Design, ingegnerizzazione e realizzazione di un sistema di dialogo basato su LLM nel dominio delle tecnologie assistive

Tesi di Laurea Magistrale

Relatore: Prof. Alessandro Mazzei
Co-Relatori: Dott. Pier Felice Balestrucci, Dott. Michael Oliverio
Candidato: Dott. Stefano Vittorio Porta

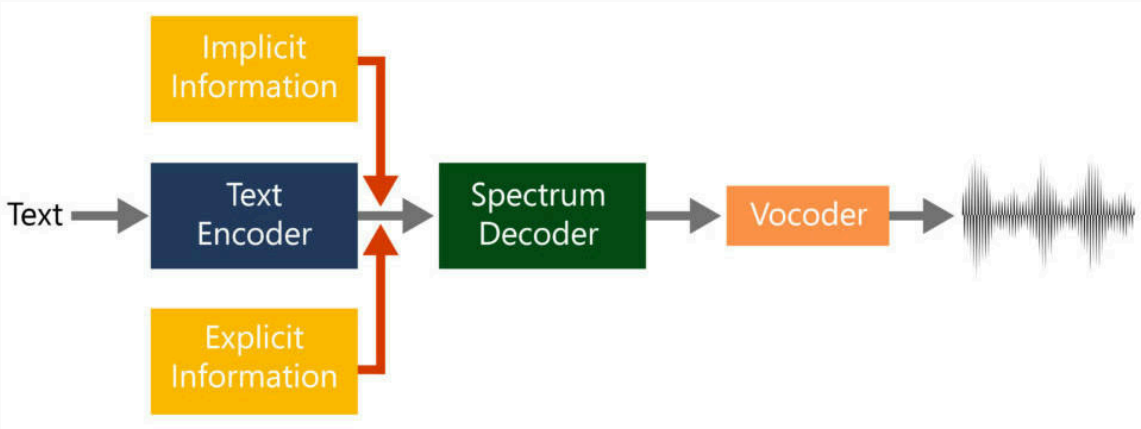
2 Aprile 2025

Università degli Studi di Torino, Dipartimento di Informatica - Anno Accademico 2023/2024

1. Contesto

Iniziamo ad ambientarci:

- **Lettura di contenuti testuali**: da 30 anni sono disponibili sistemi di sintesi vocale integrati in smartphone e computer.
- Essenziali per le persone con **disabilità visive**: consentono di accedere a contenuti testuali in modo autonomo e senza l'ausilio di un lettore umano.

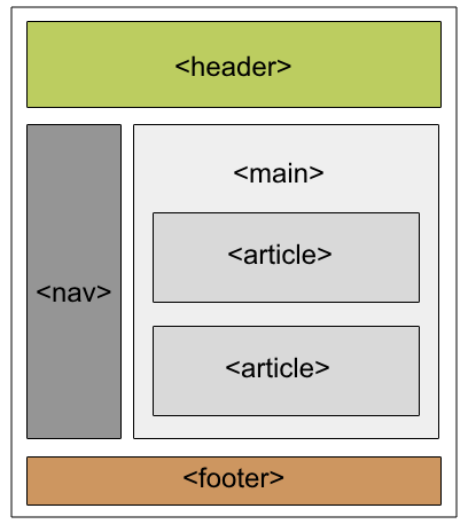


- Pagine web. I sistemi TTS sono in grado di
  - Leggere il testo
  - Interpretare la struttura del documento
  - Questo permette di seguire il flusso di lettura per fornire un'esperienza di qualità

1.2 Problema!

Tutto funziona, a patto che…

- La pagina web sia strutturata in modo semantico



- Le immagini siano accompagnate da un testo alternativo

```
<img alt="..."> o <img aria-label="...">
```

Queste due condizioni dipendono da chi prepara il contenuto!

1.2 Problema!

I sistemi di TTS non sono in grado di interpretare il significato di un'immagine o di un elemento puramente visivo.

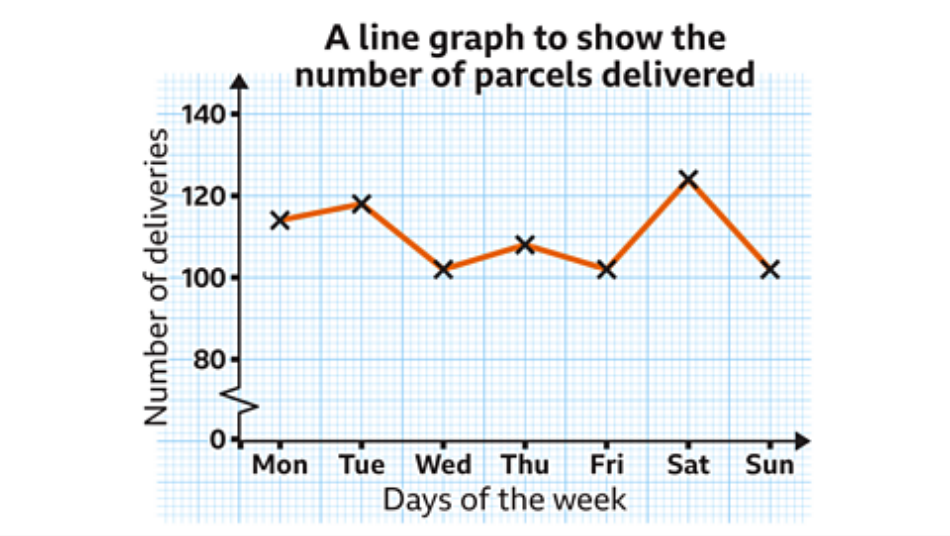
Se non viene fornita un'alternativa testuale contenente delle informazioni utili, l'utente non potrà comprendere appieno il contenuto della pagina o di uno specifico elemento!

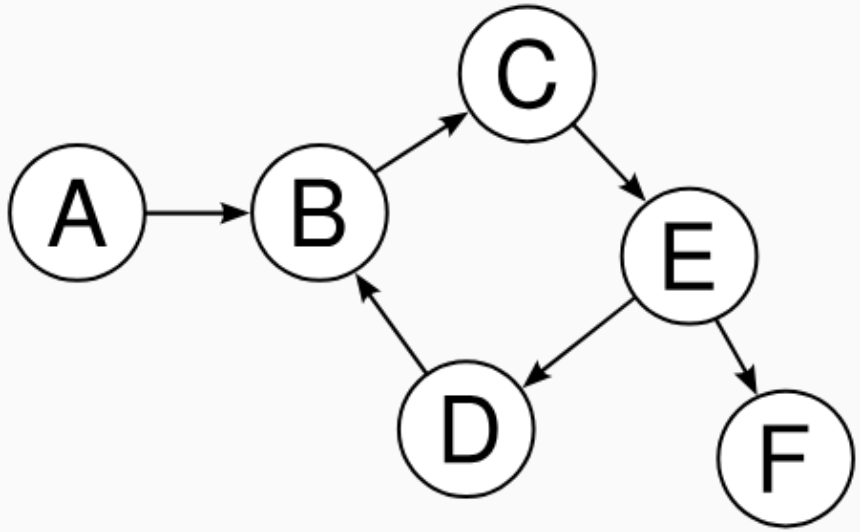
Una di quattro immagini sul web non ha una descrizione testuale o non è informativa [1].

