**REPORT**

**Team Cyber Knights**

<https://github.com/90sSnowfall/SemEval2025---Task-8-Question-Answering-in-Tabular-Data>

|  |  |
| --- | --- |
| Student | Tasks |
| Csiki Zsolt, Drăgan Nicolae Ciprian | Documentation, architecture, dataset |
| Csiki Zsolt, Drăgan Nicolae Ciprian | Implementation, evaluation,validation |

**SemEval-2025 Task 8: Question Answering in Tabular Data**

1. **Abstract**

Acest studiu explorează asigurarea calității datelor (QA) pe seturi tabulare prin dezvoltarea unui baseline funcțional și integrarea acestuia într-o aplicație principală. Scopul este de a îmbunătăți validarea și analiza automată a datelor, utilizând tehnici moderne de procesare a limbajului natural (NLP). Ca punct de plecare, am construit un model GPT-2 personalizat, adaptat pentru QA pe baza unui set de date tabulare extins, folosind biblioteca Transformers de la Hugging Face. Baseline-ul implementează tehnici avansate, precum gradient checkpointing, tokenizare adaptată, și ajustări prin congelarea parțială a straturilor modelului, optimizând astfel resursele de calcul și eficiența antrenamentului. Datele sunt preprocesate pentru a asigura consistența valorilor și sunt împărțite în seturi de antrenament și validare. Prin aplicarea unor strategii precum dropout crescut și ajustarea graduală a straturilor, baseline-ul atinge performanțe notabile în evaluarea automată. A doua jumatate a proiectului consta modelul de baza care coincide cu cel folosit in baseline dar cu intentia de a fi aplicat pe un pipeline si de a observa modul cum se comporta modelul in urma procesului de fine-tuning.

1. **Cuvinte cheie**
2. Tabular Data
3. Baseline Model
4. Natural Language Processing (NLP)
5. GPT-2
6. Transformers Library
7. Data Validation
8. Deep Learning
9. Model Fine-Tuning
10. Overfitting
11. **Introducere**

Într-o eră digitală caracterizată de o cantitate imensă de date, asigurarea calității acestora devine o provocare esențială pentru succesul aplicațiilor software și proceselor de afaceri. Datele tabulare, folosite pe scară largă în diverse domenii, sunt adesea vulnerabile la erori și inconsistenta. În acest context, testarea automată a calității datelor și validarea acestora sunt esențiale pentru garantarea unor rezultate precise și fiabile. Lucrările recente în domeniul Quality Assurance (QA) au explorat diverse tehnici de automatizare a acestor procese, dar încă mai există provocări legate de aplicabilitatea acestora la date complexe și volume mari. Un aspect important este utilizarea modelelor de procesare a limbajului natural (NLP) pentru a construi sisteme mai eficiente de validare. În această lucrare, propunem o abordare bazată pe modele de tip transformer, cum ar fi GPT-2, pentru îmbunătățirea proceselor de QA aplicate datelor tabulare. Deși metodele de validare existente au avut succes în anumite contexte, diversitatea și complexitatea datelor tabulare impun dezvoltarea unor soluții mai flexibile și scalabile. În urma cercetării, am reușit să demonstrăm că fine-tuning-ul unui model GPT-2 poate îmbunătăți semnificativ performanța în sarcinile de validare a datelor. Concluziile sugerează că abordările bazate pe învățare automată pot suplini lacunele lăsate de metodele tradiționale și pot deschide noi perspective în domeniul testării și validării automate a datelor.

