazione manuale delle etichette, abbiamo ottenuto un dataset di documenti tematicamente organizzati.

Ogni documento è coerente, contestualizzato e assegnato a uno dei diversi topic distinti. Questo set rappresenta il cuore semantico della knowledge base del chatbot.

Il grafico qui di seguito, invece, mostra i 10 topic tematici più ricorrenti emersi nel dataset dopo le operazioni specificate di merging ed etichettatura manuale.



I risultati confermano come l'esperienza Erasmus sia fortemente caratterizzata da aspetti burocratici e organizzativi, con una prevalenza di discussioni relative a: borse di studio e pagamenti, graduatorie e tempistiche, certificazioni linguistiche e livelli richiesti.

Particolarmente rilevante è anche la presenza di topic legati all'orientamento alla scelta della destinazione, che riflette le incertezze iniziali degli studenti.

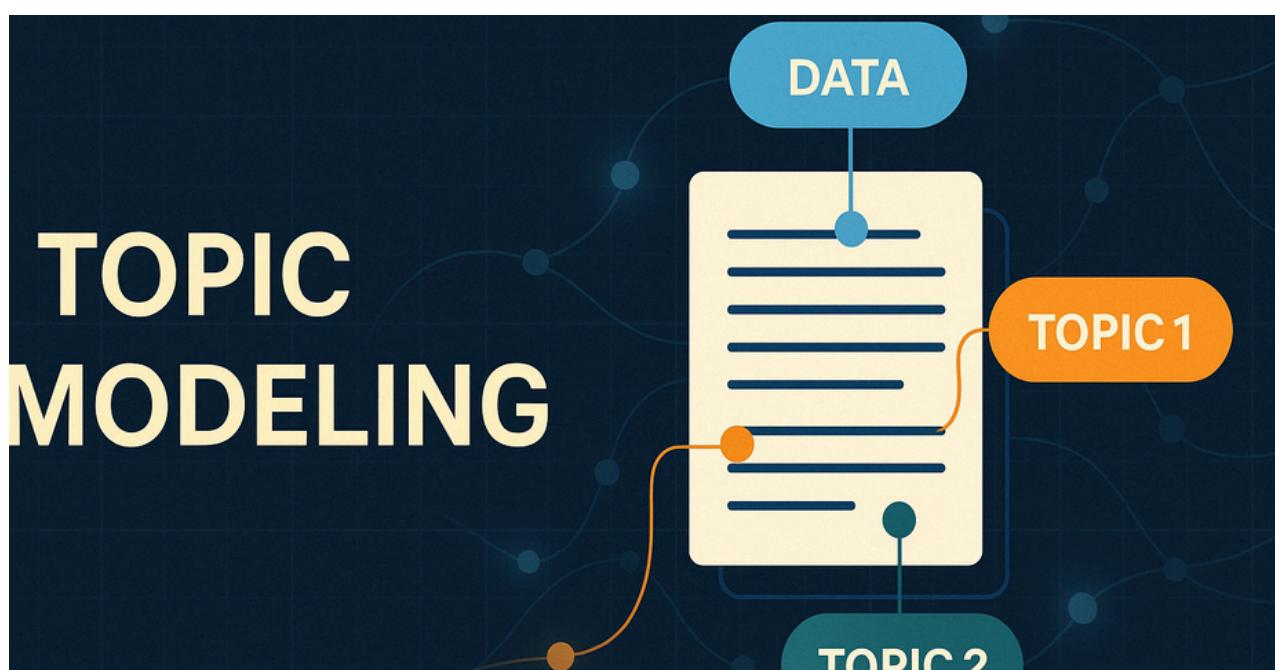
La distribuzione tematica restituisce una fotografia autentica delle preoccupazioni più sentite, fornendo indicazioni preziose per la personalizzazione delle risposte da parte del chatbot.

Accanto ai topic centrali, però, il dataset mostra anche una coda lunga di tematiche meno frequenti, ma comunque significative. Tra queste troviamo:

- richieste specifiche su orari e scadenze di iscrizione,
- problemi relativi a spostamenti locali a Cagliari,
- dubbi circoscritti su paesi specifici, come la Francia
- domande di tipo sociale, come la ricerca di gruppi Erasmus.

Questi topic, pur meno ricorrenti, rappresentano una componente essenziale per offrire risposte personalizzate e complete: coprono casi marginali ma reali, spesso trascurati nei documenti ufficiali.

La loro presenza dimostra che la knowledge base finale è ricca, sfaccettata e rappresentativa dell'intera esperienza Erasmus.



<b>Etichetta tematica</b>	<b>Documento (estratto)</b>	<b># Messaggi</b>
Lettera di accettazione e comunicazioni con l'università	Ma per compilare il learning agreement bisogna	8
Lettera di accettazione e comunicazioni con l'università	Raga il contributo UE è quello che varia da 250 a	3
Borse di studio e requisiti economici	Scusate non mi ricordo la comunicazione di vinc	3
Certificazioni linguistiche e livelli richiesti	Raga ma è normale che la Baita non mi abbia ancora fissat	6
Criteri di assegnazione delle università Erasmus	dove la trovo la data precisa di inizio e fine per il	6

La tabella, invece, mostra una selezione di documenti rappresentativi dei topic più ricorrenti individuati dal modello.

Ogni riga corrisponde a una breve conversazione, aggregata tramite merging, e successivamente etichettata manualmente.

I contenuti spaziano dai temi burocratici, come le lettere di accettazione e i contributi Erasmus, fino alle difficoltà legate alle certificazioni linguistiche o alle tempistiche di pubblicazione delle graduatorie.

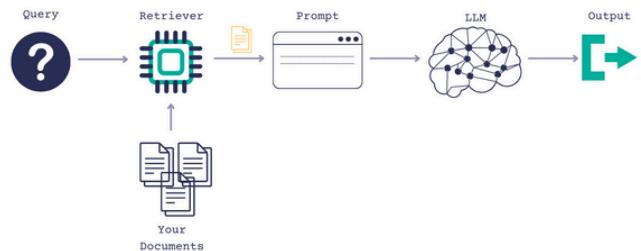
Questi esempi evidenziano come il topic modeling, una volta supportato dal contesto semantico, riesca a restituire unità testuali significative, interpretabili e ad alta rilevanza pratica.

In conclusione, il processo finora descritto ha rappresentato la svolta operativa del nostro progetto: non solo abbiamo validato l'importanza del contesto nell'analisi testuale, ma abbiamo anche costruito la base informativa ideale per alimentare un sistema di Retrieval-Augmented Generation (RAG), in grado di indicizzare e recuperare documenti sulla base del topic tematico riconosciuto. Rispetto alla prima fase, il nuovo approccio si è dimostrato più robusto sotto ogni aspetto:

- Testi più lunghi, coerenti e leggibili
- Topic più facilmente etichettabili
- Maggiore precisione nel clustering
- Dataset realmente utilizzabile nella fase RAG

In questo senso, quindi, il topic modeling ha smesso di essere un'analisi esplorativa per diventare una componente strategica del sistema di risposta, rendendo il chatbot capace di capire cosa cercare, all'interno di un vasto dataset a sua disposizione..

# ARCHITETTURA DEL SISTEMA DI RISPOSTA (RAG)



## OBIETTIVO E CONTESTO D'USO

A conclusione del progetto, l'obiettivo finale è stato quello di integrare le informazioni organizzate nei documenti tematici in un sistema di risposta automatica, capace di interagire con gli studenti Erasmus in maniera chiara, utile e personalizzata.

Per raggiungere questo traguardo, è stata adottata una pipeline basata su **Retrieval-Augmented Generation (RAG)**, una delle soluzioni più efficaci e trasparenti oggi disponibili per sistemi di question answering su basi conoscitive specifiche.

Il vantaggio di un'architettura RAG risiede nella sua capacità di coniugare il recupero documentale semantico con la generazione linguistica naturale, evitando le "hallucinations" tipiche dei modelli puramente generativi e ancorando le risposte a contenuti verificabili.

## PRIMA FASE: PREPARAZIONE DEI DATI E CREAZIONE DEI DOCUMENTI

I dati vengono letti da un file CSV contenente conversazioni già pre-elaborate e segmentate in blocchi temporali. Ogni blocco rappresenta un contesto conversazionale ("documento") con associato un timestamp e i partecipanti.

Ciascun blocco viene poi trasformato in un oggetto **Document**, contenente il testo aggregato e alcune metainformazioni come la data e i mittenti. Questi documenti saranno la base per l'indicizzazione e il recupero.

## SECONDA FASE: PIPELINE DI INDICIZZAZIONE

Viene creata una **pipeline Haystack** che si occupa di:

- Pulizia del testo tramite **DocumentCleaner**;
- **Conversione del testo in embedding** semanticci tramite un modello transformer (**bge-large-en-v1.5**) dove vengono inseriti anche i metadati utili alla comprensione del contesto;

- Memorizzazione dei documenti nel **InMemoryDocumentStore** con il **DocumentWriter**, indicizzandoli sia tramite BM25 (keyword-based) che tramite embedding vettoriali. L'embedding converte i dati in rappresentazioni numeriche e li memorizza in un database vettoriale. Questo processo crea una libreria di conoscenze che i modelli di intelligenza artificiale generativa possono comprendere.

La **pipeline di indicizzazione** è composta da diversi componenti:

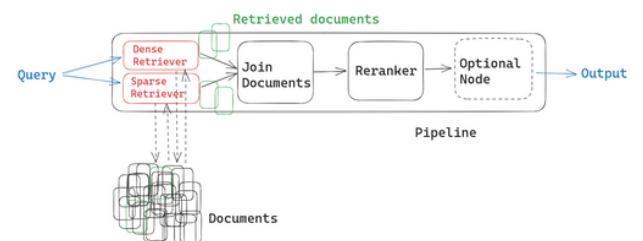
**DocumentCleaner**: esegue una pulizia di base dei documenti

**SentenceTransformersDocumentEmbedder**: rappresenta ogni documento come un vettore (catturandone il significato) utilizzando un modello specificato. Anche i metadati vengono incorporati, poiché contengono informazioni rilevanti (parametro **metadata\_fields\_to\_embed**).

**DocumentWriter** salva semplicemente i documenti nel **document store**.

Questo, serve per memorizzare le rappresentazioni numeriche dei documenti, che ci permette di semplificare la comprensione e l'elaborazione dei dati per il sistema l'LLM

InMemoryDocumentStore viene utilizzato per memorizzare i dati dei filmati con i relativi incorporamenti, che li memorizzano nella memoria RAM del dispositivo.



## TERZA FASE: PIPELINE DI RAG

La **pipeline RAG** individua i documenti rilevanti per la query dell'utente e li passa al LLM per generare una risposta fondata.

Qui abbiamo scelto di sviluppare una pipeline ibrida supportata da un modello Re-Ranker per mostrare una soluzione sofisticata per la creazione di sistemi RAG.

La **pipeline ibrida** si riferisce alla combinazione di più metodi di recupero per migliorare le prestazioni complessive. Nel contesto dei sistemi di ricerca, una pipeline di recupero ibrida esegue sia la tradizionale **ricerca** basata su parole chiave (con **BM25**) che la **ricerca vettoriale densa**, classificando successivamente i risultati con un modello cross-encoder. Questa combinazione consente al sistema di ricerca di sfruttare i punti di forza di diversi approcci, fornendo risultati più accurati e diversificati.

La pipeline di RAG è composta da più fasi:

- **Text Embedder**: converte la domanda dell'utente in un vettore semantico.
- **Retriever ibrido**: combina BM25 (lessicale) ed embedding retriever (semantico) per trovare i messaggi più pertinenti.
- **Document Joiner**: unisce i documenti recuperati in un contesto coerente.
- **Ranker**: riordina i documenti in base alla similarità rispetto alla domanda.
- **Prompt Builder**: costruisce un prompt strutturato, includendo contesto e domanda ("Dato questo prompt, risponderai esclusivamente in italiano..")
- **LLM Generator**: un modello di linguaggio (Zephyr-7b-beta) genera la risposta finale in italiano.

Abbiamo scelto Zephyr per via delle ottime prestazioni. infatti, questo modello è una versione ottimizzata di Mistral 7B V.01 che si concentra sull'utilità e supera in prestazioni molti modelli più grandi nei benchmark MT-Bench e AlpacaEval; il modello è stato ottimizzato dal team Hugging Face.

## ■ QUARTA FASE: TEST E VALUTAZIONE DEL RAG

La valutazione è stata fatta in due modi diversi:

- In una **prima fase**, abbiamo voluto controllare la coerenza nella risposta alle domande passando in funzione alla costruzione del RAG la funzione **ask\_question()** con 3 domande di cui 2 inerenti al contesto e l'ultima non facente parte delle informazioni passate dai nostri documenti.

```
questions = [
    "Vorrei avere informazioni sulla scadenza dell'invio della modulistica",
    "Studio economia, a chi devo mandare la modulistica?",
    "Quando si schiudono le uova di fenicottero?"
]
```

La risposta non è stata soddisfacente per la prima domanda, in quanto non era esplicativa, mentre per la seconda domanda è stato citato il nome del capo di ufficio ISMOKA per SEGP (Dott. Poddesu). Alla terza domanda non è stata fornita risposta, come richiesto nel prompt builder in caso di domanda incoerente ai documenti con cui è stato arricchito il modello.

- In una **seconda fase** abbiamo provato a valutare quantitativamente le risposte con una valutazione automatica delle risposte generate, confrontandole con un insieme di risposte attese (ground truth). Sono state utilizzate due tipologie di metriche:
  - **BLEU score**: valuta l'overlap tra n-grammi della risposta generata e quella attesa.
  - **ROUGE scores**: misura la qualità del **recall** e della **precision** nei contenuti generati, tramite ROUGE-1, ROUGE-2 e ROUGE-L.

I risultati sono pressoché deludenti, con un punteggio molto basso per entrambi gli score:

===== Risultati della valutazione =====

Corpus BLEU Score: 0.0107  
Average ROUGE-1: 0.1051  
Average ROUGE-2: 0.0259  
Average ROUGE-L: 0.0749

Questo punteggio viene riscontrato per una serie di motivi:

- Le risposte attese che vengono passate al modello possono differire parecchio da quella generata. Essendo questi sistemi valutati su delle metriche quantitative, non viene data valutata l'interpretazione di queste risposte, generando dei punteggi che potrebbero essere fuorvianti.
- I documenti che vengono passati fanno riferimento, per la maggior parte, a conversazioni informali e quindi possono essere poco esplicativi per domande facilmente reperibili dai bandi di concorso.

Infatti, guardando visivamente alle risposte generate sulle 20 domande per la valutazione quantitativa, notiamo come in alcune viene afferrata la richiesta della domanda e viene data una risposta pressoché soddisfacente, mentre in altre viene data una risposta sbagliata per errore di comprensione del documento. A diverse domande di prova non viene data una risposta perché all'interno dei documenti non si fa riferimento a quei determinati argomenti.

# CONCLUSIONI

Il progetto che abbiamo portato avanti si è sviluppato come un vero e proprio percorso di analisi e trasformazione costante dei dati: da conversazioni destrutturate e frammentate a una base informativa organizzata, interrogabile e pronta per alimentare un chatbot intelligente dedicato al mondo Erasmus.

A guidare tutto ciò è stato un obiettivo chiaro, quello costruire un sistema che riuscisse a estrarre valore reale dall'esperienza degli studenti, valorizzando il sapere pratico e informale contenuto nei messaggi condivisi tra pari.

Per farlo, abbiamo costruito una pipeline articolata in più fasi, ognuna delle quali ha contribuito a rafforzare la coerenza e la profondità del dataset finale.

Il vero salto di qualità nel lavoro svolto, però, si è compiuto solo nel momento in cui abbiamo deciso di non trattare più i messaggi come unità isolate tra loro, ma di ricomporli in blocchi che fossero quanto più coerenti possibile. Aggregarli nel tempo, ricostruire il contesto conversazionale, ha, infatti cambiato il modo in cui il sistema ha potuto leggerli, interpretarli e restituirli.

È stato a quel punto che l'analisi ha smesso di essere frammentaria e ha iniziato ad essere sempre più esplicativa, in grado di riconoscere non solo parole, ma significati, contesti, esigenze.

Abbiamo scelto, in virtù di questa consapevolezza, di non integrare fonti ufficiali, perché ci interessava verificare fino a che punto la sola voce degli studenti potesse essere sufficiente a raccontare l'esperienza Erasmus.

Volevamo testare la qualità informativa di una fonte informale e capire se, partendo esclusivamente da testimonianze spontanee, fosse possibile ricostruire con fedeltà i temi, i problemi e le domande che emergono durante il percorso di mobilità.

Quello che è emerso con questo approccio non è solo un elenco di topic o un insieme di entità, ma una mappa strutturata, piena, della realtà di chi intende andare in Erasmus, generata dalla sua parte più viva: chi la vive.

La chat a nostra disposizione si è quindi rivelata uno spazio denso, autentico e ricco dal punto di vista informativo, capace di toccare una quantità molto variegata di argomenti, dai documenti alle scadenze, dalle lingue ai dubbi emotivi, dalle istruzioni pratiche ai consigli tra studenti stessi.

Abbiamo dimostrato che anche un corpus informale, se ben trattato, può generare conoscenza strutturata e che un insieme di messaggi disordinati può diventare un'intelligenza collettiva, pronta ad aiutare altri studenti, a guidarli, a rispondere a ciò che non si trova scritto nei regolamenti, se validamente trattati.