---

[1] WebAIM, «The WebAIM Million: The 2024 Report on the Accessibility of the Top 1,000,000 Home Pages». [Online]. Disponibile su: https://webaim.org/projects/million/

Se per le immagini possiamo utilizzare tecniche di **Computer Vision** o **Reti Neurali** per generare automaticamente un testo alternativo, per le rappresentazioni grafiche di dati (grafici, diagrammi, mappe) non è così semplice.





- Il Progetto NoVAGraphS si propone di rendere più accessibili questi contenuti, mediante la costruzione di sistemi di dialogo (Chatbot) per individui non vedenti.
- Con essi è possibile interagire per ottenere informazioni sui dati presenti in grafi o strutture simili, per avere una comprensione **profonda** del contenuto.
- Il Progetto originale fa fondamento su AIML [1] (Artificial Intelligence Markup Language), un linguaggio di markup per la creazione di chatbot.

[1] «Artificial Intelligence Markup Language». [Online]. Disponibile su: http://www.aiml.foundation/doc.html

```
<category>
  <pattern>MI CHIAMO *</pattern>
  <template>
    Ciao <star/>, piacere di conoscerti!
  </template>
</category>
```

Codice 1: Esempio di AIML per la gestione di un saluto

1.6 Limitazioni di AIML

- Le strategie di wildcard e pattern matching restano **prevalentemente letterali**: Se una frase si discosta dal pattern previsto, il sistema fallisce il matching
- La **gestione del contesto** (via `<that>`, `<topic>`, `<star>`, ecc.) è rudimentale
- L'integrazione (via `<sraix>`) con **basi di conoscenza esterne** (KB, database, API) è possibile implementando funzioni personalizzate, ma è di difficile gestione
- Le risposte generate sono **statiche e predefinite**, e non possono essere generate dinamicamente in base a dati esterni o a contesti più ampi in modo automatico

- Sviluppare un sistema di dialogo che superi le limitazioni di AIML evidenziate
- Integrare tecniche di **Natural Language Understanding** (NLU) e **Retrieval-Augmented Generation** (RAG) per migliorare l'esperienza d'uso
- Assicurare una elevata facilità di estensione e personalizzazione per diversi domini e applicazioni

2. Natural Language Understanding

Il primo elemento dello stack di NLP rispetto ad AIML che vogliamo migliorare è il riconoscimento delle intenzioni dell'utente.

- Non useremo più un sistema basato su pattern matching ed espressioni regolari
- Riconosceremo la categoria di interazione affidandoci ad un classificatore basato su LLM
- Le parti variabili della frase (slot) verranno estratte tramite un sistema di Named Entity Recognition (NER)

- Essendo un task supervisionato, bisogna partire con l'etichettatura dei dati.
- Il dataset utilizzato proviene dalle precedenti pubblicazioni del progetto NoVAGraphS.
- Contiene 350 interazioni degli utenti prodotte durante precedenti sperimentazioni nel dominio degli automi a stati finiti.
- L'annotazione dei dati:
  - Inizialmente è stata effettuata automaticamente
  - Successivamente è stata completamente riveduta ed effettutata manualmente
- Sono usati due livelli di granularità per la classificazione:
  - 7 **classi principali**
  - Sono state introdotte le **classi secondarie** per ogni classe principale, per un totale di 33.

Classe	Scopo	Numero di Esempi
transition	Domande che riguardano le transizioni tra gli stati	77
automaton	Domande che riguardano l'automa in generale	48
state	Domande che riguardano gli stati dell'automa	48
grammar	Domande che riguardano la grammatica riconosciuta dall'automa	33
theory	Domande di teoria generale sugli automi	15
start	Domande che avviano l'interazione con il sistema	6
off_topic	Domande non pertinenti al dominio che il sistema deve saper gestire	2

Tabella 1: Classi principali per la classificazione delle domande

Sottoclassi	Scopo	Numero di Esempi
description	Descrizioni generali sull'automa	14
description_brief	Descrizione generale (breve) sull'automa	10
directionality	Domande riguardanti la direzionalità o meno dell'intero automa	1
list	Informazioni generali su nodi e archi	1
pattern	Presenza di pattern particolari nell'automa	9
representation	Rappresentazione spaziale dell'automa	13

Tabella 2: Classi secondarie per la classe primaria dell'**Automa**

2.3 Data Augmentation

- Del dataset originale sono rimasti solo 229 esempi, divisi in classi sbilanciate.
- Per assicurare che il modello abbia buone prestazioni, è stato necessario aumentare il numero di esempi.
- Sono state generate 851 nuove domande, utilizzando LLM alle quali è stato fornito un insieme di quesiti di una certa classe.
- Per le domande off-topc è stato adoperato il dataset SQUAD [1]

[1] P. Rajpurkar, J. Zhang, K. Lopyrev, e P. Liang, «SQuAD: 100,000+ Questions for Machine Comprehension of Text», in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, J. Su, K. Duh, e X. Carreras, A c. di, Association for Computational Linguistics, nov. 2016, pp. 2383–2392. doi: 10.18653/v1/D16-1264.

- Partendo LLM pre-addestrate e con una buona padronanza della lingua inglese, l'addestramento è piuttosto rapido e non richiede molte risorse.
- È stato eseguito un fine-tuning per adattare il modello alla classificazione delle domande.
- In questo modo il modello apprende le particolarità e sfumature dello scenario applicativo specifico.
- La metrica massimizzata è stata la **F1 score** (media armonica fra Precision e Recall)

$$\mathrm{F1} = 2 \cdot \frac{\mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}$$

I modelli utilizzati per il fine-tuning sono stati:

- `google-bert/bert-base-uncased`, versione da 110 milioni di parametri [1] (Google)
- `distilbert/distilbert-base-uncased` [2], versione distillata di BERT, con circa il 40% in meno di parametri [3]
- `google/mobilebert-uncased` [4] con 25 milioni di parametri; per dispositivi mobili
- `google/electra-small-discriminator` [5], da 14 milioni di parametri

---

[1] *Bert uncased model by Google*. [Online]. Disponibile su: https://huggingface.co/google-bert/bert-base-uncased

[2] *Distilbert base model*. [Online]. Disponibile su: https://huggingface.co/distilbert/distilbert-base-uncased

[3] V. Sanh, L. Debut, J. Chaumond, e T. Wolf, «DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter», 2020. doi: 10.48550/arXiv.1910.01108.

[4] *MobileBERT on Huggingface*. [Online]. Disponibile su: https://huggingface.co/google/mobilebert-uncased

[5] *Electra on Huggingface*. [Online]. Disponibile su: https://huggingface.co/google/electra-small-discriminator

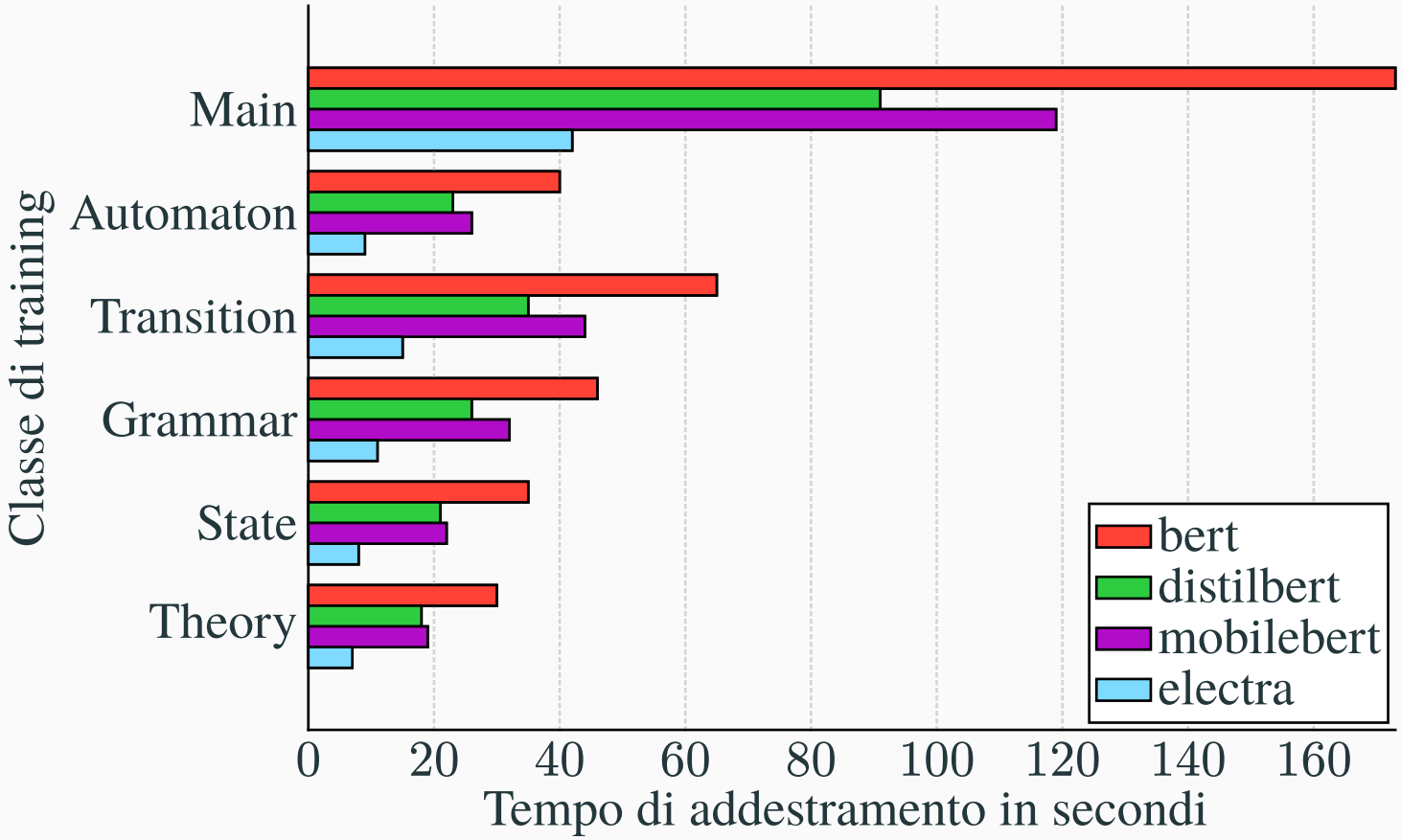


Figura 1: Tempo di addestramento per BERT, DistilBERT, MobileBERT ed ELECTRA

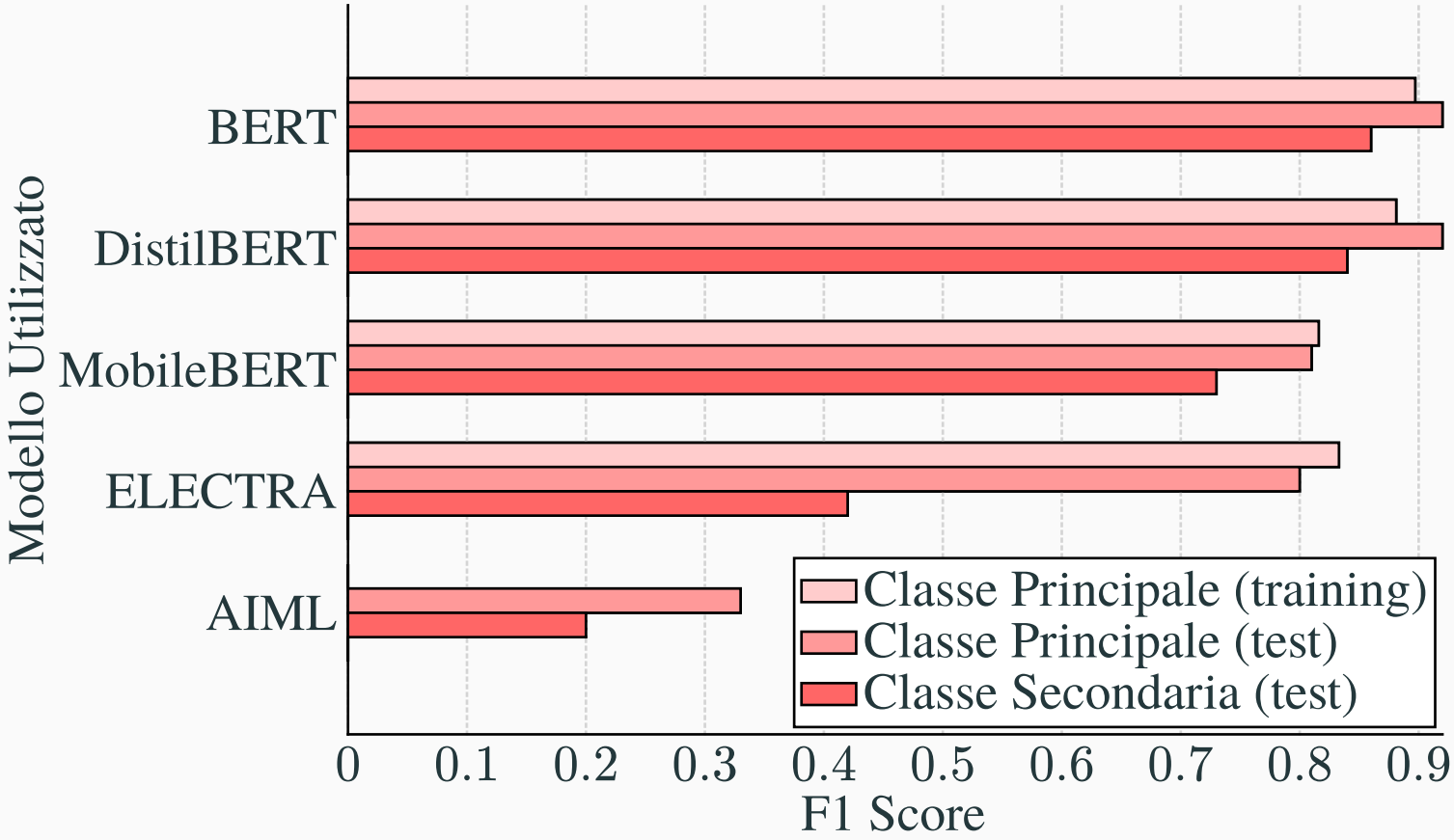


Figura 2: Performance su test set bilanciato da 468 ulteriori domande confrontato con AIML

	Performance AIML			Performance BERT			
	Precision	Recall	F1-score	Precision	Recall	F1-score	Esempi
Automaton	0.52	0.19	0.27	0.93	0.93	**0.93**	75
Grammar	0.90	0.13	0.23	0.78	0.83	**0.81**	70
Off-Topic	0.26	0.82	0.39	1.00	0.96	**0.98**	100
Start	1.00	0.53	0.69	1.00	0.90	**0.95**	40
State	0.17	0.19	0.18	0.96	1.00	**0.98**	43
Theory	0.00	0.00	0.00	0.57	0.57	**0.57**	30
Transition	0.67	0.28	0.40	0.97	0.99	**0.98**	110
Accuracy			0.35			**0.92**	468
Macro avg	0.44	0.27	0.27	0.89	0.88	**0.88**	468
Weighted avg	0.53	0.35	0.33	0.92	0.92	**0.92**	468