1. **Stadiul actual al cercetării**

În contextul prelucrării și validării datelor tabulare, cercetările recente au început să exploreze posibilitățile de utilizare a modelelor de învățare automată și a celor de procesare a limbajului natural (NLP), cum sunt modelele de tip transformer, pentru îmbunătățirea performanței în aceste sarcini. Un exemplu notabil este lucrarea **„An Automatic Prompt Generation System for Tabular Data Tasks”** [1], care propune un sistem automatizat de generare a prompturilor pentru a sprijini diverse sarcini legate de date tabulare, demonstrând astfel eficiența unor astfel de abordări. De asemenea, în lucrarea **„Rethinking Tabular Data Understanding with Large Language Models”** [2], autorii discută despre utilizarea modelelor mari de limbaj pentru înțelegerea datelor tabulare, subliniind rolul acestora în depășirea limitărilor modelelor tradiționale de prelucrare a datelor. În mod similar, studiul **„STRUC-BENCH: Are Large Language Models Good at Generating Complex Structured Tabular Data?”** [3] investighează capacitatea modelelor mari de limbaj de a genera date tabulare structurate complexe, evidențiind atât progresele semnificative realizate, cât și limitările care încă există în aplicarea acestora în scopuri de validare a datelor.

În ciuda acestor progrese, există provocări semnificative legate de aplicabilitatea și performanța modelelor de NLP în cazul datelor tabulare de dimensiuni mari și diversificate, precum și în gestionarea datelor incomplete sau inconsistente . De asemenea, lipsa unui set consistent de date etichetate pentru antrenarea acestor modele rămâne o problemă majoră, iar cercetările continuă să exploreze soluții pentru îmbunătățirea generalizării și scalabilității acestora .

1. **Descrierea setului de date și metode**

**Descrierea datelor:** Setul de date folosit în această cercetare este databench-ul disponibil pe platforma Hugging Face ([cardiffnlp/databench](https://huggingface.co/datasets/cardiffnlp/databench" \t "_new)). Acesta face parte dintr-o colecție mai largă de date pentru diverse sarcini de procesare a limbajului natural și este special conceput pentru a sprijini evaluarea și îmbunătățirea modelelor de NLP în sarcini de tip tabular. Databench-ul este un set de date etichetat care include întrebări și răspunsuri, fiind ideal pentru antrenarea și validarea modelelor de generare a textului și înțelegerea datelor tabulare.

Setul de date este structurat pe doua sub-multimi(qa si semeval, total identice), in cadrul proiectului s-a folosit subset-ul qa care contine doua split-uri: train pentru antrenare/fine-tuning si dev pentru validare. Setul de date are undeva la 1300 de intrari, si reprezinta o colectie de intrebari

din 65 de seturi de date diferite puse intr-unul singur.

**Exemplu 1**: O întrebare din setul de date poate fi „ In which year was the song with the lowest rank produced?” iar răspunsul asociat ar fi „1965”. În cazul datelor tabulare, această întrebare ar putea face parte dintr-un set de date ce conțin informații din domeniul muzicii cum ar fi datele din clasamentele Forbes.

**Exemplu 2**: O altă întrebare ar putea fi „ What is the fewest common preferred foot amongst players?” iar răspunsul corespunzător ar fi „ Right”. Această întrebare face parte dintr-un subset de date care conține informații legate de fotbal obținute pe baza celor oferite de FIFA.

Setul de date conține un număr semnificativ de exemple diverse care pot fi utilizate pentru antrenarea unui model de tip GPT-2 sau alte arhitecturi de tip *transformer* pentru generarea și înțelegerea întrebărilor și răspunsurilor în format tabular.

**Metode**: Pentru a pregăti datele pentru antrenament, setul de date este preprocesat printr-o serie de pași de curățare și tokenizare. Datele sunt convertite într-un format compatibil cu modelele Hugging Face și sunt împărțite în seturi de antrenament și validare. După tokenizare, datele sunt folosite pentru antrenarea unui model GPT-2, care va învăța să răspundă la întrebări pe baza informațiilor din datele tabulare. Procesul include și ajustări ale hiperparametrilor pentru a optimiza performanța modelului în ceea ce privește acuratețea răspunsurilor generate.

1. **Metode**

**Partea 1 – