Universitatea Națională de Știință și Tehnologie POLITEHNICA București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Robot telefonic inteligent cu sumarizarea conversațiilor telefonice și estimarea satisfacției clientului

Lucrare de dizertație

prezentată ca cerință parțială pentru obținerea titlului de *Master* în domeniul *Inginerie electronică, telecomunicații și tehnologii informaționale*
programul de studii de masterat *Tehnologii multimedia în aplicații de biometrie și securitatea informației (BIOSINF)*

Conducător(i) științific(i):
Conf. dr. ing. Horia CUCU
Dr. ing. Alexandru Lucian GEORGESCU

Absolvent:
Antonia DUMITRU

2026

Universitatea Națională de Știință și Tehnologie POLITEHNICA București
Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei
Programul de masterat **BIOSINF**

Anexa 2

TEMA LUCRĂRII DE DISERTAȚIE

a masterandului **DUMITRU V.Ş. Antonia, 421-BIOSINF**

1. Titlul temei: Robot telefonic inteligent cu sumarizarea conversațiilor telefonice și estimarea satisfacției clientului

2. Descrierea temei:
Context
Sistemele de tip voicebot sunt utilizate frecvent în centrele de suport pentru a automatiza relația cu clienții. Totuși, analiza conversațiilor rămâne adesea manuală și consumatoare de timp. Modelele lingvistice mari pot contribui la automatizarea acestei analize prin generarea de rezumate, extragerea intențiilor și estimarea satisfacției clienților.
Descriere proiect
Proiectul urmărește dezvoltarea unui robot telefonic inteligent care realizeaza:
1. Extragerea intențiilor utilizatorului în limba română și 2.Evaluarea conversației folosind LLM-uri pentru:
(i) generarea un sumar concis al conversației și (ii)
estimarea nivelul de satisfacție al clientului (pozitiv / neutru / negativ) pe baza întregii discuții.
Proiectul va integra modulele într-o aplicație vocală demonstrativă construită pe infrastructură
Twilio, Zevo STT, LLM, Zevo TTS.

3. Contribuția originală:
Activități principale
A1. Sumarizarea principalelor LLM-uri care pot procesa texte în limba română.
A2. Colectarea/generarea unei baze de date de conversații robot-utilizator în limba română.
A3. Crearea și rafinarea de prompt-uri pentru extragerea automată a intenției.
A4. Crearea și rafinarea de prompt-uri pentru sumarizare conversațională
. A5. Crearea și rafinarea de prompt-uri pentru estimarea satisfacției
. A6. Evaluarea mai multor LLM-uri (API și locale) pe setul de conversatii
. A7. Integrarea Twilio Zevo STT Zevo TTS LLM într-o aplicație demo funcțională
. A8. Evaluarea acurateței vs. resurse vs. timp de răspuns și documentarea rezultatelor.
*Cerințe: *
Programare Python,
Cunoștințe de NLP și LLM prompting,
Familiaritate cu API-uri LLM și cu tehnologii STT/TTS,
Noțiuni de analiza conversatiilor și prelucrare text,
Experiență în lucrul cu containere Docker și medii Linux,
Familiaritate cu platforma Twilio și cu serviciile Zevo Tech.

4. Resurse și bibliografie:
Liu, P. et al. – Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in NLP, ACM Computing Surveys, 2023.
Bhandari, Prabin et al. A Survey on Prompting Techniques in LLMs. 2023.
Vatsal, Shubham, Dubey, Harsh. A Survey of Prompt Engineering Methods in Large Language Models for Different NLP Tasks. 2024.
Goyal, Tanya, Junyi Jessy Li, and Greg Durrett. News summarization and evaluation in the era of gpt-3. arXiv preprint arXiv:2209.12356 (2022).
Retkowski, Fabian, et al. "Summarizing speech: A comprehensive survey." Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing. 2025.
Lin, Ying-Chun, et al. "Interpretable user satisfaction estimation for conversational systems with large language models." Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2024.
ICVSI, Laborator 6, Biosinf, UNSTPB.
Twilio – Programmable Voice API Documentation.

5. Data înregistrării temei: 2026-01-15 18:54:00

Conducător(i) lucrare,
Conf. dr. ing. Horia CUCU

Dr. Lucian Georgescu

Student,
DUMITRU V.Ş. Antonia

Responsabil program master,
Prof. dr. ing. Dragoş BURILEANU

Decan,
Prof. dr. ing. Mihnea UDREA

Cod Validare: **2508a843ad**

Declarație de onestitate academică

Prin prezenta declar că lucrarea cu titlul *Robot telefonic inteligent cu sumarizarea conversațiilor telefonice și estimarea satisfacției clientului*, prezentată în cadrul Facultății de Electronică, Telecomunicații și Tehnologia Informației a Universității Naționale de Știință și Tehnologie POLITEHNICA București drept cerință parțială pentru obținerea titlului de *Master* în domeniul *Inginerie electronică, telecomunicații și tehnologii informaționale*, programul de studii de masterat *Tehnologii multimedia în aplicații de biometrie și securitatea informației (BIOSINF)*, este scrisă de mine și nu a mai fost prezentată niciodată la o facultate sau instituție de învățământ superior din țară sau străinătate.

Declar că toate sursele utilizate, inclusiv cele de pe Internet, sunt indicate în lucrare ca referințe bibliografice. Fragmentele de text din alte surse, reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și fac referire la sursă. Reformularea în cuvinte proprii a textelor scrise de către alți autori face referire la sursă. Înțeleg că plagiatul constituie infracțiune și se sancționează conform legilor în vigoare.

Declar că toate rezultatele simulărilor, experimentelor și măsurătorilor pe care le prezint ca fiind făcute de mine, precum și metodele prin care au fost obținute, sunt reale și provin din respectivele simulări, experimente și măsurători. Înțeleg că falsificarea datelor și rezultatelor constituie fraudă și se sancționează conform regulamentelor în vigoare.

București, 15.06.2026

Absolvent: Antonia DUMITRU

ADumitru

(semnătura în original)

CUPRINS

LISTA TABELELOR

LISTA FIGURILOR

LISTA ABREVIERILOR

RNN	Recurrent Neural Netweork
LSTM	Long Short-Term Memory
LLM	Large Language Model
NLP	Natuaral Language Processing
IVR	Interactive Voice Respons
MLM	Masked Language Modeling
BERT	Bidirectional Encoder Representations from Transformers
ReLU	Rectified Linear Unit
API	Application Programming Interface
MMLU	assive Multitask Language Understanding
TTFT	Time To First Token
AI	Artificial Intelligence
AIME	American Invitational Mathematics Examination
GPQA	raduate-Level Google-Proof Q&A
RoPE	Rotary Positional Embedding
NoPE	No Positional Embedding
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
GDPR	General Data Protection Regulation
CoT	Chain of Thought
CPaaS	Communications Platform as a Service
GPT	Generative Pre-trained Transformer

1. INTRODUCERE

Lucrarea de față are ca scop implementarea unui robot telefonic care să proceseze și să gestioneze conversațiile din cadrul call-center-urilor. Instituțiile statului, dar și companiile private, pun la dispoziția cetățenilor și a clienților lor acest serviciu pentru a putea îmbunătăți serviciile oferite, pentru a oferi suport tehnic și pentru a lămuri eventualele neclarități care pot apărea. De cele mai multe ori, produsele și serviciile pe care le utilizăm zi de zi ne cauzează probleme pe care nu știm să le rezolvăm sau care nu pot fi rezolvate de către utilizatori, iar una dintre cele mai rapide soluții este apelarea companiei care furnizează aceste servicii.

1.1. Motivație

Industria centrelor de relații cu clienții reprezintă unul dintre cele mai mari sectoare economice la nivel global, angajând milioane de operatori și gestionând zilnic sute de milioane de interacțiuni vocale. Piața globală a inteligenței artificiale aplicată în call-centere a fost evaluată la 3,23 miliarde de dolari în 2024 și este preconizată să atingă aproximativ 25,84 miliarde de dolari până în 2034 [1], reflectând presiunea tot mai mare a companiilor de a automatiza procesele de suport și de a îmbunătăți experiența clienților fără a crește costurile operaționale.

LLM-urile (Large Language Models) au transformat în ultimii ani modul în care sistemele software înțeleg și generează text în limbaj natural. Capacitatea acestora de a extrage intenții, de a analiza sentimente și de a genera rezumate coerente le face soluții utile pentru automatizarea proceselor repetitive care apar în relațiile cu clienții. Prin aplicații precum chatboți sau asistenți vocali, analiza sentimentelor și transcriere automată, procesarea limbajului natural îmbunătățește eficiența operațională și poate crește nivelul de satisfacție clienților [2].

Cu toate acestea, majoritatea cercetărilor existente se concentrează pe limbile cele mai uzuale, în special engleza. Limba română rămâne o limbă cu resurse limitate în contextul NLP, fapt care ridică întrebări despre cât de bine se comportă modelele de mari dimensiuni pe sarcini specifice în română, comparativ cu modele mai mici specializate pentru această limbă. Inițiativa *Vorbești Românește?* a introdus în 2024 un benchmark pentru evaluarea modelelor lingvistice românești [3], avansând semnificativ capacitățile LLM-urilor pentru limba română, însă evaluarea acestora în scenarii specifice, cum ar fi call-centerele, rămâne un domeniu care trebuie explorat.

1.2. Obiective

Numărul foarte mare de interacțiuni gestionate zilnic de call-centere generează volume semnificative de date conversaționale. Aceste conversații conțin informații valoroase despre problemele întâmpinate de clienți, nivelul lor de satisfacție și eficiența serviciilor oferite de companii.

Analiza manuală a unui astfel de volum de date este dificilă și costisitoare, motiv pentru care utilizarea modelelor lingvistice pentru extragerea automată a informațiilor relevante poate prezenta interes pentru companii. Astfel vor fi definite trei sarcini specifice conversațiilor de call-center în limba română:

- Construirea unui dataset sintetic de 100 de conversații telefonice în limba română, distribuit pe câteva domenii distincte, cu adnotări pentru intenție, satisfacție și rezumate de referință.
- Evaluarea și compararea tehnicilor de prompt engineering pe cele trei sarcini, identificând configurația optimă pentru fiecare model și sarcină.
- Implementarea unui robot telefonic funcțional care integrează modulele de Speech-to-Text, procesare LLM și Text-to-Speech într-un pipeline complet, cu posibilitatea de conectare la rețeaua telefonică prin platforma Twilio.
- Evaluarea performanței sistemului atât pentru procesarea discuțiilor bazate pe text, cât și a celor bazate pe înregistrări transcrise și compararea diferențelor în rezultate dintre cele două.

1.3. Aplicabilitate

Sistemul propus răspunde unei nevoi reale din industria serviciilor: reducerea încărcării operatorilor umani prin automatizarea interacțiunilor din call-centere. Studiile arată că sistemele bazate pe inteligență artificială pot reduce timpul mediu de procesare a unui apel cu aproximativ 25%, îmbunătățind atât eficiența operațională, cât și timpul de răspuns [4].

Spre deosebire de sistemele IVR tradiționale, care ghidează utilizatorul prin meniuri rigide, robotul telefonic propus în această lucrare este capabil să înțeleagă limbajul natural și să genereze răspunsuri adaptate contextului. O contribuție importantă este și utiizarea limbii române, precum și evaluarea comparativă a mai multor modele în condiții identice.

În plus, aceste sisteme pot transforma conversațiile telefonice din simple interacțiuni de suport într-o sursă valoroasă de informații pentru organizații. Intențiile clienților, nivelul de satisfacție și rezumatele automate pot fi utilizate pentru monitorizarea calității serviciilor, identificarea problemelor recurente și luarea unor decizii mai bune la nivel de business.

2. CERCETARE

2.1. Identificare LLM-uri

Înțelegerea și generarea limbajului natural reprezintă una dintre cele mai solicitate direcții de cercetare și aplicare în domeniul inteligenței artificiale. Agenții utilizați în mod constant în sarcini uzuale și nu numai, se bazează pe capacitatea de a interpreta corect informațiile oferite de utilizatori și de a transforma acest limbaj natural în acțiuni și raspunsuri concrete [5].

Până în 2017, printre cele mai utilizate modele pentru înțelegerea limbajului natural se numărau rețelele neuronale recurente (RNN) și LSTM-urile (long short-term memory) [6], ambele având performanțe remarcabile și fiind considerate state-of-the-art în domeniu [7]. Principalele limitări ale acestor modele constau în dificultatea de modelare a secvențelor lungi și în procesarea secvențială a textului, astfel Vaswani et al. [8] propun arhitectura Transformer, bazată pe mecanismul de self-attention [9], care permite accesul direct la toate pozițiile dintr-o secvență, eliminând procesarea secvențială și reducând costurile asociate prelucrării textului.

Odată cu introducerea Transformer-ului, a devenit posibilă construirea de modele de limbaj de dimensiuni foarte mari (Large Language Models), capabile să învețe relații complexe și contexte pe termen lung din cantități mari de text. Printre primele modele de acest tip se numără GPT-1, dezvoltat de OpenAI în 2018. Radford et al. [10] au propus o abordare semi-supervizată pentru sarcinile de procesare a limbajului natural, combinând pre-antrenarea nesupervizată cu fine-tuning-ul supervizat. GPT-1 utilizează o arhitectură Transformer [8] și este antrenat să prezică următorul cuvânt dintr-o secvență, fapt care îi permite să genereze text, fiind adaptat ulterior în îndeplinirea unor diverse sarcini.

Arhitecturile de tip Transformer pot fi clasificate în trei categorii, în funcție de modul care procesează și generează informația:

- Encoder-Only (doar codificator),
- Decoder-Only (doar decodor),
- Encoder-Decoder (și codificator și decodor).

În lucrarea *Attention Is All You Need* [8], care introduce pentru prima dată arhitectura de tip Transformer, este utilizată o structură compusă atât dintr-un codificator (encoder) și un decodor (decoder), așa cum este ilustrat în Figura 2.1.

Codificatorul, responsabil de înțelegerea și reprezentarea textului de intrare, este alcătuit din mai multe straturi identice, fiecare strat fiind format, la rândul său, din două substraturi:

- un substrat de *multi-head self-attention*;
- un substrat complet conectat de tip *feed-forward* [11].

Niciunul dintre aceste substraturi nu codifică în mod explicit ordinea sau poziția token-urilor din secvența de intrare. Pentru ca modelul să poată ține cont de această informație, este adăugat un vector de positional embeddings corespunzător poziției fiecărui token [12]. Primul substrat

se bazează pe mecanismul de atenție cu produs scalar, definit prin relația:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

unde Q reprezintă matricea interogărilor (Query), K matricea cheilor (Key), V matricea valorilor (Value), iar d_k este dimensiunea vectorilor cheie. Aceste trei matrici sunt obținute la ieșiriea stratului anterior. În cazul mecanismului de multi-head attention, acest proces este aplicat în paralel, fiecare token interacționând simultan cu celelalte token-uri.

Fiecăruia dintre cele două substraturi îi sunt adăugate conexiuni reziduale, urmate de un mecanism de normalizare, cu scopul de a atenua problema dispariției gradientului [13], frecvent întâlnită în arhitecturile adânci. Această problemă apare ca urmare a micșorării a gradientului în timpul antrenării pe perioade îndelungate. În arhitectura originală, codificatorul este compus din șase straturi, iar componenta feed-forward este formată din două straturi complet conectate, utilizând funcția de activare ReLU [14].

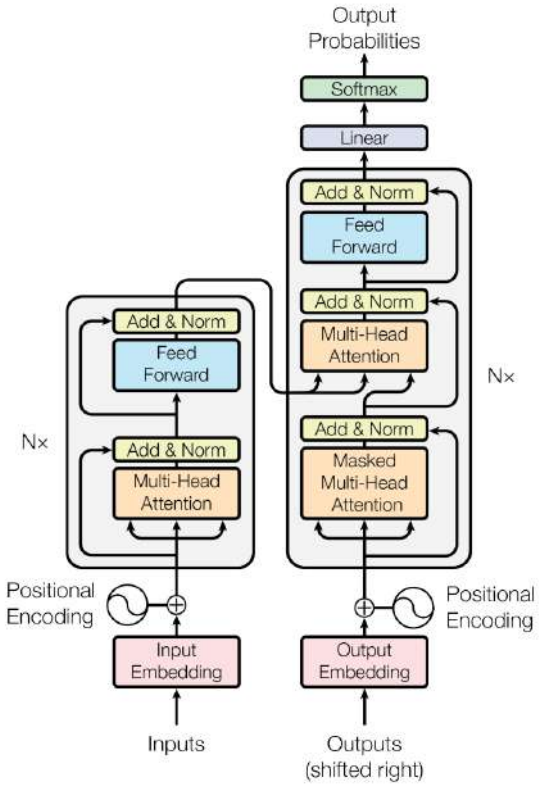

Figura 2.1: Arhitectura de tip Transformer Encoder-Decoder [8]

Structura decodorului este similară cu cea a codificatorului, însă include un substrat suplimentar de atenție care preia informația provenită de la ieșirea codificatorului. De asemenea, substratul de atenție al decodorului este modificat astfel încât, prin mascarea token-urilor viitoare și deplasarea acestora cu o poziție, predicția fiecărei poziții să depindă doar de cele anterioare [8].

Totuși, în funcție de scopul fiecărui model, una dintre cele două componente ale arhitecturii Transformer poate fi eliminată. Cele mai cunoscute modele care utilizează doar codificatorul sunt cele din familia BERT (Bidirectional Encoder Representations from Transformers). Acestea sunt antrenate folosind o metodă specifică numită Masked Language Modeling (MLM), care

constă în mascarea anumitor token-uri din textul de intrare, astfel încât modelul să învețe să prezică cuvintele lipsă [15]. BERT a înregistrat performanțe superioare într-o gamă largă de sarcini de procesare a limbajului natural.

Pe de altă parte, modelele de tip decodor, precum GPT, sunt antrenate folosind metode de Causal Language Modeling, care prezic token-ul următor pe baza tuturor token-urilor anterioare [16]. Cu toate acestea, la baza, un model encoder-only nu diferă semnificativ de unul decoder-only. Principala diferență constă în recursivitate: modelele encoder-only nu dispun de acest mecanism și, prin urmare, nu pot fi utilizate într-un mod auto-regresiv [17].

În prezent, utilizatorii au acces la o multitudine de agenți inteligenți performanți, care pot fi utilizați în orice moment și într-o gamă largă de contexte. Dezvoltarea rapidă a acestor sisteme a permis integrarea lor în activitățile de zi cu zi, unde pot răspunde la întrebări uzuale și oferi recomandări adaptate nevoilor utilizatorului.

Un aspect mai puțin dezvoltat al acestor sisteme îl reprezintă înțelegerea și generarea de răspunsuri în limba română [3]. Deși LLM-uri consacrate, precum GPT sau Gemini, pot fi utilizate direct prin interfețele lor sau apelate prin API și oferă suport pentru limba română, numărul modelelor lingvistice locale antrenate sau adaptate pentru această limbă rămâne limitat. În continuare, sunt analizate și comparate mai multe LLM-uri capabile să proceseze texte în limba română.

2.2. Analiza LLM-urilor

Pentru o analiză cât mai bună a modelelor lingvistice de mari dimensiuni, au fost selectați atât agenți performanți disponibili prin intermediul API-urilor, cât și modele care pot fi utilizate local. Astfel, din tabelul 2.1 pot fi extrase informații de baza cu privire la costurile, latența, parametrii și performanța modelelor selectate.

Evaluarea performanței modelelor se va realiza pe baza benchmark-ului MMLU, singurul utilizat pentru testarea tuturor modelelor selectate. MMLU (Massive Multitask Language Understanding) [18] reprezintă un set de date frecvent utilizat în evaluarea modelelor pentru sarcinile de procesare a limbajului natural. Deși Gema et al. [19] evidențiază anumite erori ale acestui benchmark, precum întrebările ambigue sau răspunsurile incorecte, MMLU rămâne totuși un set de date cu informații variate din domenii multiple.

Pentru măsurarea latenței modelelor, una dintre metodele cele mai cunoscute este TTFT (Time to First Token), adică timpul necesar modelului pentru a genera primul token de ieșire [20]. Deși pentru multe dintre modele acest indicator nu este raportat oficial, pasionații de AI au realizat teste independente pentru a-l evalua de-a lungul timpului.

Pentru a putea compara modelele din punct de vedere al costurilor de utilizare, sunt analizate costurile de interfață, care indică prețul per milion de tokeni de intrare și de ieșire. În cazul modelelor RoGemma și RoMistral, care reprezintă adaptări neoficiale ale modelelor de bază, sunt luate în considerare costurile originale declarate pentru Gemma și Mistral. Totuși, această modalitate de raportare a costurilor nu reflectă întotdeauna modul în care utilizatorii interacționează cu aceste sisteme în practică. Pentru publicul larg, majoritatea furnizorilor oferă abonamente lunare care permit accesul la modele, în special la variantele mai performante. Multe dintre acestea sunt disponibile și în versiuni gratuite, cu anumite limitări legate de numărul de interogări sau de procesarea imaginilor.

Tabelul 2.1: Analiza comparativă a LLM-urilor pentru procesarea limbii române.

Model	Parametri	TTFT	Cost/1M de tokeni [I/0]	MMLU
RoMistral-7b-Instruct	7B	0.13*	$0.25$0.25	49.33
RoGemma-7b-Instruct	7B	0.13*	$0.15/$0.15	52.37
Command R7B	7B	0.15s	$0.25 / $1	65.2
GPT-4.1 mini	N/A	5s	$0.4/$1.6	87.5
Gemini 2.5 Flash	N/A	2s	$0.3/$1	89.8

2.2.1. Modele bazate pe API-uri

Din categoria modelelor care poti fi apelate prin API face parte ChatGPT-4.1 mini, dezvoltat de OpenAI și lansat în aprilie 2025. Modelele din versiunea 4.1 prezintă performanțe ridicate în sarcini de codare, respectarea instrucțiunilor utilizatorului și înțelegerea contextelor lungi. Deși este un model de dimensiuni reduse, GPT-4.1 mini obține scoruri foarte bune pe benchmark-uri de raționament și cunoștințe generale, depășind varianta anterioară, GPT-4o [21].

În ceea ce privește numărul de parametri, sursele oficiale nu au publicat detalii concrete despre arhitectura sau dimensiunea modelului, însă experții și pasionații estimează că GPT-4.1 mini ar avea aproximativ 8 miliarde de parametri, similar celorlalte modele menționate în tabelul prezentat. GPT-4.1 mini prezintă costuri reduse, de aproximativ $0.4 pentru un milion de tokeni de intrare și $1.6 pentru un milion de tokeni de ieșire. Dintre toate modelele din tabelul 2.1, GPT-4.1 mini este singurul pentru care OpenAI a publicat date privind latența: aproximativ 5 secunde până la generarea primului token.

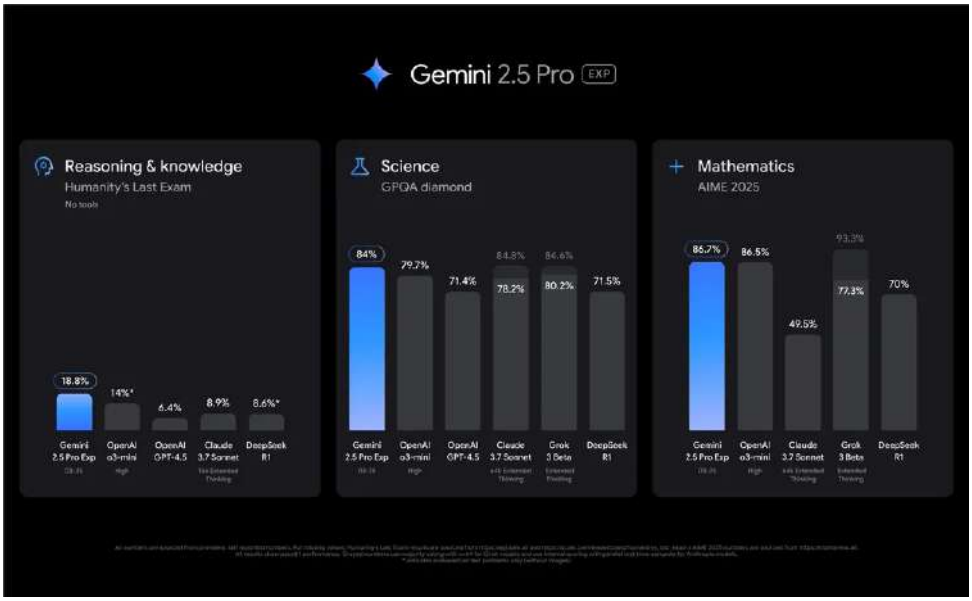

Figura 2.2: Performanțele lui Gemini 2.5 pro [22]

Un alt model popular este Gemini 2.5, lansat în martie 2025. Conform metricilor de evaluare disponibile, acesta depășește GPT-4.1 mini în ceea ce privește performanța pe setul de date MMLU, obținând un scor de aproximativ 90% [22]. Modelul se remarcă în special în sarcini care implică raționament matematic și programare. Fiind totodată un model multimodal performant, dispune de capabilități extinse de înțelegere a limbajului natural așa cum se poate observa în Figura 2.2.

Costurile pentru această variantă sunt mai avantajoase comparativ cu GPT-4.1 mini, fiind de $0.3 pentru tokenii de intrare și $1 pentru tokenii de ieșire [23]. Asemănător cu modelul precedent, creatorii nu au făcut publice detalii privind numărul de parametri sau latența acestuia. În ceea ce privește timpul până la generarea primului token, testele individuale efectuate de utilizatori indică aproximativ 2 secunde [24].

Gemini 2.5 Pro depășește modele precum DeepSeek, Grok sau Claude în sarcini de știință, matematică și cunoștințe generale. Performanțele au fost măsurate pe baza seturilor de date AIME (American Invitational Mathematics Examination), care a devenit un standard în evaluarea modelelor lingvistice pentru sarcini de matematică, și GPQA (Graduate-Level Google-Proof Q&A), utilizat pentru evaluarea raționamentului științific (chimie, biologie, fizică). Având în vedere performanțele ridicate ale modelelor concurente, Gemini 2.5 Pro reușește totuși să le întrecă cu câteva procente în toate cele trei tipuri de sarcini menționate [25].

Aya Expanse este un model multilingvistic dezvoltat de Cohere, lansat în decembrie 2024, care oferă suport pentru 23 de limbi, inclusiv româna. Deși performanțele sale pe benchmark-ul MMLU sunt moderate (53,7%), comparativ cu alte modele similare, acesta se remarcă în două domenii specifice: matematică la nivel școlar (MGSM), unde obține cel mai bun scor, 67%, și traducere automată, cu 57,2% [26]. Totuși, costurile asociate utilizării modelului Aya Expanse sunt mai ridicate în raport cu cele ale modelelor Gemini sau GPT, ajungând la aproximativ $0.5 per 1M de tokeni de intrare și $1.5 per 1M de tokeni de ieșire [27]. Aya este singurul model pentru care nu a fost încă publicată nicio analiză a latenței și nici nu a fost testat de comunitatea de pasionați.

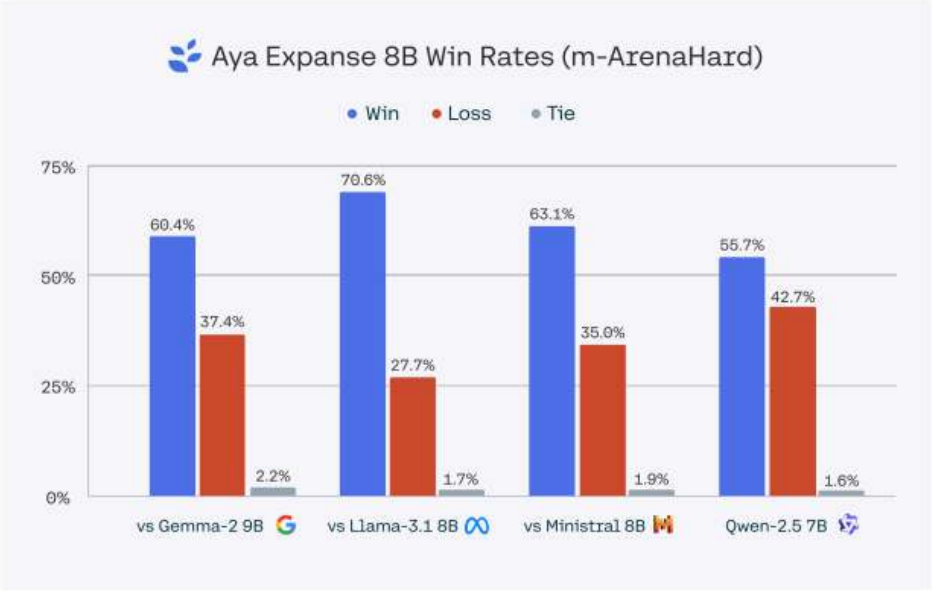

Figura 2.3: Analiza Modelului Aya Expanse cu alte modele Open Source [26]

În Figura 2.3 este ilustrată performanța modelului Aya Expanse 8B în comparație cu alte modele, evaluată pe același set de date, m-ArenaHard. Acest benchmark conține întrebări mai dificile, adesea ambigue sau neclare, fiind conceput pentru a testa capacitatea modelelor de a face față unor sarcini de raționament reale, în care utilizatorul nu oferă cele mai clare indicații. Pentru acest benchmark, Aya Expanse a fost evaluat pe rând în raport cu fiecare dintre modelele menționate — Mistral, Llama, Gemma și Qwen [26]. Evaluarea a fost realizată asemănător unui concurs, utilizând o metrică de *head-to-head win rate*, în care ambelor modele li se oferă același prompt, iar câștigătorul este ales pe baza calității răspunsului generat.

Începând cu data de 4 aprilie 2026, cei de la Cohere au eliminat de pe piață modelul Aya Expanse. Astfel, a fost preluat, în această lucrare, un model nou dezvoltat de aceeași companie. Command R7B este cel mai mic și rapid model din seria R a Cohere, lansată în decembrie 2024. Modelul dispune de 7 miliarde de parametri și o fereastră de context de 128 de mii de tokeni, fiind optimizat pentru aplicații cu cerințe stricte de latență, precum chatboți și asistenți de cod [28]. Modelul este antrenat pe 23 de limbi, inclusiv română, și excelează în sarcini de tip RAG (Retrieval-Augmented Generation).

Arhitectura sa combină straturi de atenție de tip sliding window cu straturi de atenție globală, fără utilizarea embeddings-urilor poziționale clasice. Tehnica sliding window îi permite modelului să acceseze informații dintr-un context extins, fără a pierde complet informațiile recente [29]. În locul embeddings-urilor poziționale tradiționale, modelul utilizează mecanisme alternative, precum RoPE (Rotary Positional Embeddings), pentru a încorpora informația de poziție în procesul de atenție.

Yang et al. propun în lucrarea *RoPE to NoPE and Back Again: A New Hybrid Attention Strategy* [30] un mecanism hibrid de atenție, analizând comparativ trei abordări: Rotary Position Embedding (RoPE), No Positional Embedding (NoPE) și Query-Key Normalization (QK-Norm), evidențiind avantajele și limitările fiecăreia în modelarea contextelor lungi. RoPE (Rotary Positional Embeddings) introduce informația de poziție în mod explicit, prin rotații aplicate vectorilor în spațiul de atenție [31]. Modelul codifică astfel relațiile între tokeni, ceea ce îl ajută să capteze dependențele locale.

În contrast, NoPE (No Positional Embedding) [30] elimină complet informația poziționlă din anumite straturi de atenție, permițând modelului să se bazeze exclusiv pe conținutul semantic al tokenilor. În aceste straturi, atenția nu se mai bazează pe ordinea cuvintelor, permițând modelului să învețe interacțiuni globale mai flexibile atunci când este combinat cu alte mecanisme. Pe baza acestor observații, autorii propun o arhitectură hibridă care alternează straturi de atenție locală (RoPE) cu straturi de atenție globală (NoPE).

Command A este un model mai mare și mai performant din aceeași familie, fiind optimizat pentru sarcini complexe de raționament, înțelegere avansată a limbajului și utilizare în scenarii enterprise. Datorită unui număr mai mare de parametri și unei capacități superioare de generalizare, Command A obține rezultate semnificativ mai bune pe benchmark-uri de tip MMLU și GPQA, comparativ cu Command R7B.

În plus, Command A este mai robust în sarcini care necesită consistență pe contexte lungi și performează mai bine în scenarii complexe de tip RAG, unde este necesară combinarea informațiilor din mai multe surse. În schimb, acest lucru vine cu un cost mai mare de calcul și o latență mai ridicată comparativ cu Command R7B, care este optimizat în principal pentru viteză și eficiență. Totuși capacitățile celui din urmă sunt suficiente pentru cerințele acestui proiect.

2.2.2. Modele apelate local

Mistral și Gemma sunt două modele lingvistice open-source utilizate frecvent în sarcini de înțelegere și generare a textului și sunt adesea menționate în comparațiile dintre LLM-uri. Deși ambele sunt modele multilingvistice care obțin rezultate solide pe o varietate de limbi, specializarea unui model pentru o limbă specifică se dovedește adesea utilă, dacă nu chiar necesară. Masala et al. [3] analizează în lucrarea lor diferențele de performanță dintre modelele de bază, lansate inițial de Mistral și Google, și variantele acestora specializate pentru înțelegerea și generarea limbii române, RoMistral și RoGemma.

În ceea ce privește performanța, RoMistral obține un scor de aproximativ 49% pe setul

de date MMLU, în timp ce RoGemma atinge 52%. Ambele modele au fost testate ulterior pe benchmark-uri suplimentare, precum GSM8K — care conține probleme de matematică de nivel primar [32], Winogrande — utilizat pentru evaluarea capacității de *commonsense reasoning* a modelelor lingvistice [33], și HellaSwag (HS) [34], un benchmark similar, axat tot pe raționament de tip commonsense.

Deși pe MMLU RoGemma depășește RoMistral cu câteva procente, în celelalte trei benchmark-uri menționate RoMistral înregistrează rezultate semnificativ mai bune, așa cum se poate observa și în Tabelul 2.2. Rezultatele arată că ambele modele au fost evaluate în principal pe sarcini uzuale, precum matematică de bază, raționament de tip commonsense și cunoștințe generale. Totuși, aplicația propusă nu necesită înțelegerea unor concepte științifice sau matematice avansate.

Tabelul 2.2: RoMistral vs. RoGemma [3]

Model	**HS**	**Wino**	**GSM8K**
RoMistral-7b-Instruct	62.88	70.03	32.42
RoGemma-7b-Instruct	56.32	66.97	25.98

Aceste seturi de date sunt disponibile doar în limba engleză, așadar, pentru a putea fi aplicate modelelor, este necesară traducerea acestora cu ajutorul unui model lingvistic. Totuși, este importantă și evaluarea pe un benchmark original în limba română; în acest scop a fost utilizat LiRo [35], din care au fost extrase seturi de date precum LaRoSeDa [36] și XQuAD-ro. Cele două benchmark-uri se bazează pe sarcini diferite, contribuind astfel la o evaluare mai diversificată a modelelor. Astfel, sunt evidențiate atât similaritatea semantică (unde scorurile obținute au fost de aproximativ 85% pentru clasificarea mai multor clase și 98% pentru o singură clasă), cât și capacitatea de a răspunde la întrebări (unde modelele au înregistrat de aproximativ 88%) [3].

Atât Gemma, cât și Mistral prezintă costuri reduse de utilizare, aproximativ $0.25 per milion de tokeni pentru Mistral și $0.15 per milion de tokeni pentru Gemma, atât pentru textul de intrare, cât și pentru cel de ieșire.

În ceea ce privește latența, Mistral a reușit să reducă timpul până la generarea primului token de la valori de aproape 11 secunde la sub o secundă. Desigur, acest timp poate varia în funcție de factori precum lungimea promptului, dimensiunea răspunsului sau numărul de cereri procesate simultan [37]. Cu toate acestea, testele raportate în cadrul benchmark-ului Baseten prezintă stabilitatea modelului, indiferent de configurația experimentală. În toate scenariile testate, timpul până la primul token a rămas sub pragul de o secundă.

Rezultate similare sunt raportate și pentru modelul Gemma 7B, rulat pe motorul de inferență Groq LPU, care atinge un timp de aproximativ 0.13 secunde până la primul token și o viteză de aproximativ 814 token-uri pe secundă [38]. Trebuie menționat însă că aceste rezultate provin din medii diferite de testare: Mistral a fost evaluat pe sisteme GPU optimizate, în cadrul platformei Baseten, în timp ce Gemma a fost testat pe o arhitectură specializată, Groq LPU. Din acest motiv, o comparație directă între cele două modele din punct de vedere al latenței nu este complet echitabilă, iar valorile raportate ar trebui privite mai degrabă ca indicatori de performanță în contexte specifice, nu ca o ierarhie strictă între modele.

2.3. Definirea indicatorilor de performanță

În cadrul acestui proiect, indicatorii cheie de performanță (KPI) sunt definiți pentru a măsura eficiența robotului telefonic în următoarele trei domenii: detectarea intențiilor utilizatorului, estimarea satisfacției și generarea de rezumate coerente ale conversațiilor.

2.3.1. Detectarea intențiilor utilizatorului

Primul pas al agentului telefonic constă în înțelegerea cerinței utilizatorului. Pentru aceasta, este necesară identificarea unor tipuri standard de intenții pentru care un client ar putea apela la serviciile call-center. Astfel, robotul poate clasifica fiecare cerere în clasa corespunzătoare. Pentru fiecare domeniu în parte va fi definită o listă proprie de intenții posibile. Etichetele au fost selectate pe baza scenariilor frecvente din call-centerele românești și acoperă situații precum:

- reclamații
- probleme tehnice sau administrative
- solicitări de informații
- alte solicitări neclasificate

Evaluarea performanței de clasificare utilizează doi indicatori principali: acuratețea și scorul F1 macro. **Acuratețea** reflectă procentul de clasificări corecte, unde eticheta prezisă trebuie să coincidă exact cu eticheta gold din vocabularul controlat. **Scorul F1** macro calculează media aritmetică a scorurilor F1 obținute pentru fiecare clasă în parte. Un indicator suplimentar specific acestui experiment este **acuratețea fără ordine**, care măsoară proporția cazurilor în care modelul are dificultăți în a determina care este intenția principală și care este cea secundară. Astfel este eliminată ordinea inetnțiilor din conversație.

Evaluarea detectării intențiilor este realizată la nivel de conversație completă, nu la nivel de replică individuală.

2.3.2. Estimarea satisfacției clientului

Un aspect de interes pentru orice companie este nivelul de satisfacție al clienților. Pe lângă calitatea produselor sau a serviciilor oferite, un rol important îl joacă și calitatea interacțiunilor din cadrul call-center-ului. Clienții apelează la asistență prin intermediul convorbirilor telefonice, căutând suport cu diferite probleme. Totuși, nu toți clienții încheie apelul într-o notă pozitivă. Astfel, apelurile evaluate cu un nivel scăzut de satisfacție pot fi analizate ulterior pentru a identifica motivul nemulțumirii. În acest scop, fiecare apel este evaluat pe o scală de satisfacție, definită după cum urmează:

- **Pozitiv** – client satisfăcut, exprimare clară și respectuoasă.
- **Neutru** – client neutru, fără exprimarea clară a satisfacției sau nemulțumirii.
- **Negativ** – client nesatisfăcut, exprimare direct nemulțumită, frustrată sau critică.

Aceste trei categorii pot întâmpina, totuși, cazuri limită în care determinarea nivelului real de satisfacție al clientului devine mai dificilă, necesitând o înțelegere contextuală:

- **Politicos, dar nemulțumit:** Clientul utilizează un limbaj formal și respectuos, însă mesajul său exprimă nemulțumire față de produs sau față de modul în care a decurs apelul.
- **Sarcasm și comportament pasiv-agresiv:** Folosirea unor termeni aparent pozitivi, dar cu intenție ironică, care semnalează frustrare sau nemulțumire.

– **Ambivalență (satisfacție mixtă):** Situații în care clientul apreciază atitudinea robotului sau a operatorului, dar este profund nemulțumit de răspunsul oferit.
– **Utilizarea expresiilor:** Unele expresii care pot indica nemulțumire, fără a folosi neapărat cuvinte negative (*Mi-ați facut capul calendar*, *Ma scoateți din minți*).

Pentru validarea clasificării nivelului de satisfacție al consumatorului vor fi utilizați următorii indicatori de performanță: acuratețea și scorul F1.

Având în vedere distribuția dezechilibrată a claselor de satisfacție, în special pentru categoriile mai rare, evaluarea performanței va lua în considerare scorul F1, care combină precizia și reamintirea într-o singură metrică. În acest context, scorul macro-F1 este considerat mai important decât acuratețea globală, deoarece tratează în mod egal toate clasele, indiferent de frecvența lor.

Similar clasificării intențiilor, evaluarea satisfacției este realizată la nivel de conversație, luând în considerare evoluția interacțiunii și tonul general al clientului. Datorită caracterului subiectiv al noțiunii de satisfacție, etichetele asociate acestui criteriu sunt considerate un ground truth operațional, definit conform regulilor stabilite în cadrul acestui studiu.

2.3.3. Generarea rezumatelor conversațiilor

Pentru o analiză mai eficientă a convorbirilor, validarea automată a clasificărilor și gestionarea cazurilor limită, este adesea necesară interpretarea manuală a datelor. Un mod eficient de a facilita acest proces îl reprezintă generarea de rezumate ale conversațiilor. Astfel, rezumatele pot fi structurate în următoarele categorii:

– **Scurt** – concentrat pe ideile principale ale conversației.
– **Mediu** – oferă context suplimentar și principalele acțiuni sau probleme discutate.
– **Lung** – oferă detalii relevante și evoluția completă a conversației.

Alegerea tipului de rezumat este influențată de nivelul de satisfacție al clientului. În cazul conversațiilor asociate unui nivel de satisfacție pozitiv sau neutru, nu este necesară generarea unui rezumat detaliat; interacțiunea s-a desfășurat fără probleme. În schimb, pentru conversațiile evaluate cu un nivel de satisfacție negativ sau încadrate în cazuri-limită, detaliile devin importante, iar o sumarizare mai amplă este necesară pentru identificarea și analiza problema. Vor fi definite astfel constrangerile stilistice și lungimea rezumatelor generate:

– **Scurt**: 20–40 de cuvinte; ideile principale ale conversației.
– **Mediu**: 40–70 de cuvinte; punctele cheie, cărora li se adaugă observații suplimentare.
– **Lung**: 60–100 de cuvinte; detalii relevante despre desfășurarea conversației.

Pentru evaluarea fidelității dintre conversația telefonică originală și rezumatul generat de agentul inteligent vor fi utilizați următorii indicatori de performanță: ROUGE și BERTScore. ROUGE reprezintă o metodă uzuală pentru evaluarea calității rezumatelor, bazată pe suprapunerea lexicală dintre textul sursă și cel generat. Dintre variantele disponibile, vor fi utilizate ROUGE-1, care măsoară suprapunerea la nivel de cuvinte individuale, ROUGE-2, care evaluează câte perechi de cuvinte se regăsesc în ambele texte, și ROUGE-L, care determină cea mai lungă secvență comună dintre original și rezumat [39].

Deși metricile ROUGE oferă o evaluare foarte bună a gradului de acoperire lexicală și a alinierii cu informațiile din conversația inițială, acestea nu iau în calcul și sensul semantic al informațiilor. Pentru a adresa această limitare, va fi utilizat scorul BERTScore, care evaluează similaritatea semantică dintre cele două texte, asigurând astfel corespondența dintre acestea.

3. DESCRIERE DATASET

Deși aproape toate companiile care oferă produse sau servicii dispun de un call center, aceste conversații nu sunt publicate din motive legale. Aceste interacțiuni conțin frecvent date cu caracter sensibil, precum informații de identificare personală, adrese, date ale cardurilor bancare, iar înregistrările audio pot fi considerate date biometrice. Conform legislației privind protecția datelor, în special Regulamentului General privind Protecția Datelor (GDPR), colectarea, stocarea și distribuirea acestor informații sunt strict reglementate, iar publicarea lor fără consimțământul explicit și informat al persoanelor implicate este interzisă. Chiar și în cazul în care sunt aplicate diferite tehnici de anonimizare, există riscul de reidentificare a persoanelor pe baza contextului, mai ales în situații specifice sau neobișnuite. Pe lângă aceste constrângeri legislative, un alt factor limitativ este disponibilitatea redusă a seturilor de date relevante, majoritatea acestora fiind în limba engleză, ceea ce îngreunează dezvoltarea și evaluarea sistemelor destinate limbii române.

Park și Gates analizează în lucrarea lor, *Towards Real-Time Measurement of Customer Satisfaction Using Automatically Generated Call Transcripts* [40], limbajul și comportamentul clienților care au apelat la call-center-ul unei companii din domeniul automotive. Autorii au analizat relația dintre starea de spirit finală a consumatorului (preluată pe baza răspunsurilor unui chestionar final) și frecvența cuvintelor care să indice sentimentele aferente stării de spirit, utilizate de acesta în timpul apelului. Indicatorul utilizat, notat cu *diff*, a fost calculat ca diferență dintre numărul de cuvinte pozitive și numărul de cuvinte negative exprimate (cuvinte pozitive − cuvinte negative).

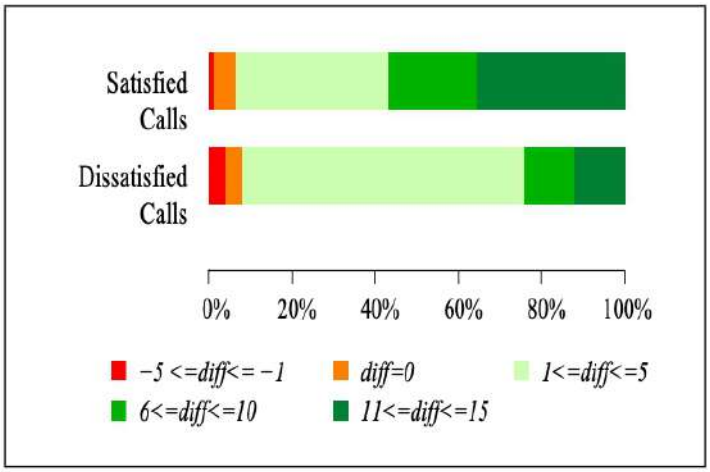

Figura 3.1: Relația dintre satisfacția clientului și cuvintele utilizate

Pe baza rezultatelor acestei analize, ilustrate în Figura 3.1, se poate observa că atât clienții satisfăcuți, cât și cei nesatisfăcuți tind să utilizeze mult mai multe cuvinte pozitive în timpul

unei conversații telefonice, indiferent de nivelul lor real de mulțumire. Doar 8% dintre clienții frustrați folosesc un număr egal sau mai mare de cuvinte negative comparativ cu cele pozitive, în timp ce un procent similar de clienți satisfăcuți (6.4%) au un vocabular preponderent negativ.

Prin urmare, diferența dintre numărul de cuvinte pozitive și negative din apelurile rezolvate, comparativ cu cele care nu au fost rezolvate, este mică. Acest fapt poate fi explicat prin politețea formală, prezența ironiei și modul în care clienții își descriu problemele folosind un limbaj uzual, fără un vocabular agresiv. Pentru aceasta, utilizarea unui LLM care să extragă, din replicile clientului, adevărata stare de spirit și satisfacția vis-a-vis de serviciile oferite poate fi utilă.

O abordare asemănătoare acestei lucrări este documentată de Shah et al. [41], autorii realizând o analiză a utilizării tehnicilor de procesare a limbajului natural în automatizarea serviciilor call-center. Sistemul propus de aceștia vizează aceleași sarcini precum cele menționate anterior: clasificarea automată a intențiilor, evaluarea satisfacției și generarea de rezumate pentru eficientizarea activității post-apel (*After-Call Work*).

Printre concluziile principale ale studiului se numără faptul că modelele tradiționale bazate pe cuvinte-cheie sau reguli fixe nu reușesc să gestioneze ambiguitatea limbajului natural, în timp ce arhitecturile moderne bazate pe Transformer și LLM-uri oferă o înțelegere mult mai bună.

3.0.1. Generarea Conversațiilor

În acest scop, vor fi generate 100 de scenarii cu ajutorul unui LLM. Pentru a ne asigura că evaluarea modelelor alese este cât se poate de obiectivă, va fi folosit un alt model, pe lângă cele menționate, mai exact Claude.

Claude 3 este o familie de LLM-uri care au atins performanțe state-of-the-art și a fost dezvoltată de compania Anthropic. Această serie include trei variante optimizate pentru diferite nevoi: Haiku (viteză), Sonnet (echilibru) și Opus (performanță maximă). Diferenta de baza dintre aceste modele se poate observa in Figura 3.2. Claude 3 este recunoscut pentru capacitatea de a urma instrucțiuni complexe, pentru fereastra de context extinsă care permite procesarea documentelor mari și pentru acuratețea ridicată în sarcini de extracție structurată a datelor [42]. Acesta utilizează Constitutional AI [43], o metodă de antrenare care ghidează comportamentul modelului pe baza unui set de principii etice predefinite, asigurând răspunsuri mai sigure, mai puțin subiective și cu mai puține halucinații.

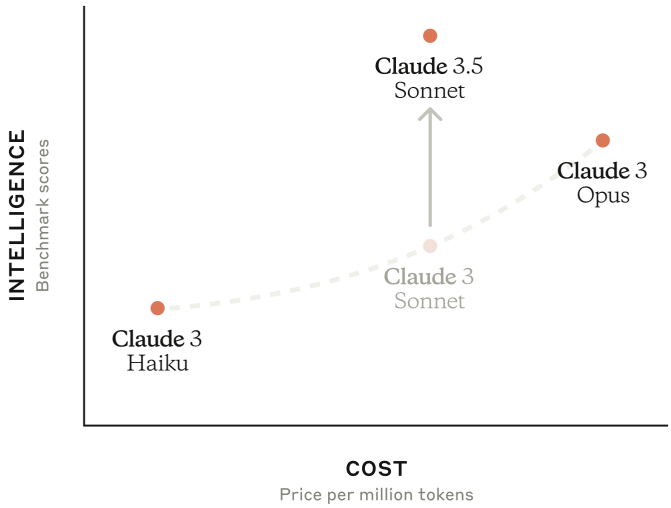

Figura 3.2: Performanțele modelelor din familia Claude 3 [44]

Conversațiile necesare setului de date dedicat acestui proiect nu necesită o complexitate ridicată. Cuvintele utilizate sunt uzuale, fără folosirea unor termeni academici, iar contextul este unul ușor de procesat. Astfel, nu este nevoie de un model avansat, ci mai degrabă de unul rapid, motiv pentru care a fost ales modelul Haiku.

Aceste date sintetice prezintă avantaje în controlarea scenariilor și asigură o acoperire largă a subiectelor și intențiilor; dispare astfel și problema legislativă a datelor sensibile. Pe de altă parte, trade-off-ul major al acestor conversații îl reprezintă nuanța reală a conversațiilor. Indiferent cât de bine ar reuși algoritmul să genereze conversații complexe și cu diferite stări emoționale ale clientului, acestea nu vor fi 100% precise. În viața reală, aceste conversații sunt uneori greu de transcris din cauza sunetelor de fundal sau a calității slabe a echipamentelor, astfel încât și transcrierile pot fi problematice. Atitudinea oamenilor nu este întotdeauna politicoasă, recurgând uneori la injurii și țipete; aceste stări și exprimări sunt dificil de generat cu ușurință sau acuratețe de către LLM-uri.

Această limitare este acceptată în mod conștient, întrucât scopul setului de date nu este reproducerea exactă a tuturor particularităților interacțiunilor umane, ci crearea unui cadru suficient de complex pentru evaluarea capacității de înțelegere și raționament a modelelor lingvistice.

Pentru a ne asigura, pe cât posibil, că modelul generează conversații cât mai apropiate de realitate, trebuie luate în calcul câteva aspecte:

- complexitatea conversațiilor
- lungimea conversațiilor
- intențiile multiple
- variabilitatea limbajului

Astfel, vor fi definite 5 domenii diferite pentru care se vor genera scenarii specifice: domeniul bancar, telecomunicații, domeniul medical, retail/e-commerce și servicii publice. Fiecare dintre aceste domenii dispune de produse și servicii diferite, care permit acoperirea mai multor situații și intenții. Pentru fiecare dintre acestea vor fi alocate 20 de scenarii, care vor fi ulterior împărțite în conversații simple și complexe.

Interacțiunile umane nu sunt întotdeauna simple și ușor de gestionat; oamenii prezintă probleme diferite și, de cele mai multe ori, indiferent de motivul pentru care apelează serviciul de call-center, atitudinea clientului reprezintă cel mai dificil aspect. Din acest motiv, o atenție mai mare trebuie acordată interacțiunilor complexe și situațiilor cu un grad ridicat de dificultate emoțională.

Tabelul 3.1: Distribuția setului de date: Scenarii și complexitate conversații

Domeniu Scenariu	**Nr. Tot. Conversații**	**Simple**	**Complexe**
Banking	20	6	14
Medicină	20	6	14
Retail / E-commerce	20	6	14
Telecom	20	6	14
Servicii Publice	20	6	14

Așadar, în Tabelul 3.1 sunt definite cele 5 tipuri principale de subiecte, împărțite în 30% conversații simple și 70% conversații complexe. Cele 6 conversații simple vor conține solicitări clare și întrebări directe, care nu necesită interacțiuni lungi sau răspunsuri complexe. Pe de altă parte, cele 14 conversații complexe vor include situații dificile, limbaj mai puțin uzual sau mai puțin politicos, intenții multiple, mai mulți pași pentru rezolvarea problemei, întrebări diverse

și o stare emoțională negativă a utilizatorului.

În plus, pe lângă complexitate, conversațiile vor fi împărțite și în funcție de satisfacția clientului, definită în capitolul anterior (pozitivă, negativă, neutră). O importanță sporită trebuie acordată distribuției dintre cele trei niveluri de satisfacție, având în vedere că nivelul negativ prezintă cel mai mare interes și nu este întotdeauna cel mai ușor de identificat. Prezența ironiei sau a formulărilor tăioase poate pune modelul lingvistic într-o situație ambiguă. Multe cuvinte uzuale, care în mod normal nu au o conotație negativă, pot fi utilizate într-un sens negativ, iar tonul unei persoane nu poate fi determinat întotdeauna doar din cuvintele utilizate. Un utilizator poate fi nesatisfăcut, dar să își păstreze un limbaj adecvat și politicos. Din acest motiv, este necesar un context suficient de complex pentru a putea analiza corect atitudinea clientului.

Astfel, conversațiile simple din fiecare categorie vor avea în medie 10–15 linii de dialog și vor fi împărțite între conversațiile pozitive și neutre. Pentru îmbunătățirea serviciilor de call-center, pozitivitatea și neutralitatea nu reprezintă o sursă importantă de informații. De aceea, atenția asupra acestor niveluri de satisfacție nu ar trebui să fie foarte ridicată. Din cele 6 conversații simple, 3 vor fi pozitive și 3 neutre.

Pe de altă parte, conversațiile complexe vor conține aproximativ 25 de linii de dialog. Pentru a nu aloca în totalitate conversațiile complexe nivelului de satisfacție negativ, dintre cele 14 scenarii rămase, 2 vor fi neutre și 12 negative. Toate aceste dialoguri vor mima conversații reale cu operatori umani; operatorii automatizați dispun de replici predefinite, iar interacțiunea client–robot este, în general, mai puțin solicitantă decât cea complet umană.

Un alt aspect al acestor conversații îl reprezintă intențiile multiple. În cazul cel mai simplu, clientul apelează având o singură intenție, însă există situații în care acesta formulează mai multe întrebări de la început sau adaugă ulterior o nouă solicitare. Acest aspect trebuie luat în considerare de către model; aceste cazuri vor fi tratate în același mod și vor avea două sau mai multe etichete asociate, în funcție de numărul de intenții.

Tabelul 3.2: Lista intențiilor specifice alocate per domeniu

Domeniu	**Intenții incluse în dataset**
Banking	problemă sold, card pierdut, problemă transfer, problemă credit, tranzacție greșită, problemă schimb valutar, card blocat, tranzacție suspectă
Medicină	problemă programare, anulare programare, problemă rețetă, rezultate analize, consultație anulată, problemă facturare, problemă asigurare, reclamație personal
Retail	comandă întârziată, retur produs, reclamație produs, produs lipsă stoc, comandă greșită, anulare comandă, problemă livrare, problemă garanție
Telecom	activare eșuată, problemă semnal, factură greșită, reziliere contract, portare eșuată, problemă internet, problemă roaming, problemă modificare abonament
S. Publice	acte incomplete, programare ghișeu, sesizare problemă, dosar respins, contestație decizie, problemă plată taxă, informații program, reclamație serviciu

Limbajul din aceste conversații va fi unul colocvial, uzual. Conversațiile formale, deși de preferat în fiecare situație, sunt scenarii simple care nu pun în valoare performanța modelului. De aceea, limbajul clientului va fi unul colocvial, care va conține în plus ambiguitate și, eventual, și greșeli gramaticale, pentru a pune la încercare raționamentul LLM-ului. În funcție de zona de proveniență a unei persoane, se poate observa în vorbirea acesteia utilizarea regionalismelor, a unor expresii care sugerează neplăcere sau, pur și simplu, o anumită manieră de exprimare; și

aceste aspecte trebuie luate în calcul în conversațiile generate.

Pentru construirea setului de date utilizat în evaluarea sistemului, a fost dezvoltat un script în Python care generează automat conversații telefonice sintetice între clienți și operatori, folosind modelul lingvistic de mari dimensiuni menționat anterior, Claude, prin intermediul unei chei API. Astfel, dialogurile sunt salvate direct în folderul de lucru al proiectului. Generarea presupune definirea intențiilor pe baza cărora modelul trebuie să creeze dialogurile, așa cum sunt prezentate în tabelul 3.2. Fiecare domeniu prezintă câte 8 posibile intenții, iar în cadrul generării acestea au fost împărțite în mod aproximativ egal pentru a asigura că fiecare intenție are cel puțin două conversații.

Conversațiilor complexe le-au fost alocate detalii suplimentare cu privire la tipul complexității. Mai precis, a fost definită o matrice cu zece tipologii de conversații complexe care să acopere situații reale:

- **Multi-intenție**: Clientul inițiază apelul cu o intenție principală, dar pe parcursul dialogului introduce o a doua solicitare.
- **Ezitantă**: Clientul comunică folosind pauze, reformulări și un discurs fragmentat.
- **Problemă nerezolvată**: Interacțiunea se încheie fără o soluție tehnică sau procedurală din partea operatorului.
- **Pasat între departamente**: Simulează experiența negativă a redirecționărilor multiple.
- **Dialect familiar**: Utilizarea unui registru informal sau a expresiilor regionale.
- **Context implicit**: Clientul face referire la interacțiuni anterioare sau presupune că operatorul deține deja informații de context.
- **Negare condiționată**: Formulare de cereri prin excludere (ce nu dorește clientul) sau condiții specifice.
- **Reclamație implicită**: Nemulțumirea nu este verbalizată prin termeni agresivi, ci este transmisă prin ton ironic, răspunsuri scurte sau punctuație tăioasă în dialog.
- **Ambiguă**: Situație în care solicitarea inițială este neclară, necesitând întrebări de clarificare din partea operatorului.
- **Informații contradictorii**: Clientul semnalează primirea unor date diferite în apeluri precedente.

Conversațiile sunt salvate individual în format JSON, conținând un identificator unic, intenția și lista replicilor. Ulterior, conversațiile au fost citite și evaluate manual pentru a nota problemele de bază și pentru a îmbunătăți generarea acestora. În acest scop a fost păstrată și intenția în partea de generare, pentru a putea determina dacă modelul a înțeles corect cum trebuie alocate, alese și respectate aceste intenții. După rafinarea finală a acestor conversații, intențiile au fost șterse pentru a nu influența pasul de adnotare.

În urma acestor verificări, s-a observat necesitatea mai multor schimbări; niciuna dintre variante nu respectă complet cerințele impuse. În primul rând, în promptul inițial au fost definite cele trei componente principale ale dialogurilor: nivelul de satisfacție, intenția și complexitatea. Modelul a reușit să respecte în totalitate doar nivelul de satisfacție și complexitatea conversațiilor (simplă sau complexă), împărțindu-le așa cum era prevăzut. A reușit să respecte în proporție mare și tipologia conversațiilor complexe.

Totuși, a avut probleme destul de mari când a venit vorba de respectarea intențiilor. Deși acestea erau deja predefinite, în anumite conversații adăuga intenții noi, care nu respectau lista impusă sau, de cele mai multe ori, alegea să combine intențiile între ele, chiar și atunci când nu era impusă o intenție multiplă. Chiar dacă unica intenție a clientului era raportarea unei reclamații, LLM-ul introducea și o altă intenție ulterioară (precum blocarea cardului), care venea ca o completare a scenariului: clientul reclama că i-au fost luați banii de pe card în mod

fraudulos, operatorul deschidea un dosar de fraudă, dar pe lângă aceasta îi era blocat cardul utilizatorului ca metodă de siguranță. Astfel, modelul cataloga această conversație ca având două intenții: banii luați fraudulos de pe card și blocarea cardului.

Limbajul nu respecta deloc realismul uman; multe dintre replici păreau infantile, iar modul în care operatorul vorbea nu era deloc profesional și respectuos, așa cum se întâmplă în call-center în realitate (*Salut, o secundă că nu te aud... Gata, cu Cristina vorbiți.*). Modul de conversație al clientului respecta mult mai mult realitatea, dar se repetau adesea aceleași replici care indicau frustrare, replici pe care în realitate oamenii nu le-ar utiliza în acel fel (*Deci trebuie să stau pe drumuri din cauza voastră. Super! Mulțumesc pentru nimic!*).

Având în vedere aceste rezultate nefavorabile, a fost reluată generarea și au fost impuse mult mai multe restricții în acest prompt. A doua oară, aceste intenții au fost mult mai bine respectate, dar o problemă foarte des întâlnită se regăsea în continuare: greșelile gramaticale, de logică și utilizarea unor cuvinte greșite, spre exemplu: *Vă salut la serviciul clienți BankRom. Cu plăcere vă ajut.*. Modelul se încurca în ordinea cuvintelor în propoziție sau utiliza replici pe care oamenii nu le-ar fi folosit în viața reală.

Pentru a grăbi verificarea și corectarea acestor conversații, a mai fost generat un script care să corecteze aceste erori. Acestuia i-a fost dată o sarcină simplă — corectarea conversațiilor sintetice între un client și un operator uman — și toate detaliile necesare pentru înțelegerea contextului conversațiilor.

Astfel, ultimul pas al sarcinii de generare a fost o ultimă verificare manuală. Scriptul de corectare a îmbunătățit drastic discuțiile dintre operator și client; marea majoritate a greșelilor de exprimare au fost corectate, singurele modificări implementate fiind:

- eliminarea expresiilor exagerate
- îmbunătățirea logicii unor propoziții
- accentuarea satisfacției acolo unde era necesar

Deși procesul de corectare a redus semnificativ erorile evidente, trebuie menționat faptul că datele rămân în continuare sintetice. Astfel, stilul conversațional poate reflecta tiparele modelului lingvistic, introducând un anumit grad de subiectivitate prin modul specific de structurare a dialogului și a replicilor, aspect care rămâne valabil pentru o parte din conversațiile setului de date.

Chiar dacă aceste măsuri de verificare nu elimină complet limitările asociate datelor sintetice, ele contribuie la obținerea unui set de conversații suficient de realist pentru a permite evaluarea comparativă a mai multor modele lingvistice în cadrul sarcinilor analizate în această lucrare.

3.0.2. Adnotarea Conversațiilor

A doua parte a procesului de creare a setului de date o reprezintă adnotarea conversațiilor. În acest scop a fost implementat un al doilea script Python dedicat adnotării automate a conversațiilor, folosind același model lingvistic — Claude. Scriptul parcurge fișierele JSON generate anterior și adaugă metadate relevante pentru analiza conversațiilor, incluzând:

- domeniul conversației
- nivelul de complexitate
- intenția sau intențiile clientului
- nivelul de satisfacție
- motivul satisfacției
- rezumatul conversației

Procesul de adnotare semi-automată a fost aplicat asupra celor 100 de conversații generate, parcurgând două etape distincte:

– adnotarea automată prin Claude API
– corecția sistematică pe baza distribuției și a listei de intenții definite anterior

În prima etapă, modelul a generat automat câmpurile pentru intenție, satisfacție, complexitate și rezumat pentru fiecare conversație. Totuși, adnotările inițiale nu prezentau rezultate satisfăcătoare: fie erau adăugate mai multe intenții acolo unde nu era nevoie (eroare întâlnită și anterior, în partea de generare; modelul confunda acțiunile ulterioare discuției cu intenția inițială cu care clientul apela serviciile), fie era confundat nivelul de satisfacție.

Ulterior, pentru a simplifica corectarea acestor erori, a fost implementat un script de corecție care compară adnotările adăugate automat cu valorile de referință stabilite în faza de proiectare și generare a setului de date. Din totalul de 100 de conversații, 47 au fost adnotate corect de la început, în timp ce 53 au necesitat cel puțin o corecție.

La nivel de domeniu, cel mai problematic s-a dovedit domeniul serviciilor publice, unde 16 din cele 20 de conversații au necesitat corecții. Acest rezultat poate fi explicat prin natura mai ambiguă a interacțiunilor din acest domeniu, unde solicitările clienților pot implica proceduri administrative, documente sau etape multiple ale procesului birocratic. Pe lângă acestea, serviciile publice oferite de stat sunt de foarte multe tipuri, iar pentru conversațiile din acest proiect nu a fost definit un anumit tip de servicii publice (cum ar fi serviciul de taxe și impozite), așadar conversațiile sunt generale, adaptabile mai multor situații.

Domeniul medical a avut nevoie de corectare pentru 10 conversații, în timp ce domeniile banking, retail și telecom au înregistrat performanțe similare, cu câte 9 conversații corectate fiecare. Acestea din urmă, care au prezentat cele mai puține erori, sunt și cele în care se oferă cel mai frecvent servicii de tip call-center, iar clienții se lovesc mult mai des de probleme atunci când apelează la ele. De asemenea, termenii din aceste domenii pot fi mult mai ușor de procesat de către model, fiind mai ușor de înțeles în context.

Distribuția erorilor poate fi împărțită în funcție de cele trei categorii de condiții definite: intenții, satisfacție și complexitate. Erorile de intenție au fost cele mai frecvente, afectând 35 de conversații. În timpul adnotării, modelul tindea să identifice intenții multiple acolo unde clientul exprimase o singură solicitare, incluzând acțiunile inițiate de operator ca și cum ar fi fost cereri ale clientului.

Erorile de complexitate au fost cele mai puține, afectând 9 conversații, și s-au distribuit relativ echilibrat între supraevaluare și subevaluare. Modelul reușește, în general, să distingă în mod corect între scenariile simple și cele complexe. Confuzia apare în special în situațiile în care conversațiile includ elemente suplimentare de context sau mai multe etape ale rezolvării problemei, ceea ce poate conduce la interpretarea lor ca fiind mai complexe decât au fost generate inițial.

Erorile de satisfacție au afectat 16 conversații și, de cele mai multe ori, era vorba despre același tip de confuzie: 14 conversații au fost clasificate ca pozitive în loc de neutre, iar 2 ca neutre în loc de negative. Nicio conversație nu a fost supraevaluată în sens invers. Impresia inițială, înainte de generarea acestui set de date, a fost că ar trebui acordată o importanță sporită conversațiilor negative, fiind cele cărora compania ar trebui să le ofere o atenție sporită. Totuși, diferența dintre conversațiile neutre și cele pozitive se dovedește a fi relativ mică și adesea dificil de identificat chiar și pentru un evaluator uman, întrucât interacțiunile pot fi corecte din punct de vedere procedural, dar lipsite de un nivel ridicat de satisfacție.

În acest context, confuzia dintre nivelurile pozitiv și neutru nu afectează semnificativ utilitatea practică a sistemului. În aplicații reale de tip call-center, conversațiile negative sunt cele care necesită intervenție sau analiză suplimentară, deoarece ele indică existența unei probleme nerezolvate sau a unei experiențe negative a clientului. Prin urmare, capacitatea modelului de a identifica corect conversațiile negative este mai relevantă operațional decât

distincția dintre cazurile neutre și cele pozitive.

Chiar dacă modelul prezintă o tendință de supraevaluare ușoară a satisfacției, această limitare are un impact redus asupra obiectivului principal al sistemului, acela de a semnala interacțiunile problematice care necesită investigații ulterioare.

Lucrări precum cea a lui Wang et al. [45] scot în evidență problema neutralității atunci când vine vorba de nivelul de satisfacție al consumatorilor. Aceștia analizează recenziile lăsate de clienții hotelurilor pe platforma TripAdvisor, utilizând modele de deep learning precum BERT și LSTM pentru a clasifica opiniile clienților în trei categorii: pozitive, neutre și negative. Studiul evidențiază faptul că, în mod similar multor altor seturi de date din domeniul recenziilor online, distribuția claselor este dezechilibrată, recenziile pozitive fiind semnificativ mai numeroase decât cele neutre sau negative.

Rezultatele experimentale arată că ambele modele analizate întâmpină dificultăți în identificarea corectă a recenziilor neutre. Autorii observă că această categorie este în mod particular problematică deoarece diferențele lingvistice dintre opiniile neutre și cele pozitive sunt adesea subtile, ceea ce face ca modelele de clasificare să confunde frecvent aceste două tipuri de sentiment. Chiar și după aplicarea unor tehnici de echilibrare a setului de date, precum under-sampling, performanța pentru clasa neutră rămâne instabilă.

Wilson, Wiebe și Hoffmann [46] au realizat un studiu în domeniul analizei sentimentelor, care introduce distincția dintre polaritatea prioritară și polaritatea contextuală a termenilor. Polaritatea prioritară reprezintă sentimentul asociat unui cuvânt în mod izolat, așa cum apare acesta într-un lexicon de sentiment, în timp ce polaritatea contextuală descrie valoarea reală a sentimentului atunci când cuvântul este utilizat într-o propoziție sau într-un context specific. Autorii demonstrează că un cuvânt cu polaritate negativă sau pozitivă poate deveni neutru în funcție de structura propoziției sau de prezența unor elemente precum negația sau modificatorii de intensitate.

Autorii propun un proces de analiză în două etape. În prima etapă este determinat caracterul subiectiv sau obiectiv al expresiei analizate, iar abia ulterior, în cazul expresiilor considerate subiective, este clasificată polaritatea sentimentului.

Un aspect important este rolul clasei neutre în sistemele de analiză a sentimentului. Spre deosebire de abordări binare, care se concentrează exclusiv pe polaritatea pozitivă și negativă, includerea unei clase neutre permite o reprezentare mai realistă a interacțiunilor lingvistice. Totuși, această categorie este adesea dificil de identificat automat, deoarece diferențele lingvistice dintre enunțurile neutre și cele pozitive pot fi subtile.

Singura parte a adnotării care nu poate fi catalogată ca fiind greșită este partea de generare a rezumatului. Claude a reușit în proporții mari să respecte condițiile impuse cu privire la lungimea rezumatului și a reușit să înțeleagă contextul suficient de bine încât rezumatul prezentat să aibă sens. Totuși, o limitare care nu a putut fi corectată este nivelul de complexitate al rezumatului. Oamenii sunt obișnuiți să realizeze rezumate cât mai succinte, utilizând cuvinte simple și propoziții foarte scurte (poate chiar idei principale scrise sub formă de listă). Modelul nu a respectat în totalitate acest aspect, având în continuare propoziții complete, elegante și utilizând cuvinte mai complexe comparativ cu cele folosite în mod uzual în rezumate. Dar, cum acest aspect nu reprezintă o problemă de bază, creativitatea scrierii rezumatului pe care modelul o prezintă nu a fost limitată mai mult decât a fost necesar.

4. PROMPT ENGINEERING

Prompt engineering-ul reprezintă un element important în obținerea unor rezultate bune, atunci când sunt utilizate modelele lingvistice. Deși acestea dispun de capacități avansate de înțelegere și generare a limbajului natural, performanța lor depinde foarte mult de modul în care cerințele utilizatorului sunt formulate și cât de multe detalii sunt oferite.

Spre deosebire de metodele clasice de antrenare sau fine-tuning, care necesită o cantitate foarte mare de date și resurse computaționale, prompt engineering-ul permite adaptarea comportamentului modelului prin definirea explicită a sarcinii, a constrângerilor și a formatului răspunsului. Astfel, același model poate fi utilizat pentru sarcini diferite, fără a fi modificați parametrii săi interni.

În cadrul acestui proiect, prompt engineering-ul este utilizat pentru trei tipuri de sarcini distincte: detectarea intențiilor utilizatorului, estimarea nivelului de satisfacție și generarea de rezumate ale conversațiilor. Fiecare dintre aceste sarcini impune cerințe diferite asupra modului de formulare a promptului și sunt impuse niveluri de constrângere diferite. În literatura de specialitate sunt descrise mai multe strategii de proiectare a prompturilor, fiecare cu avantaje și limitări specifice.

4.1. Metode de prompt engineering

Zero-shot prompting presupune formularea unei instrucțiuni directe, fără furnizarea unor exemple de intrare și ieșire [47]. Modelul este lăsat să rezolve sarcina pe baza cunoștințelor dobândite în timpul antrenării. Deși simplă și rapidă, această metodă nu garantează că răspunsul oferit va corespunde așteptărilor utilizatorului, dar este utilizată cel mai adesea pentru măsurarea performanțelor LLM-urilor. Lucrări precum cea a lui Radford et al. [48] demonstrează capacitățile extraordinare ale modelelor state-of-the-art în generarea unor rezultate foarte bune doar pe baza unui zero-shot prompting. Aceștia subliniază importanța antrenării unui model lingvistic pe un set cât mai mare de date pentru a putea facilita și îmbunătăți răspunsurile generate atunci când instrucțiunile sunt puține.

Kojima et al. [49] propun o abordare diferită, introducând conceptul de Zero-shot-CoT (*Zero-shot Chain-of-Thought*), care presupune utilizarea unui prompt simplu, precum *„Let's think step by step"* (*„Hai să ne gândim pas cu pas"*), ce reușește să ghideze modelul în generarea unui proces cu mai multe etape. Spre deosebire de metoda originală Chain-of-Thought, care necesită exemple, Zero-shot-CoT este independentă de sarcină (*task-agnostic*).

Tehnica presupune două etape — extracția raționamentului și extracția răspunsului — și a dus la creșterea performanței pe seturi de date dificile, cum ar fi MultiArith (de la 17.7% la 78.7%) și GSM8K (de la 10.4% la 40.7%), folosind modelul InstructGPT.

Few-shot prompting extinde abordarea zero-shot prin includerea în prompt a unui număr redus de exemple reprezentative. Cu ajutorul acestora, modelul poate respecta structura și formatul răspunsului așteptat, ceea ce conduce, în general, la o performanță îmbunătățită a

rezultatului final.

Un studiu realizat de Brown et al. [50] pe modelul GPT-3 cu 175 de miliarde de parametri a demonstrat că, prin creșterea dimensiunii modelului, poate fi îmbunătățită semnificativ performanța în regim few-shot, acesta devenind competitiv chiar și cu sistemele optimizate prin fine-tuning. În testele de cultură generală precum TriviaQA, GPT-3 a atins o precizie de 71.2% în setarea few-shot, stabilind un nou standard de performanță pentru sistemele de tip closed-book. Ulterior, lucrări precum cea a lui Kojima et al. [49] au analizat în paralel aceste metode, punând în balanță diferențele dintre strategii și demonstrând modul în care LLM-urile au avansat în ultimii ani, adesea fiind capabile să rezolve sarcini complexe fără a mai necesita exemple numeroase sau prompturi complicate.

Role prompting constă în atribuirea unui rol explicit modelului la începutul promptului, de exemplu *„ești un expert în analiza conversațiilor telefonice”*. Modelul este astfel orientat către un anumit stil de răspuns, relevant pentru domeniul dorit. Conform studiului realizat de Wang et al. [51], succesul acestei metode depinde de îmbinarea instrucțiunilor de sistem cu fragmente de dialog specifice, proces numit *dialogue engineering*. Această abordare permite modelului să treacă de la un răspuns generic la unul personalizat, menținând în același timp coerența factuală și fidelitatea față de trăsăturile personajului interpretat (ex: expert bancar), cum ar fi vocabularul specific.

În lucrarea lor, autorii introduc framework-ul RoleLLM și setul de date RoleBench, care demonstrează capacitățile complexe ale acestui tip de prompting. Rezultatele experimentale arată că modelele ajustate prin tehnici de tip *Context-Instruct* pot depăși performanțele modelelor standard în ceea ce privește bogăția detaliilor despre personajul mimat, având un nivel de interacțiune mult mai natural. O altă observație care reiese pe baza testelor, dar care este des întâlnită și în cadrul celorlalte metode de prompting, este că modelele de dimensiuni mai mari sunt mult mai robuste atunci când primesc informații contradictorii și pot menține „masca” personajului mult mai bine.

Abordarea prompturilor prin utilizarea exemplelor se lovește totuși de o limitare: numărul de exemple care pot fi incluse într-un prompt este restricționat de fereastra de context a modelului, care acceptă de obicei între 10 și 100 de exemple. Pentru a depăși această limitare, Hao et al. [52] introduc **structured prompting**, care presupune organizarea promptului în secțiuni clar delimitate, precum contextul sarcinii, reguli de decizie, constrângeri și formatul așteptat al outputului. Structurarea cerinței și a răspunsului reduce ambiguitatea și determină modelul să respecte mult mai ușor toate cerințele impuse.

Experimentele conduse pe un set divers de sarcini arată că structured prompting îmbunătățește performanța față de in-context learning și reduce varianța răspunsurilor în funcție de alegerea exemplelor din prompt. Metoda a fost aplicată pe modele de dimensiuni foarte mari, confirmând că modelele de limbaj de mari dimensiuni, precum GPT-3 (175B) sau PaLM (540B), reușesc să prezinte rezultate foarte bune atunci când le sunt oferite mai multe detalii și sunt ghidate către răspunsul dorit.

Constrained prompting introduce restricții explicite asupra răspunsului generat, cum ar fi formatul de ieșire, numărul de cuvinte sau lista valorilor acceptate. Spre deosebire de prompturile libere, care îi permit modelului să fie creativ și să genereze un rezultat după propriile convingeri, constrained prompting ghidează răspunsul prin specificarea unor restricții structurale sau stilistice.

Experimentele conduse de Lu et al. [53] pe modelele GPT-3, BLOOM și OPT au evidențiat că LLM-urile întâmpină dificultăți semnificative în respectarea constrângerilor numerice și că specificarea formatului dorit în prompt minimizează riscul unor outputuri nestructurate. Deși eficiența metodei depinde în continuare de cunoștințele de bază ale modelului și de capacitatea

acestuia de a urma instrucțiunile, s-a observat că modelele de scară mare sunt mult mai capabile să gestioneze aceste cerințe. Lucrarea propune și strategii de atenuare bazate pe in-context learning, deschizând direcții importante de cercetare pentru îmbunătățirea controlului modelelor generative.

Pentru a putea evalua cât mai bine evoluția performanței prompturilor pe baza sarcinilor anterioare, vor fi definite versiuni diferite, fiecare cu un anumit grad de complexitate, începând de la cea mai simplă și ajungând la cea mai complexă.

4.2. Definirea Prompturilor

Prompturile vor fi aplicate mai întâi asupra unui subset de date pentru a putea determina mai rapid punctele care necesită îmbunătățire. Subsetul ales va conține 10 conversații, selectate astfel încât să acopere toate cele 5 domenii. Pentru fiecare domeniu vor fi alese câte două conversații: una simplă și una complexă. Această selecție va permite evaluarea comportamentului modelelor atât în scenarii directe, cât și în situații care implică ambiguitate, contexte complicate sau multiple etape de rezolvare.

Fiecare domeniu va dispune de propriul set de prompturi ce vor fi testate pentru a putea analiza în detaliu comportamentul fiecărui model în parte. Ulterior acestor teste, pe baza rezultatelor obținute, vor fi evaluate prompturi complete — care să integreze toate cele trei sarcini: intenție, satisfacție și rezumat — pe cele mai performante modele. Astfel, pentru sarcina de identificare a intențiilor sunt definite patru versiuni de prompt engineering, în timp ce pentru clasificarea satisfacției și generarea rezumatului vor fi testate doar trei prompturi. Sarcina de detectare a intenției clientului conține substanțial mai multe clase, comparativ cu determinarea satisfacției (care prezintă doar trei clase) sau generarea rezumatului (care nu dispune de nicio clasă); așadar, o atenție sporită a fost acordată testării intențiilor prin adăugarea unui prompt suplimentar față de celelalte două sarcini.

Versiunile inițiale ale prompturilor de **detectare a intenției** din această lucrare vor fi bazate pe zero-shot prompting, fără structurare explicită sau exemple. Modelului îi va fi oferită o instrucțiune foarte simplă, fără prea multe detalii: *„extrage intenția clientului”*. Prin acest prim prompt, modelul va avea libertatea să catalogheze fiecare conversație așa cum consideră că este mai bine. În această situație, trebuie luat în calcul faptul că, cel puțin în cazul intențiilor, modelul poate să nu găsească cuvintele exacte din lista oficială de intenții, ci ar putea folosi sinonime sau alte metode de a exprima aceeași idee.

Al doilea tip de prompt va introduce role prompting și constrained prompting ca prim nivel de ghidare a modelului. Spre deosebire de varianta inițială, modelului îi este atribuit un rol explicit — acela de agent specializat în analiza conversațiilor telefonice din call-center-uri, adaptat chiar pentru domeniul relevant. În plus, modelului îi este furnizată o listă explicită de intenții posibile, specifică fiecărui domeniu, ceea ce transformă problema dintr-una deschisă într-una de clasificare într-un spațiu controlat de etichete. Prin acest rol și prin constrângerea vocabularului, modelul este orientat către un anumit stil de răspuns și către o perspectivă specifică de analiză a dialogului, fără a i se impune încă o structură rigidă de output.

Promptul trei extinde abordarea anterioară prin introducerea structured prompting. Spre deosebire de versiunea precedentă, promptul este organizat în secțiuni distincte, cu titluri clare — SARCINA, REGULI, INTENȚII DISPONIBILE, CONVERSAȚIE — iar regulile definesc explicit ordinea intențiilor, limita de etichete și diferențierea dintre acțiunile clientului și cele ale operatorului.

Ultimul tip de prompt reprezintă varianta cea mai avansată, care combină toate tipurile de prompting utilizate anterior cu few-shot prompting. Sunt introduse exemple concrete care

ilustrează explicit modul corect de mapare între o replică a clientului și eticheta corespunzătoare din lista de intenții. Formatul exemplelor este consistent cu formatul răspunsului cerut, astfel încât modelul să poată învăța direct din structura promptului cum să răspundă. Scopul acestei versiuni este de a minimiza erorile reziduale identificate în etapele anterioare — în special confuzia dintre intenții similare și ordonarea greșită a etichetelor — și de a obține un comportament cât mai predictibil al modelului. Distribuția tipurilor de prompturi este regăsită în tabelul 4.1.

Versiune	**Tipuri de prompting**
V1	Zero-shot
V2	Zero-shot + Role + Constrained prompting
V3	Zero-shot + Role + Constrained + Structured
V4	Zero-shot + Role + Constrained + Structured + Few-shot

Tabelul 4.1: Evoluția strategiilor de prompt engineering pentru clasificarea intenției

Fiecare versiune de prompt va fi evaluată pe baza rezultatelor obținute pe subsetul menționat și vor fi identificate tipare de erori recurente, precum răspunsuri greșite, interpretări ambigue ale intenției sau nerespectarea constrângerilor impuse. Pe baza acestor observații, prompturile vor fi rafinate progresiv prin introducerea de reguli explicite, structuri mai clare și restricții suplimentare asupra outputului.

Pasul final al etapei de prompting va fi reprezentat de testarea acestor prompturi pe întreg setul de date pentru modelele apelate prin API, pentru a putea observa comportamentul acestora atunci când volumul de solicitări crește semnificativ și este necesară procesarea unui set extins de date. Pentru modelele locale, care prezintă o latență mult mai mare și necesită mai multe resurse hardware, va fi rulat doar pe jumătate din setul total. Vor putea fi analizate astfel, pentru toate modelele și pentru fiecare prompt în parte, latența, acuratețea și mai ales nivelul de halucinație, care poate apărea pe parcursul procesării unui număr mare de conversații.

Versiunea inițială a prompturilor de **satisfacție** adoptă aceeași abordare zero-shot, fără niciun context suplimentar oferit modelului. Promptul furnizează modelului instrucțiunea de a analiza conversația și de a determina nivelul de satisfacție al clientului, solicitând un răspuns dintr-un singur cuvânt. Astfel, se poate stabili o linie de bază pentru comparația cu versiunile ulterioare, testând în ce măsură modelul poate clasifica corect satisfacția clientului fără definiții explicite ale claselor, exemple de referință sau indicații cu privire la etichetele acceptate.

Versiunea a doua introduce role prompting și constrained prompting. Modelului îi este atribuit rolul de expert în analiza satisfacției clienților în conversații de call-center, iar spațiul de răspuns este controlat prin definiții explicite ale fiecărei clase și prin reguli de decizie menite să reducă confuzia dintre categorii. Principala dificultate în această sarcină o reprezintă distincția dintre clasa *neutru* și celelalte două clase, unde clientul poate accepta o soluție fără a exprima satisfacție, dar tonul sau replicile sale pot indica o frustrare implicită — de exemplu, prin exprimări de tipul *„ce să fac”* sau prin ironie.

Versiunea a treia reprezintă cea mai avansată variantă, combinând role prompting, constrained prompting, structured prompting și few-shot prompting. Structura promptului este organizată în secțiuni distincte — CONVERSAȚIE, SARCINĂ, DEFINIȚII, REGULI, EXEMPLE — iar adăugarea exemplelor concrete urmărește să ghideze modelul în diferențierea cazurilor ambigue, în special a granițelor dintre clasa neutru și clasa negativ, considerată cel mai dificil caz de clasificare din această sarcină.

Prompturile de rezumat prezintă o structură de evoluție similară cu cele de satisfacție, cu o particularitate importantă: tipul rezumatului generat — scurt, mediu sau lung — este

condiționat de nivelul de satisfacție al clientului, determinat anterior. Astfel, o conversație cu satisfacție pozitivă va genera un rezumat scurt, una neutră va genera un rezumat mediu, iar una negativă un rezumat lung, care să surprindă nuanțele contextului și ale frustrării clientului.

Versiune	Tipuri de prompting	Sarcina
V1	Zero-shot	Satisfactie, Rezumat
V2	Zero-shot + Role + Constrained	Satisfactie, Rezumat
V3	Zero-shot + Role + Constrained + Structured + Few-shot	Satisfactie, Rezumat

Tabelul 4.2: Evoluția strategiilor de prompt engineering pentru clasificarea satisfacției și generarea rezumatului

Versiunea întâi este zero-shot și testează capacitatea modelului de a genera rezumate fără nicio constrângere explicită, lăsând acestuia libertatea deplină de a decide lungimea și structura răspunsului. Singura cerință prezentă este solicitarea unui rezumat al conversației, fără indicații privind formatul sau conținutul minim.

Versiunea a doua adaugă role prompting și constrained prompting. Modelului îi este atribuit rolul de expert în sumarizarea conversațiilor de call-center, iar promptul include cerințe explicite de lungime în cuvinte, limba de redactare și conținut minim obligatoriu. Cerințele de conținut vizează menționarea problemei principale și a rezultatului final al conversației, apropiindu-se de o structură impusă.

Versiunea a treia combină toate tehnicile anterioare cu few-shot prompting, introducând exemple concrete de conversații și rezumatele lor corespunzătoare, câte unul pentru fiecare tip de rezumat — scurt, mediu și lung. Exemplele urmăresc să calibreze modelul atât în privința lungimii, cât și a stilului narativ așteptat, oferindu-i un șablon concret de urmat pentru fiecare nivel de satisfacție a clientului.

4.3. Prompt design pentru detectarea intențiilor utilizatorului

Detectarea intențiilor utilizatorului reprezintă prima etapă funcțională a sistemului de analiză și constituie baza tuturor deciziilor ulterioare. În acest context, modelul lingvistic trebuie să realizeze o clasificare controlată, bazată pe conținutul unei conversații naturale, adesea ambigue sau incomplete. Pentru a putea determina sensibilitatea modelelor la modul în care le sunt adresate instrucțiunile, fiecare variantă de prompt engineering va dispune de două subversiuni, care vor diferi prin stilul de adresare și prin limbaj. Pe baza rezultatelor obținute, va fi selectată varianta optimă, urmând ca testările ulterioare să fie continuate exclusiv pe aceasta.

4.3.1. Testarea modelelor locale

4.3.1.1 Prima versiune de prompting - definire, testare, rafinare

Pentru prima versiune de prompting, pentru a respecta structura menționată anterior, a fost folosită metoda de *zero-shot prompting*. Așadar, descrierea cerinței a fost compusă din câteva propoziții care îi cereau modelului să analizeze conversația și să extragă intenția consumatorului. În plus, îi mai era cerut să se rezume la un singur cuvânt sau o expresie scurtă, cum se poate observa în anexa A.1.

Pe baza primei rulări cu acest prompt au putut fi extrase câteva observații. În primul rând, așa cum era de așteptat, niciunul dintre cele două modele locale nu a generat răspunsuri care să se potrivească cu lista predefinită. Un prompt zero-shot fără o structură clară sau constrângeri stricte de format produce rezultate care nu pot fi folosite în mod direct pentru o clasificare automată. Ambele modele au avut o acuratețe formală de 0% pe subsetul de test, toate cele 10 conversații fiind clasificate incorect (neputând fi mapate pe clasele fixe), dar motivele pentru care au fost catalogate greșit sunt destul de diferite.

Modelul RoMistral a generat răspunsuri foarte scurte, uneori formate dintr-un singur cuvânt sau chiar incomplete (de tipul *„Întrebare”*, *„Întoarcere”* sau *„Rezolvarea problemei”*). În câteva situații, modelul a intrat într-o buclă de repetiție, generând sintagma *„Întrebare:...”* până la atingerea limitei de tokeni. Deși din punct de vedere semantic modelul a intuit corect contextul în doar 3 din 10 cazuri, acest comportament indică faptul că modelul nu poate interpreta corect structura promptului fără instrucțiuni mai clare și mai restrictive.

RoGemma a înțeles bine contextul semantic, dar a folosit mult prea multe cuvinte în generarea răspunsului. În loc de o etichetă unică, modelul a generat descrieri detaliate. De exemplu, pentru un card pierdut, a explicat pe larg că *„intenția clientului este de a raporta un card de debit pierdut și de a primi asistență în blocarea acestuia...”*. Analizând aceste răspunsuri brute, se poate observa o diferență foarte mare între acuratețea formală (care a fost 0%) și cât de bine a înțeles modelul, de fapt, discuțiile. Deși niciun răspuns nu a putut fi mapat automat pe etichetele fixe, 9 din cele 10 răspunsuri conțineau intenția corectă, doar că erau scrise sub formă de text liber.

A doua variantă a promptului a fost simplificată, folosind un limbaj mult mai concis. Modelului i-a fost dată conversația și i s-a cerut un singur lucru: *Rezumă în 2-3 cuvinte ce a cerut clientul:*. Deși promptul este mult mai scurt și mai simplu decât cele inițiale, a prezentat rezultate mult mai bune. Cum era de așteptat, modelele nu au intuit nici de data aceasta intențiile din lista noastră, dar au prezentat o îmbunătățire semantică.

RoGemma a clasificat toate cele zece conversații corect din punct de vedere semantic. Totuși, o limitare care a apărut în acest prompt este latența, care a crescut semnificativ. Deși în prompt a fost introdusă cerința de a folosi 2-3 cuvinte, modelul a ignorat complet sugestia și a acționat contrar, generând răspunsuri mai lungi și mult mai detaliate decât în varianta inițială.

RoMistral, în schimb, a avut o evoluție mult mai bună față de varianta inițială: a reușit să clasifice nouă din zece răspunsuri corect semantic. Noua formulare a promptului a eliminat complet acele bucle de repetiție și răspunsurile goale. Modelul a oferit rezumate scurte și la obiect, exact cum i-a fost cerut.

Model	**Corect semantic V1**	**Corect semantic V1.2**	**Latența V1 (s)**	**Latența V1.2 (s)**
RoMistral	3/10	9/10	9.2	12.2
RoGemma	9/10	10/10	18.7	35.2

Tabelul 4.3: Comparație rezultate zero-shot prompting – detectare intenție, modele locale

Tabelul 4.3 pune în balanță rezultatele celor două modele. Deși RoGemma clasifică mult mai bine intențiile, indiferent de prompt, generarea răspunsurilor durează mai mult, iar cerința impusă de a respecta lungimea răspunsului generat nu este respectată. RoMistral, pe de altă parte, are nevoie de prompturi specifice pentru a da rezultate bune, dar respectă mult mai mult cerințele și este mult mai rapid.

Testarea pe întregul set de 100 de conversații a confirmat și observațiile obținute anterior pe subsetul inițial. RoGemma a demonstrat un comportament consistent pe întregul set: toate

cele 100 de răspunsuri au conținut informație semantică utilă, fără apariția unor răspunsuri goale sau a buclelor de repetiție.

În cazul modelului RoMistral, comportamentul a fost ușor mai variabil. Opt dintre răspunsuri au fost prea scurte sau prea generale pentru a fi utilizate eficient, iar într-un caz răspunsul a fost complet gol. Cu toate acestea, comparativ cu prima versiune de prompt — unde problemele de format afectau aproximativ 60% din răspunsuri — varianta a doua a dus la o îmbunătățire semnificativă. Latența medie a fost comparabilă cu cea a modelului RoGemma.

4.3.1.2 A doua versiune de prompting

Versiunea a doua de prompt pentru modelele locale a adăugat rolul de agent specializat și lista completă de intenții disponibile pentru fiecare domeniu, ca prim mecanism de ghidare a modelului. Spre deosebire de versiunea anterioară, în care modelului i se solicita pur și simplu să identifice intenția clientului fără niciun context suplimentar, în această versiune modelul trebuie să clasifice intențiile pe baza listei oferite. Au fost testate două variante ale acestui prompt, cu roluri și formulări ușor diferite, pentru a evalua sensibilitatea modelelor la modul în care este definit contextul ocupațional, așa cum se poate observa în Anexa A.2.

Diferența dintre cele două prompturi este la nivel de limbaj și adresare și se rezumă la gradul de strictețe și control aplicat modelului. Primul prompt adoptă o abordare mai descriptivă și mai narativă, finalizându-se cu o întrebare directă care oferă modelului mai multă flexibilitate. În schimb, al doilea prompt utilizează un ton imperativ și strict operațional, înlocuind întrebarea cu o comandă sintactică explicită.

Rezultatele prezentate în Tabelul 4.4 indică faptul că introducerea tehnicilor de *role prompting* și *constrained prompting* a avut efecte diferite asupra celor două modele locale. Dintre acestea, RoGemma a beneficiat cel mai mult de pe urma informațiilor suplimentare introduse în prompt. A doua versiune a promptului a obținut cea mai bună performanță pe subsetul de 10 conversații, cu o acuratețe de 70% și un scor F1 Macro de 0.556. Creșterea scorului F1 sugerează că modelul a reușit să echilibreze mai bine clasificarea între diferitele clase de intenții, nu doar să crească numărul total de predicții corecte.

Model & Prompt	**N**	**Accuracy**	**F1 Macro**
RoGemma V2-1	10	60%	0.472
RoGemma V2-2	10	70%	0.556
RoGemma V2-2	50	54%	0.423
RoMistral V2-1	10	50%	0.333
RoMistral V2-2	10	40%	0.250
RoMistral V2-1	50	40%	0.315

Tabelul 4.4: Rezultate V2 role + constrained prompting pentru detectarea intenției — modele locale

În cazul RoMistral, efectul noilor constrângeri a fost limitat. Prima variantă a obținut o acuratețe de 50%, în timp ce a doua variantă a scăzut la 40%, împreună cu o reducere a scorului F1 Macro de la 0.333 la 0.250. Aceste rezultate sugerează că modelul este mai sensibil la formulările restrictive. Spre deosebire de RoGemma, care pare să utilizeze definițiile și instrucțiunile pentru a delimita mai bine clasele, RoMistral nu reușește să transforme informațiile suplimentare din prompt într-o îmbunătățire a performanței.

Extinderea evaluării la un subset de 50 de conversații confirmă tendințele observate pe setul inițial. Pentru RoGemma, acuratețea scade de la 70% la 54%, iar F1 Macro de la 0.556 la

0.423. Deși această scădere indică o capacitate mai redusă de generalizare pe un volum mai mare și mai variat de date, modelul continuă să depășească performanțele RoMistral. În cazul acestuia din urmă, rezultatele rămân relativ constante, cu o acuratețe de 40% și un scor F1 Macro de 0.315 pe cele 50 de conversații, ceea ce sugerează un comportament stabil, dar limitat din punct de vedere al performanței.

Per ansamblu, a doua versiune de prompting se dovedește mai eficientă pentru RoGemma decât pentru RoMistral. Rezultatele arată că aceste modele pot răspunde foarte diferit la aceleași tehnici de proiectare a prompturilor. În timp ce RoGemma valorifică informațiile suplimentare oferite prin rol și constrângeri explicite, RoMistral nu reușește să beneficieze în aceeași măsură de acestea, ceea ce evidențiază importanța adaptării strategiilor de prompting la particularitățile fiecărui model.

4.3.1.3 A treia versiune de prompting

Versiunea a treia conține tehnici de role prompting, *structured prompting* și *constrained prompting*, oferind modelelor un cadru mai clar pentru interpretarea sarcinii. Cele două variante diferă prin două aspecte: ordinea în care sunt prezentate conversația și regulile, respectiv numărul de reguli incluse. Prima variantă plasează conversația la finalul promptului și include patru reguli de bază, în timp ce a doua variantă adaugă două reguli suplimentare: una care permite deducerea intenției din context și alta care impune alegerea celei mai apropiate etichete din listă atunci când nicio variantă nu se potrivește exact.

Model & Prompt	**N**	**Accuracy**	**Intenție în răspuns**	**F1 Macro**
RoGemma V3-1	10	70%	90%	0.556
RoGemma V3-2	10	70%	90%	0.538
RoGemma V3-1	50	52%	76%	0.397
RoMistral V3-1	10	40%	90%	0.256
RoMistral V3-2	10	40%	90%	0.262
RoMistral V3-1	50	46%	56%	0.384

Tabelul 4.5: Rezultate V3 role + constrained + structured prompting pentru detectarea intenției

Ambele modele au obținut aceleași scoruri de acuratețe pentru cele două variante de prompt pe subsetul de 10 conversații: 70% pentru RoGemma și 40% pentru RoMistral. Modificarea ordinii informațiilor și adăugarea regulilor suplimentare din varianta a doua nu influențează semnificativ performanța clasificării.

O problemă frecventă la ambele modele este returnarea sistematică a două etichete, chiar și pentru conversațiile cu o singură intenție. Toate conversațiile din setul de date cu 10 conversații au o singură intenție gold, însă modelele adaugă adesea o a doua etichetă, plasând uneori intenția corectă pe poziția a doua. Coloana *Gold în răspuns* din tabelul 4.5 indică proporția cazurilor în care eticheta corectă apărea în răspuns, indiferent de poziție. Pe subsetul ales, ambele modele și ambele variante ating 90%, ceea ce sugerează că modelele identifică în general intenția corectă, dar nu respectă constrângerea de a returna o singură etichetă sau este confuz cu privire la definiția intenției. În cazul RoMistral, 5 dintre cele 6 clasificări greșite conțineau eticheta corectă pe poziția a doua.

Extinderea evaluării la 50 de conversații a produs o scădere a performanței pentru ambele modele. RoGemma a obținut o acuratețe de 52%, comparativ cu 70% pe subsetul inițial, iar proporția rezultatelor care aveau intenția adevarată printre intențiile multiple detectate a scăzut

la 76%, ceea ce indică și erori reale de identificare, nu doar de formatare. Pe setul extins au fost observate răspunsuri de tip *placeholder* (de exemplu, *intentie_principala, intentie_secundara*) în locul etichetelor reale, precum și situații în care modelul a generat mai mult de două intenții, ignorând constrângerile impuse.

RoMistral a obținut o acuratețe de 56%, semnificativ mai mică decât cea a RoGemma. Această diferență indică un număr mai mare de erori de identificare efectivă, nu doar de formatare. Pe subsetul extins au apărut și cazuri în care modelul a reprodus fragmente din dialog în loc să genereze etichetele cerute. Comparativ cu RoMistral, RoGemma demonstrează o înțelegere semantică mai bună a conversațiilor și obține scoruri mai ridicate pe ambele subseturi.

4.3.1.4 A patra versiune de prompting

Ultima versiune adaugă few-shot prompting la cadrul definit în varianta precedentă, prin includerea unor exemple concrete de conversații cu intențiile corespunzătoare, câte două exemple per domeniu. Cele două variante diferă prin tipul exemplelor utilizate: varianta 1 folosește exemple scurte, cu o singură replică a clientului, în timp ce varianta 2 include dialoguri complete operator-client, mai apropiate de structura conversațiilor reale din dataset, așa cum se poate vedea în A.4. Regulile și constrângerile rămân identice cu cele din testele anterioare.

Model & Prompt	**N**	**Accuracy**	**Intenție în răspuns**	**F1 Macro**
RoGemma V4-1	10	70%	90%	0.556
RoGemma V4-2	10	60%	80%	0.429
RoGemma V4-1	50	58%	88%	0.481
RoMistral V4-1	10	50%	70%	0.385
RoMistral V4-2	10	50%	90%	0.333
RoMistral V4-1	50	56%	78%	0.510

Tabelul 4.6: Rezultate V4 few-shot prompting pentru detectarea intenției — modele locale

Pe subsetul de 10 conversații, RoGemma obține 70% acuratețe cu prima variantă și 60% cu varianta a doua, ceea ce indică faptul că exemplele scurte funcționează mai bine decât dialogurile complete pentru acest model. Proporția cazurilor în care intenția corectă era una dintre intențiile returnate scade de la 90% la 80% între cele două variante, sugerând că dialogurile complete introduc mai multă confuzie în identificarea intenției, nu doar în formatarea răspunsului. O problemă persistentă la ambele variante este tendința modelului de a returna mai multe intenții decât este necesar — deși toate conversațiile au o singură intenție, RoGemma adaugă sistematic o a doua etichetă. Scorul F1 Macro confirmă această tendință: scade de la 0.556 la 0.429 între cele două variante, penalizând inconsistența clasificării pe fiecare clasă în parte.

RoMistral obține 50% pentru ambele variante, similar cu varianta fără few-shot. Dialogurile complete ajută RoMistral să identifice intenția corectă, chiar dacă nu o plasează întotdeauna pe prima poziție — proporția cazurilor în care intenția corectă apare în răspuns crește de la 70% la 90% între cele două variante. Scorul F1 Macro mai scăzut al variantei 2 (0.333 față de 0.385) arată că, deși intenția corectă este mai des prezentă în răspuns, clasificarea finală este mai puțin precisă, modelul identificând intenția dar prioritizând-o greșit.

Față de varianta precedentă, versiunea few-shot aduce îmbunătățiri pentru ambele modele pe subsetul extins de 50 de conversații. RoGemma crește de la 52% la 58%, iar proporția răspunsurilor care conțin intenția corectă se menține ridicată, la 88%, confirmând că modelul identifică în general corect intenția, dar continuă să genereze etichete suplimentare. RoMistral

crește de la 46% la 56%, cel mai bun rezultat al său din întreaga etapă de testare, cu un F1 Macro de 0.510.

4.3.2. Testarea modelelor apelate prin API

4.3.2.1 Prima veriune de prompting

Prima versiune de prompt pentru detectarea intenției a reprezentat un experiment zero-shot, utilizând exact aceleași prompturi ca și în cazul modelelor locale.

Model & Prompt	N	Acc. formală	Acc. semantică	F1 Macro
GPT-4.1-mini V1-1	10	0%	90%	0.000
GPT-4.1-mini V1-2	10	0%	90%	0.000
GPT-4.1-mini V1-2	100	0%	60%	0.000
Gemini-2.5-flash V1-1	10	0%	20%	0.000
Gemini-2.5-flash V1-2	10	0%	80%	0.000
Gemini-2.5-flash V1-2	100	0%	59%	0.000
command-r7b V1-1	10	0%	30%	0.000
command-r7b V1-2	10	0%	90%	0.000
command-r7b V1-2	100	0%	49%	0.000

Tabelul 4.7: Rezultate V1 zero-shot pentru detectarea intenției — modele API.

Acuratețea formală a fost 0% pentru toate cele trei modele și ambele variante; toate predicțiile au fost clasificate ca *alta_solicitare*, eticheta de fallback returnată atunci când răspunsul generat nu se potrivește cu niciuna dintre etichetele controlate. În absența listei de intenții disponibile în prompt, modelele nu aveau cum să producă formatul așteptat.

Analiza răspunsurilor brute oferă însă o imagine diferită. Varianta 2, cu formatul de completare directă, a produs răspunsuri semantic corecte în 9 din 10 cazuri pentru GPT și command R7B, respectiv 8 din 10 pentru Gemini. Varianta 1 a produs rezultate semantice semnificativ mai slabe pentru Gemini și Command R7B (20%, respectiv 30%) ambele modele generând răspunsuri prea generale de tipul *Reclamație* sau *Solicitare*, fără să identifice domeniul specific al cererii. GPT a menținut 90% și pentru această variantă, dovedindu-se mai robust la schimbările de formulare.

Pe tot setul de 100 de conversații, acuratețea semantică a scăzut la 60% pentru GPT, 59% pentru Gemini și 49% pentru Command. Distincția dintre acuratețea formală (0%) și cea semantică (49–90%) ilustrează limitarea fundamentală a zero-shot prompting pentru sarcini de clasificare cu vocabular controlat. Modelele înțeleg corect intenția clientului, dar nu pot produce eticheta exactă fără să o cunoască. Spre deosebire de estimarea satisfacției, unde clasele sunt intuitive și universale, etichetele de intenție sunt specifice domeniului și nu pot fi deduse din context.

4.3.2.2 A doua versiune de prompting

Versiunea a doua de prompt a introdus lista explicită de intenții disponibile, role prompting și constrained prompting. Această schimbare a produs un salt major față de prima variantă: acuratețea formală a trecut de la 0% la valori între 60% și 90% pe subsetul de 10 conversații. GPT cu varianta 1 și Gemini cu varianta 2 obțin cel mai bun rezultat — 90% acuratețe și F1 Macro de 0.818 — cu câte o singură eroare, ambele pe același caz dificil din domeniul serviciilor

publice. Command rămâne cel mai slab dintre cele trei modele, cu 60% pentru varianta 1 și 70% pentru varianta 2.

Model & Prompt	N	Accuracy	F1 Macro
GPT-4.1-mini V2-1	10	90%	0.818
GPT-4.1-mini V2-2	10	80%	0.667
GPT-4.1-mini V2-1	100	73%	0.673
Gemini-2.5-flash V2-1	10	80%	0.667
Gemini-2.5-flash V2-2	10	90%	0.818
Gemini-2.5-flash V2-1	100	68%	0.615
command-r7b V2-1	10	60%	0.444
command-r7b V2-2	10	70%	0.538
command-r7b V2-1	100	51%	0.416

Tabelul 4.8: Rezultate V2 role + constrained prompting pentru detectarea intenției — modele API

Comportamentul celor două variante diferă între modele. GPT obține rezultate mai bune cu varianta 1 (90% față de 80%), în timp ce Gemini se descurcă mai bine cu varianta a doua (90% față de 80%). Command urmează același tipar ca Gemini, varianta 2 aducând o îmbunătățire. Aceste diferențe sugerează că GPT interpretează mai eficient instrucțiunile concise, în timp ce Gemini și Command beneficiază de regulile suplimentare de deducere a intenției din context.