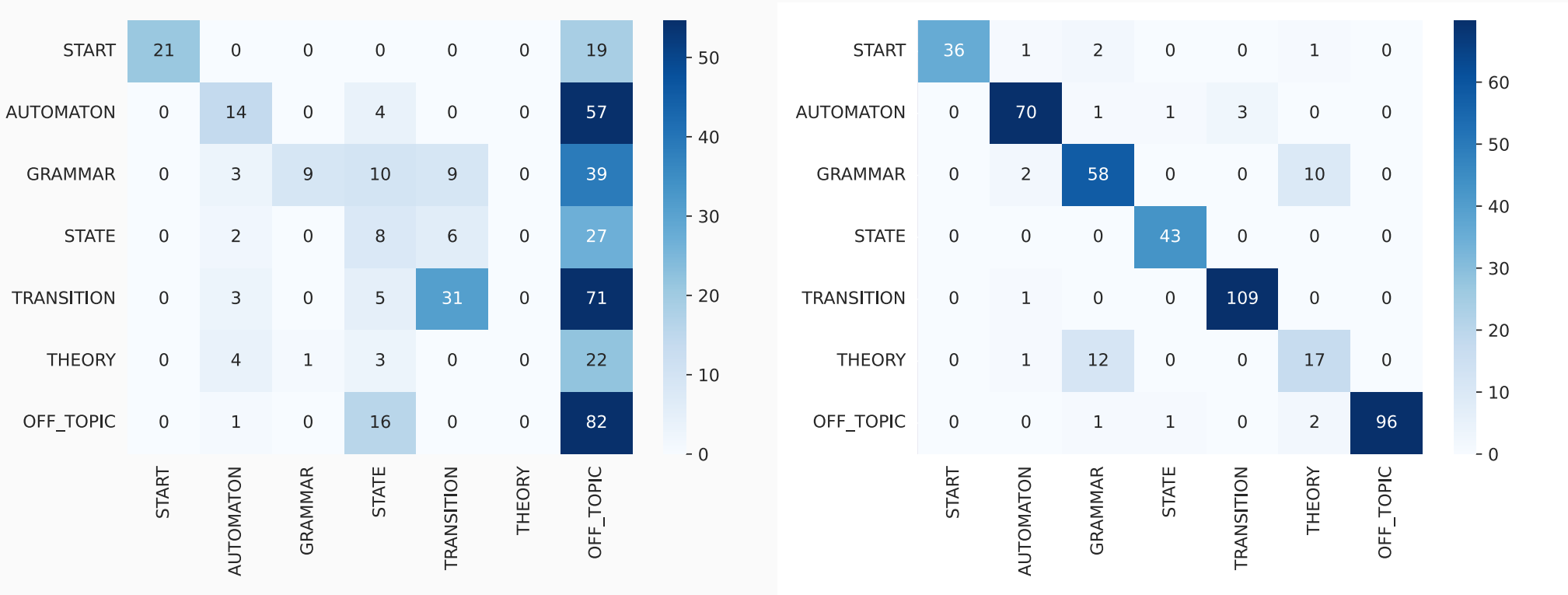


Figura 3: Matrici di confusione per le classi principali con AIML e BERT

3. Retrieval Augmented Generation

- Per fornire risposte complete e pertinenti, il sistema deve essere in grado di interrogare una base di conoscenza esterna.
- Esistono diverse tecniche, che dipendono dal tipo di dati e dalla loro rappresentazione:
  - Basi di conoscenza strutturate: SQL, SPARQL, ecc.
  - Corpus: ricerca full-text, TF-IDF, embeddings…
  - API e servizi esterni: REST, JMESPath

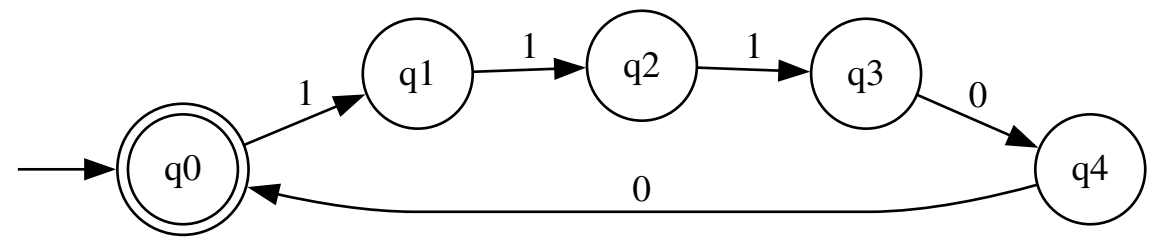
```
digraph FSA {
    rankdir=LR;
    node [shape = circle];
    q0 [shape = doublecircle];
    q1; q2; q3; q4;

    start [shape=none, label=""];
    start -> q0;

    q0 -> q1 [label = "1"];
    q1 -> q2 [label = "1"];
    q2 -> q3 [label = "1"];
    q3 -> q4 [label = "0"];
    q4 -> q0 [label = "0"];
}
```

Snippet 1: Rappresentazione in formato Graphviz dell'automa a stati finiti utilizzato come input per le domande.

- Dobbiamo assicurarci che il sistema sia in grado di rispondere in modo non solo coerente e pertinente, ma anche efficace alle domande degli utenti.
- Per effettuare le valutazioni sono stati utilizzati diversi LLM
- Una volta generate tutte le risposte con i vari modelli, sono state tutte annotate
- I risultati delle annotazioni hanno fornito dettagli essenziali per la scelta del modello finale



Se il tema della domanda è riguardante le *transizioni uscenti da un nodo*, prima di generare la risposta, il sistema recupera le i dettagli utili e li presenta al modello.

```
The transitions exiting from the node are the following:
- From qo to q1, with label '1'
```

Gli annotatori sono liberi di evidenziare nelle risposte frammenti problematici semplicemente selezionandoli. Sono stati definiti quattro generi di errori:

- **INCORRECT**$^{\text{I}}$: la risposta contiene informazioni che contraddicono i dati forniti o che sono chiaramente sbagliate.
- **NOT_CHECKABLE**$^{\text{NC}}$: la risposta contiene informazioni che non possono essere verificate con i dati forniti.
- **MISLEADING**$^{\text{M}}$: la risposta contiene informazioni fuorvianti o che possono essere interpretate in modo errato.
- **OTHER**$^{\text{O}}$: la risposta contiene errori grammaticali, stilistici o di altro tipo.

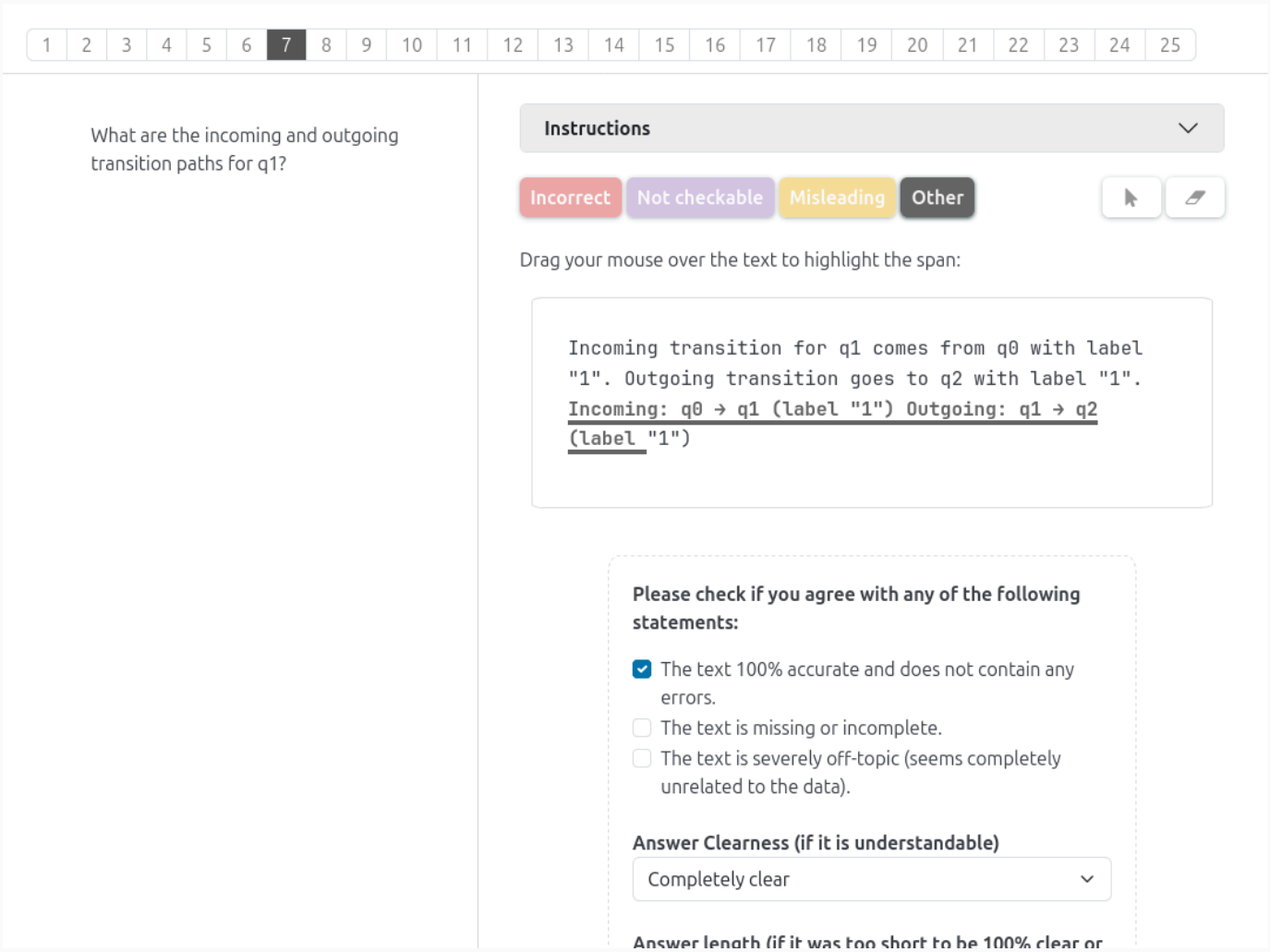


Figura 5: Interfaccia di Factgenie

Oltre ad evidenziare parti problematiche delle risposte, è stato richiesto agli annotatori anche di fornire delle valutazioni qualitative su alcune metriche:

- **Chiarezza della risposta**: se è comprensibile e ben strutturata;
- **Lunghezza della risposta**: se la comprensione della risposta è facilitata dalla sua lunghezza (o brevità);
- **Utilità percepita della risposta**: se la risposta è utile e fornisce informazioni rilevanti;
- **Apprezzamento generale**: se la risposta è apprezzata o gradita.

In totale, 12 annotatori hanno partecipato alla valutazione delle risposte generate dai modelli. Riguardo gli annotatori:

- 8 su 12 sono studenti del Dipartimento di Informatica;
- 2 hanno un background ingegneristico;
- I rimanenti 2 provengono dal Dipartimento di Storia e Biologia.

L'età media è di 28 anni, con un range tra i 21 e i 68 anni.

Ogni volontario ha valutato un sottoinsieme delle risposte, garantendo comunque un overlap sulle 75 risposte totali.

Oltre agli annotatori umani, sono stati utilizzati anche due modelli di LLM (`GPT-o3-mini` e `GPT-4.5`) per svolgere automaticamente una valutazione simile a quella dei volontari (ε_{hum}).

Modello	Incorrect		Not Checkable		Misleading		Other		Globale	
	ε_{hum}	$\varepsilon_{4.5}$	ε_{hum}	$\varepsilon_{4.5}$	ε_{hum}	$\varepsilon_{4.5}$	ε_{hum}	$\varepsilon_{4.5}$	ε_{hum}	$\varepsilon_{4.5}$
Deepseek	**20.0%**	**13.3%**	6.67%	**0.0%**	**26.7%**	6.7%	66.7%	**0.0%**	83.0%	**33.3%**
Gemma2	26.7%	26.7%	**0.0%**	**0.0%**	33.3%	**0.0%**	**20.0%**	6.7%	**30.6%**	**33.3%**
Llama3.1	33.3%	33.3%	6.67%	**0.0%**	**26.7%**	6.7%	46.7%	**0.0%**	67.3%	53.3%
GPT-4o	20.0%	**6.67%**	13.3%	**0.0%**	20.0%	**0.0%**	40.0%	**0.0%**	36.0%	**6.67%**
GPT-o3-mini	**0.0%**	13.3%	**0.0%**	**0.0%**	**13.3%**	**0.0%**	**20.0%**	**0.0%**	**18.6%**	20.0%

Tabella 3: Percentuali di *risposte contenenti almeno un errore*, secondo le annotazioni umane (ε_{hum}) e le valutazioni automatiche ($\varepsilon_{4.5}$). Più basso è il valore, migliore è la qualità delle risposte.

Metrica	LLM Commerciale		LLM Open-Weights	
Chiarezza	GPT-o3-mini	93%	Deepseek-r1:8b	69%
Lunghezza	GPT-o3-mini	96%	Deepseek-r1:8b	78%
Utilità	GPT-o3-mini	98%	Deepseek-r1:8b	82%
Apprezzamento	GPT-o3-mini	95%	Deepseek-r1:8b	68%

Tabella 4: Percentuali di valutazione delle risposte generate dai modelli LLM commerciali e open-weights.

3.4 Risultati

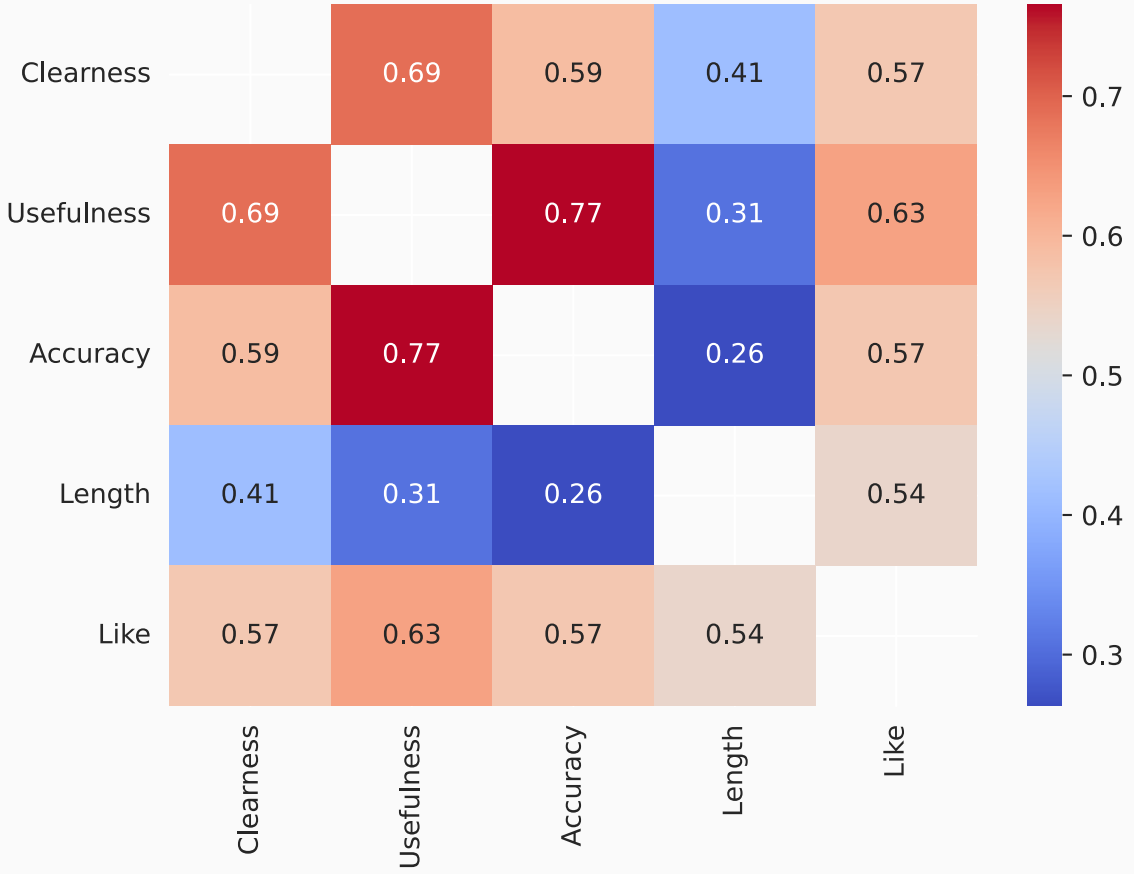


Diagramma 1: Correlazione tra le metriche di valutazione delle risposte per ogni modello.