Domeniul serviciilor publice rămâne cel mai dificil pentru toate modelele și ambele variante, cu acurațețe de 0-50% pe subsetul de 10 conversații. Celelalte patru domenii sunt clasificate corect în proporție de 50-100%, erorile apărând preponderent în conversațiile complexe.

Pe baza rezultatelor de pe subset, varianta 1 a fost selectată pentru GPT, iar varianta 2 pentru Gemini și Command. Totuși, pentru simplificarea comparației, tot setul de 100 de conversații a fost testat cu varianta 1 pentru toate modelele.

Extinderea la 100 de conversații a evidențiat o scădere a performanței față de subsetul mic: GPT a coborât la 73%, Gemini la 68%, iar Command la 51%. Erorile sunt distribuite preponderent în conversațiile complexe și în domeniul serviciilor publice, unde acurațețea ajunge la 30% pentru Command și 50% pentru celelalte două modele. Domeniile banking și telecom prezintă cele mai multe confuzii între intenții semantice similare: de exemplu, *card_pierdut* vs.*card_blocat* sau *problema_roaming* vs.*problema_internet*, ceea ce indică faptul că distincțiile fine dintre etichete rămân dificile chiar și cu lista de intenții disponibilă.

4.3.2.3 A triea versiune de prompting

Față de versiunea a doua, adăugarea structured prompting nu a adus îmbunătățiri consistente pentru modelele API. GPT scade de la 90% la 70% cu varianta 1, iar Command rămâne la valori similare. Singurul model care menține performanța este Gemini, cu 90% acurațețe pe varianta 1, același scor ca cel mai bun rezultat al său din versiunea 2.

GPT și Command obțin rezultate mai bune cu a două varianta de formulare a promptului, în timp ce Gemini performează mai bine cu varianta 1, sugerând că organizarea secțiunilor în prompt influențează diferit cele trei modele.

Un tipar comun tuturor modelelor și ambelor variante este performanța slabă pe domeniul serviciilor publice: 0% pentru toate cele trei modele în varianta 2 și cel mult 50% în varianta 1. Erorile din acest domeniu implică în mod repetat aceleași perechi de intenții confundate:

Model & Prompt	N	Accuracy	F1 Macro
GPT-4.1-mini V3-1	10	70%	0.583
GPT-4.1-mini V3-2	10	80%	0.667
GPT-4.1-mini V3-1	100	69%	0.599
Gemini-2.5-flash V3-1	10	90%	0.818
Gemini-2.5-flash V3-2	10	80%	0.727
Gemini-2.5-flash V3-1	100	65%	0.532
command-r7b V3-1	10	70%	0.538
command-r7b V3-2	10	60%	0.472
command-r7b V3-1	100	56%	0.455

Tabelul 4.9: Rezultate V3 role + constrained + structured prompting pentru detectarea intenției — modele API

programare_ghiseu vs. *acte_incomplete* sau *sesizare_problema* vs. *reclamatie_serviciu* — ceea ce indică o ambiguitate semantică între aceste etichete, nu doar o limitare a modelelor.

Pe tot setul de 100 de conversații cu varianta 1, performanța scade pentru toate modelele: GPT ajunge la 69%, Gemini la 65%, iar Command la 56%. Erorile sunt distribuite similar cu V2, cu concentrări în conversațiile complexe și în domeniile banking și telecom, unde intenții semantice apropiate continuă să producă confuzii. Scorurile pe 100 de conversații sunt comparabile sau ușor mai mici față de V2, ceea ce confirmă că structurarea explicită a promptului nu aduce un avantaj clar pentru această sarcină în cazul modelelor API.

4.3.2.4 A patra versiune de prompting

Spre deosebire de modelele locale, unde few-shot prompting nu a adus îmbunătățiri consistente, pentru Command-r7b această versiune produce cele mai bune rezultate. Pe subsetul de 10 conversații, Command atinge 90% acuratețe cu varianta 1 și un F1 Macro de 0.818, cel mai bun rezultat al său din întreaga etapă de testare. Varianta 2 scade la 80% acuratețe și F1 Macro de 0.667, similar cu variantele anterioare, ceea ce indică că exemplele scurte funcționează mai bine decât dialogurile complete pentru acest model.

Model & Prompt	N	Accuracy	F1 Macro
GPT-4.1-mini V4-1	10	70%	0.583
GPT-4.1-mini V4-2	10	70%	0.583
GPT-4.1-mini V4-1	100	69%	0.602
Gemini-2.5-flash V4-1	10	80%	0.667
Gemini-2.5-flash V4-2	10	80%	0.667
Gemini-2.5-flash V4-1	100	63%	0.536
command-r7b V4-1	10	90%	0.818
command-r7b V4-2	10	80%	0.667
command-r7b V4-1	100	70%	0.632

Tabelul 4.10: Rezultate V4 few-shot prompting pentru detectarea intenției — modele API

GPT și Gemini nu beneficiază de pe urma exemplelor în același mod. GPT rămâne la 70% și F1 Macro de 0.583 pentru ambele variante, iar Gemini la 80% și 0.667, scoruri similare cu varianta 3. Faptul că F1 Macro rămâne constant între cele două variante pentru ambele modele

indică faptul că tipul exemplelor nu influențează distribuția erorilor pe clase. Diferența față de cel mai bun rezultat al acestor modele din V2 (90% pentru ambele) se menține, sugerând că exemplele few-shot nu compensează performanța obținută cu un prompt mai simplu și mai direct.

Pe tot setul de 100 de conversații cu varianta 1, Command înregistrează cel mai bun rezultat al său la această scară, 70% acuratețe și F1 Macro de 0.632, depășind pentru prima dată GPT (69%, F1 0.602) și apropiindu-se de Gemini (63%). Diferența mai pronunțată dintre acuratețe și F1 Macro la Gemini (63% față de 0.536) indică o distribuție mai dezechilibrată a erorilor între clase comparativ cu GPT și Command. Tiparele de eroare rămân consistente cu versiunile anterioare: banking-ul continuă să producă confuzii între intenții semantice apropiate, iar serviciile publice rămân cel mai dificil domeniu pentru toate modelele, cu acuratețe între 50% și 60%.

4.4. Prompt design pentru estimarea satisfacției clientului

Estimarea nivelului de satisfacție al clientului reprezintă o sarcină aproape la fel de dificilă precum detectarea intențiilor, întrucât satisfacția nu este întotdeauna exprimată explicit și necesită o interpretare contextuală a întregii conversații. În cadrul interacțiunilor reale din call-center, nemulțumirea poate fi exprimată indirect, prin ironie, resemnare sau remarci aparent neutre.

4.4.1. Testarea modelelor locale

4.4.1.1 Prima versiune de prompting

Prima versiune de prompting pentru estimarea satisfacției a utilizat zero-shot prompting, așa cum se poate vedea în anexa A.5. Cele două variante testate diferă în modul de formulare a cererii. Varianta 1 solicită explicit un răspuns dintr-o listă de trei clase: pozitiv, neutru sau negativ și folosește o exprimare foarte clară, cerându-i modelului să analizeze cu atenție conversația. Varianta 2, în schimb, folosește un format de completare directă, fără enumerarea claselor posibile și utilizează un limbaj direct, lipsit de context complet.

Prima variantă a produs o acuratețe globală de 60% pentru ambele modele pe subsetul de 10 conversații, dar analiza per clasă evidențiază faptul că nicio conversație etichetată ca neutră nu a fost clasificată corect. Atât RoMistral, cât și RoGemma au catalogat toate conversațiile ca pozitive sau negative, fără nicio predicție neutră, în ciuda faptului că 4 din cele 10 conversații aveau această etichetă. Clasele pozitiv și negativ au fost identificate corect în 100% din cazuri, ceea ce înseamnă că acuratețea de 60% reflectă exclusiv erorile asociate clasei neutre. Scorul macro-F1 de 0.5 subliniază aceeași problemă.

Model & Prompt	N	Accuracy	F1 Macro	Pozitiv	Neutru	Negativ
RoMistral V1-1	10	60%	0.508	100%	0%	100%
RoMistral V1-2	10	0%	0.000	0%	0%	0%
RoMistral V1-1	50	62%	0.509	100%	0%	85%
RoGemma V1-1	10	60%	0.500	100%	0%	100%
RoGemma V1-2	10	20%	0.121	0%	50%	0%
RoGemma V1-1	50	70%	0.589	100%	6%	100%

Tabelul 4.11: Rezultate V1 zero-shot pentru estimarea satisfactiei — modele locale

A doua variantă, care nici nu menționează tipurile de satisfacție, a produs rezultate mult mai slabe, așa cum era de așteptat. RoMistral a obținut 0% acuratețe, cu 8 din 10 răspunsuri clasificate ca fiind necunoscute, modelul generând răspunsuri lungi și explicații, comportament nesurprinzător având în vedere lipsa de indicații clare. RoGemma a obținut 20% acuratețe, însă din alte motive. Acesta a reușit să clasifice corect doar o parte dintre conversațiile neutre. Ambele modele s-au dovedit sensibile la modul de formulare al promptului și la nivelul de detaliu oferit.

Prima versiune de prompt a fost testată ulterior pe un subset mai mare, de 50 de conversații. RoGemma a atins 70% acuratețe și un scor F1 Macro de 0.589. De această dată a reușit să clasifice corect 6% dintre conversațiile neutre, mai precis 1 din cele 16 conversații neutre. Clasele pozitivă și negativă au fost identificate corect aproape în totalitate. RoMistral a reușit să îmbunătățească acuratețea doar cu două procente. Scorul F1 Macro a crescut până la 0.509, însă clasa neutră a rămas la 0%.

Modelele locale înțeleg diferența dintre conversațiile pozitive și cele negative, însă nu reușesc să recunoască în mod corespunzător clasa neutră fără definiții explicite sau exemple suplimentare. Simpla enumerare a celor trei opțiuni nu este suficientă pentru a ghida modelul către o clasificare corectă a satisfacției neutre.

4.4.1.2 A doua versiune de prompting

Versiunea a doua de prompt a introdus role prompting și constrained prompting, adăugând modelului un rol de expert în analiza satisfacției și definiții explicite pentru fiecare clasă, așa cum este prezentat în anexa A.6. Cele două variante diferă prin nivelul de detaliu al definițiilor: varianta inițială oferă descrieri scurte și reguli generale, pe când a doua include expresii concrete pentru fiecare clasă și o regulă critică pentru diferențierea între neutru și negativ.

Cele două modele au reacționat diferit la această versiune, confirmând încă o dată sensibilitatea ridicată a modelelor locale la formularea promptului.

Adăugarea definițiilor în prima variantă de prompt nu a produs nicio îmbunătățire pentru RoGemma, menținând aceleași rezultate ca în varianta zero-shot: 60% acuratețe și clasa neutră la 0%. În schimb, a doua variantă de prompt, care conține descrieri mai bine detaliate, a produs o îmbunătățire semnificativă: acuratețea a crescut la 80%, iar scorul F1 Macro a ajuns la 0.802. În plus, clasa neutră a fost recunoscută în 75% din cazuri (3 din 4 conversații neutre clasificate corect). Distribuția predicțiilor include 4 conversații neutre, 4 negative și 2 pozitive. RoGemma a avut nevoie de exemple lingvistice concrete pentru a diferenția clasa neutră de celelalte două.

Model & Prompt	N	Accuracy	F1 Macro	Pozitiv	Neutru	Negativ
RoGemma V2-1	10	60%	0.508	100%	0%	100%
RoGemma V2-2	10	80%	0.802	67%	75%	100%
RoGemma V2-2	50	70%	0.685	64%	50%	90%
RoMistral V2-1	10	30%	0.154	100%	0%	0%
RoMistral V2-2	10	40%	0.367	100%	0%	33%
RoMistral V2-2	50	44%	0.385	100%	0%	40%

Tabelul 4.12: Rezultate V2 role + constrained prompting pentru estimarea satisfacției — modele locale

Spre deosebire de RoGemma, prima variantă de prompt a produs rezultate mai slabe pentru RoMistral comparativ cu prompt-ul zero-shot: acuratețea a scăzut la 30%, iar toate cele 10 conversații au fost clasificate ca pozitive, indiferent de conținut. A doua formulare a promptului

a adus o îmbunătățire parțială: acuratețea a ajuns la 40%, cu clasa negativă recunoscută în 33% din cazuri, însă nicio conversație neutră nu a fost identificată.

Pe subsetul de 50 de conversații, performanța scade ușor pentru RoGemma, acuratețea ajungând la 70%. F1 Macro a crescut la 0.685, iar 50% dintre conversațiile neutre au fost clasificate corect. Totuși, modelul întâmpină dificultăți și în cazul clasei pozitive, identificând corect doar 64% dintre cazuri. RoMistral clasifică bine toate conversațiile pozitive, însă are în continuare probleme în identificarea clasei neutre, dar mai ales a celei negative.

4.4.1.3 A treia versiune de prompting

Versiunea a treia de prompt a adăugat few-shot prompting peste structura deja prezentă în subcapitolul anterior, incluzând exemple concrete de conversații cu etichetele corespunzătoare. De asemenea, promptul a fost structurat mai clar, așa cum este prezentat în anexa A.7. Prima formulare a folosit exemple scurte, cu o singură replică per clasă, iar a doua a inclus dialoguri mai lungi, cu accent explicit pe diferența dintre neutru și negativ. Totuși, această exemplificare și introducerea unor indicații mai precise nu au produs, din păcate, rezultatele dorite.

RoGemma a obținut o acuratețe de 70% și un scor F1 Macro de 0.707 în prima formulare, pe subsetul de 10 conversații, un rezultat ușor mai slab decât în varianta de prompt care nu conține few-shot. Atât clasa negativă, cât și cea neutră au înregistrat scăderi în acuratețe. Varianta a doua a promptului a produs o scădere și mai pronunțată, exemplele mai lungi dezorientând modelul în loc să clarifice granița dintre neutru și negativ.

Model & Prompt	N	Accuracy	F1 Macro	Pozitiv	Neutru	Negativ
RoGemma V3-1	10	70%	0.707	100%	50%	67%
RoGemma V3-2	10	50%	0.489	100%	0%	67%
RoGemma V3-1	50	60%	0.615	100%	31%	55%
RoMistral V3-1	10	30%	0.389	67%	0%	33%
RoMistral V3-2	10	30%	0.324	67%	25%	0%
RoMistral V3-1	50	36%	0.382	79%	0%	35%

Tabelul 4.13: Rezultate V3 few-shot prompting pentru estimarea satisfacției — modele locale

RoMistral a performat mult mai slab decât în versiunile anterioare. Varianta 1 a obținut 30% acuratețe; modelul a preluat formatul secțiunii *SATISFACȚIE IDENTIFICATĂ* din prompt și a generat text în loc să returneze un singur cuvânt pentru clasificare. Varianta a doua a obținut același scor de 30%, dar cu un tipar diferit: clasa negativă nu a fost clasificată corect în niciun caz.

Pe subsetul de 50 de conversații, RoMistral nu a reușit să clasifice corect aproape jumătate dintre conversații (21 din 50 de răspunsuri au fost clasificate ca fiind necunoscute). Promptul lung, combinat cu exemplele few-shot, a depășit capacitatea modelului de a produce un răspuns simplu și concis. RoGemma a avut, în schimb, rezultate mai bune, menținând o acuratețe de 60%, dar a întâmpinat din nou dificultăți în identificarea negativității și a neutralității.

Adăugarea unui prompt few-shot nu aduce beneficii modelelor locale pentru sarcini de clasificare, ci introduce zgomot suplimentar. A doua formulare a variantei de prompt, care conține role prompting și constrained prompting, rămâne cea mai bună variantă.

4.4.2. Testarea modelelor apelate prin API

4.4.2.1 Prima versiune de prompting

Modelele apelate prin API au utilizat aceleași prompturi de testare precum modelele locale. Asemănător testelor pentru RoGemma și RoMistral, cele două formulări ale promptului zero-shot au generat rezultate diferite.

A doua formulare a promptului, care folosea un format de completare directă fără a specifica clasele posibile, a produs rezultate inutilizabile pentru GPT și Command R7B; ambele modele au generat text liber în loc să răspundă cu un singur cuvânt, rezultând în 10 din 10 răspunsuri clasificate ca fiind necunoscute. Gemini, totuși, a utilizat un singur cuvânt pentru clasificare și a obținut 40% acuratețe, dar în continuare clasa neutră nu a fost clasificată deloc. Modelele API tind să completeze fraze deschise cu explicații și nu folosesc răspunsuri concise. Prima formulare a promptului, care specifică explicit lista de clase, a fost superioară și a fost utilizată ulterior pentru întregul set.

Model & Prompt	N	Accuracy	F1 Macro	Pozitiv	Neutru	Negativ
GPT-4.1-mini V1-1	10	80%	0.806	100%	50%	100%
GPT-4.1-mini V1-2	10	0%	0.000	0%	0%	0%
GPT-4.1-mini V1-1	100	74%	0.727	100%	68%	70%
Gemini-2.5-flash V1-1	10	70%	0.669	100%	25%	100%
Gemini-2.5-flash V1-2	10	40%	0.367	100%	0%	33%
Gemini-2.5-flash V1-1	100	80%	0.717	100%	40%	92%
command-r7b V1-1	10	50%	0.448	67%	0%	100%
command-r7b V1-2	10	0%	0.000	0%	0%	0%
command-r7b V1-1	100	72%	0.593	60%	12%	100%

Tabelul 4.14: Rezultate V1 zero-shot pentru estimarea satisfacției — modele API

În prima variantă, GPT a obținut cel mai bun rezultat pe subsetul de 10 conversații, având 80% acuratețe și scorul F1 Macro de 0.806, clasa neutră fiind identificată în jumătate din cazuri. Această performanță este comparabilă cu cea mai bună configurație a modelelor locale, mai exact varianta a doua de prompting pentru RoGemma, și superioară tuturor celorlalte modele API pe acest subset. Gemini a obținut 70% acuratețe, iar Command R7B a performat mai slab, cu 50% acuratețe. F1 Macro subliniază același dezechilibru: GPT atinge 0.8, în timp ce Gemini și Command R7B obțin scoruri mai scăzute, 0.67 respectiv 0.44.

Pe întregul set de 100 de conversații, Gemini a atins o acuratețe de 80%, cel mai ridicat scor, datorită performanței bune pe clasa negativă, 92% din conversațiile negative fiind clasificate corect. Totuși, F1 Macro de 0.717 rămâne sub GPT, care a obținut 0.727 cu o distribuție mai echilibrată pe clase. Clasa neutră rămâne problematică pentru toate modelele: GPT a identificat-o în 68% din cazuri, Gemini în 40%, iar Command R7B în doar 12%. Command R7B tinde să catalogheze conversațiile ca fiind negative mai des, având un bias pentru această clasă (82 din 100 de conversații clasificate ca negative). În consecință, F1 Macro este mai scăzut, ajungând la 0.593, semnificativ sub celelalte două modele.

Latența rămâne un avantaj clar al modelelor API față de cele locale: GPT și Command R7B au răspuns în sub 1 secundă per conversație, comparativ cu 7–15 secunde pentru modelele locale. Gemini a necesitat în medie aproximativ 6 secunde, în principal din cauza timpului TTFT ridicat.

4.4.2.2 A doua versiune de prompting

A doua versiune de prompting, care adaugă în plus role prompting și constrained prompting, a produs rezultate diferite pentru cele trei modele.

Command R7B a înregistrat cea mai mare creștere în rezultatele generale odată cu introducerea specificațiilor privind nivelul de satisfacție al clientului. Varianta întâi, care conține definiții scurte și câteva reguli de bază, a avut o acuratețe de 80%, comparativ cu promptul *zero-shot*. Și scorul F1 Macro a crescut de la 0.448 la 0.800. De asemenea, clasa neutră a fost clasificată corect pe subsetul de 10 conversații. Varianta a doua a prezentat rezultate mai slabe; modelul a clasificat 7 din 10 conversații ca neutre, inclusiv conversații pozitive și negative. Indicațiile suplimentare din acest prompt au introdus un bias către clasa neutră.

GPT, pe de altă parte, fluctuează în rezultate, prezentând o diferență destul de mare între scorurile F1 ale celor două variante (0.707 și 0.806). Modelul identifică foarte bine clasele pozitive și negative în a doua versiune de prompting, însă întâmpină în continuare dificultăți în identificarea conversațiilor neutre.

Gemini nu a prezentat nicio îmbunătățire sesizabilă pe subsetul de 10 conversații în niciuna dintre cele două variante, menținând o acuratețe de 70% și un scor F1 de 0.669, similar cu varianta *zero-shot*. Totuși, clasa neutră a fost clasificată greșit de mai multe ori, spre deosebire de celelalte două clase.

Model & Prompt	**N**	**Accuracy**	**F1 Macro**	**Pozitiv**	**Neutru**	**Negativ**
GPT-4.1-mini V2-1	10	70%	0.707	100%	50%	67%
GPT-4.1-mini V2-2	10	80%	0.806	100%	50%	100%
GPT-4.1-mini V2-1	100	70%	0.699	100%	68%	63%
Gemini-2.5-flash V2-1	10	70%	0.669	100%	25%	100%
Gemini-2.5-flash V2-2	10	70%	0.669	100%	25%	100%
Gemini-2.5-flash V2-1	100	78%	0.629	100%	12%	100%
command-r7b V2-1	10	80%	0.800	67%	100%	67%
command-r7b V2-2	10	70%	0.676	33%	100%	67%
command-r7b V2-1	100	74%	0.779	87%	100%	60%

Tabelul 4.15: Rezultate V2 role + constrained prompting pentru estimarea satisfației — modele API

Pe tot setul de 100 de conversații a fost testată prima variantă de prompt, deoarece a avut cele mai echilibrate rezultate pentru toate cele trei modele. Command R7B a obținut cel mai bun rezultat general, având cel mai mare scor F1 Macro (0.779) și reușind să identifice corect toate conversațiile neutre. Totuși, performanța pe clasa negativă a scăzut la 60%, ceea ce sugerează că modelul tinde să favorizeze eticheta neutră în unele situații ambigue.

GPT a rămas cel mai echilibrat model per ansamblu, clasificând corect toate conversațiile pozitive și 68% dintre cele neutre, însă performanța pe clasa negativă a scăzut la 63%. Gemini a obținut cea mai mare acuratețe globală, 78%, datorită identificării foarte bune a claselor pozitivă și negativă, ambele cu 100%, dar a continuat să întâmpine dificultăți în detectarea clasei neutre, clasificând corect doar 12% dintre aceste conversații.

Rezultatele arată că niciun model nu reușește să balanseze perfect cele trei clase. Command R7B înțelege mai bine neutralitatea atunci când primește definiții clare, GPT oferă cele mai stabile rezultate între clase, iar Gemini tinde să favorizeze extremele pozitiv și negativ în detrimentul cazurilor neutre.

4.4.2.3 A treia versiune de prompting

Few-shot prompting produce rezultate complet diferite pentru cele trei modele. Ambele variante de prompt au produs aceleasi rezultate pentru GPT, care prezinta cele mai bune rezultate ale sale, echilibrand foarte bine cele trei clase. Toate conversațiile pozitive și negative au fost identificate corect, însă clasa neutră este recunoscută doar în jumătate dintre cazuri. Scorul F1 Macro ajunge la 0.806, cel mai ridicat rezultat obținut pe subsetul de 10 conversații.

Gemini a prezentat o scădere a performanței; ambele variante au avut 60% acuratețe, iar clasa neutră nu a fost identificată deloc. Scorul F1 Macro a scăzut semnificativ până la aproximativ 0.5 pentru ambele versiuni. Și Command R7B a prezentat un comportament similar, fiind încurcat de exemplele oferite în prompturi; scorul său F1 a scăzut chiar până la intervalul 0.26-0.33.

Exemplele few-shot îl ajută doar pe GPT să genereze rezultate mai bune. Pentru Gemini și Command R7B, versiunile anterioare rămân superioare, acestea prezentând comportamente similare modelelor locale.

Model & Prompt	N	Accuracy	F1 Macro	Pozitiv	Neutru	Negativ
GPT-4.1-mini V3-1	10	80%	0.806	100%	50%	100%
GPT-4.1-mini V3-2	10	80%	0.806	100%	50%	100%
GPT-4.1-mini V3-2	100	83%	0.816	100%	80%	80%
Gemini-2.5-flash V3-1	10	60%	0.508	100%	0%	100%
Gemini-2.5-flash V3-2	10	60%	0.500	100%	0%	100%
Gemini-2.5-flash V3-2	100	78%	0.642	100%	16%	98%
command-r7b V3-1	10	20%	0.333	33%	0%	33%
command-r7b V3-2	10	20%	0.267	0%	0%	67%
command-r7b V3-2	100	14%	0.124	0%	0%	23%

Tabelul 4.16: Rezultate V3 few-shot prompting pentru estimarea satisfacției — modele API

Pe întregul set de 100 de conversații, GPT a obținut cele mai bune rezultate din toate experimentele realizate pentru această sarcină. Modelul a atins 83% acuratețe și un scor F1 Macro de 0.816, reușind să clasifice corect toate conversațiile pozitive și 80% dintre cele neutre și negative.

Gemini a menținut o acuratețe ridicată pe setul extins - 78% - datorită performanței foarte bune pe clasa negativă, unde a identificat corect 98% dintre conversații. Totuși, clasa neutră a rămas problematică, cu doar 16% dintre exemple clasificate corect. Modelul continuă să favorizeze clasificările extreme, pozitiv și negativ, chiar și atunci când primește exemple suplimentare în prompt.

4.5. Prompt design pentru generarea rezumatelor conversatiilor

Generarea automată a rezumatelor a fost introdusă în sistem ca mecanism de sprijin pentru analiza și interpretarea interacțiunilor telefonice și facilitează verificarea rezultatelor robotului telefonic. Investigațiile suplimentare ale conversațiilor pot fi, astfel, realizate mult mai ușor, citirea întregii conversații nefiind utilă în acest context.

Pentru această componentă a sistemului trebuie luat în calcul nivelul de detaliu al rezumatului în funcție de satisfacția clientului. În cazul interacțiunilor încadrate ca pozitive sau

neutre, rezumatul are un rol pur informativ și este limitat la prezentarea de bază a cererii și a rezultatului obținut. În schimb, pentru conversațiile clasificate drept negative sau considerate cazuri limită, rezumatul trebuie să ofere un context mai amplu, incluzând descrierea problemei, modul în care a decurs dialogul și cauza principală a nemulțumirii clientului.

4.5.1. Testarea modelelor locale

Modelele locale, RoMistral și RoGemma, au fost testate pentru trei variante diferite de prompt engineering pentru generarea rezumatului, așa cum au fost definite în tabelul 4.2. Fiecare variantă are, la rândul ei, două versiuni de prompturi, care se bazează pe aceeași structură, dar sunt formulate diferit.

Acest pas intermediar poate sublinia detalii cu privire la modul în care structura prompturilor influențează rezultatul generat. Fiecare versiune va fi testată inițial pe un subset cu zece conversații, iar ulterior fiecare model va fi testat pe un subset mai mare, cu 50 de conversații, folosind formularea care a avut cele mai bune rezultate pentru fiecare dintre cele trei variante. Aceste subseturi cu 50 de conversații au fost alese având în vedere latența ridicată cu care sunt rulate modelele locale; rularea întregului set ar fi fost mult prea îndelungată.

4.5.1.1 Prima versiune de prompting

Promptul inițial dispune de două scenarii zero-shot. Ambele folosesc exprimări similare, diferența de bază dintre ele fiind ordinea în care îi sunt oferite modelului instrucțiunea și conversația. Primul prompt îi oferă întâi instrucțiunea (*Rezumă această conversație*) și abia apoi îi este oferit dialogul, pe când celălalt îi oferă întâi conversația și ulterior cerința.

Ambele modele au obținut scoruri ROUGE-1 în intervalul 0.33-0.37 și BERTScore F1 în intervalul 0.72-0.74, ceea ce sugerează că, deși suprapunerea lexicală cu rezumatul de bază (cel generat în partea de adnotare) este destul de mică, modelele au înțeles semantic conversația. Scorul ROUGE-2, între 0.12 și 0.15, sugerează că modelele formulează rezumatele diferit față de referință, chiar dacă informația transmisă este similară.

Rezumatele generate au avut între 74 și 90 de cuvinte. Rezultatele nu sunt neașteptate, având în vedere lipsa constrângerilor de lungime. Chiar și fără aceste constrângeri, un avantaj al acestora este faptul că modelele nu au detaliat prea mult conversația și au înțeles, în mare parte, că un rezumat nu trebuie să fie foarte lung.

Model & Prompt	N	ROUGE-1	ROUGE-L	BERT F1	Cuv. medii
RoGemma V1-1	10	0.332	0.223	0.727	90.1
RoGemma V1-2	10	0.347	0.255	0.731	90.2
RoGemma V1-2	50	0.331	0.233	0.722	90.7
RoMistral V1-1	10	0.366	0.260	0.736	74.7
RoMistral V1-2	10	0.343	0.244	0.729	73.9
RoMistral V1-1	50	0.329	0.234	0.730	67.9

Tabelul 4.17: Rezultate V1 zero-shot pentru generarea rezumatului — modele locale

Toate cele trei scoruri utilizate ca metode de evaluare a acestei sarcini au fost apropiate. Pentru RoGemma, versiunea a doua a adus o îmbunătățire de câteva sutimi pentru fiecare scor în parte, comparativ cu RoMistral, care a prezentat rezultate mai bune în urma primului prompt. Diferențele dintre cele două scenarii de zero-shot sunt mici pentru ambele modele, ceea

ce indică faptul că schimbarea ordinii informațiilor în prompt nu are un impact semnificativ asupra calității rezumatului.

Analizând răspunsurile brute ale modelelor în această etapă, se observă o tendință de a include informații redundante în rezumate și de a dezvolta excesiv anumite aspecte ale conversației. În plus, o parte dintre rezumate se încheie abrupt, deoarece modelele depășesc limita de tokeni disponibilă, generând conținut mai amplu decât este necesar.

Extinderea testării la un subset de 50 de conversații a confirmat observațiile inițiale. Atât RoGemma, cât și RoMistral au prezentat scoruri asemănătoare celor originale, cu doar câteva sutimi mai mici.

Totuși, o diferență destul de mare între modele este numărul de cuvinte generate pentru fiecare rezumat, așa cum se poate observa în tabelul 4.17. RoMistral are tendința de a genera rezumate mult mai scurte, de aproximativ 70 de cuvinte, comparativ cu RoGemma, care folosește în medie 90 de cuvinte pentru fiecare rezumat.

4.5.1.2 A doua versiune de prompting

Versiunea a doua de prompt a introdus role prompting și constrained prompting, atribuind modelului rolul de expert în sumarizarea conversațiilor de call-center și specificând explicit tipul de rezumat și lungimea în cuvinte. Cele două variante diferă prin nivelul de detaliu al cerințelor, cum se poate observa în anexa A.9 — varianta 1 include tipul de rezumat adaptat satisfacției și o listă de cerințe de bază, iar varianta 2 adaugă limitari explicite în legătură cu fluxul conversației: motivul apelului, acțiunile operatorului și rezultatul final.

Constrângerile de lungime au ridicat scorurile pentru RoMistral cu câteva sutimi, dar diferența dintre cele două formulări nu este majoră. Explicațiile cu privire la fluxul în care trebuie extrasă informația din conversație nu au adus îmbunătățiri. Cu prima versiune de prompt, RoMistral a înregistrat cel mai bun rezultat al său, având un scor ROUGE-1 de aproape 0.4 și ROUGE-L de 0.3, creșteri de aproape 4 sutimi comparativ cu promptul zero-shot. A reușit de asemenea să încadreze 40% din conversații în limitele impuse.

Model & Prompt	N	ROUGE-1	ROUGE-L	BERT F1	În limite
RoGemma V2-1	10	0.341	0.247	0.722	30%
RoGemma V2-2	10	0.343	0.262	0.737	30%
RoGemma V2-2	50	0.317	0.223	0.722	38%
RoMistral V2-1	10	0.395	0.307	0.746	40%
RoMistral V2-2	10	0.375	0.283	0.749	50%
RoMistral V2-1	50	0.357	0.253	0.737	64%

Tabelul 4.18: Rezultate V2 role + constrained prompting pentru generarea rezumatului — modele locale

RoGemma, pe de altă parte, a prezentat rezultate asemănătoare cu cele din varianta zero-shot; niciunul dintre scoruri nu s-a îmbunătățit sesizabil. Multe dintre rezultate au rămas la fel sau au scăzut cu o sutime. Acest model nu beneficiază de pe urma unor cerințe și limitări și, de această dată, a continuat să genereze rezumate mai lungi.

Și de această dată, ambele modele au generat rezumate lungi și incomplete, atingând limita de tokeni disponibilă. Mai mult, pe lângă tendința de a include detalii excesive, în unele conversații modelele au repetat integral sau parțial conținutul generat anterior, ceea ce a condus la răspunsuri semnificativ mai lungi decât ar fi fost necesar. Această limitare este

relevantă într-un context real de utilizare, deoarece repetarea informației este determinată de comportamentul modelului și nu de modul de formulare a promptului, fiind astfel mai dificil de controlat prin tehnici de prmpt engineering.

Testarea pe subsetul de 50 de conversații a produs o scădere a scorurilor ROUGE pentru ambele modele. Totuși, scorul BERT a rămas constant, indicând că, deși modelele reacționează diferit atunci când este oferit un volum mai mare de date, generând rezumate diferite, ele își păstrează componenta semantică în continuare. Respectarea limitelor de lungime s-a îmbunătățit semnificativ; pentru RoMistral, 32 din 50 de rezumate au fost încadrate corect, adică 64%, așa cum poate fi observat în tabelul 4.18, cel mai bun rezultat înregistrat până la acest moment. RoGemma a continuat să genereze rezumate mai lungi decât i-a fost impus, având doar 19 conversații din 50 în limitele de lungime.

4.5.1.3 A treia versiune de prompting

Versiunea a treia de prompt a adăugat few-shot prompting prin includerea unui exemplu concret pentru fiecare tip de rezumat, alături de structured prompting prin organizarea explicită a promptului în secțiuni distincte. Ambele variante folosesc aceleași dialoguri demonstrative, diferența constând în forma rezumatului exemplificat, precum in anexa A.10: varianta 1 prezintă rezumate în exprimare liberă, în timp ce varianta 2 include rezumate cu structură explicită — motiv apel, acțiuni operator, rezultat — reflectând același contrast dintre constrângerile de conținut și cele de structură aplicat anterior în V2.

Model & Prompt	N	ROUGE-1	ROUGE-L	BERT F1	În limite
RoGemma V3-1	10	0.168	0.113	0.634	30%
RoGemma V3-2	10	0.209	0.150	0.656	30%
RoGemma V3-2	50	0.204	0.139	0.654	40%
RoMistral V3-1	10	0.235	0.166	0.692	50%
RoMistral V3-2	10	0.259	0.180	0.708	60%
RoMistral V3-2	50	0.233	0.164	0.695	62%

Tabelul 4.19: Rezultate V3 few-shot prompting - generarea rezumatului — modele locale

Tabelul 4.19 subliniază că adăugarea unor exemple în prompt a produs rezultate mult mai slabe față de varianta 2 pentru ambele modele. RoGemma a scăzut de la ROUGE-1 de 0.34 la 0.16, iar BERTScore F1 a coborât sub 0.65. RoMistral a scăzut de la ROUGE-1 de 0.39 la 0.23-0.26. Aceasta este cea mai slabă performanță înregistrată pentru ambele modele în întreaga etapă de testare.

Analizând rezultatele brute ale modelelor iese în evidență influența negativă pe care exemplele au avut-o asupra modelelor. RoMistral a învățat din exemplele few-shot să genereze rezumate scurte, toate având aproximativ 40 de cuvinte, dar a aplicat această tehnică pentru toate tipurile de rezumat, inclusiv pentru cele lungi. Varianta 2, cu rezumate puțin mai lungi și mai concis structurate, a atenuat parțial această problemă, ridicând scorul ROUGE-1 la 0.259. RoGemma a continuat să genereze rezumate de aproximativ 80 de cuvinte, similar cu versiunile anterioare, dar calitatea lexicală a scăzut. Totuși, RoGemma nu a prezentat diferențe sesizabile între cele două versiuni.

Analiza rezumatelor brute evidențiază faptul că, deși RoGemma demonstrează o bună înțelegere a contextului conversațional, modelul manifestă tendința de a reproduce exemplele few-shot incluse în prompt și de a le adăuga la finalul rezumatului generat. Ca urmare, numeroase

răspunsuri depășesc limitele de lungime stabilite. În contrast, RoMistral nu prezintă acest comportament, ceea ce conduce la o respectare mai bună a constrângerilor impuse.

În această etapă, rezumatele nu mai sunt trunchiate din cauza depășirii numărului maxim de tokeni generați. Cu toate acestea, în cazul RoGemma, exemplele few-shot reproduse în răspuns apar frecvent incomplete, deoarece generarea se oprește înainte de finalizarea lor.

Pentru modelele locale de 7 miliarde de parametri, exemplele few-shot în sarcina de rezumare introduc mai mult zgomot decât beneficii. În loc să ghideze modelul către un format mai bun, exemplele scurte au determinat modele precum RoMistral să uniformizeze lungimea rezumatelor, ignorând informația despre tipul solicitat.

4.5.2. Testarea modelelor apelate prin API

Testarea modelelor apelate prin API a produs rezultate mai bune decât cele obținute de modelele locale, atât din punct de vedere al scorurilor, cât și al respectării constrângerilor de lungime. Latența modelelor API este semnificativ mai scăzută comparativ cu cea a modelelor locale.

4.5.2.1 Prima versiune de prompting

Prima versiune de prompt dispune de două scenarii zero-shot. Cele două variante folosesc exprimări similare, diferența de bază dintre ele fiind ordinea în care îi sunt oferite modelului instrucțiunea și conversația.

Model & Prompt	N	ROUGE-1	ROUGE-L	BERT F1	Cuv. medii
GPT-4.1-mini V1-1	10	0.389	0.288	0.747	89.7
GPT-4.1-mini V1-2	10	0.402	0.286	0.754	71.7
GPT-4.1-mini V1-2	100	0.390	0.270	0.747	78.1
Gemini-2.5-flash V1-1	10	0.339	0.256	0.737	161.5
Gemini-2.5-flash V1-2	10	0.419	0.288	0.752	92.9
Gemini-2.5-flash V1-2	100	0.389	0.276	0.748	109.0
command-r7b V1-1	10	0.349	0.247	0.731	91.8
command-r7b V1-2	10	0.413	0.309	0.759	78.1
command-r7b V1-2	100	0.388	0.274	0.745	82.0

Tabelul 4.20: Rezultate V1 zero-shot pentru generarea rezumatului — modele API

Toate cele trei modele au înțeles bine sensul conversațiilor, prezentând scoruri BERT de aproximativ 0.75. Deși scorurile ROUGE rămân în continuare moderate, varianta a doua reușește să le îmbunătățească pentru Gemini și Command R7B, în timp ce pentru GPT diferența dintre cele două variante este neglijabilă. Similar modelelor locale, schimbarea ordinii informațiilor în prompt nu are un impact semnificativ asupra calității rezumatului.

Înainte de începerea testării modelelor apelate prin API, lui Gemini i-a fost mărit numărul de tokeni acceptați, datorită răspunsurilor foarte scurte pe care le genera inițial. Limitarea inițială producea răspunsuri incomplete, spre deosebire de GPT și Command, care nu au necesitat această modificare. Pe baza acestui parametru, Gemini a generat rezumate foarte lungi și detaliate, atingând în prima variantă de prompt aproximativ 160 de cuvinte în medie.

GPT și Command, ale căror limite de tokeni nu au fost modificate, au generat uneori răspunsuri incomplete din cauza acestor constrângeri, multe propoziții sau cuvinte fiind întrerupte

brusc. Totuși, ideea de bază a conversației este prezentă în toate rezumatele generate de aceste modele, aspect vizibil și în scorurile BERT ridicate. În general, elementele excluse de aceste limitări au fost în principal detalii suplimentare, mai puțin esențiale.

Extinderea testării la întregul set de 100 de conversații a confirmat observațiile inițiale, cu scoruri similare celor de pe subset. Numărul mediu de cuvinte per rezumat a crescut ușor pentru toate modelele, Gemini continuând să genereze cele mai lungi rezumate, în timp ce GPT rămâne cel mai stabil în ceea ce privește lungimea.

4.5.2.2 A doua versiune de prompting

Versiunea a doua de prompt, care introduce role prompting și constrained prompting, a adus îmbunătățiri semnificative în ceea ce privește respectarea limitelor de lungime, față de prima versiune. Cel mai notabil rezultat a fost obținut de GPT cu varianta 2, care a respectat limitele în toate cele 10 cazuri din subset — prima dată în întreaga etapă de testare când un model atinge 100% la această metrică. Gemini a înregistrat o îmbunătățire importantă pe ROUGE-L, care a crescut de la 0.301 în prima variantă la 0.342 în a doua, iar BERTScore a atins 0.770, cel mai ridicat scor din toate experimentele de rezumare. Command R7B a menținut scoruri competitive, respectând limitele în 8 din 10 cazuri.

Model & Prompt	**N**	**ROUGE-1**	**ROUGE-L**	**BERT F1**	**În limite**
GPT-4.1-mini V2-1	10	0.403	0.311	0.754	90%
GPT-4.1-mini V2-2	10	0.399	0.309	0.759	100%
GPT-4.1-mini V2-2	100	0.381	0.275	0.752	98%
Gemini-2.5-flash V2-1	10	0.416	0.301	0.763	70%
Gemini-2.5-flash V2-2	10	0.426	0.342	0.770	80%
Gemini-2.5-flash V2-2	100	0.400	0.290	0.759	62%
command-r7b V2-1	10	0.395	0.293	0.756	70%
command-r7b V2-2	10	0.411	0.314	0.763	80%
command-r7b V2-2	100	0.351	0.247	0.735	89%

Tabelul 4.21: Rezultate V2 role + constrained prompting - generarea rezumatului - API

Spre deosebire de modelele locale, unde varianta 2 nu a adus îmbunătățiri consistente față de varianta 1, pentru modelele API aceasta s-a dovedit mai bună pentru toate cele trei modele, atât ca scoruri, cât și ca respectare a lungimii. Din acest motiv, varianta 2 a fost selectată pentru testarea pe întregul set de 100 de conversații.

De această dată, rezumatele brute au respectat mult mai bine limitele impuse, iar marea majoritate nu au mai fost incomplete nici în cazul GPT, nici în cazul Command. Nu au mai fost incluse detalii redundante, iar structura sumarizărilor a fost mult mai clară și mai concisă. Doar în câteva cazuri GPT s-a oprit brusc. Aceste situații nu au influențat scorul BERT; informația esențială se regăsește în rezumat, chiar dacă acesta nu este complet.

Testarea pe întregul set a evidențiat un comportament diferit între modele. GPT a reușit să respecte limitele impuse în 98 din 100 de cazuri, cu scoruri ROUGE stabile. Gemini a încadrat doar 62 din 100 de rezumate în limite, lungimea medie crescând la 82 de cuvinte. Command R7B a respectat constrângerile în 89 din 100 de cazuri, deși scorurile de calitate au rămas ușor mai scăzute comparativ cu GPT și Gemini.

4.5.2.3 A treia versiune de prompting

Versiunea a treia de prompt a adăugat few-shot prompting a generat rezultate superioare față de celelalte două versiuni. Spre deosebire de modelele locale, unde exemplele few-shot au produs o scădere semnificativă a scorurilor, pentru modelele API această versiune a reprezentat cea mai bună performanță din întreaga etapă de testare. Toate cele trei modele au înregistrat îmbunătățiri ale scorurilor ROUGE față de versiunile anterioare. GPT cu varianta 1 a obținut scoruri globale de 0.441 pentru ROUGE-1 și 0.778 pentru BERTScore, respectând totodată constrângerile de lungime pentru toate cele 10 conversații. Gemini a atins cel mai ridicat scor ROUGE-1 din toate experimentele, 0.462, cu varianta a doua de prompt. Command R7B a menținut scoruri similare cu celelalte două modele, având un scor ROUGE-1 de 0.394 și respectând limitele în 9 din 10 cazuri. BERTScore rămâne stabil între variante pentru toate modelele, independent de formularea promptului.

Model & Prompt	N	ROUGE-1	ROUGE-L	BERT F1	În limite
GPT-4.1-mini V3-1	10	0.441	0.358	0.778	100%
GPT-4.1-mini V3-2	10	0.364	0.280	0.749	100%
GPT-4.1-mini V3-1	100	0.401	0.292	0.758	100%
Gemini-2.5-flash V3-1	10	0.460	0.337	0.767	70%
Gemini-2.5-flash V3-2	10	0.462	0.351	0.774	80%
Gemini-2.5-flash V3-1	100	0.416	0.299	0.764	67%
command-r7b V3-1	10	0.394	0.311	0.763	90%
command-r7b V3-2	10	0.380	0.267	0.753	80%
command-r7b V3-1	100	0.381	0.275	0.754	93%

Tabelul 4.22: Rezultate V3 few-shot prompting - generarea rezumatului - API

Comparând cele două variante pe subsetul de 10 conversații, prima variantă obține rezultate mai bune pentru GPT și Command R7B, în timp ce pentru Gemini varianta a doua prezintă scoruri superioare — același comportament observat și în versiunea V2. Gemini răspunde mai bine la structurarea explicită a outputului, în timp ce GPT și Command R7B beneficiază mai mult de libertatea oferită de constrângerile de conținut fără structură impusă.

Asemănător promptului precedent, modelele nu se mai confruntă cu rezumate incomplete la fel de des; singurele situații în care acest comportament persistă sunt conversațiile cu feedback negativ, care necesită un nivel mai ridicat de detaliere. Informațiile importante și detaliile relevante sunt deja prezente în aceste rezumate, însă modelele tind să le extindă suplimentar. GPT se oprește uneori brusc în aceste conversații, păstrând totuși răspunsul în limitele impuse. Această limitare nu este problematică în acest caz, având în vedere rezultatele semnificativ mai bune pe care le obține în celelalte metrici de evaluare.

Pentru testarea pe întregul set de 100 de conversații a fost utilizată prima variantă, care reușește să echilibreze cel mai bine toate scorurile. GPT a menținut respectare a limitelor de lungime pentru toate conversațiile in set, cu un ROUGE-1 de 0.401. Gemini a încadrat 67 din 100 de dialoguri în limite, atingând un ROUGE-1 de 0.416. Command R7B a respectat constrângerile în 93% dintre cazuri, cu un ROUGE-1 de 0.381.

4.5.3. Concluzii cu privire la strageiile de prompt engineering

Experimentele realizate pe cele trei sarcini — detectarea intențiilor, estimarea satisfacției și generarea rezumatelor — subliniază la câteva concluzii generale care pot fi utile în proiectarea

prompturilor pentru sarcini similare de clasificare și generare de text.

Complexitatea optimă a promptului depinde mai degrabă de model decât de sarcină. Modelele mai mari, accesate prin API (GPT-4.1-mini, Gemini-2.5-flash), obțin în mod constant rezultate bune cu prompturi de complexitate medie — varianta a doua de prompt, care combină role prompting și constrained prompting — fără a avea neapărat nevoie de exemple few-shot de clasificare sau de o structurare mai rigidă. În schimb, modelele locale de 7 miliarde de parametri sunt mult mai sensibile la modul de formulare a promptului și răspund mai puțin stabil la tehnici avansate: în cazul RoMistral și RoGemma, few-shot prompting a dus chiar la o scădere a performanței în sarcina de rezumare.

Zero-shot prompting nu este suficient atunci când este necesar un vocabular controlat. Atât timp cât lista de etichete nu este inclusă explicit în prompt, modelele înțeleg corect sensul conversației, dar nu reușesc să producă eticheta dorită. În experimente, acuratețea formală a fost 0% pe sarcina de detecție a intențiilor în regim zero-shot pentru toate modelele, deși înțelegerea semantică a fost mult mai bună, situată între 60–90%. Această diferență între *înțelegerea mesajului* și *respectarea formatului* este importantă pentru sistemele de clasificare automate.

Clasa neutră este, în mod constant, cea mai dificil de identificat în sarcina de estimare a satisfacției. Indiferent de model sau de varianta de prompt, conversațiile neutre au fost frecvent confundate fie cu cele pozitive, fie cu cele negative. Definițiile mai explicite, împreună cu exemple de expresii concrete (precum *ce să fac* sau formulări de resemnare), au ajutat într-o oarecare măsură, dar nu au eliminat complet problema. În general, modelele tind să favorizeze extremele, comportament observat atât la modelele locale, cât și la cele accesate prin API.

Constrângerile de lungime nu sunt respectate automat doar pentru că sunt specificate în prompt. Introducerea unor limite explicite de cuvinte a îmbunătățit semnificativ respectarea acestora în cazul modelelor API (GPT atingând 98–100% pe versiunile V2 și V3), însă modelele locale au continuat să depășească frecvent limitele, chiar și cu instrucțiuni clare. Acest lucru sugerează că respectarea constrângerilor structurale depinde mai mult de capacitatea modelului decât de formularea promptului.

Sensibilitatea la formulare este mai pronunțată în cazul modelelor locale. Aceleași instrucțiuni, exprimate ușor diferit, au dus la variații de 10–30% în acurateță pentru RoMistral și RoGemma, în timp ce modelele API au fost mult mai stabile între variantele de prompt. Această robustețe reprezintă un avantaj important în scenarii reale, unde prompturile nu sunt întotdeauna perfect standardizate.

5. EVALUAREA MODELELOR

Structura finală a aplicației presupune alegerea unui model de dimensiuni mari care să poată satisface cerințele predeterminate. În urma testelor realizate pot fi extrase concluzii cu privire la performanțele fiecărui tip de model, la cea mai bună metodă de formulare a prompturilor pentru fiecare tip de LLM, precum și la avantajele și dezavantajele fiecăruia.

În practică, robotul telefonic va primi conversația în timp real și va extrage inițial intenția utilizatorului, iar ulterior, la finalul conversației, va determina nivelul de satisfacție și va genera rezumatul. Deși toate aceste sarcini presupun construirea unui singur prompt final, pentru alegerea celei mai bune formulări fiecare componentă a fost testată separat, așa cum a fost prezentat în capitolul anterior. Totuși, acest capitol va evidenția concluziile necesare pentru alegerea modelului și a prompturilor potrivite sarcinilor analizate și va prezenta performanțele modelelor selectate pe baza acurateții, latenței și a diferențelor dintre acestea.

Pentru evaluarea performanței au fost utilizate acuratețea și scorul F1 pentru sarcinile de detectare a intenției și estimare a satisfacției, iar pentru generarea rezumatelor au fost folosite scorurile ROUGE și BERTScore. Latența a fost măsurată, pentru modelele locale, în numărul de secunde dintre trimiterea promptului și generarea ultimului token. Pentru modelele apelate prin API au fost măsurate atât timpul până la primul token (*Time To First Token*), cât și durata completă de generare a răspunsului.

Următoarele subcapitole vor realiza o analiză generală a performanțelor fiecărui model pentru fiecare sarcină, a diferențelor dintre modelele locale și cele apelate prin API și, în final, o analiză a întregului pipeline, prin intermediul căreia va fi ales modelul cu cele mai bune rezultate.

5.1. Evaluarea modelelor - extragerea intentiei

Prima versiune de prompting, deși prezintă cele mai slabe rezultate din cauza lipsei unei liste definite de intenții din care să se aleagă, subliniază totuși nivelul ridicat de înțelegere semantică a modelelor. Atunci când au generat răspunsurile, deși nu au utilizat aceleași cuvinte, LLM-urile au demonstrat că identifică cu o precizie foarte mare intenția de bază a utilizatorului. Odată cu trecerea la a doua variantă de prompting, care introduce lista explicită de intenții, acuratețea modelelor a crescut semnificativ, atingând în cazul modelului GPT un maxim de 73%.

Tabelul 5.1 compară performanțele obținute pe toate versiunile de prompturi de extragere a intenției pentru fiecare model. Modelele apelate prin API precum Gemini și GPT, au demonstrat o performanță optimă fără a necesita instrucțiuni detaliate. Versiunea a doua a fost suficient de clară pentru acestea, în timp ce versiunile ulterioare au produs confuzie. Aceste modele nu necesită o strictețe rigidă în configurarea promptului, obținând rezultate superioare atunci când li se oferă o libertate mai mare de interpretare.

În schimb, modelele locale, dar și modelul Command, au înregistrat cele mai bune rezultate pe ultima versiune de prompt, care include definirea rolului, lista de intenții și exemple de

Model	Versiune	Acc. strict	F1 Macro	Latență mediană (s)
RoMistral	V1	0%	0.000	13.70
RoMistral	V2	40%	0.315	11.34
RoMistral	V3	46%	0.384	14.32
RoMistral	V4	56%	0.510	11.55
RoGemma	V1	0%	0.000	13.47
RoGemma	V2	54%	0.423	13.99
RoGemma	V3	52%	0.397	16.48
RoGemma	V4	58%	0.481	16.66
GPT-4.1-mini	V1	0%	0.000	0.529
GPT-4.1-mini	V2	73%	0.673	0.567
GPT-4.1-mini	V3	69%	0.599	0.648
GPT-4.1-mini	V4	69%	0.602	0.597
Gemini-2.5-flash	V1	0%	0.000	3.441
Gemini-2.5-flash	V2	68%	0.615	6.955
Gemini-2.5-flash	V3	65%	0.532	7.306
Gemini-2.5-flash	V4	63%	0.536	7.737
command-r7b	V1	0%	0.000	0.590
command-r7b	V2	51%	0.416	0.536
command-r7b	V3	56%	0.455	0.516
command-r7b	V4	70%	0.632	0.591

Tabelul 5.1: Comparație performanță prompturi V1–V4 pentru detectarea intenției.

tipul few-shot. Command este singurul model apelat prin API a cărui performanță a crescut progresiv, de la 51.00% în versiunea a doua până la 70.00% în versiunea a patra, comparativ cu GPT și Gemini, ale căror metrici stagnează sau scad pe măsură ce crește complexitatea promptului.

Un aspect relevant pentru arhitecturile locale este acuratețea fără ordine, care măsoară proporția cazurilor în care eticheta corectă apare în răspunsul brut al modelului, indiferent de poziționarea sau formatarea sa. RoGemma atinge un scor de 92.00% la a patra versiune, comparativ cu doar 58.00% în cazul acurateții stricte, în timp ce RoMistral obține 78.00% față de 56.00% acuratețe strictă. Modelele locale înțeleg corect intenția utilizatorului, însă întâmpină dificultăți de aliniere la formatul cerut, returnând eticheta ca o intenție secundară sau într-o structură diferită. De asemena, mare problemă a acestor modele este percepția mai multor intenții acolo unde nu este cazul. Chiar dacă una dintre ele se dovedește a fi corectă, acțiunile operatorului ca și raspuns la întrebarea clientului sunt adesea catalogate ca și intenții.

În privința latenței, mediana este mai reprezentativă decât media în cazul modelelor locale, unde câteva cazuri extreme de generare repetitivă pot să influențeaze semnificativ media. A treia variantă de prompt pentru RoGemma, de exemplu, înregistrează o mediană de 16.5 secunde, față de o medie de 1714 secunde, subliniind că problema nu este sistematică, ci concentrată în câteva conversații complexe care au declanșat bucle de generare. Mediana latenței pentru RoMistral se menține relativ constantă între 11 și 14 secunde, în timp ce pentru modelele API valorile sunt sub 0.65 secunde pentru GPT și command-r7b, și sub 8 secunde pentru Gemini.

Tabelul 5.2 sintetizează cel mai bun prompt identificat pentru fiecare model și performanța globală asociată. GPT prezintă cele mai bune rezultate dintre toate cele cinci modele, având cel mai ridicat scor F1, de 0.673. Modelele accesate prin API reușesc să obțină scoruri apropiate între ele, chiar dacă acestea nu sunt pe deplin satisfăcătoare. În schimb, modelele locale înregistrează performanțe semnificativ mai reduse, scorurile acestora apropiindu-se cu dificultate de pragul de

0.5. Chiar dacă modelele locale și cele accesate prin API au fost testate pe eșantioane diferite de date, în faza finală se pot observa rezultate similare chiar și pe un set de date mai restrâns, format din doar 10 conversații.

Model	Prompt	Accuracy	F1 Macro	Latență mediană	N
RoGemma	V4	58%	0.481	16.66s	50
RoMistral	V4	56%	0.510	11.55s	50
GPT-4.1-mini	V2	73%	0.673	0.567s	100
Gemini-2.5-flash	V2	68%	0.615	6.955s	100
command-r7b	V4	70%	0.632	0.591s	100

Tabelul 5.2: Cel mai bun prompt per model pentru detectarea intenției

Scorurile F1 dar si acuratetea sunt puternic infleuntate de performantele modelelor pentru fiecare domeniu in parte de aceea este utila o analiza a fiecarui domeniu in parte. Asa cum se poate observa in tabelul 5.3, in mare parte din cazuri, dificultatea intampinata in domeniul serviciilor publice face ca acuratetea sa nu atinga un prag acceptabil. Domeniul serviciilor publice, nefiind ceva clar definit nici in cadrul proiectului dar nici pe baza informatiilor pe care modelele au fost antrenate, scoate in evidenta dificultatea de adaptare a modelelor lingvistice la domenii si situatii mai putin uzuale.

Modelele accesate prin API se descurcă foarte bine în domeniile medicină, retail, telecomunicații și banking, însă întâmpină dificultăți în clasificarea corectă a intențiilor din domeniul serviciilor publice. Pe de altă parte, modelele locale prezintă rezultate mai fluctuante între domenii. Deși și în cazul acestora domeniul serviciilor publice înregistrează performanțe mai reduse, diferențele față de domenii precum banking sau medicină nu sunt la fel de pronunțate. Se poate observa că, pentru RoGemma, domeniul serviciilor publice obține rezultate mai bune decât cel al telecomunicațiilor, un aspect neașteptat în raport cu tendința generală a celorlalte modele. O posibilă explicație este reprezentată de exemplele few-shot utilizate în prompturi, care par să fi facilitat identificarea intențiilor din domeniul serviciilor publice de către modelele locale, dar care, în același timp, ar fi putut introduce confuzii în cazul celorlalte domenii, mai familiare și mai bine reprezentate în datele de antrenare ale modelelor.

Model	Banking	Medicina	Retail	Telecom	Servicii publice
RoGemma V4-1	60%	70%	60%	40%	60%
RoMistral V4-1	60%	60%	30%	80%	50%
GPT-4.1-mini V2-1	70%	75%	80%	80%	60%
Gemini-2.5-flash V2-1	75%	80%	60%	75%	50%
command-r7b V4-1	70%	80%	75%	75%	50%

Tabelul 5.3: Acuratete per domeniu pentru cel mai bun prompt de detectare a intentiei

Erorile de clasificare nu sunt aleatorii, ci urmează tipare logice dictate de asemănarea semantică a etichetelor intențiilor. Cele mai frecvente confuzii apar între intenții cu o suprapunere ridicată a cuvintelor utilizate sau care par funcțional similare. Pentru a înțelege mai bine tiparele erorilor de clasificare, au fost generate matrici de confuzie pentru fiecare domeniu, utilizând modelul GPT în cea mai bună versiune a sa, V2_v1.

Banking (14/20 corecte, 70%) înregistrează cele mai multe erori pe clasa *tranzacție greșită*, confundată sistematic cu *problemă transfer* în câteva cazuri. Ambele implică o sumă transferată

incorect, diferența de bază dintre ele fiind că prima sugerează o eroare a clientului care a efectuat în mod eronat transferul, iar a doua indică o problemă tehnică a procesării acestuia. *Tranzacție suspectă* este confundată cu alte trei clase, sugerând o ambiguitate a situațiilor de fraudă. *Problemă sold* este confundată cu *card blocat* și *tranzacție suspectă*, deși aceste etichete sunt distincte pentru un operator uman. În urma unei tranzacții suspecte sau a unei probleme a soldului, din motive de siguranță, operatorul poate bloca cardul clientului; această acțiune, după cum s-a putut observa și în alte analize, îl determină pe model să considere blocarea cardului ca fiind intenția principală.

Medicină (15/20 corecte, 75%) prezintă confuzii concentrate în jurul gestionării programărilor. *Anulare programare* este clasificată ca *problemă programare*, deși ambele au la bază o programare problematică, acestea diferă prin acțiunea clientului; în primul caz, utilizatorul dorește să realizeze o anulare, în timp ce în al doilea caz problema apare din partea clinicii, iar clientul contactează unitatea medicală pentru clarificări. *Reclamație personal* și *consultație anulată* generează câte o eroare, ambele fiind încadrate în clasa *problemă programare*, ceea ce indică o suprapunere semantică în jurul interacțiunilor cu personalul medical, deși, în mod conceptual, aceste acțiuni sunt distincte.

Retail (16/20 corecte, 80%) este unul dintre cele mai bine clasificate domenii, alături de telecomunicații. Erorile sunt izolate: *reclamație produs* este confundată cu *problemă garanție* într-un singur caz, *comandă întârziată* generează confuzii între *problemă livrare* și *reclamație produs*, iar *retur produs* apare o dată clasificat ca *comandă greșită*. Toate aceste clasificări eronate nu se îndepărtează totuși de semantica conversației: o comandă întârziată este, într-o oarecare măsură, o problemă de livrare, iar returul unui produs poate fi catalogat, de asemenea, ca o comandă greșită. Deși oamenii pot face distincții mai fine între aceste concepte, un model de limbaj operează într-o manieră mai simplificată și tinde să le asocieze pe baza similarității semantice.

Servicii publice (12/20 corecte, 60%) este domeniul cu cele mai multe erori, confirmat și de analiza pe domenii. Confuziile principale apar între *reclamație serviciu* și *sesizare problemă*, precum și între *sesizare problemă* și *reclamație serviciu*, indicând o ambiguitate în ambele direcții între aceste două clase. *Contestație decizie* este confundată cu *dosar respins* într-un caz, iar *acte incomplete* cu *informații program* și *programare ghișeu*. Acest domeniu este cel mai vag dintre toate; structurile statului și serviciile oferite de acestea sunt de multe tipuri și nu pot fi niciodată limitate la un singur cadru bine delimitat. GPT probabil a avut și cele mai puține date de antrenare care să se apropie de acest domeniu fictiv, iar confuziile nu sunt neașteptate, având în vedere cât de neclar este doemniul și cât de vagi sunt definite aceste etichete.

Telecomunicații (16/20 corecte, 80%) prezintă erori concentrate pe intenții cu suprapunere tehnică. *Problemă roaming* este confundată cu *problemă internet* într-un caz, ceea ce este explicabil prin similaritatea dintre acestea, chiar dacă un operator uman poate face diferența între situațiile în care roamingul este activ sau relevant. *Factură greșită* și *problemă modificare abonament* generează câte o eroare, ultima fiind confundată cu *factură greșită* într-un caz în care clientul menționează costul noului abonament. Și de această dată, modelul pare să înțeleagă bine intențiile, însă similaritatea dintre etichete îl determină să aleagă o clasificare diferită.

Fără definiții clare pentru fiecare etichetă în parte, modelele vor continua să clasifice conversațiile în mod eronat sau, mai corect spus, după o logică diferită de cea umană. În multe cazuri, nici pentru o persoană alegerea intenției corecte nu este clară fără o delimitare clară a claselor și fără o instruire explicită privind criteriile de diferențiere dintre acestea.

5.2. Evaluarea modelelor — estimarea satisfacției

Sarcina de estimare a satisfacției prezintă o dificultate diferită față de cea a detectării intenției. Distribuția dezechilibrată a claselor din dataset (15 conversații pozitive, 25 neutre și 60 negative) favorizează modelele care clasifică preponderent clasa negativă, mascând performanța reală pe clasele minoritare. Din acest motiv, F1 Macro este metrica principală de evaluare, penalizând modelele care ignoră una sau mai multe clase.

Clasa neutră reprezintă dificultatea centrală a acestei sarcini pentru toate cele cinci modele testate. Conversațiile neutre, în care clientul acceptă soluția fără a exprima nicio emoție vizibilă, sunt adesea confundate fie cu cele pozitive, fie cu cele negative. Granița dintre neutru și negativ sau pozitiv poate fi ambiguă; un client care spune simplu *am înțeles, la revedere* poate fi interpretat atât ca acceptare calmă, cât și ca resemnare față de o situație nerezolvată.

Model	**Versiune**	**Accuracy**	**F1 Macro**	**Latență mediană (s)**
RoMistral	V1	60%	0.508	6.40
RoMistral	V2	40%	0.367	7.61
RoMistral	V3	30%	0.389	10.05
RoGemma	V1	60%	0.500	8.61
RoGemma	V2	80%	0.802	10.29
RoGemma	V3	70%	0.707	12.37
GPT-4.1-mini	V1	74%	0.727	0.476
GPT-4.1-mini	V2	70%	0.699	0.553
GPT-4.1-mini	V3	83%	0.816	0.663
Gemini-2.5-flash	V1	80%	0.717	6.016
Gemini-2.5-flash	V2	78%	0.629	6.421
Gemini-2.5-flash	V3	78%	0.642	8.183
command-r7b	V1	72%	0.593	0.398
command-r7b	V2	74%	0.779	0.417
command-r7b	V3	14%	0.124	0.591

Tabelul 5.4: Comparație performanță prompturi V1–V3 pentru estimarea satisfacției. Pentru modelele locale este prezentată varianta optimă per versiune. Latența este raportată ca mediană.

Tabelul 5.4 compară toate versiunile de prompt engineering pentru fiecare model în parte. În general, modelele API prezintă performanțe superioare față de modelele locale, atât în termeni de acuratețe, cât și de stabilitate între versiuni.

RoMistral a avut rezultate mai slabe odată cu evoluția versiunilor de prompt și adăugarea detaliilor. Dacă prima variantă atinge un nivel moderat de performanță, versiunile ulterioare prezintă scăderi semnificative, ajungând la clasificare corectă de 40% în a doua variantă și doar 30% în a treia. RoMistral a fost mult mai sensibil la schimbările din prompturi și nu beneficiat de pe urma detaliilor suplimentare. RoGemma are un comportament mai stabil și mai performant decât RoMistral. Modelul atinge cel mai bun rezultat în a doua versiune a promptingului (80% acuratețe și scor F1 de 0.802).

GPT este modelul cu cele mai bune rezultate generale în cadrul acestei sarcini. Versiunea a treia obține cea mai ridicată performanță globală, cu o acuratețe de 83% și F1 de 0.816, depășind atât variantele anterioare, cât și toate modelele locale. Se observă o îmbunătățire consistentă de la prima până la a treia versiune, ceea ce indică o capacitate bună de adaptare la rafinarea și creșterea complexității prompturilor.

Gemini prezintă performanțe relativ stabile între versiuni, cu valori apropiate ale acurateții (78-80%). Totuși, se observă o scădere a scorului F1, ceea ce sugerează o sensibilitate moderată la formularea promptului, fără variații extreme. Command prezintă cea mai mare instabilitate dintre toate modelele evaluate. Deși primele două variante ating performanțe relativ competitive, versiunea a treia înregistrează o scădere destul de mare.

Model	Prompt	Accuracy	F1 Macro	Latență mediană	N
RoGemma	V2-2	80%	0.802	10.29s	50
RoMistral	V1-1	60%	0.508	6.40s	50
GPT-4.1-mini	V3	83%	0.816	0.663s	100
Gemini-2.5-flash	V1	80%	0.717	6.016s	100
command-r7b	V2	74%	0.779	0.417s	100

Tabelul 5.5: Cel mai bun prompt per model pentru estimarea satisfacției

Tabelul 5.5 sintetizează cel mai bun prompt per model. GPT conduce cu F1 Macro de 0.816 și este singurul model care menține un echilibru acceptabil pe toate cele trei clase. Gemini obține cel mai bun rezultat cu promptul cel mai simplu; versiunile mai complexe nu aduc îmbunătățiri, iar pentru clasa neutră înregistrează o scădere progresiva. Command cu a doua variantă de prompt identifică toate conversațiile neutre corect, dar recunoaște clasa negativa doar în 60% din cazuri.

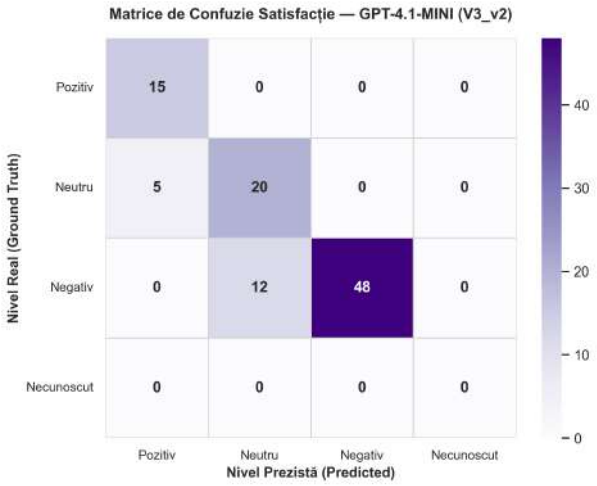

Figura 5.1: Matrice de confuzie pentru GPT-4.1-mini V3 — estimarea satisfacției

GPT depășește RoGemma pe toate clasele, însă RoGemma cu a doua versiune de prompting obține un F1 Macro de 0.802, superior Gemini (0.717) și de Command (0.779), demonstrând că un model local cu un prompt bine structurat poate concura cu modelele API pe această sarcină. Diferența de baza rămâne latența: sub o secundă pentru GPT și Command, față de 10-12 secunde pentru modelele locale. Spre deosebire de sarcina de detectare a intenției, unde decalajul dintre modele era mai pronunțat, estimarea satisfacției pare o sarcină unde echilibrarea promptului contează mai mult decât capacitatea modelului.

Ca și în cazul intențiilor, este util de analizat erorile pe care modelele le fac; de aceea, și de această dată, vor fi vizualizate rezultatele date de GPT în cea mai bună variantă a sa (versiunea a treia de prompting) prin intermediul unei matrice de confuzie. Având în vedere

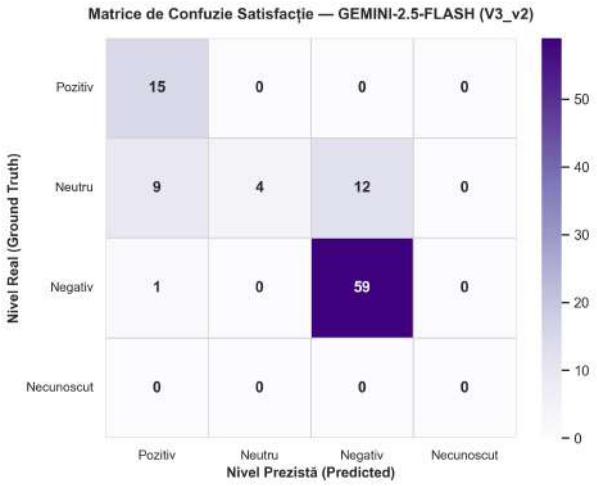

Figura 5.2: Matrice de confuzie pentru Gemini-2.5-flesh V3 — estimarea satisfacției

numărul redus de clase din această sarcină, se poate realiza ușor și rapid o comparație a acestor rezultate cu cele ale unui model local. Pentru aceasta a fost generată o matrice de confuzie și pentru RoGemma în varianta a doua de prompting, acesta având cele mai bune rezultate în cadrul modelelor locale.

GPT demonstrează cel mai echilibrat comportament dintre toate modelele testate. Clasifică corect toate cele 15 conversații pozitive și 20 din 25 neutre, cu 5 confuzii spre pozitiv și niciuna spre negativ. Principala sursă de eroare rămâne granița dintre neutru și negativ: 12 din 60 de conversații negative sunt clasificate ca neutre, ceea ce reflectă dificultatea de a distinge între un client supărat sau resemnat și unul cu adevărat neutru.

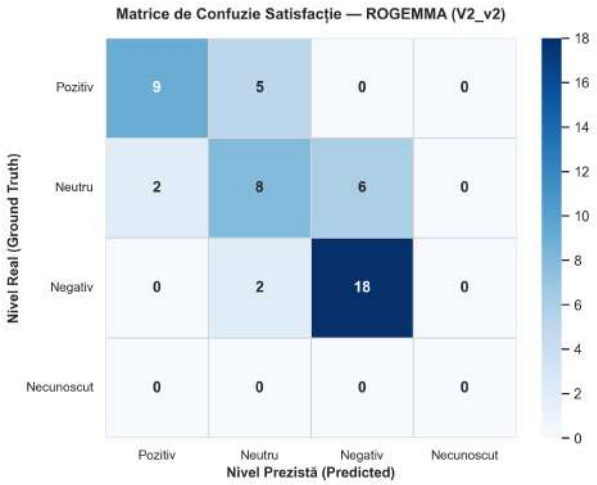

Figura 5.3: Matrice de confuzie pentru RoGemma V2-2 — estimarea satisfacției

Gemini, pe de altă parte, cum se poate observa în Figura 5.2, clasifică 59 din cele 60 de conversații negative corect, dar prezintă probleme în determinarea clasei neutre; Gemini clasifică greșit 21 din 25 de conversații neutre, 9 ca pozitive și 12 ca negative, în timp ce GPT ratează doar 5. Deși Gemini atinge o acuratețe globală similară (78% față de 83%), aceasta provine aproape exclusiv din clasificarea corectă a negativului (59/60). F1 Macro de 0.642 față de 0.816 al GPT subliniază această diferență.

Totuși, în situații reale, pentru o companie, clasificarea eronată a unei conversații negative este mai costisitoare decât clasificarea eronată a uneia neutre, deci Gemini ar fi mai util pentru un caz aplicat de utilizare a LLM-urilor în sarcini de sentiment analysis. Cum, totuși, ne dorim ca acest experiment să utilizeze cel mai echilibrat model, GPT reprezintă varianta mai bună.

RoGemma este singurul model local care reușește să recunoască clasa neutră într-o proporție destul de bună: 8 din 10 conversații neutre din subset sunt clasificate corect. Clasa pozitivă este recunoscută în 9 din 14 cazuri. Negativul este clasificat corect în toate cele 18 cazuri, fără nicio eroare. Rezultatele lui RoGemma sunt comparabile cu cele ale lui GPT, dar de data aceasta clasificând corect clasa negativă în proporții complete, ceea ce îl face o opțiune bună pentru situațiile în care robotul telefonic necesită rularea locală a LLM-urilor. Singura problemă rămâne în continuare latența.

5.3. Evaluarea modelelor - generarea rezumatului

Sarcina de generare a rezumatului introduce o metodă suplimentară de evaluare față de celelalte două sarcini: pe lângă calitatea semantică a textului generat, măsurată prin ROUGE și BERTScore, este evaluată și respectarea limitelor de lungime alese pe baza nivelului de satisfacție al clientului (rezumat scurt pentru conversații pozitive, mediu pentru cele neutre și lung pentru cele negative).

O observație importantă este distincția dintre scorul BERT și scorurile ROUGE. Toate cele cinci modele obțin valori ridicate de BERTScore F1, înscrise în intervalul 0.65-0.76, indicând o înțelegere semantică bună a conversației și o aliniere ridicată cu informația din rezumatul de referință. Scorurile ROUGE-2, în schimb, rămân scăzute (între 0.07 și 0.18), ceea ce sugerează că modelele formulează rezumatele diferit față de referință, chiar dacă transmit în esență aceeași informație.

Această distincție este importantă pentru interpretarea metricilor: un model cu ROUGE scăzut, dar BERTScore ridicat, nu generează rezumate slabe calitativ, ci rezumate formulate diferit de adnotările stabilite. Adnotările fiind generate inițial de un alt model lingvistic (Claude), era de așteptat să existe o variație naturală între stilurile de exprimare și nivelul de creativitate al modelelor.

Model (Versiune)	**ROUGE-1**	**ROUGE-L**	**BERT F1**	**În limite**	**Latență (s)**
RoMistral V1	0.329	0.234	0.730	27/50	35.50
RoMistral V2	0.357	0.253	0.737	32/50	88.38
RoMistral V3	0.233	0.164	0.695	31/50	31.31
RoGemma V1	0.331	0.233	0.722	20/50	37.02
RoGemma V2	0.317	0.223	0.722	19/50	41.20
RoGemma V3	0.204	0.139	0.653	20/50	102.35
GPT-4.1-mini V1	0.390	0.270	0.747	75/100	1.654
GPT-4.1-mini V2	0.381	0.275	0.752	98/100	1.865
GPT-4.1-mini V3	0.401	0.292	0.758	100/100	1.615
Gemini-2.5-flash V1	0.389	0.276	0.748	12/100	6.457
Gemini-2.5-flash V2	0.400	0.290	0.759	62/100	5.519
Gemini-2.5-flash V3	0.416	0.299	0.764	67/100	4.534
command-r7b V1	0.388	0.274	0.745	67/100	1.707
command-r7b V2	0.351	0.247	0.735	89/100	1.621
command-r7b V3	0.381	0.275	0.754	93/100	1.542

Tabelul 5.6: Comparație performanță prompturi V1–V3 pentru generarea rezumatului. Latența este raportată ca mediană.

Atunci când vine vorba de arhitecturile locale, modelul RoMistral obține cele mai bune rezultate pe a doua configurație de prompting, care include toate structurile de prompt engineering, atingând un ROUGE-1 de 0.357 și un ROUGE-L de 0.253. Versiunea a treia, bazată pe exemple few-shot, a produs rezultate semnificativ mai slabe: modelul a învățat din exemple să genereze în mare parte rezumate scurte și a aplicat greșit această lungime comprimată pentru toate tipurile de conversatii. Astfel scorul ROUGE-1 a scăzut la 0.233 și a avut o performanță foarte slabă pe subsetul conversțiilor negative, așa cum se poate observa în tabelul 5.6; acestea necesitau rezumate lungi (între 60 și 100 de cuvinte), iar răspunsurile generate de model (de doar 20-40 de cuvinte) au omis unele informații.

RoGemma a ignorat aproape complet constrângerile de lungime pe toate cele trei versiuni de prompturi, generând constant între 75 și 90 de cuvinte, indiferent de dimensiunea solicitată. Acest comportament se observă printr-o rată scăzută de încadrare în limite (de exemplu, doar 19/50 de cazuri pentru versiunea a doua, echivalentul a 38%).

Model	**Prompt**	**ROUGE-1**	**ROUGE-L**	**BERT F1**	**În limite**	**Latență**
RoGemma	V2	0.317	0.223	0.722	38%	41.20s
RoMistral	V2	0.357	0.253	0.737	64%	88.38s
GPT-4.1-mini	V3	0.401	0.292	0.758	100%	1.615s
Gemini-2.5-flash	V3	0.416	0.299	0.764	67%	4.534s
command-r7b	V3	0.381	0.275	0.754	93%	1.542s

Tabelul 5.7: Cel mai bun prompt per model pentru generarea rezumatului

Așa cum se poate observa în Tabelul 5.7, toate cele trei modele comerciale apelate prin API înregistrează cele mai bune performanțe pe versiunea a treia de prompt, care combină definirea rolului, structura strictă a rezumatelor și câte un exemplu few-shot adaptat fiecărei limitări de lungime.

Dintre acestea, Gemini obține cele mai ridicate scoruri semantice, cu un ROUGE-1 de 0.416 și un BERTScore F1 de 0.764. Totuși, performanța sa este afectată de nerespectarea constrângerilor de lungime, doar 67% dintre rezumate încadrându-se în limitele stabilite. Modelul tinde să genereze texte mai lungi și mai descriptive decât cele solicitate, în special pentru conversațiile pozitive.

GPT obține scoruri semantice foarte apropiate de cele ale lui Gemini, însă este singurul model care respectă limitele de lungime în toate cele 100 de conversații evaluate. Acest echilibru între fidelitatea semantică și respectarea instrucțiunilor îl face cea mai stabilă soluție pentru această sarcină. Command prezintă, la rândul său, rezultate competitive. Deși scorurile ROUGE și BERTScore sunt ușor inferioare celor obținute de GPT și Gemini, acesta respectă constrângerile de lungime în 93% dintre cazuri și oferă cea mai redusă latență dintre toate modelele evaluate.

Comparația directă dintre modelele locale și cele accesate prin API evidențiază diferențe importante atât în ceea ce privește calitatea rezumatelor generate, cât și capacitatea de adaptare la cerințele promptului. Modelele API utilizează mai eficient exemplele few-shot și își ajustează lungimea rezumatelor în funcție de contextul conversației, în timp ce modelele locale întâmpină dificultăți atunci când complexitatea promptului crește. Acest aspect este vizibil mai ales în cazul RoGemma, unde introducerea exemplelor few-shot a condus simultan la scăderea scorurilor semantice și la creșterea latenței.

Diferențele sunt evidente și din perspectiva timpului de rulare. Modelele locale necesită între 35 și 102 secunde pentru generarea unui rezumat, în timp ce GPT și Command răspund în

aproximativ una-două secunde. Chiar dacă Gemini produce rezumate de calitate comparabilă, latența sa rămâne semnificativ mai ridicată. Dintre toate modelele evaluate, GPT oferă cel mai bun compromis între calitatea semantică a rezumatelor, respectarea constrângerilor de lungime și timpul de răspuns.

Crescând numărul de tokeni, este probabil să se producă o schimbare în clasamentul modelelor, permițând atât lui GPT, cât și lui Command să genereze răspunsuri mai lungi. Totuși, această limitare este utilă în a ne asigura că rezumatele nu depășesc limitele impuse. Având în vedere mediul controlat al conversațiilor și limitele de lungime, acest comportament este acceptat; într-o situație reală, unde conversațiile pot fi mult mai complexe și mai lungi decât cele generate în acest proiect, tehnica de generare a rezumatului poate fi regândită și se poate genera o listă cu punctele cheie, extragerea unor cuvinte sau replici importante etc.

Același comportament poate fi valabil și pentru modelele locale, care se opresc brusc în continuare atunci când generează rezumatele din cauza acestor limitări. În unele situații, deși rezumatele se încadrează foarte bine în limite, sumarizarea este incompletă, deși mai era loc de adăugare a unor informații. Și de această dată, informațiile rezumate sunt prezente, dar nivelul de creativitate și limitările de tokeni produc rezultate nedorite.

Utilizarea unei liste cu un număr predefinit de puncte, care să ghideze modelele spre a completa mai mult decât a genera, s-ar dovedi utilă. Astfel, nu ar mai trebui luat în calcul nivelul de creativitate al modelului și tendințele lor de a genera răspunsuri mult mai lungi. Utilizarea unui șablon pentru generarea unei liste cu ideile principale ar face nu doar evaluarea manuală mult mai simplă, dar și informațiile redundante ar fi eliminate.

5.4. Comparație modele API vs. modele locale

Următorul pas în analiza evaluării modelelor îl reprezintă testarea acestora prin intermediul unui prompt complet, care să conțină toate cele trei sarcini, asemănător unui caz real. Având în vedere latența ridicată cu care rulează modelele locale, dar și capacitățile dispozitivului în care acestea sunt stocate și rulate, pentru evaluarea lor se va utiliza un subset de 10 conversații, asemănător testelor realizate în capitolul de prompt engineering. Modelele apelate prin API au metrici mai bune când vine vorba de latență și, deși prezintă costuri mai mari comparativ cu rularea modelelor locale, sunt mult mai eficiente. Astfel, acestea vor fi rulate atât pe un subset de 10 conversații, cât și pe întregul set de date. Rularea pe un subset a modelelor din urmă a fost aleasă pentru a facilita o analiză corectă a tuturor celor cinci modele, pe același set de date, de aceeași dimensiune. Rezultatele sunt prezentate în Tabelul 5.8.

Model	Acc. Int.	F1 Sat.	ROUGE-L	BERT F1	In limite	Latenta
RoGemma	60%	0.372	0.262	0.736	30%	71.0s
RoMistral	50%	0.508	0.307	0.746	40%	54.4s
GPT-4.1-mini	90%	0.806	0.339	0.772	90%	3.6s
Gemini-2.5-flash	90%	0.669	0.310	0.766	70%	15.0s
command-r7b	80%	0.800	0.318	0.769	80%	3.0s

Tabelul 5.8: Comparatie modele locale vs. API pe subset comun de 10 conversatii

Pe subsetul comun de 10 conversații, se pot observa diferențele de performanță dintre modelele alese. La detectarea intenției, GPT și Gemini ating 90% acuratețe, iar Comman 80%, față de 50-60% pentru modelele locale. Această diferență de aproape 40% subliniază capacitatea

superioară a modelelor API de a gestiona ambiguitatea semantică a etichetelor și de a selecta corect dintr-o listă de intenții specifice domeniului.

La estimarea satisfacției, diferența este și mai pronunțată la nivelul scorului F1: GPT și Command ating aproximativ 0.8, aproape dublu față de RoGemma. RoMistral obține un F1 de 0.508 datorită performanței bune pe clasele pozitivă și negativă, însă clasa neutră rămâne la fel de problematică. Modelele apelate prin API gestionează mult mai bine echilibrul dintre cele trei clase. La generarea rezumatului, avantajul modelelor API este vizibil atât în scorurile ROUGE-L, cât și în respectarea limitelor de lungime; GPT respectă aceste limite în aproape 90% din cazuri, față de 30-40% pentru modelele locale.

Diferența cea mai semnificativă din perspectiva unui sistem real rămâne însă latența. Modelele locale necesită aproximativ un minut pentru a procesa o singură conversație prin toate cele trei sarcini, în timp ce GPT și Command răspund în 3-4 secunde. Acest aspect face modelele locale incompatibile cu procesarea în timp real, însă ele pot fi considerate pentru scenarii offline sau în absența accesului la API-uri.

Model	**F1 Int.**	**F1 Sat.**	**ROUGE-L**	**BERT F1**	**În limite**	**Latență**
GPT-4.1-mini	0.706	0.756	0.288	0.764	91%	4.3s
Gemini-2.5-flash	0.638	0.742	0.295	0.760	66%	20.8s
command-r7b	0.566	0.745	0.279	0.756	76%	2.9s

Tabelul 5.9: Performanța modelelor API pe tot setul de 100 de conversații — pipeline complet

Tabelul 5.9 prezintă performanțele celor trei modele API pe întregul set de conversații, rulând pipeline-ul complet cu prompturile câștigătoare pentru fiecare sarcină. Având în vedere latența ridicată a modelelor locale și resursele computaționale necesare pentru procesarea întregului set de date, această etapă a evaluării a fost realizată exclusiv pentru modelele apelate prin API.

GPT obține cele mai echilibrate performanțe, având 76% acuratețe la detectarea intenției, 77% la estimarea satisfacției, un scor F1 Macro de 0.756 și 91% din rezumate încadrate în limitele de lungime corecte. Distribuția pe domenii pentru intenție este relativ uniformă, iar în cazul satisfacției, toate cele trei clase sunt bine acoperite, confirmând că GPT gestionează echilibrat clasele minoritare ale datasetului.

Command se remarcă prin cea mai mică latență totală și prin cel mai bun echilibru la estimarea satisfacției pe clasele pozitivă și neutră, 87% și respectiv 88%. Clasa negativă, care reprezintă 60% din dataset, este însă acoperită într-o proporție mai mică, sugerând că modelul tinde să subestimeze frustrările implicite ale clienților. La detectarea intenției obține 62%, mai slab decât GPT și Gemini.

Gemini prezintă o acuratețe ridicată când vine vorba de detectarea satisfacției (81%), dar un F1 Macro de doar 0.556, indicând un bias puternic către clasa negativă dominantă din dataset, recunoscută în 90% din cazuri, în timp ce clasa neutră atinge doar 48%. La rezumat, Gemini obține cele mai bune scoruri ROUGE-L (0.413) și BERTScore (0.760), însă cu doar 66% din rezumate încadrate în limitele de lungime, datorită tendinței de a genera texte mai lungi decât cele solicitate. Latența sa totală de 20.8 secunde per conversație este semnificativ mai mare decât a celorlalte două modele API, din cauza duratei ridicate de așteptare până la primul token.

Rezultatele finale sunt puternic influențate și de volumul mare de prompturi care au fost trimise unul după altul și de dimensiunea fiecărei conversații. Scăderea acurateții și a celorlalte scoruri este, așadar, o consecință a dimensiunii setului de date. Într-un caz real, robotul telefonic

va apela aceste prompturi la intervale inegale de timp, unele dintre ele fiind foarte apropiate, iar altele la intervale mari de timp.

În ansamblu, GPT oferă cel mai bun raport între performanța globală și stabilitatea rezultatelor, urmat de Command pentru scenarii în care latența este prioritară. Gemini rămâne competitiv pe sarcina de rezumat, dar prezintă limitări importante în detectarea satisfacției și în ceea ce privește latența.

5.4.1. Testarea modelelor cu înregistrări

Dacă în subcapitolul anterior evaluarea modelelor a fost realizată utilizând conversații sub formă de text, în acest subcapitol, odată cu implementarea modulelor de STT (speech-to-text) și TTS (text-to-speech), va fi testat un subset de conversații înregistrate, care vor fi trecute prin sistemul de transcriere a convorbirii și ulterior procesate de către modelele lingvistice.

Pentru a simula fluxul de date pe care robotul telefonic trebuie să îl proceseze, a fost implementată o componentă de procesare a acestui subset prin transformarea lui în dialoguri înregistrate. Utilizând biblioteca *edge-tts*, care oferă acces la mai multe modele neurale de generare vocală dezvoltate de Microsoft, a fost configurat un sistem multi-speaker care să pună la dispoziție două voci diferite, una pentru client iar cealaltă pentru operator; astfel, operatorul a fost reprezentat de vocea Alina, iar clientul de vocea Emil.

Fișierele audio generate în format MP3 au fost convertite la WAV 16 kHz mono cu ajutorul utilitarului *ffmpeg*, pentru a fi compatibile cu formatul cerut de API-ul Zevo STT. Transcrierea funcționează prin trimiterea audio în chunk-uri de 4096 bytes în timp real, simulând ritmul real de redare printr-un sleep calculat pentru fiecare chunk.

Serverul Zevo returnează răspunsuri partiale pe parcursul vorbirii, și u rezultat complet la finalul unui fragment. Implementarea acumulează toate fragmentele finalizate, pentru a asigura o transcriere completă a conversației. Textul rezultat este continuu, fără punctuație sau delimitatori între replici. Deși pentru un operator uman acest aspect poate fi problematic, nu ar trebui să reprezinte o problemă pentru model.

Model	**Acc. Intenție**	**F1 Satisfacție**	**BERTScore**	**În limite**	**Latență**
GPT-4.1-mini	80%	0.756	0.764	90%	4.4s
Gemini-2.5-flash	80%	0.500	0.748	60%	15.0s
command-r7b	80%	0.452	0.741	50%	2.5s
RoMistral	50%	0.000	0.710	20%	52.2s
RoGemma	—	—	—	—	—

Tabelul 5.10: Rezultate pipeline audio pe subsetul de 10 conversații înregistrate

Tabelul 5.10 surprinde rezultatele celor cinci modele în urma testării pe un subset de zece conversații audio. Astfel, modelele au procesat aceleași conversații testate și în partea de prompt engineering, doar că de această dată transcrise de sistemul STT. Transcrierea automată poate produce erori de recunoaștere, în special în cazul numelor proprii sau al termenilor din domenii specifice. Aceste erori se propagă mai departe în pipeline și pot influența rezultatele finale.

Toate cele trei modele apelate prin API au înregistrat rezultate foarte bune în detectarea intenției, mai precis o acuratețe de 80%. Totuși, analizând în detaliu clasificările realizate de modelele apelate prin API, se poate observa că, deși acestea au identificat foarte bine intenția principală, au adăugat în foarte multe cazuri o intenție suplimentară.

Pentru aceleași conversații, fiecare model consideră că mai este prezentă o intenție. Spre exemplu, într-o conversație din domeniul retail, toate cele trei modele au clasificat, pe lângă intenția corectă *comandă greșită*, și o altă etichetă, *retur produs* sau *problemă livrare*. Returnarea produsului vine ca o urmare a comenzii greșite; totuși, din punct de vedere semantic, etichetele nu au fost plasate greșit. S-ar putea considera util ca o conversație să conțină mai multe etichete într-un caz real.

Cele două conversații clasificate ca fiind eronate în cazul GPT și Gemini provin din domeniul serviciilor publice, unde modelele au identificat intenția, dar au clasat-o ca fiind secundară. Această clasificare a multor conversații ca având intenții multiple se poate rezolva în faza de implementare a pipeline-ului, având în vedere că intenția este extrasă în faza inițială a conversației și pe baza acesteia sunt alese direcțiile conversației.

Satisfacția scade totuși, mai ales pentru Gemini și Command, care ajung la scoruri F1 Macro mult mai slabe decât în faza de testare a conversațiilor text: Command are un scor de 0.452, iar Gemini de 0.500, reușind să clasifice corect în jur de jumătate dintre conversațiile din subset. Lipsa semnelor de punctuație și a structurii dialogului a determinat o scădere a performanței acestora. Totuși, GPT și-a menținut scorul destul de bine, ajungând la 0.756, foarte aproape de performanțele sale din evaluările anterioare.

Scorul BERT rămâne ridicat pentru toate modelele, demonstrând și de această dată că nu au probleme în generarea unui rezumat și în înțelegerea semantică a conversației. În multe rezumate s-a remarcat totuși un limbaj destul de elaborat, iar multe dintre acestea ofereau informații inutile, precum numele operatorului și al companiei — detalii pe care operatorul uman le oferă la începutul conversației. Acest comportament nu este neașteptat; testarea prompturilor a prezentat rezultate asemănătoare, mai ales pentru prompturile cu rezultate mai slabe, unde modelele nu reușeau să fie suficient de concise.

Latența totală a pipeline-ului este influențată în principal de etapa de transcriere a conversației, având o mediană de aproximativ 300 de secunde pentru transcriere, comparativ cu doar câteva secunde pentru analiza realizată de modelele lingvistice. Această durată nu este neașteptată, având în vedere că fișierele au aproximativ cinci minute fiecare. Într-un scenariu real, sistemul STT procesează fluxul audio în timp real, pe măsură ce clientul vorbește. Dintre toate modelele API, Gemini a prezentat și de această dată cea mai ridicată latență, ajungând la aproximativ 15 secunde per conversație, comparativ cu GPT și Command, care înregistrează 4.4 secunde, respectiv 2.5 secunde.

Pe de altă parte, modelele locale prezintă dificultăți mult mai semnificative în acest scenariu de testare. RoMistral identifică corect intenția în 50% din cazuri, un rezultat acceptabil în sine, dar nu reușește să estimeze satisfacția în nicio conversație — toate răspunsurile sunt marcate *necunoscut*. Textul continuu produs de STT nu oferă indicii suficiente pentru a distinge tonul și emoțiile prezente în conversație. În schimb, deși modelul nu respectă limitele de lungime impuse, rezumatele sunt complete și clar structurate, și deși oferă informații redundante, înțeleg contextul discuției.

RoGemma nu a putut fi evaluat în acest scenariu. Transcrierea completă a unei conversații de câteva minute depășește fereastra de context disponibilă a modelului, declanșând bucle de generare care nu se finalizează într-un interval de timp rezonabil.

Din perspectiva unei implementări reale, aceste rezultate sugerează că performanța sistemului este mai puternic limitată de calitatea componentelor de preprocesare audio decât de modelul lingvistic utilizat, ceea ce evidențiază importanța optimizării pipeline-ului end-to-end.

5.4.2. Concluzii cu privire la evaluarea modelelor

Analiza comparativă a celor cinci modele pe întregul pipeline evidențiază câteva concluzii generale care depășesc performanțele individuale ale fiecărei sarcini.

Diferența dintre modelele API și cele locale este cea mai mare la detectarea intențiilor (30–40 de procente), moderată la estimarea satisfacției și mai redusă la generarea rezumatului, unde scorul BERTScore al modelelor locale rămâne competitiv. Acest lucru sugerează că sarcinile de clasificare cu vocabular controlat și spațiu larg de etichete beneficiază cel mai mult de capacitatea modelelor mari, în timp ce sarcinile de sumarizare sunt mai puțin sensibile la dimensiunea modelului.

GPT a demonstrat că un model care menține rezultate constante între variantele de prompt este mai de încredere decât un model care atinge performanțe ridicate doar în configurații punctuale, cum este cazul Command pe V3 la satisfacție sau RoGemma pe V2. Într-un sistem real, această stabilitate este importantă, deoarece prompturile sunt frecvent ajustate și iterațiile sunt inevitabile.

Analiza matricilor de confuzie arată că, deși Gemini înregistrează un număr mai mic de erori totale, acestea sunt concentrate în special pe clasa neutră, care are un impact redus asupra deciziilor operaționale. GPT, în schimb, distribuie erorile mai uniform, inclusiv către clasa negativă, care are cel mai mare potențial de impact. Prin urmare, alegerea modelului optim depinde de costul relativ al fiecărui tip de eroare în contextul aplicației.

Scăderile de performanță la sarcina de satisfacție pentru Gemini și Command în scenariul audio nu sunt cauzate de limitări ale modelelor lingvistice, ci de lipsa punctuației și a structurii conversaționale din transcrierea STT. Îmbunătățirile aduse modelului LLM au un impact mai mic decât optimizarea componentei de transcriere, ceea ce face ca STT-ul să fie un punct foarte important în pipeline.

Latența exclude utilizarea modelelor locale în scenarii de timp real, indiferent de performanța lor pe date text. Un timp de ordinul minutelor per conversație pentru modelele locale, comparativ cu câteva secunde pentru GPT sau Command, reprezintă o diferență care nu poate fi compensată prin ajustări de prompt sau optimizări hardware în condițiile resurselor utilizate.

6. IMPLEMENTAREA PIPELINE-ULUI

Arhitectura software propusă pentru automatizarea și analiza fluxurilor vocale din cadrul acestei lucrări este proiectată sub forma unui pipeline modular, ghidat de evenimente în timp real. Structura de ansamblu a sistemului, ilustrată în Figura 6.1, evidențiază componentele robotului telefonic.

Acesta a fost proiectat modular, astfel încât fiecare componentă să poată fi înlocuită independent fără modificări majore asupra întregului sistem. De exemplu, serviciul de Speech-to-Text poate fi înlocuit cu un alt furnizor, iar modelul lingvistic poate fi schimbat cu o versiune mai performantă sau specializată pentru un anumit domeniu. Astfel, testarea tehnologiilor poate fi realizată mult mai ușor, iar sistemul poate fi adaptat mult mai simplu la diferite cerințe.

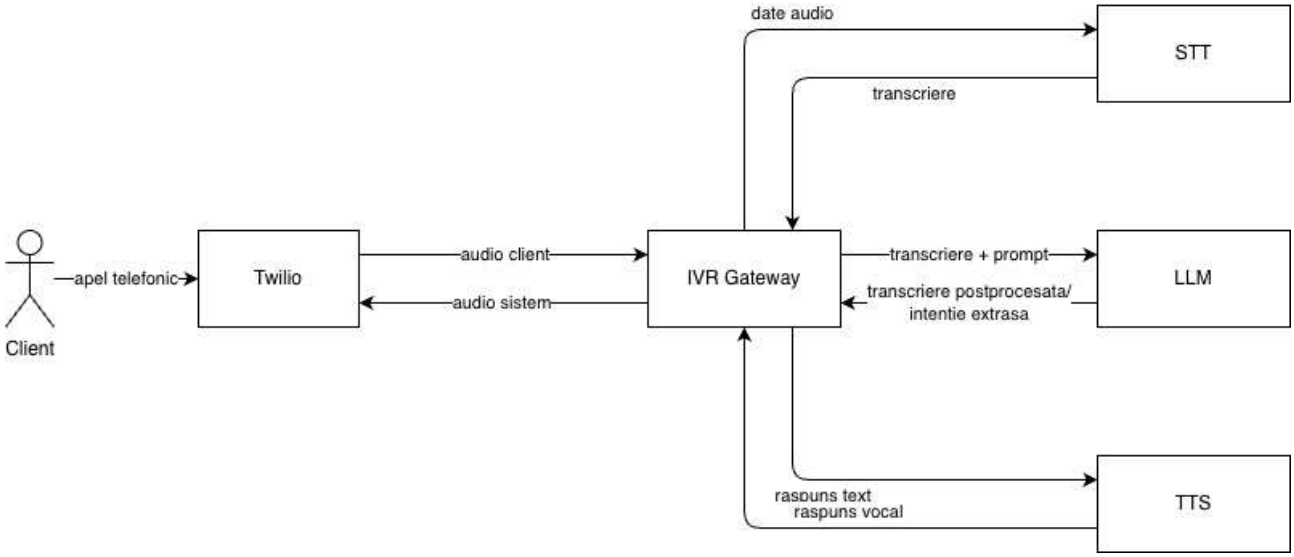

Figura 6.1: Sturctura pipeline-ului

Clientul apelează telefonic numărul obținut prin intermediul Twilio, o platformă globală de comunicații în cloud de tip Platform as a Service(CPaaS) care permite companiilor să integreze funcții de interacțiune vocală direct în aplicațiile lor prin intermediul REST API și WebSocket.

În loc să construiască o infrastructură telefonică de la zero, companiile pot utiliza Twilio pentru automatizarea comunicării prin SMS-uri, apeluri telefonice și e-mailuri. În cadrul sistemului propus, apelul este preluat de un server Quart, care returnează un răspuns ce instruiește Twilio să inițieze un flux WebSocket bidirecțional către server. Prin acest canal, Twilio transmite fluxul audio al clientului la o frecvență de 8 kHz, iar serverul trimite înapoi răspunsurile vocale generate de asistent, codificate în același format.

Acest apel este procesat prin intermediul unui Gateway IVR, care face legătura dintre celelalte sisteme ale robotului. Sistemul de Speech-to-Text transcrie în timp real replicile

clientului. Robotul telefonic este instruit ca, atunci când este apelat, să adreseze mai întâi câteva întrebări de bază cu privire la datele clientului, pentru a putea identifica contul acestuia și pentru a facilita identificarea mai rapidă a problemei. Ulterior, robotul îl întreabă pe client cu ce îl poate ajuta, moment în care acesta își exprimă problema pentru care a sunat. Replica robotului este transformată în semnal audio prin intermediul sistemului Text-to-Speech. După această etapă, LLM-ul intervine și extrage intenția utilizatorului, iar pe baza acesteia generează răspunsuri, soluții și replici adaptate contextului conversației.

Având în vedere diversitatea cazurilor pentru care o persoană poate suna într-un call-center, un set predefinit de replici nu ar fi util, iar un șablon de întrebări sau direcții conversaționale ar fi, de asemenea, mult prea restrictiv. Astfel, GPT, modelul final ales în acest experiment, va genera replici pe baza răspunsurilor clientului și a contextului conversației. Chiar dacă generarea următoarei replici poate dura puțin mai mult decât utilizarea unor răspunsuri predefinite, având în vedere valoarea redusă a TTFT-ului pentru GPT, timpul de așteptare rămâne acceptabil. Acest schimb de dialoguri continuă până în momentul în care clientul afirmă că dorește să încheie conversația; până atunci, LLM-ul va continua să genereze răspunsuri și să justifice acțiunile sau soluțiile pe care le-a propus.

Separarea componentelor STT, LLM și TTS permite procesarea fiecărei etape în mod independent și monitorizarea latenței introduse de fiecare modul. Experiența utilizatorului într-un sistem telefonic este influențată direct de timpul de răspuns al sistemului. Experimentele realizate în cadrul acestei lucrări au evidențiat faptul că etapa de transcriere introduce cea mai mare parte a întârzierii totale, în timp ce analiza efectuată de modelul lingvistic necesită doar câteva secunde. Prin urmare, performanța întregului sistem depinde într-o măsură mai mare de eficiența componentei de recunoaștere vocală decât de timpul de inferență al modelului lingvistic.

6.1. Modulul de procesare vocală (STT și TTS)

Pentru realizarea interfeței vocale a robotului telefonic, arhitectura sistemului integrează serviciile specializate pentru limba română oferite de platforma *Zevo-Tech*. Integrarea cuprinde două componente fundamentale: un modul de transcriere în timp real de tip Speech-to-Text și un modul de sinteză vocală de tip Text-to-Speech. Ambele componente sunt integrate în aplicație prin intermediul unor conexiuni WebSocket securizate, asigurând transmiterea și recepția pachetelor de date.

Serviciul de Speech-to-Text primește fluxul audio transmis de Twilio și generează în timp real transcrierea replicilor, care este ulterior utilizată de modelul lingvistic. Modelele lingvistice operează asupra textului, așadar este necesară o componentă care să transforme semnalul audio înainte ca acesta să ajungă la model. Calitatea transcrierii influențează performanța etapelor ulterioare, erorile de recunoaștere putând afecta detectarea intenției, estimarea satisfacției sau generarea rezumatului.

Serviciul Zevo returnează rezultate parțiale pe parcursul conversației și actualizează continuu transcrierea pe măsură ce utilizatorul vorbește. Acest mecanism permite detectarea rapidă a finalului de enunț și contribuie la o interacțiune mai naturală și mai cursivă. În momentul în care este detectată o pauză de două secunde în vorbirea utilizatorului, segmentul audio este transmis către sistemul de transcriere pentru procesare. Aceasta este realizată la nivelul serverului printr-un mecanism de Voice Activity Detection (VAD), care identifică perioadele de inactivitate vocală și declanșează transmiterea segmentului audio către sistemul de transcriere. Această durată poate fi ajustată în funcție de scenariul de utilizare, întrucât două secunde pot fi insuficiente sau excesive în contexte reale.

Detecția pauzei în vorbire este mai eficientă decât utilizarea unei ferestre de timp predefinite pentru colectarea replicii. Fereastra poate fi problematică, fiind fie prea lungă pentru replici scurte, fie prea scurtă pentru explicații detaliate oferite de utilizator.

Serviciul de Text-to-Speech transformă textul generat de modelul lingvistic într-un semnal audio care este redat utilizatorului prin intermediul apelului telefonic. Rolul acestei componente este de a permite interacțiunea naturală dintre utilizator și sistem, fără necesitatea afișării răspunsurilor în format text. Calitatea vocii sintetizate și timpul de generare influențează direct nivelul de satisfacție al utilizatorului și gradul de naturalețe al conversației. În cadrul implementării a fost utilizată vocea *Maria*, adaptată pentru limba română, care este mai calitativă decât celelalte voci și mai apropiată de vorbirea umană. Ca orice sistem de sinteză vocală, pot apărea erori de pronunție pentru anumite cuvinte, iar replicile mai lungi pot genera uneori artefacte de vorbire. Totuși, aceste limitări nu afectează semnificativ înțelegerea răspunsurilor acestuia.

6.2. Modulul de generare a răspunsurilor (LLM)

Componenta centrală de procesare a replicilor utilizatorului și de generare a soluțiilor folosește modelul GPT-4.1-mini, accesat prin API. Conversația este organizată în patru etape care au ca scop determinarea unor informații de bază, înainte de înțelegerea intenției consumatorului:

- **Identificarea clientului** — înainte de a afla motivul apelului de la client, robotul colectează datele de identificare specifice domeniului prin replici complet predefinite, fără a apela LLM-ul. Pentru domeniul bancar, de exemplu, sunt solicitate numele complet și ultimele patru cifre ale cardului; pentru medicină, numele și data nașterii; pentru retail, numele și adresa de email asociată contului. Aceste întrebări au fost puse la începutul conversației pentru a fi mai ușor și mai rapid de procesat informațiile cu privire la client, astfel încât replicile ulterioare să aibă legătură numai cu intenția clientului.
- **Determinarea intenției** — după identificarea clientului, transcrierea primei replici relevante este trimisă modelului împreună cu lista celor opt intenții predefinite pentru domeniul respectiv. Dacă intenția nu poate fi identificată cu certitudine, robotul solicită clientului să reformuleze. După două încercări consecutive fără succes, apelul este redirecționat automat către un operator uman.
- **Colectarea detaliilor** — odată ce intenția este identificată, modelul verifică dacă toate informațiile necesare pentru rezolvarea problemei au fost deja menționate în conversație. Fiecare intenție are asociat un set specific de detalii obligatorii — numărul comenzii pentru o problemă de livrare, data și suma pentru o tranzacție suspectă, numărul dosarului pentru un dosar respins. Modelul formulează întrebări de clarificare pe baza contextului, câte una pe rând, până când toate informațiile sunt complete.
- **Oferirea soluției și confirmarea** — soluția este generată dinamic de model, determinată de un nivel de dificultate ales aleator la începutul fiecărei conversații: simplu (30%), mediu (25%) sau complex (45%). Nivelul de dificultate influențează tonul și completitudinea răspunsului — un caz simplu primește o soluție concretă și imediată, unul mediu o rezolvare parțială cu explicarea limitărilor, iar unul complex doar înregistrarea cazului fără o soluție directă, generând în mod intenționat o experiență mai puțin satisfăcătoare. Pentru intențiile cu răspunsuri invariante față de context — precum blocarea unui card pierdut sau anularea unei programări — sunt utilizate replici predefinite, eliminând un apel LLM inutil. Conversația rămâne

deschisă până când clientul exprimă explicit dorința de a încheia, detectată prin potrivire cu expresii predefinite precum *la revedere*, *nu mai am întrebări* sau *asta era tot*.

La finalul fiecărei conversații este rulată o analiză care să extragă nivelul de satisfacție al clientului pe baza întregului dialog și să genereze un rezumat adaptat ca lungime nivelului de satisfacție detectat. Ambele rezultate, împreună cu istoricul complet al dialogului și metadatele conversației, sunt salvate într-un fișier `JSON` pentru monitorizare și analiză ulterioară

6.3. Modulul de extragere a intenției

Structura input-ului. Modulul primește dialogul acumulat până în acel moment, formatat ca șir de replici fără punctuație și fără includerea persoanei care a vorbit (client sau robot), împreună cu lista celor opt intenții predefinite pentru domeniu. Promptul atribuie modelului rolul de expert în clasificarea intențiilor pentru call-center-ul din domeniul respectiv, impune constrângeri explicite - maxim două intenții separate prin virgulă, exclusiv din lista furnizată - și include exemple few-shot cu conversații și etichetele corespunzătoare.

Structura output-ului. Modelul returnează una sau două etichete din vocabularul controlat. Dacă nicio etichetă nu este recunoscută, intenția este marcată automat ca *alta_solicitare*, declanșând redirecționarea către un operator uman după două încercări consecutive.

Colectarea detaliilor. Odată intenția identificată, modulul verifică ce informații necesare pentru rezolvarea problemei lipsesc din conversație. Fiecare intenție are asociat un set specific de detalii obligatorii definit în lista cu aceste detalii — numărul comenzii pentru o problemă de livrare, data și suma pentru o tranzacție suspectă. Dacă informații lipsesc, robotul le solicită câte una pe rând, evitând să bombardeze clientul cu mai multe întrebări simultan.

6.4. Modulul de estimare a satisfacției

Structura input-ului. Modulul este apelat o singură dată, la finalul conversației, după ce clientul și-a exprimat dorința de a încheia apelul. Estimarea satisfacției pe parcursul conversației ar putea fi eronată, deoarece tonul clientului se poate schimba semnificativ între prima și ultima replică. Promptul include definiții explicite pentru fiecare clasă, cu expresii tipice: *pozitiv* pentru un client vizibil mulțumit, *neutru* pentru unul care nu exprimă emoții (*ok*, *am înțeles*), și *negativ* pentru unul frustrat, chiar dacă acceptă formal soluția (*ce să fac*, ironie implicită).

Structura output-ului. Modelul returnează un singur cuvânt dintre cele trei clase. Extragerea se face prin căutare directă în răspunsul normalizat; dacă niciuna nu este găsită, satisfacția este marcată ca *necunoscut*. Valoarea obținută determină direct tipul rezumatului generat în pasul următor: pozitiv produce un rezumat scurt (20–40 cuvinte), neutru unul mediu (40–70 cuvinte), iar negativ unul lung (60–100 cuvinte). O clasificare greșită a satisfacției se propagă astfel și în rezumat, un compromis acceptabil față de complexitatea unui sistem care ar estima satisfacția independent de sumarizare.

6.5. Modulul de generare a rezumatului

Structura input-ului. Modulul primește dialogul complet și tipul de rezumat determinat de satisfacția estimată. Promptul specifică tipul (*SCURT*, *MEDIU* sau *LUNG*), intervalul de lungime în cuvinte, limba de redactare și conținutul minim obligatoriu: menționarea problemei

principale și a rezultatului final. Modelului i se interzice explicit să introducă informații care nu apar în conversație.

Structura output-ului. Modelul returnează text liber în limba română. Lungimea este validată prin numărarea cuvintelor și compararea cu intervalul stabilit. Metrica *în limite* a fost introdusă deoarece un rezumat prea scurt pentru o conversație negativă complexă omite informații relevante, iar unul prea lung pentru o conversație pozitivă simplă devine redundant față de scopul său practic de documentare rapidă.

6.6. Gestionarea soluțiilor

Soluțiile oferite clientului în faza de rezolvare a problemei urmează un mecanism hibrid. Pentru 21 de intenții cu răspunsuri invariante față de contextul specific al conversației — precum blocarea unui card pierdut sau confirmarea anulării unei programări — robotul folosește replici predefinite, eliminând un apel LLM inutil și reducând latența. Pentru celelalte 19 intenții care depind de detaliile colectate precum o programare la ghișeu sau o problemă de schimb valutar, soluția este generată dinamic de model. Astfel, cazurile cu risc operațional ridicat sunt tratate prin răspunsuri fixe, în timp ce scenariile deschise beneficiază de flexibilitatea modelului lingvistic.

În ambele cazuri, răspunsul este modulat de nivelul de dificultate alesaleator la începutul conversației: simplu (30%), mediu (25%) sau complex(45%). Ponderile reflectă distribuția din dataset-ul de evaluare și asigură că robotul generează o varietate realistă de experiențe. Dacă toate câmpurile obligatorii sunt complet determinate și intenția este marcată ca fiind deterministă, este selectată automat soluția predefinită. În caz contrar, răspunsul este asociat modelului lingvistic, care are acces la istoricul conversației și la setul de detalii extrase anterior.

Modulul de generare a soluțiilor este integrat direct în fluxul conversațional, fiind apelat imediat după etapa de colectare a detaliilor.

6.7. Testarea robotului telefonic

În cadrul acestei secțiuni a fost evaluat pipeline-ul complet al robotului telefonic. Pentru aceasta, au fost inițiate apeluri către numărul de telefon pus la dispoziție prin platforma Twilio și au fost testate mai multe scenarii pentru fiecare domeniu. Etapa inițială de testare a evidențiat o serie de probleme de funcționare și limitări ale sistemului, care au fost ulterior analizate și remediate pentru optimizarea pipeline-ului.

Primele scenarii testate cuprind trei conversații din domenii diferite — telecom, retail și servicii publice. Având în vedere structura replicilor și modul în care conversațiile evoluează, intențiile au fost identificate corect în proporții mari sau au fost alese intenții asemănătoare ca semnificație. Problema de bază a fost estimarea satisfacției, care implicit a determinat generarea unui rezumat ce nu respectă în totalitate cerințele impuse.

6.7.1. Analiza preliminara a sistemului

Prima conversație analizată aparține domeniului telecom. Aflat în vacanță la Milano, clientul a contactat serviciul de relații cu clienții deoarece nu putea utiliza conexiunea la internet mobil în roaming. Dialogul a devenit tensionat încă din prima parte a interacțiunii, deoarece operatorul a insistat asupra obținerii unei adrese exacte, informație pe care clientul nu o putea

furniza. Această solicitare repetată a generat confuzie și nemulțumire, clientul neînțelegând relevanța acesteia pentru problema semnalată.

Situația s-a deblocat în momentul în care operatorul a oferit recomandări tehnice concrete privind verificarea setărilor telefonului. După activarea roaming-ului de date și selectarea manuală a unei rețele disponibile, conexiunea la internet a fost restabilită, iar problema a fost rezolvată. Conversația s-a încheiat într-un mod politicos, clientul confirmând că serviciul funcționează din nou.

Deși modelul a clasificat satisfacția drept *neutră*, analiza manuală a dialogului sugerează că experiența clientului a fost mai degrabă negativă. Pe parcursul conversației apar mai multe semnale explicite de frustrare, clientul exprimându-și nemulțumirea față de întrebările repetitive și față de timpul petrecut fără a primi o soluție concretă. Clasificarea eronată pare să fie cauzată de faptul că modelul acordă o importanță mai mare rezultatului final al interacțiunii decât experienței utilizatorului pe durata întregului dialog.

Principalul factor care a influențat negativ satisfacția a fost rigiditatea operatorului. Solicitarea repetată a unei adrese exacte pentru o problemă care s-a dovedit a fi legată de configurarea roaming-ului a fost inutilă. În plus, înainte de oferirea soluției, operatorul a sugerat că problema necesită o investigație suplimentară și un timp mai îndelungat de rezolvare, ceea ce a accentuat nemulțumirea clientului.

Rezolvarea efectivă a apărut abia după ce clientul a solicitat în mod explicit informații despre alte variante de asistență și despre posibile verificări pe care le poate efectua singur. Chiar dacă apelul s-a încheiat într-o manieră politicoasă, iar problema tehnică a fost rezolvată, replica finală a clientului — prin care acesta remarcă timpul pierdut în prima parte a conversației — indică faptul că experiența generală nu poate fi considerată una complet satisfăcătoare. Din acest motiv, conversația ar fi trebuit încadrată în categoria *negativă*, nu *neutră*.

A doua conversație analizată aparține domeniului serviciilor publice. Clientul a contactat serviciul de asistență cu scopul de a solicita clarificări și de a contesta o decizie de respingere a dosarului său de subvenție. Spre deosebire de prima interacțiune analizată, parcursul acestui dialog a fost mult mai fluid, eficient și lipsit de blocaje sau momente de tensiune.

Situația s-a desfășurat într-un mod constructiv încă de la început. Operatorul a colectat datele necesare identificării și numărul deciziei contestate, înregistrând solicitarea fără întârziere. Pe lângă procesarea contestației, acesta a oferit explicații cu privire la pașii următori și la termenul estimat de soluționare. În plus, a sugerat transmiterea unor documente justificative suplimentare prin e-mail, oferind astfel clientului o modalitate concretă de a sprijini analiza cazului și, eventual, de a accelera procesul de reevaluare.

Deși modelul a clasificat satisfacția ca fiind negativă, analiza dialogului indică o experiență pozitivă. Eroarea pare să provină din interpretarea izolată a replicilor finale ale clientului, în special a expresiei „nu cred că este destul de clar", care, în absența punctuației și a structurii conversaționale, poate fi interpretată ca o formă de nemulțumire. În contextul complet al discuției („nu, cred că este destul de clar, mulțumesc frumos"), particula „nu" funcționează ca răspuns la întrebarea operatorului, iar clientul confirmă de fapt că a înțeles pașii explicați.

Încadrarea corectă a acestei conversații în categoria satisfacției pozitive este susținută de eficiența rezolvării și de calitatea ghidajului oferit de operator, precum și de formulările finale ale clientului, care sunt politicoase și de mulțumire. Interacțiunea se încheie într-un ton pozitiv, iar comportamentul operatorului rămâne empatic și profesionist până la final.

A treia conversație analizată aparține domeniului retail. Clientul a contactat serviciul de relații cu clienții pentru a solicita informații privind statusul unei comenzi care nu fusese livrată la timp. Acest apel s-a desfășurat într-o manieră eficientă, transparentă și axată pe soluționarea rapidă a problemei.

Deși comanda era într-adevăr blocată, operatorul a oferit o explicație clară și onestă, asumând întârzierea ca fiind cauzată de o problemă la firma de curierat. Pentru a nu lăsa clientul fără un răspuns concret, operatorul a propus deschiderea unei solicitări de status suplimentare către departamentul logistic.

De această dată, modelul a clasificat corect satisfacția drept *pozitivă*. Spre deosebire de erorile de interpretare din cazurile anterioare, algoritmul a identificat corect tonul calm și cooperant al clientului. Sintagma *nu nu este perfect în regulă*, deși din punct de vedere strict literal conține o dublă negație care ar putea induce în eroare sistemul, a fost corect integrată în contextul său real, reprezentând un răspuns la întrebarea operatorului. De această dată, utilizarea cuvântului *perfect* poate ghida modelul către starea de spirit pozitivă a clientului. Acesta s-a arătat mulțumit de implicarea operatorului, exprimându-și aprecierea (*am înțeles mulțumesc frumos*) și închizând apelul pe un ton pozitiv.

Atitudinea orientată către client a operatorului a jucat un rol important în nivelul de satisfacție al clientului. Acesta a oferit o acțiune imediată — transmiterea solicitării de status — și o garanție de monitorizare constantă. Această abordare i-a oferit clientului siguranța că situația sa este tratată cu prioritate.

6.7.2. Decizii de îmbunătățire pe baza testelor preliminare

Pe baza celor trei conversații au fost identificate și corectate următoarele probleme concrete în pipeline-ul de analiză automată:

1. Detectarea satisfacției pe transcrieri STT fără punctuație: O problemă a apărut în cazul conversației din servicii publice, unde clientul a spus *nu cred că este destul de clar* ca răspuns la întrebarea operatorului — o negație care, în context, indica de fapt confirmare. Fără punctuație, modelul nu putea distinge această frază de o nemulțumire reală. Au fost implementate două corecții: în primul rând, înainte de analiză, dialogul este procesat printr-un apel LLM de adăugare a punctuației și majusculelor; în al doilea rând, promptul de satisfacție a fost completat cu instrucțiunea explicită că textul provine dintr-o transcriere automată și că ambiguitățile trebuie interpretate strict contextual.

2. Reguli mai clare de clasificare pozitiv vs. neutru vs. negativ: Clasificarea inițială confunda cazurile în care clienții încheiau politicos, dar exprimau niveluri diferite de satisfacție, din cauza unor reguli prea generale. Pentru corectare, au fost introduse criterii mai clare: aprecierea explicită indică satisfacție pozitivă, încheierea neutră fără entuziasm este considerată neutru, iar frustrarea urmată de acceptare este etichetată ca negativ.

3. Soluții concrete în faza de rezolvare: În toate cele trei conversații, robotul tindea să genereze răspunsuri vagi de tipul *procesarea poate dura câteva săptămâni*. Cauza era instrucțiunea asociată nivelului de dificultate complex, care limita formularea unor soluții directe. Această regulă a fost ajustată, astfel încât modelul să ofere întotdeauna cel puțin o acțiune concretă pe care clientul o poate realiza imediat, inclusiv în scenarii complexe.

4. Detectarea corectă a închiderii conversației: Pipeline-ul declanșa uneori închiderea conversației prematur, înainte ca ultima replică a robotului să fie redată complet. Problema a fost rezolvată, conversația fiind considerată încheiată doar după confirmarea finalizării redării audio, nu pe baza unui timeout fix.

În urma modificărilor implementate, fluxul conversațiilor a fost îmbunătățit semnificativ și reproduce într-o măsură mai mare interacțiunile reale din centrele de suport. Robotul oferă răspunsuri mai clare și soluții mai bine formulate, reducând tendința de a furniza răspunsuri evazive. Totuși, atenția sporită acordată clasei neutre a introdus un bias în procesul de clasificare, astfel încât o parte dintre conversațiile cu caracter pozitiv au fost încadrate în mod eronat ca

fiind neutre.

În cadrul promptului au fost adăugate informații suplimentare și instrucțiuni mai detaliate privind clasificarea acestei clase. Cu toate acestea, rezultatele au indicat o degradare a performanței, motiv pentru care, în ultima iterație a experimentelor, promptul utilizat pentru estimarea satisfacției clientului a fost simplificat. Deși exemplele few-shot au fost păstrate, descrierile extinse ale clasei neutre au generat confuzie și au determinat modelul să acorde o importanță disproporționată acestei categorii. Pentru a reduce acest efect, definițiile claselor au fost simplificate, iar indicații de tipul *„dacă clientul încheie conversația politicos, dar fără entuziasm sau apreciere explicită, clasifică interacțiunea ca fiind neutră"* au fost eliminate.

6.7.3. Dashboard de monitorizare

Pentru vizualizarea și monitorizarea conversațiilor procesate de robotul telefonic, a fost implementat un dashboard interactiv utilizând biblioteca Streamlit. Interfața oferă o vedere de ansamblu asupra activității sistemului și permite analiza detaliată a fiecărei interacțiuni înregistrate.

Panoul principal afișează statistici agregate: numărul total de conversații, distribuția pe clase de satisfacție (pozitiv, neutru, negativ) și repartiția pe domenii și niveluri de dificultate, vizualizate prin grafice de tip donut și histogramă. Secțiunea de filtrare permite restrângerea rezultatelor după domeniu, satisfacție, dificultate și interval de date, facilitând analiza comparativă pe subseturi specifice.

Conversații (15 rezultate)

ID	Domeniu	Dificultate	Satisfacție	Intenție	Nr. replici	Data
20260615_193258	banking	complexa	negativ	tranzactie_suspecta	21	15.06.2026 19:32
20260615_192500	banking	simpla	neutru	card_blocat	11	15.06.2026 19:25
20260615_192150	banking	complexa	negativ	tranzactie_gresita	17	15.06.2026 19:21
20260615_191439	medicina	complexa	pozitiv	problema_reteta	21	15.06.2026 19:14
20260615_190757	medicina	complexa	negativ	problema_programare	15	15.06.2026 19:07
20260615_190122	medicina	complexa	negativ	rezultate_analize	13	15.06.2026 19:01
20260615_185520	retail	medie	neutru	comanda_gresita	21	15.06.2026 18:55
20260615_184701	retail	complexa	negativ	problema_livrare	17	15.06.2026 18:47
20260615_183917	servicii_publice	complexa	neutru	dosar_respins	19	15.06.2026 18:39
20260615_183126	servicii_publice	simpla	neutru	problema_plata_taxa	15	15.06.2026 18:31

Figura 6.2: Tabelul cu lista conversațiilor

Tabelul central, așa cum se poate vedea în Figura 6.2, listează toate conversațiile disponibile, cu metadatele asociate: identificator, domeniu, dificultate, satisfacție, intenție detectată și număr de replici. Selectarea unei conversații din tabel deschide o secțiune de vizualizare detaliată, care prezintă rezumatul generat automat și dialogul complet, cu replicile operatorului și ale clientului marcate distinct.

Figura 6.3 ilustrează modul în care fiecare conversație poate fi analizată individual; operatorul uman poate selecta în mod specific doar conversațiile negative și poate citi rezumatul pe care modelul l-a produs. Dacă acesta nu oferă suficiente informații utile, conversația este redată în continuare printr-un schimb de replici ușor de urmărit.

Dashboardul citește datele direct din fișierele JSON salvate de pipeline în directorul local, fără a necesita o bază de date separată, și poate fi actualizat în timp real.

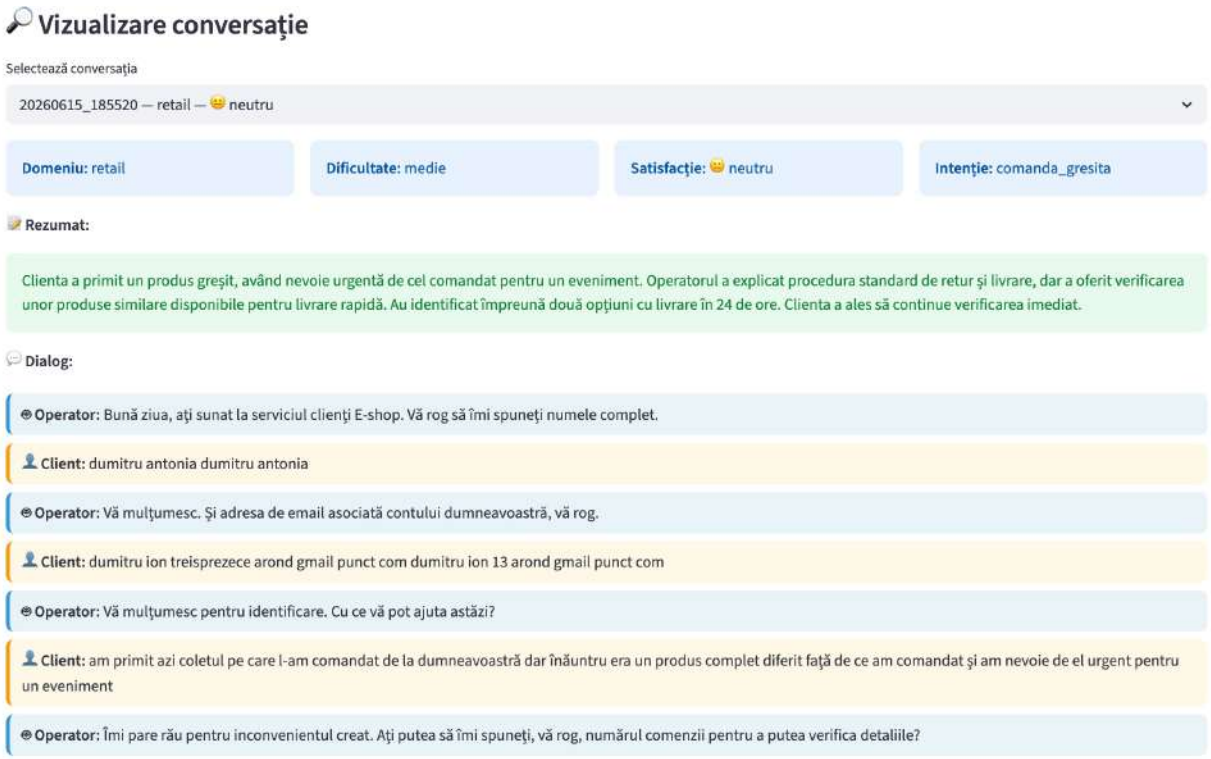

Figura 6.3: Conversația completă și rezultatele acesteia

6.7.4. Ultimă etapa de testare

În etapa finală de evaluare au fost testate 15 conversații telefonice, acoperind toate cele cinci domenii analizate, cu câte trei conversații pentru fiecare domeniu. Distribuția etichetelor de satisfacție a fost de 4 conversații pozitive, 5 neutre și 6 negative și poate fi analizată mult mai clar în Figura 6.4. Durata medie a unei conversații complete a fost de 354,7 secunde, valoare influențată în principal de timpul necesar transcrierii replicilor clientului.

Domeniul telecomunicații a inclus scenarii corespunzătoare intențiilor *activare_esuata*, *problema_semnal* și *reziliere_contract*. Primele două conversații au fost clasificate corect ca pozitive, iar cea de-a treia ca negativă. Toate intențiile au fost identificate corect. Conversația privind rezilierea contractului a evidențiat o limitare a sistemului: clientul, nemulțumit de perioada de preaviz de 30 de zile, a solicitat rezilierea imediată, însă operatorul nu a putut oferi o soluție directă. Clasificarea negativă este justificată, întrucât conversația s-a încheiat abrupt, printr-o replică ce exprimă în mod clar frustrarea clientului.

Domeniul serviciilor publice a cuprins scenarii de tip *programare_ghiseu*, *problema_plata_taxa* și *dosar_respins*. Primele două conversații au fost clasificate corect ca neutre. În cazul scenariului *dosar_respins*, clientul și-a exprimat nemulțumirea și presiunea financiară resimțită, însă fără formulări explicit negative. Din acest motiv, modelul a estimat satisfacția ca fiind neutră, deși o clasificare negativă ar fi fost de asemenea justificată. Intenția a fost identificată ca *dosar_respins* în loc de *contestatie_decizie*; totuși, cele două categorii sunt suficient de apropiate semantic încât eroarea are un impact redus asupra utilității practice a sistemului.

Domeniul retail a inclus intențiile *retur_produs*, *problema_livrare* și *comanda_gresita*. Conversația privind returul unui produs a fost clasificată corect ca pozitivă, clientul exprimând apreciere explicită pentru soluția oferită. În cazul livrării întârziate, intenția a fost identificată drept *problema_livrare* în loc de *comanda_intarziata*, diferență minoră între două categorii

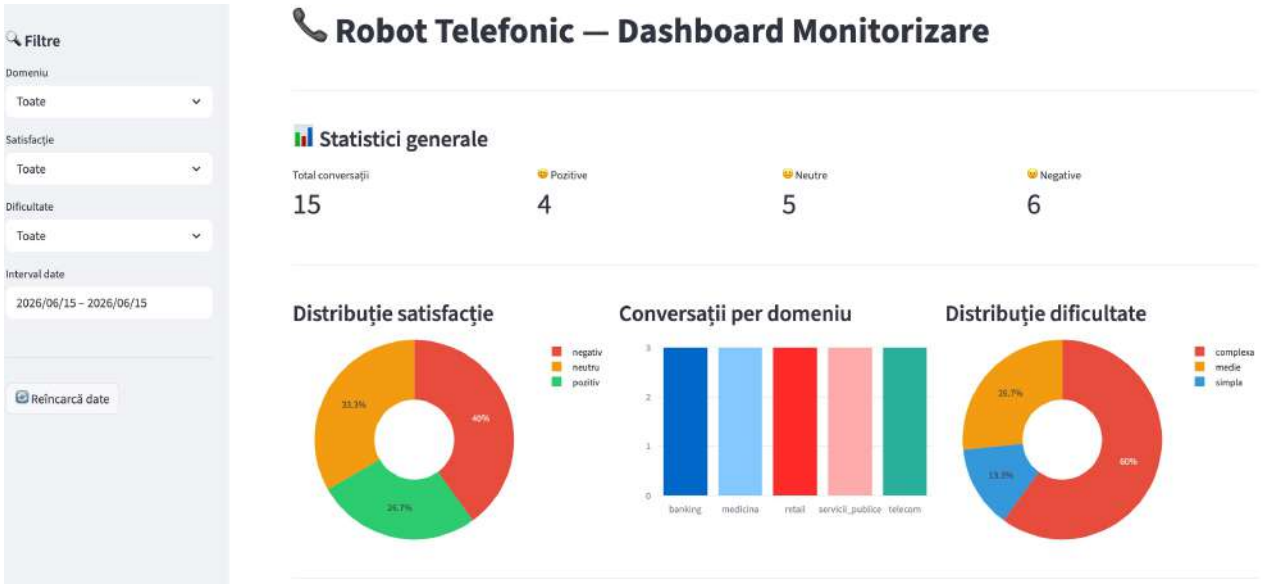

Figura 6.4: Dashboard Streamlit

similare. Conversația referitoare la o comandă greșită a evidențiat aceeași dificultate observată și în domeniul serviciilor publice: deși clientul era nemulțumit, acesta nu și-a exprimat frustrarea într-un mod suficient de explicit pentru a fi clasificat ca negativ. În consecință, modelul a estimat satisfacția ca fiind neutră. Aceste exemple confirmă faptul că delimitarea dintre clasele neutră și negativă rămâne dificilă în situațiile în care nemulțumirea este exprimată indirect sau într-o manieră moderată.

Domeniul medical a generat o conversație clasificată pozitiv (*problema_reteta*), una neutră (*anulare_programare*) și una negativă (*rezultate_analize*). Intenția *anulare_programare* a fost identificată ca *problema_programare*, eroare minoră care nu afectează semnificativ interpretarea solicitării. Conversația referitoare la întârzierea rezultatelor analizelor a fost clasificată corect ca negativă, clientul menționând existența unei intervenții chirurgicale programate în următoarele zile și exprimând în mod explicit nemulțumirea față de situația întâmpinată.

Domeniul bancar a inclus scenariile *card_blocat*, *tranzactie_gresita* și *tranzactie_suspecta*. Solicitarea privind deblocarea cardului a fost rezolvată cu succes și clasificată corect ca neutră. În această conversație a fost observat un artefact al sistemului de recunoaștere vocală, care a introdus în mod eronat cuvântul *„nu”* în replica finală a clientului. Acest fenomen a fost întâlnit și în alte conversații, STT-ul inserând ocazional cuvinte scurte, în special negații, fără corespondent în semnalul audio original. Conversația referitoare la un transfer greșit a fost clasificată ca negativă, deși tonul clientului era mai degrabă resemnat decât nemulțumit; o clasificare neutră ar fi putut fi considerată la fel de plauzibilă. În ultimul scenariu, intenția identificată a fost *tranzactie_suspecta*, în locul unei categorii alternative apropiate semantic, fără impact relevant asupra rezultatului final.

Analiza latențelor confirmă observațiile obținute în etapele anterioare ale evaluării. Latența mediană a modulului STT a fost de 12,98 secunde per replică, iar media de 14,04 secunde, reflectând timpul necesar transcrierii unor fragmente audio cu durate cuprinse între 10 și 30 de secunde. Pentru componenta LLM, latența mediană a fost de 2,17 secunde, iar media de 2,58 secunde per tur de dialog. Valorile mai ridicate observate ocazional sunt asociate cu acumularea unui context conversațional mai extins. Deși aceste latențe limitează utilizarea sistemului în scenarii complet interactive, ele sunt acceptabile pentru analiza post-apel, reprezentând principalul caz de utilizare vizat.

Per ansamblu, identificarea intențiilor a fost realizată corect în 11 din cele 15 conversații analizate. Cele patru erori observate au implicat intenții apropiate semantic și nu afectează semnificativ utilitatea practică a sistemului. Clasificarea satisfacției s-a dovedit mai dificilă, principalele erori apărând la granița dintre clasele neutră și negativă. Rezultatele confirmă că estimarea satisfacției rămâne cea mai provocatoare componentă a pipeline-ului, în special în situațiile în care frustrarea clientului este exprimată implicit sau într-o manieră moderată.

6.7.5. Concluzii ale testării robotului telefonic

Rezultatele testării end-to-end a robotului telefonic indică un sistem funcțional în ceea ce privește identificarea intențiilor, cu performanțe bune chiar și în scenarii din domenii diferite. În majoritatea cazurilor, intențiile au fost detectate corect sau au fost înlocuite cu categorii semantic apropiate, fără impact semnificativ asupra interpretării generale a conversației.

Principala limitare observată în toate etapele de testare rămâne clasificarea satisfacției utilizatorului. Modelul tinde să confunde în mod special clasele *neutru* și *negativ* în situațiile în care frustrarea este exprimată implicit sau moderat, fără formulări explicite. Acest aspect confirmă sensibilitatea ridicată a sarcinii și necesitatea unor reguli de clasificare mai stricte sau a unor seturi de antrenare mai bine calibrate.

Din punct de vedere al experienței conversaționale, pipeline-ul a evoluat semnificativ după aplicarea modificărilor, generând interacțiuni mai naturale, mai coerente și mai orientate spre soluție. Reducerea răspunsurilor evazive și introducerea unor acțiuni concrete a îmbunătățit calitatea generală a dialogului.

În ceea ce privește performanța sistemului, latențele observate sunt acceptabile pentru analizarea ulterioară a conversațiilor, însă reprezintă în continuare o limitare pentru utilizarea în timp real. Sistemul demonstrează un echilibru bun între acuratețe, robustețe și utilizabilitate, cu direcții clare de îmbunătățire în zona de clasificare a satisfacției și optimizare a latenței.

7. CONCLUZII

Această lucrare a propus evaluarea comparativă a cinci modele lingvistice de mari dimensiuni pe trei sarcini specifice conversațiilor de call-center în limba română: detectarea intenției clientului, estimarea satisfacției acestuia la finalul discuției și generarea unui rezumat al acesteia. Experimentele au fost realizate pe un dataset de 100 de conversații, generate tot prin intermediul LLM-urilor, distribuite pe cinci domenii, cu adnotări manuale verificate, și au inclus o analiză performanțelor tehnicilor de prompt engineering cu complexități diferite.

Rezultatele subliniază superioritatea modelelor apelate prin API față de modelele locale pe toate cele trei sarcini, GPT-4.1-mini obținând cele mai bune performanțe globale: 73% acuratețe la detectarea intenției (F1 Macro 0.673), 83% acuratețe la estimarea satisfacției (F1 Macro 0.816) și BERTScore de 0.758 cu 100% rezumate încadrate în limitele de lungime impuse. Rezultatele modelelor locale sunt în proporții mari comparabile cu cele ale modelelor API, acestea reușind să înțeleagă contextul conversației foarte bine în aproape toate cazurile. Principala limitare a modelelor locale rămâne latența: 10–17 secunde față de sub o secundă pentru GPT și Command.

Domeniul serviciilor publice s-a dovedit cel mai dificil pentru toate modelele, cu acuratețe între 50% și 60%, fapt datorat ambiguității termenilor utilizați în cadrul discuțiilor, dar și similarității dintre intenții. Acest domeniu nu este clar definit, iar modelele, cel mai probabil, l-au întâlnit pentru prima dată, comparativ cu alte domenii precum medicina și telecomunicațiile, care sunt destul de uzuale. Clasa neutră reprezintă dificultatea principală a sarcinii de estimare a satisfacției pentru aproape toate modelele; conversațiile simple neutre sunt adesea confundate cu cele pozitive, iar cele complexe și neutre cu cele negative.

Evaluarea pipeline-ului audio a demonstrat că integrarea STT introduce o degradare moderată de aproximativ 10 procente pe ambele sarcini de clasificare, menținând în același timp performanțe bune la generarea rezumatului (90% în limite).

7.1. Contribuții personale

Lucrarea încearcă să testeze în ce măsură modelele lingvistice actuale pot fi utilizate eficient într-un scenariu real de tip call-center, în special pentru limba română și pentru sarcini multiple executate într-un singur flux de procesare. Principalele contribuții ale acestei lucrări sunt:

- **Dataset sintetic adnotat** — construirea unui set de date de 100 de conversații telefonice în limba română, distribuit echilibrat pe cinci domenii și trei niveluri de satisfacție, cu adnotări pentru intenție, satisfacție și rezumat, verificate manual.
- **Analiză sistematică de prompt engineering** — evaluarea a patru versiuni de prompturi pe cinci modele și trei sarcini, identificând tiparele de comportament specifice fiecărui model și configurația optimă per sarcină.
- **Pipeline complet de analiză vocală** — implementarea unui sistem end-to-end care integrează transcriere STT (Zevo), procesare LLM (GPT-4.1-mini) și sinteză TTS (Zevo) pentru analiza automată a conversațiilor telefonice înregistrate.

– **Robot telefonic funcțional** — implementarea unui agent conversațional capabil să gestioneze apeluri telefonice reale prin integrarea cu platforma Twilio, cu flux conversațional structurat în patru faze și mecanism de dificultate aleatoare pentru generarea de conversații diverse.
– **Evaluare comparativă locală vs. API** — analiza diferențelor de performanță și latență dintre modelele locale de 7 miliarde de parametri și modelele comerciale de mari dimensiuni, pe o limbă cu resurse limitate.

7.2. Direcții viitoare

Rezultatele obținute în cadrul acestei lucrări deschid mai multe direcții de îmbunătățire a aplicației, atât la nivel de metodologie, cât și la nivel de aplicabilitate.

Îmbunătățirea modelelor prin fine-tuning pe date de call-center în limba română. Modelele locale RoMistral și RoGemma au demonstrat o înțelegere semantică bună a intențiilor, însă întâmpină dificultăți legate de respectarea constrângerilor de format și de latență. Antrenarea suplimentară (fine-tuning) pe un set de date real de conversații din call-centere, alături de definiții clare ale domeniilor și intențiilor predefinite ar putea îmbunătăți performanțele modelelor.

Extinderea taxonomiei de intenții și a mecanismului de clasificare. Setul actual de 40 de etichete acoperă scenariile frecvente, însă un sistem real trebuie să gestioneze o varietate mai mare de solicitări. O posibilă direcție este introducerea unei clasificări ierarhice sau a unei clase dedicate pentru intenții necunoscute, dar și a unor definiții mult mai riguroase pentru fiecare intenție. După cum s-a putut observa în multe dintre răspunsurile oferite de LLM-uri, unele etichete sunt asemănătoare și uneori nu este clar nici pentru o persoană care dintre ele ar trebui aleasă.

Evaluarea pe date reale și diversificarea corpusului. Dataset-ul sintetic utilizat nu surprinde complet variabilitatea limbajului natural (ezitări, formulări incomplete, variații regionale sau code-switching română–engleză), iar zgomotul nu este prezent în niciuna dintre conversații. Colectarea și adnotarea unui set de date reale din call-centere ar permite o evaluare mai realistă și ar oferi o imagine mai fidelă asupra performanței în producție.

Optimizarea latenței pentru scenarii de utilizare în timp real. Latența totală a pipeline-ului (STT + LLM + TTS) atinge în prezent durate ridicate, ceea ce poate afecta experiența utilizatorului în scenarii de tip call-center în timp real. Principalul factor de întârziere nu este reprezentat de modelele lingvistice, ci de etapa de transcriere automată (STT) și de integrarea cu infrastructura de comunicație (Twilio), în timp ce componenta LLM contribuie relativ puțin la timpul total de răspuns.

Extinderea evaluării către scenarii de tip real-time și interacțiuni conversaționale lungi. O direcție suplimentară de cercetare constă în evaluarea sistemului pe conversații mult mai lungi și mai dinamice, în care contextul se acumulează pe parcursul mai multor schimburi de replici dintre utilizator și operator. În astfel de scenarii, clientul își poate schimba intenția pe parcursul conversației sau pot apărea și ambiguități contextuale, care nu sunt complet surprinse de setul actual de date format din dialoguri scurte și independente.

BIBLIOGRAFIE

[1] Invensis, *Impact of AI on Call Centers: 7 Key Impacts in 2025*, Invensis Blog. [Online] Disponibil: `https://www.invensis.net/blog/impact-of-ai-on-call-centers` [Accessed: Jun. 2026].

[2] The Team at CallMiner, *The Future of AI Call Center Automation in 2025 and Beyond*, CallMiner Blog. [Online] Disponibil: `https://callminer.com/blog/the-future-of-ai-call-center-automation-in-2025-and-beyond` [Accessed: Jun. 2026].

[3] Mihai Masala, Denis C Ilie-Ablachim, Alexandru Dima, Dragos Corlatescu, Miruna Zavelca, Ovio Olaru, Simina Terian, Andrei Terian, Marius Leordeanu, Horia Velicu et al., „" Vorbe\c {s} ti Rom\^ ane\c {s} te?" A Recipe to Train Powerful Romanian LLMs with English Instructions", *arXiv preprint arXiv:2406.18266*, vol. nr. 2024.

[4] GoodCall, *Call Center Automation Trends Transforming Customer Service*, GoodCall Blog. [Online] Disponibil: `https://www.goodcall.com/voice-ai/call-center-automation-trends` [Accessed: Jun. 2026].

[5] Sophia White, Nathan Carter, „Natural Language Understanding in Virtual Assistants: Advances and Challenges", *International Journal on Mechanical Engineering and Robotics*, vol. 13, nr. 1, pp. 9–15 2024.

[6] Sepp Hochreiter, Jürgen Schmidhuber, „Long short-term memory", *Neural computation*, vol. 9, nr. 8, MIT press, pp. 1735–1780 1997.

[7] Dzmitry Bahdanau, „Neural machine translation by jointly learning to align and translate", *arXiv preprint arXiv:1409.0473*, vol. nr. 2014.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, Illia Polosukhin, „Attention is all you need", *Advances in neural information processing systems*, vol. 30, nr. 2017.

[9] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, Yoshua Bengio, „A structured self-attentive sentence embedding", *arXiv preprint arXiv:1703.03130*, vol. nr. 2017.

[10] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever et al., „Improving language understanding by generative pre-training", vol. nr. San Francisco, CA, USA 2018.

[11] Alessandro Raganato, Jörg Tiedemann, „An analysis of encoder representations in transformer-based machine translation", *Proceedings of the 2018 EMNLP workshop BlackboxNLP: analyzing and interpreting neural networks for NLP*, pp. 287–297 2018.

[12] Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, Chunhua Shen, „Conditional positional encodings for vision transformers", *arXiv preprint arXiv:2102.10882*, vol. nr. 2021.

[13] Sepp Hochreiter, „The vanishing gradient problem during learning recurrent neural nets and problem solutions", *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, nr. 02, World Scientific, pp. 107–116 1998.

[14] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, Ivan Titov, „Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned", *arXiv preprint arXiv:1905.09418*, vol. nr. 2019.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, „Bert: Pre-training of deep bidirectional transformers for language understanding", *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)*, pp. 4171–4186 2019.

[16] Anis Redjdal, Luis Pinto, Michel Desmarais, „Optimizing Encoder-Only Transformers for Session-Based Recommendation Systems", *arXiv preprint arXiv:2410.11150*, vol. nr. 2024.

[17] Jesse Roberts, „How powerful are decoder-only transformer neural models?", *2024 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8 2024.

[18] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, Jacob Steinhardt, „Measuring massive multitask language understanding", *arXiv preprint arXiv:2009.03300*, vol. nr. 2020.

[19] Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani et al., „Are we done with mmlu?", *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5069–5096 2025.

[20] Maxwell Horton, Qingqing Cao, Chenfan Sun, Yanzi Jin, Sachin Mehta, Mohammad Rastegari, Moin Nabi, „KV Prediction for Improved Time to First Token", *arXiv preprint arXiv:2410.08391*, vol. nr. 2024.

[21] OpenAI, *Introducing GPT-4.1 in the API*, OpenAI website. [Online] Disponibil: `https://openai.com/index/gpt-4-1/` [Accesat: Ian. 2026].

[22] Google, *Gemini 2.5 Pro update: Coding, web apps with Gemini*, Google Blog. [Online] Disponibil: `https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/#gemini-2-5-pro` [Accesat: Ian. 2026].

[23] Google, *Gemini API Pricing*, Google AI Developer Documentation. [Online] Disponibil: `https://ai.google.dev/gemini-api/docs/pricing` [Accessed: Jan. 2026].

[24] Karim Saadeldin, *I Tested 6 Gemini Models for Voice AI Latency. The Results Will Change How You Build*, DEV Community. [Online] Disponibil: `https://dev.to/karimgeh/i-tested-6-gemini-models-for-voice-ai-latency-the-results-will-change-how-you-build-1kbm` [Accessed: Jan. 2026].

[25] Jonathan Yin, Arman Cohan, „Reasoning with Less: Distilling Reasoning Models with Small Datasets", vol. nr.

[26] John Dang, Shivalika Singh, Daniel D'souza, Arash Ahmadian, Alejandro Salamanca, Madeline Smith, Aidan Peppin, Sungjin Hong, Manoj Govindassamy, Terrence Zhao et al., „Aya expanse: Combining research breakthroughs for a new multilingual frontier", *arXiv preprint arXiv:2412.04261*, vol. nr. 2024.

[27] Cohere, *Pricing*, Cohere website. [Online] Disponibil: `https://cohere.com/pricing` [Accesat: Ian. 2026].

[28] Team Cohere, Arash Ahmadian, Marwan Ahmed, Jay Alammar, Milad Alizadeh, Yazeed Alnumay, Sophia Althammer, Arkady Arkhangorodsky, Viraat Aryabumi, Dennis Aumiller et al., „Command a: An enterprise-ready large language model", *arXiv preprint arXiv:2504.00698*, vol. nr. 2025.

[29] Devendra Singh Chaplot, „Albert q. jiang, alexandre sablayrolles, arthur mensch, chris bamford, devendra singh chaplot, diego de las casas, florian bressand, gianna lengyel, guillaume lample, lucile saulnier, lélio renard lavaud, marie-anne lachaux, pierre stock, teven le scao, thibaut lavril, thomas wang, timothée lacroix, william el sayed", *arXiv preprint arXiv:2310.06825*, vol. 3, nr. 2023.

[30] Bowen Yang, Bharat Venkitesh, Dwarak Talupuru, Hangyu Lin, David Cairuz, Phil Blunsom, Acyr Locatelli, „Rope to nope and back again: A new hybrid attention strategy, 2025", *URL https://arxiv. org/abs/2501.18795*, vol. nr.

[31] Byeongho Heo, Song Park, Dongyoon Han, Sangdoo Yun, „Rotary position embedding for vision transformer", *European Conference on Computer Vision*, pp. 289–305 2024.

[32] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano et al., „Training verifiers to solve math word problems", *arXiv preprint arXiv:2110.14168*, vol. nr. 2021.

[33] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, Yejin Choi, „Winogrande: An adversarial winograd schema challenge at scale", *Communications of the ACM*, vol. 64, nr. 9, ACM New York, NY, USA, pp. 99–106 2021.

[34] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, Yejin Choi, „Hellaswag: Can a machine really finish your sentence?", *arXiv preprint arXiv:1905.07830*, vol. nr. 2019.

[35] Stefan Daniel Dumitrescu, Petru Rebeja, Beata Lorincz, Mihaela Gaman, Andrei Avram, Mihai Ilie, Andrei Pruteanu, Adriana Stan, Lorena Rosia, Cristina Iacobescu et al., „Liro: Benchmark and leaderboard for romanian language tasks", *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)* 2021.

[36] Anca Maria Tache, Mihaela Gaman, Radu Tudor Ionescu, „Clustering word embeddings with self-organizing maps. application on laroseda–a large romanian sentiment data set", *arXiv preprint arXiv:2101.04197*, vol. nr. 2021.

[37] Baseten, *Benchmarking Fast Mistral 7B Inference: Benchmarking Factors for Dedicated Model Deployments*, Baseten Blog. [Online] Disponibil: `https://www.baseten.co/blog/benchmarking-fast-mistral-7b-inference/#benchmarking-factors-for-dedicated-model-deployments` [Accessed: Jan. 2026].

[38] Groq, *Groundbreaking Gemma 7B Performance Running on the Groq LPU Inference Engine*, Groq Blog. [Online] Disponibil: `https://groq.com/blog/groundbreaking-gemma-7b-performance-running-on-the-groq-lpu-inference-engine` [Accessed: Jan. 2026].

[39] Jun Ping Ng, Viktoria Abrecht, „Better summarization evaluation with word embeddings for ROUGE", *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 1925–1930 2015.

[40] Youngja Park, Stephen C Gates, „Towards real-time measurement of customer satisfaction using automatically generated call transcripts", *Proceedings of the 18th ACM conference on Information and knowledge management*, pp. 1387–1396 2009.

[41] Shariq Shah, Hossein Ghomeshi, Edlira Vakaj, Emmett Cooper, Shereen Fouad, „A review of natural language processing in contact centre automation", *Pattern Analysis and Applications*, vol. 26, nr. 3, Springer, pp. 823–846 2023.

[42] Anthropic, *The Claude 3 Model Family: Opus, Sonnet, Haiku*, rap. teh., Anthropic, Mar. 2024, Disponibil: `https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf`.

[43] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon et al., „Constitutional ai: Harmlessness from ai feedback", *arXiv preprint arXiv:2212.08073*, vol. nr. 2022.

[44] JetBrains, *Interacting with Anthropic Claude Models on Amazon Bedrock using Go*, JetBrains Guide. [Online] Disponibil: `https://www.jetbrains.com/guide/go/tutorials/bedrock_with_go/models_interaction_anthropic/` [Accessed: May 2026].

[45] Gunawan Wang, Mustafa Musa Jaber, „A Deep Learning Approach to Sentiment Analysis of Hotel Reviews: Comparing BERT and LSTM Models", *International Journal of Advances in Artificial Intelligence and Machine Learning*, vol. 2, nr. 2, pp. 67–75 2025.

[46] Theresa Wilson, Janyce Wiebe, Paul Hoffmann, „Recognizing contextual polarity in phrase-level sentiment analysis", *Proceedings of human language technology conference and conference on empirical methods in natural language processing*, pp. 347–354 2005.

[47] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, Aman Chadha, „A systematic survey of prompt engineering in large language models: Techniques and applications", *arXiv preprint arXiv:2402.07927*, vol. 1, nr. 2024.

[48] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever et al., „Language models are unsupervised multitask learners", *OpenAI blog*, vol. 1, nr. 8, p. 9 2019.

[49] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, Yusuke Iwasawa, „Large language models are zero-shot reasoners", *Advances in neural information processing systems*, vol. 35, nr. Pp. 22199–22213 2022.

[50] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell et al., „Language models are few-shot learners", *Advances in neural information processing systems*, vol. 33, nr. Pp. 1877–1901 2020.

[51] Noah Wang, Zy Peng, Haoran Que, Jiaheng Liu, Wangchunshu Zhou, Yuhan Wu, Hongcheng Guo, Ruitong Gan, Zehao Ni, Jian Yang et al., „Rolellm: Benchmarking, eliciting, and enhancing role-playing abilities of large language models", *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 14743–14777 2024.

[52] Yaru Hao, Yutao Sun, Li Dong, Zhixiong Han, Yuxian Gu, Furu Wei, „Structured prompting: Scaling in-context learning to 1,000 examples", *arXiv preprint arXiv:2212.06713*, vol. nr. 2022.

[53] Albert Lu, Hongxin Zhang, Yanzhe Zhang, Xuezhi Wang, Diyi Yang, „Bounding the capabilities of large language models in open text generation with prompt constraints", *Findings of the Association for Computational Linguistics: EACL 2023*, pp. 1982–2008 2023.

A. APENDIX 1

A.1. Prompturile utilizate pentru detectarea intenției – V1

```
def get_prompt(dialog, domeniu):
    return (
        "Analizeaza urmatoarea conversatie telefonica si "
        "identifica intentia clientului.\n\n"
        "Conversatie:\n" + dialog + "\n\n"
        "Care este intentia clientului? "
        "Raspunde cu un singur cuvant sau expresie scurta:"
    )

def get_prompt2(dialog, domeniu):
    return (
        "Conversatie:\n" + dialog + "\n\n"
        "Rezuma in 2-3 cuvinte ce a cerut clientul:"
    )
```

A.2. Prompturile utilizate pentru detectarea intenției – V2

```
def get_prompt(dialog, domeniu):
    intentii = INTENTII_DOMENII.get(domeniu, [])
    intentii_str = ", ".join(intentii)
    return (
        "Lucrezi ca analist de date intr-un call-center din domeniul " + domeniu + ". "
        "Sarcina ta zilnica este sa identifici motivul pentru care clientii suna, "
        "pe baza transcripturilor conversatiilor cu operatorii.\n\n"
        "REGULI:\n"
        "- Include DOAR ce a cerut sau intrebat clientul, nu actiunile operatorului\n"
        "- Alege DOAR din lista de intentii de mai jos\n\n"
        "INTENTII DISPONIBILE: " + intentii_str + "\n\n"
        "Conversatie:\n" + dialog + "\n\n"
        "De ce a sunat clientul? Raspunde cu una sau doua intentii din lista:"
    )

def get_prompt2(dialog, domeniu):
    intentii = INTENTII_DOMENII.get(domeniu, [])
    intentii_str = ", ".join(intentii)
    return (
        "Esti un agent specializat in analiza conversatiilor telefonice din call-center-uri "
        "din domeniul " + domeniu + ". Rolul tau este sa identifici intentia clientului.\n\n"
        "REGULI:\n"
        "- Include DOAR ce a cerut sau intrebat clientul, nu actiunile operatorului\n"
        "- Alege DOAR din lista de intentii de mai jos\n\n"
        "INTENTII DISPONIBILE: " + intentii_str + "\n\n"
        "Conversatie:\n" + dialog + "\n\n"
        "Raspunde cu una sau doua intentii din lista, separate prin virgula:"
    )
```

A.3. Prompturile utilizate pentru clasificarea intenției – V3

```
def get_prompt(dialog, domeniu):
    intentii = INTENTII_DOMENII.get(domeniu, [])
    intentii_str = ", ".join(intentii)
    return (
        "Esti un expert in clasificarea intentiilor pentru call-center-uri "
        "din domeniul " + domeniu + ".\n\n"
        "CONVERSATIE:\n" + dialog + "\n\n"
        "SARCINA: Pe baza conversatiei de mai sus, identifica intentia clientului.\n\n"
        "REGULI:\n"
        "1. Include DOAR ce a cerut sau intrebat clientul, ignora actiunile operatorului\n"
        "2. Alege EXCLUSIV din lista de intentii furnizata, nu inventa etichete noi\n"
        "3. Returneaza MAXIM doua intentii, separate prin virgula\n"
        "4. Prima intentie trebuie sa fie cea principala, cu care a sunat clientul\n\n"
        "INTENTII DISPONIBILE: " + intentii_str + "\n\n"
        "INTENTIE IDENTIFICATA:"
    )

def get_prompt2(dialog, domeniu):
    intentii = INTENTII_DOMENII.get(domeniu, [])
    intentii_str = ", ".join(intentii)
    return (
        "Esti un expert in clasificarea intentiilor pentru call-center-uri "
        "din domeniul " + domeniu + ".\n\n"
        "SARCINA: Identifica intentia sau intentiile clientului din conversatia de mai jos.\n\
    n"
        "REGULI:\n"
        "1. Include DOAR ce a cerut sau intrebat clientul,  ignora actiunile operatorului\n"
        "2. Alege EXCLUSIV din lista de intentii furnizata,  nu inventa etichete noi\n"
        "3. Returneaza MAXIM doua intentii, separate prin virgula\n"
        "4. Prima intentie trebuie sa fie cea principala, cu care a sunat clientul\n"
        "5. Daca intentia nu este exprimata explicit, deduce-o din contextul conversatiei\n"
        "6. Daca nicio eticheta nu se potriveste exact, alege cea mai apropiata din lista\n\n"
        "INTENTII DISPONIBILE: " + intentii_str + "\n\n"
        "CONVERSATIE:\n" + dialog + "\n\n"
        "INTENTIE IDENTIFICATA:"
    )
```

A.4. Prompturile utilizate pentru clasificarea intenției – V4

```
def get_prompt(dialog, domeniu):
    intentii = INTENTII_DOMENII.get(domeniu, [])
    intentii_str = ", ".join(intentii)
    exemple = EXEMPLE_FEWSHOT_LUNGI.get(domeniu, [])
    exemple_text = ""
    for dialog_ex, intentie_ex in exemple:
        exemple_text += (
            "CONVERSATIE:\n" + dialog_ex + "\n"
            "INTENTIE IDENTIFICATA: " + intentie_ex + "\n\n"
        )
    return (
        "Esti un expert in clasificarea intentiilor pentru call-center-uri "
        "din domeniul " + domeniu + ".\n\n"
        "CONVERSATIE:\n" + dialog + "\n\n"
        "SARCINA: Pe baza conversatiei de mai sus, identifica intentia clientului.\n\n"
        "REGULI:\n"
        "1. Include DOAR ce a cerut sau intrebat clientul, ignora actiunile operatorului\n"
        "2. Alege EXCLUSIV din lista de intentii furnizata, nu inventa etichete noi\n"
        "3. Returneaza MAXIM doua intentii, separate prin virgula\n"
        "4. Prima intentie trebuie sa fie cea principala, cu care a sunat clientul\n\n"
        "INTENTII DISPONIBILE: " + intentii_str + "\n\n"
        "EXEMPLE:\n" + exemple_text +
        "INTENTIE IDENTIFICATA:"
    )

def get_prompt2(dialog, domeniu):
    intentii = INTENTII_DOMENII.get(domeniu, [])
    intentii_str = ", ".join(intentii)
    exemple_lungi = {
        "banking": [
```

```
            ("OPERATOR: Buna ziua, cu ce va pot ajuta?\nCLIENT: Buna ziua, am o problema cu cardul meu, l-am pierdut ieri si nu stiu ce sa fac.\nOPERATOR: Inteleg, va ajut imediat.", "card_pierdut"),
            ("OPERATOR: Serviciul clienti, va ascult.\nCLIENT: Buna ziua, as vrea sa stiu de ce rata mea la credit a crescut luna aceasta.\nOPERATOR: Va verific contul.", "problema_credit"),
        ],
        "medicina": [
            ("OPERATOR: Clinica MedCare, buna ziua.\nCLIENT: Buna ziua, am facut analize saptamana trecuta si vreau sa aflu rezultatele.\nOPERATOR: Va caut in sistem.", "rezultate_analize"),
            ("OPERATOR: Cu ce va pot ajuta?\nCLIENT: Am o programare maine dar nu mai pot veni, as vrea sa o anulez.\nOPERATOR: Sigur, cum va numiti?", "anulare_programare"),
        ],
        "retail": [
            ("OPERATOR: Serviciul clienti, buna ziua.\nCLIENT: Buna ziua, am primit o comanda dar produsele nu sunt cele pe care le-am comandat.\nOPERATOR: Imi pare rau, va ajut.", "comanda_gresita"),
            ("OPERATOR: Cu ce va pot ajuta?\nCLIENT: Pachetul meu nu a ajuns si a trecut termenul de livrare de trei zile.\nOPERATOR: Va verific comanda.", "problema_livrare"),
        ],
        "telecom": [
            ("OPERATOR: Buna ziua, serviciul clienti.\nCLIENT: Buna ziua, am vrut sa imi port numarul la voi dar cererea a fost respinsa.\nOPERATOR: Va verific situatia.", "portare_esuata"),
            ("OPERATOR: Cu ce va pot ajuta?\nCLIENT: As vrea sa schimb abonamentul meu dar nu reusesc sa fac asta din aplicatie.\nOPERATOR: Va ajut eu.", "problema_modificare_abonament"),
        ],
        "servicii_publice": [
            ("OPERATOR: Primaria, buna ziua.\nCLIENT: Buna ziua, dosarul meu a fost respins si nu inteleg de ce.\nOPERATOR: Va caut dosarul.", "dosar_respins"),
            ("OPERATOR: Cu ce va pot ajuta?\nCLIENT: As vrea sa fac o programare la ghiseu pentru un act de identitate.\nOPERATOR: Va programez.", "programare_ghiseu"),
        ],
    }
    exemple = exemple_lungi.get(domeniu, [])
    exemple_text = ""
    for dialog_ex, intentie_ex in exemple:
        exemple_text += (
            "CONVERSATIE:\n" + dialog_ex + "\n"
            "INTENTIE IDENTIFICATA: " + intentie_ex + "\n\n"
        )
    return (
        "Esti un expert in clasificarea intentiilor pentru call-center-uri "
        "din domeniul " + domeniu + ".\n\n"
        "CONVERSATIE:\n" + dialog + "\n\n"
        "SARCINA: Pe baza conversatiei de mai sus, identifica intentia clientului.\n\n"
        "REGULI:\n"
        "1. Include DOAR ce a cerut sau intrebat clientul, ignora actiunile operatorului\n"
        "2. Alege EXCLUSIV din lista de intentii furnizata, nu inventa etichete noi\n"
        "3. Returneaza MAXIM doua intentii, separate prin virgula\n"
        "4. Prima intentie trebuie sa fie cea principala, cu care a sunat clientul\n\n"
        "INTENTII DISPONIBILE: " + intentii_str + "\n\n"
        "EXEMPLE:\n" + exemple_text +
        "INTENTIE IDENTIFICATA:"
    )
```

A.5. Prompturile utilizate pentru clasificarea saisfacției – V1

```
def get_prompt(dialog):
    return (
        "Analizeaza urmatoarea conversatie telefonica si determina nivelul de satisfactie al clientului.\n\n"
        "Conversatie:\n" + dialog + "\n\n"
        "Care este satisfactia clientului la finalul conversatiei? "
        "Raspunde cu un singur cuvant: pozitiv, neutru sau negativ:"
    )
```

```
def get_prompt2(dialog):
    return (
        "Conversatie:\n" + dialog + "\n\n"
        "Nivelul de satisfactie al clientului la finalul acestei conversatii este:"
    )
```

A.6. Prompturile utilizate pentru clasificarea saisfacției – V2

```
def get_prompt(dialog):
    return (
        "Esti un expert in analiza satisfactiei clientilor in conversatii de call-center.\n"
        "Determina nivelul de satisfactie al clientului la finalul conversatiei de mai jos.\n\
    n"
        "DEFINITII:\n"
        "- pozitiv: problema rezolvata complet, clientul multumit si o exprima clar\n"
        "- neutru: problema rezolvata tehnic dar clientul nu prezinta nicio emotie\n"
        "- negativ: clientul pleaca frustrat sau nemultumit, chiar daca implicit\n"
        "  (ironie, resemnare, replici taioase, incheie brusc conversatia)\n\n"
        "REGULI:\n"
        "- Un singur comentariu negativ urmat de acceptare NU inseamna automat negativ\n"
        "- Uita-te la TONUL GENERAL si REZULTATUL FINAL al conversatiei\n"
        "- Frustrarea implicita conteaza: bine inteleg, ce sa fac, remarci ironice\n\n"
        "Conversatie:\n" + dialog + "\n\n"
        "Raspunde DOAR cu unul dintre cuvintele: pozitiv, neutru, negativ:"
    )
def get_prompt2(dialog):
    return (
        "Esti un expert in analiza satisfactiei clientilor in conversatii de call-center.\n"
        "Determina nivelul de satisfactie al clientului la finalul conversatiei de mai jos.\n\
    n"
        "DEFINITII DETALIATE:\n"
        "- pozitiv: clientul pleaca multumit si o exprima explicit\n"
        "  Expresii tipice: multumesc mult, excelent, exact ce aveam nevoie, sunteti promti\n"
        "- neutru: problema rezolvata dar clientul nu exprima nicio emotie\n"
        "  Expresii tipice: ok, am inteles, bine, la revedere fara caldura\n"
        "- negativ: clientul pleaca frustrat, chiar daca accepta situatia\n"
        "  Expresii tipice: ce sa fac, bine..., nu am de ales, iarasi aceeasi problema, ironie
    \n\n"
        "REGULA CRITICA neutru vs negativ:\n"
        "- Accepta solutia fara entuziasm dar FARA frustrare vizibila -> neutru\n"
        "- Accepta solutia cu resemnare, ironie sau nemultumire implicita -> negativ\n\n"
        "Conversatie:\n" + dialog + "\n\n"
        "Raspunde DOAR cu unul dintre cuvintele: pozitiv, neutru, negativ:"
    )
```

A.7. Prompturile utilizate pentru clasificarea saisfacției – V3

```
def get_prompt(dialog):

    exemple_text = ""
    for clasa, (dialog_ex, satisfactie_ex) in EXEMPLE_SCURTE.items():
        exemple_text += (
            "CONVERSATIE:\n" + dialog_ex + "\n"
            "SATISFACTIE: " + satisfactie_ex + "\n\n"
        )
    return (
        "Esti un expert in analiza satisfactiei clientilor in conversatii de call-center.\n\n"
        "CONVERSATIE:\n" + dialog + "\n\n"
        "SARCINA: Pe baza conversatiei de mai sus, determina nivelul de satisfactie al
    clientului.\n\n"
        "DEFINITII:\n"
        "- pozitiv: problema rezolvata complet, clientul multumit si o exprima clar\n"
```

```
        "- neutru: problema rezolvata tehnic dar clientul nu prezinta nicio emotie\n"
        "- negativ: clientul pleaca frustrat sau nemultumit, chiar daca implicit\n"
        "  (ironie, resemnare, replici taioase, incheie brusc conversatia)\n\n"
        "REGULI:\n"
        "1. Un singur comentariu negativ urmat de acceptare NU inseamna automat negativ\n"
        "2. Uita-te la TONUL GENERAL si REZULTATUL FINAL al conversatiei\n"
        "3. Frustrarea implicita conteaza: bine inteleg, ce sa fac, remarci ironice\n\n"
        "EXEMPLE:\n" + exemple_text +
        "SATISFACTIE IDENTIFICATA:"
    )
def get_prompt2(dialog):
    exemple_text = ""
    for clasa, (dialog_ex, satisfactie_ex) in EXEMPLE_LUNGI.items():
        exemple_text += (
            "CONVERSATIE:\n" + dialog_ex + "\n"
            "SATISFACTIE: " + satisfactie_ex + "\n\n"
        )
    return (
        "Esti un expert in analiza satisfactiei clientilor in conversatii de call-center.\n\n"
        "CONVERSATIE:\n" + dialog + "\n\n"
        "SARCINA: Pe baza conversatiei de mai sus, determina nivelul de satisfactie al
    clientului.\n\n"
        "DEFINITII:\n"
        "- pozitiv: problema rezolvata complet, clientul multumit si o exprima clar\n"
        "- neutru: problema rezolvata tehnic dar clientul nu prezinta nicio emotie\n"
        "- negativ: clientul pleaca frustrat sau nemultumit, chiar daca implicit\n"
        "  (ironie, resemnare, replici taioase, incheie brusc conversatia)\n\n"
        "REGULI:\n"
        "1. Un singur comentariu negativ urmat de acceptare NU inseamna automat negativ\n"
        "2. Uita-te la TONUL GENERAL si REZULTATUL FINAL al conversatiei\n"
        "3. Frustrarea implicita conteaza: bine inteleg, ce sa fac, remarci ironice\n\n"
        "EXEMPLE (acorda atentie diferentei dintre neutru si negativ):\n" + exemple_text +
        "SATISFACTIE IDENTIFICATA:"
    )
```

A.8. Prompturile utilizate pentru generarea rezumatului – V1

```
def get_prompt(dialog, satisfactie):
    return (
        "Rezuma urmatoarea conversatie telefonica in limba romana.\n\n"
        "Conversatie:\n" + dialog + "\n\n"
        "Rezumat:"
    )
def get_prompt2(dialog, satisfactie):
    return (
        "Conversatie:\n" + dialog + "\n\n"
        "Citeste conversatia telefonica si scrie un rezumat in limba romana.\n\n"
        "Rezumat:"
    )
```

A.9. Prompturile utilizate pentru generarea rezumatului – V2

```
def get_prompt(dialog, satisfactie):
    tip_info = get_tip_rezumat(satisfactie)
    return (
        "Esti un expert in sumarizarea conversatiilor telefonice din call-center.\n"
        "Genereaza un rezumat de tip " + tip_info["tip"] + " al conversatiei de mai jos.\n\n"
        "CERINTE:\n"
        "- " + tip_info["propozitii"] + ", " + str(tip_info["min_cuv"]) + "-" + str(tip_info["
    max_cuv"]) + " cuvinte\n"
        "- Scrie in limba romana\n"
        "- Mentioneaza problema principala si rezultatul final\n"
```

```
        "- Nu adauga informatii care nu apar in conversatie\n\n"
        "Conversatie:\n" + dialog + "\n\n"
        "Rezumat " + tip_info["tip"] + ":"
    )
def get_prompt2(dialog, satisfactie):
    tip_info = get_tip_rezumat(satisfactie)
    return (
        "Esti un expert in sumarizarea conversatiilor telefonice din call-center.\n"
        "Genereaza un rezumat de tip " + tip_info["tip"] + " al conversatiei de mai jos.\n\n"
        "CERINTE:\n"
        "- Lungime: " + str(tip_info["min_cuv"]) + "-" + str(tip_info["max_cuv"]) + " cuvinte
    (" + tip_info["propozitii"] + ")\n"
        "- Scrie in limba romana si nu adauga informatii care nu apar in conversatie\n"
        "- Prezinta evenimentele in ordine cronologica"
        "- Descrie mai intai motivul apelului clientului, apoi actiunile intreprinse de
    operator si, la final, rezultatul obtinut\n\n"
        "Conversatie:\n" + dialog + "\n\n"
        "Rezumat " + tip_info["tip"] + ":"
    )
```

A.10. Prompturile utilizate pentru generarea rezumatului – V3

```
def get_prompt(dialog, satisfactie):
    tip_info = get_tip_rezumat(satisfactie)

    exemple = {
        "SCURT": (
            "CLIENT: Am pierdut cardul, il vreau blocat.\nOPERATOR: L-am blocat, va trimitem
    altul in 3 zile.",
            "Clientul a sunat pentru a bloca un card pierdut. Operatorul a blocat cardul si a
    initiat emiterea unuia nou."
        ),
        "MEDIU": (
            "CLIENT: Comanda mea nu a sosit.\nOPERATOR: Verificam, a fost o intarziere la
    curier. Va ajunge maine.\nCLIENT: Ok.",
            "Clientul a contactat call-center-ul pentru a reclama o comanda nelivrata la
    termen. Operatorul a verificat situatia si a informat clientul ca livrarea a intarziat din
     cauza curierului. Comanda urmeaza sa fie livrata a doua zi. Clientul a acceptat solutia."
        ),
        "LUNG": (
            "CLIENT: Am sunat de trei ori pentru aceeasi problema cu factura.\nOPERATOR: Imi
    pare rau, investigam.\nCLIENT: Bine, astept.",
            "Clientul a contactat call-center-ul pentru a treia oara in legatura cu aceeasi
    problema de facturare nerezolvata. Clientul si-a exprimat nemultumirea fata de lipsa unei
    solutii dupa contactele anterioare. Operatorul si-a cerut scuze si a initiat o
    investigatie interna. Clientul a acceptat sa astepte, exprimand insa o frustrare evidenta
    fata de procesul de rezolvare. Problema ramane deschisa si necesita urmarire."
        ),
    }

    tip = tip_info["tip"]
    dialog_ex, rezumat_ex = exemple[tip]
    exemplu_text = (
        "CONVERSATIE:\n" + dialog_ex + "\n"
        "REZUMAT " + tip + ":\n" + rezumat_ex + "\n\n"
    )

    return (
        "Esti un expert in sumarizarea conversatiilor telefonice din call-center.\n\n"
        "CONVERSATIE:\n" + dialog + "\n\n"
        "SARCINA: Genereaza un rezumat de tip " + tip + " al conversatiei de mai sus.\n\n"
        "CERINTE:\n"
        "- Lungime: " + str(tip_info["min_cuv"]) + "-" + str(tip_info["max_cuv"]) + " cuvinte
    (" + tip_info["propozitii"] + ")\n"
        "- Limba: romana\n"
        "- Mentioneaza problema principala si rezultatul final\n"
        "- Nu adauga informatii care nu apar in conversatie\n\n"
        "EXEMPLU:\n" + exemplu_text +
```

```
        "REZUMAT " + tip + ":"
    )
def get_prompt2(dialog, satisfactie):
    tip_info = get_tip_rezumat(satisfactie)

    exemple = {
        "SCURT": (
            "CLIENT: Am pierdut cardul, il vreau blocat.\nOPERATOR: L-am blocat, va trimitem altul in 3 zile.",
            "Clientul a solicitat blocarea unui card pierdut. Operatorul a rezolvat imediat solicitarea si a initiat emiterea unui card de inlocuire."
        ),
        "MEDIU": (
            "CLIENT: Comanda mea nu a sosit.\nOPERATOR: Verificam, a fost o intarziere la curier. Va ajunge maine.\nCLIENT: Ok.",
            "Motivul apelului: clientul a reclamat o comanda nelivrata la termen. Actiuni operator: verificarea statusului comenzii si identificarea intarzierii la curier. Rezultat: livrarea reprogramata pentru a doua zi, clientul a acceptat solutia."
        ),
        "LUNG": (
            "CLIENT: Am sunat de trei ori pentru aceeasi problema cu factura.\nOPERATOR: Imi pare rau, investigam.\nCLIENT: Bine, astept.",
            "Motivul apelului: clientul a contactat call-center-ul pentru a treia oara in legatura cu o eroare de facturare persistenta. Context: problema nu a fost rezolvata in urma contactelor anterioare, generand frustrare acumulata. Actiuni operator: initierea unei investigatii interne si solicitarea de timp suplimentar. Reactia clientului: acceptare cu resemnare evidenta, fara satisfactie fata de procesul de rezolvare. Rezultat: problema ramane deschisa, necesita monitorizare si urmarire prioritara."
        ),
    }

    tip = tip_info["tip"]
    dialog_ex, rezumat_ex = exemple[tip]
    exemplu_text = (
        "CONVERSATIE:\n" + dialog_ex + "\n"
        "REZUMAT " + tip + ":\n" + rezumat_ex + "\n\n"
    )

    return (
        "Esti un expert in sumarizarea conversatiilor telefonice din call-center.\n\n"
        "CONVERSATIE:\n" + dialog + "\n\n"
        "SARCINA: Genereaza un rezumat de tip " + tip + " al conversatiei de mai sus.\n\n"
        "CERINTE:\n"
        "- Lungime: " + str(tip_info["min_cuv"]) + "-" + str(tip_info["max_cuv"]) + " cuvinte (" + tip_info["propozitii"] + ")\n"
        "- Limba: romana\n"
        "- Mentioneaza motivul apelului, actiunile operatorului si rezultatul final\n"
        "- Nu adauga informatii care nu apar in conversatie\n\n"
        "EXEMPLU:\n" + exemplu_text +
        "REZUMAT " + tip + ":"
    )
```

B. APENDIX 2

Codul complet al implementarii poate fi accesat aici