4. Ingegnerizzazione del sistema

4.2 Compilatore

- Dal momento che dobbiamo lavorare con dei LLM, è necessario un sistema che effettui fine-tuning per ogni insieme di interazioni. Potremmo lasciare il fine-tuning a runtime, ma prepararlo in anticipo permette di risparmiare tempo e risorse.
- È sufficiente descrivere in modo sequenziale (pipeline) le operazioni da eseguire, e il «compilatore» si occuperà di preparere tutto il necessario

```yaml
- name: "question_intent_classifiers"
    type: classification
    steps:
      - type: load_csv
        name: data
        path: "./multitask_training/data_cleaned_manual_combined.csv"
        label_columns: ["Global Subject", "Question Intent"]
      - type: split_data
        name: split
        dataframe: data.dataframe
        on_column: "Global Subject"
        for_each:
          - type: train_model
            name: training
            dataframe: split.dataframe
            pretrained_model: "google/electra-small-discriminator"
            examples_column: "Question"
            labels_column: "Question Intent"
            resulting_model_name: "question_intent_{on_column}"
```

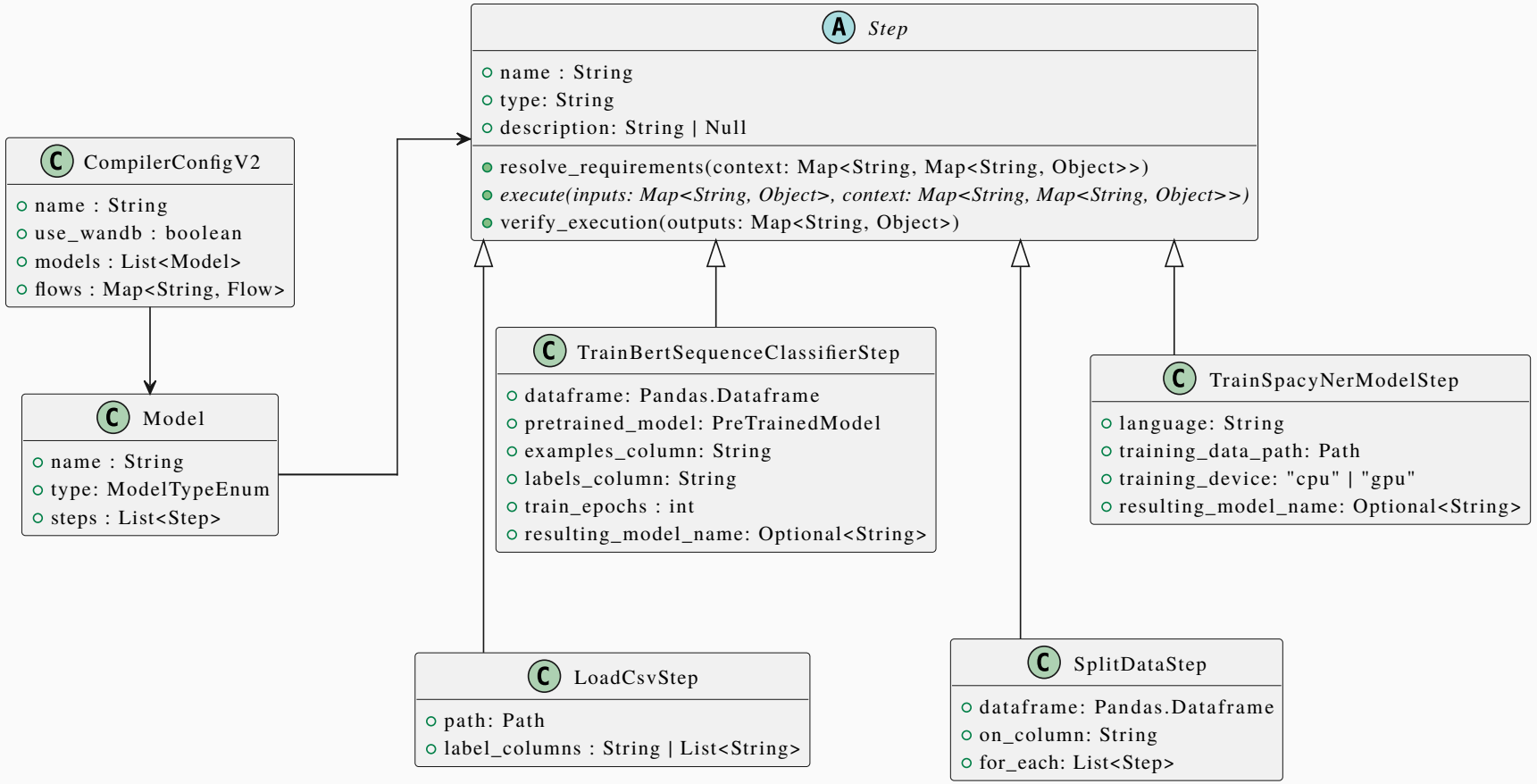


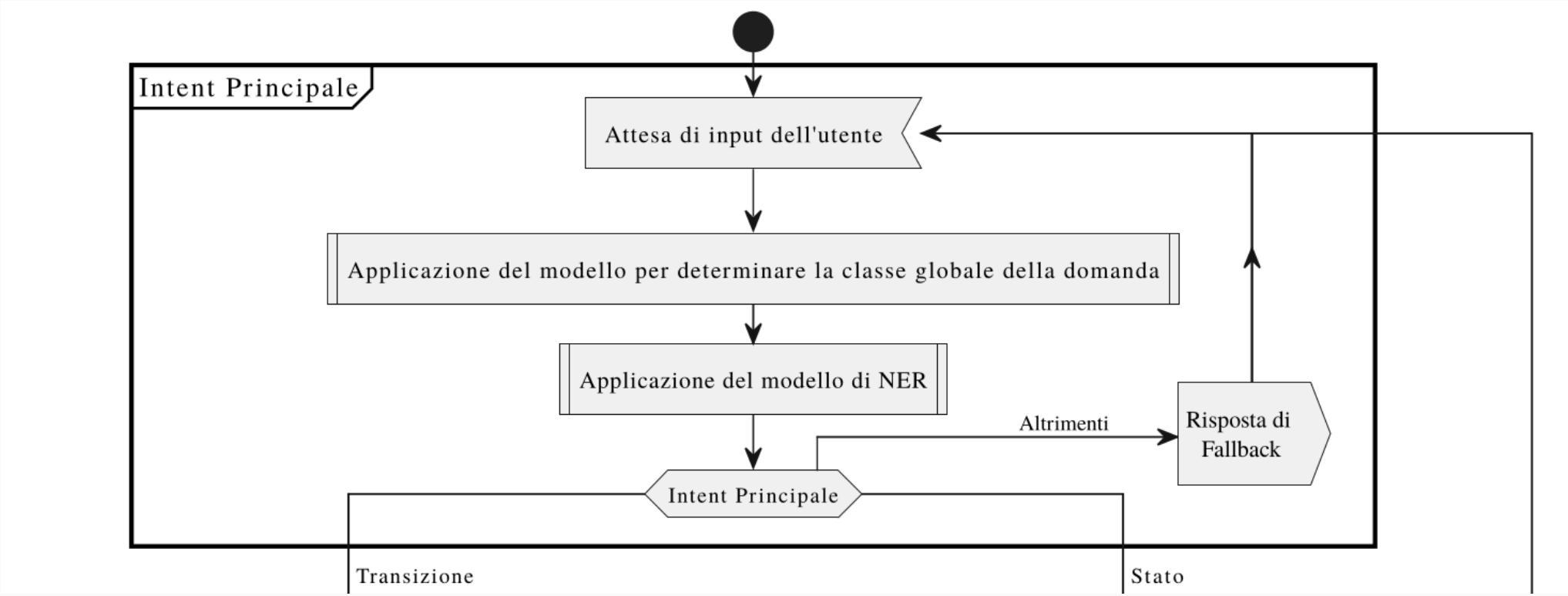
Diagramma 2: Class Diagram raffigurante le classi e proprietà utilizzate per la compilazione.

4.3 Runner

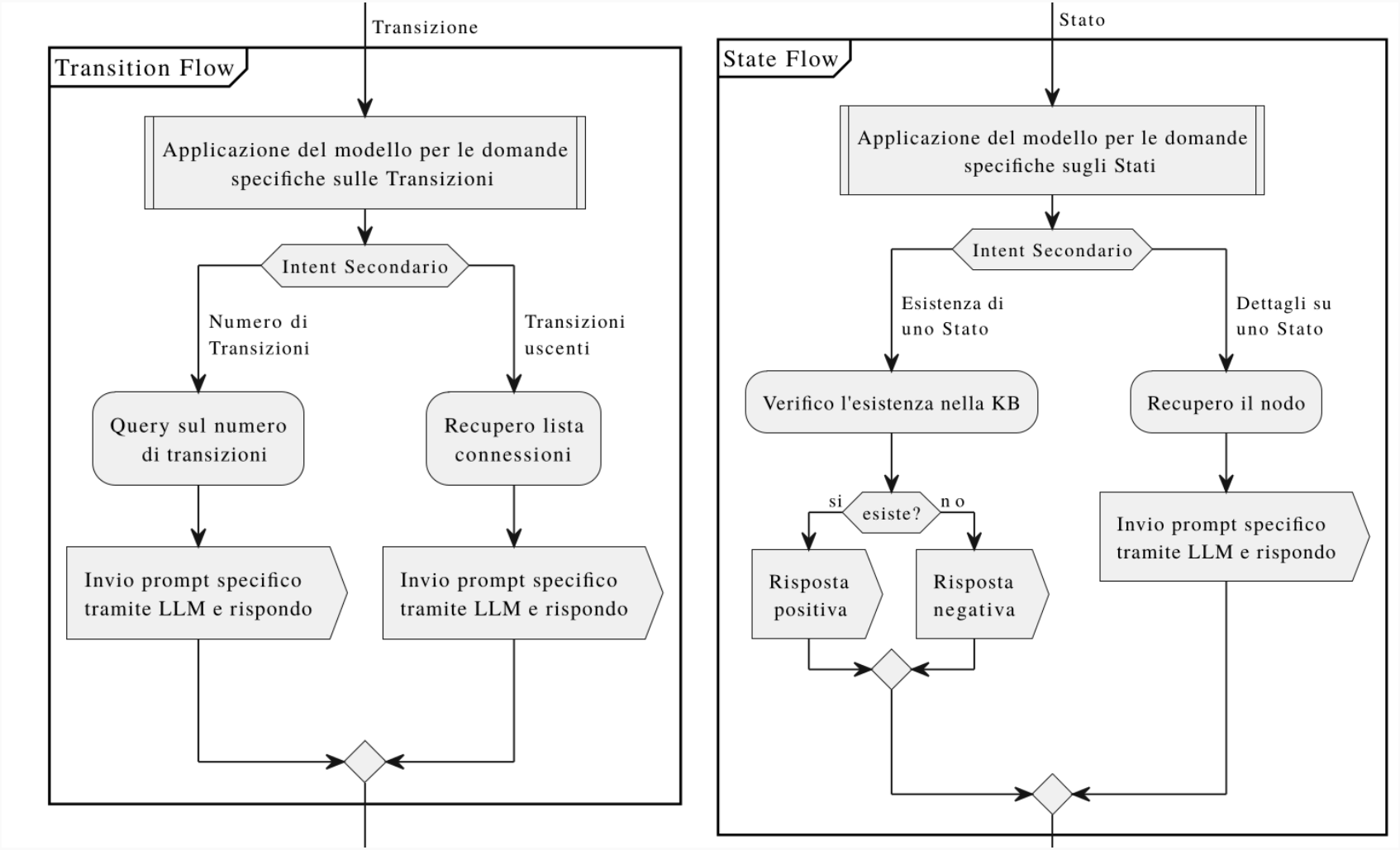
Come per AIML, è necessario un motore di esecuzione del chatbot, che si occupi di seguire il flusso di esecuzione delle interazioni e di gestire le domande degli utenti.

- Il motore di esecuzione è un automa a stati finiti
- Il percorso di interazione a ogni nodo codifica un'azione che il motore deve svolgere:
  - Lasciare la parola all'utente
  - Recuperare informazioni da risorse (DB, API, ecc.)
  - Generare una risposta (LLM, template, default, ecc.)
- Un insieme di nodi è definito come un *flusso* di interazione (flow), che può essere richiamato in qualsiasi momento.

4.4 Esempio di chatbot



4.4 Esempio di chatbot



5. Conclusioni

- **Riconoscimento Intenzioni**: pattern matching ⟹ classificatori neurali o basati su regole
- **Gestione Contesto**: `<topic>` e `<that>` ⟹ maggiore dettaglio e salvataggio contesto;
- **Dati Esterni**: `<sraix>` ⟹ integrazione nativa di moduli di retrieval;
- **Controllo del Flusso**: `<condition>` e `<srai>` (salti semplici) ⟹ branching condizionale, passaggio dinamico tra flussi di interazione.
- **Approccio ibrido**: dichiaratività di AIML e flessibilità delle reti neurali.
- **Controllo totale**: Definire con precisione l'ordine delle interazioni evita limiti tipici dei LLM, come explainability ridotta, allucinazioni e jailbreaking.

L'utilizzo degli step permette di avere la massima flessibilità, astraendo comunque il sistema da dettagli implementativi.

- Libreria di valutazione delle espressioni (Asteval) che consente di eseguire codice Python a runtime, garantendo grande flessibilità
- Assenza di protezioni predefinite contro codice malevolo rende necessaria una sandbox
- Servono controlli più stringenti per garantire sicurezza e stabilità

5.2 Sviluppi futuri

- YAML spinto al limite: ottimo per flow semplici, ma diventa complesso per espressioni avanzate
- Possibilità di introdurre un DSL dedicato per maggiore chiarezza e validazione

- Definire le configurazioni direttamente in Python abbasserebbe la barriera di ingresso per sviluppatori
- Supporto IDE (PyCharm, VS Code) migliorerebbe validazione e velocità di sviluppo
- Mantenere compatibilità YAML per utenti meno esperti

Grazie per l’attenzione!
Domande?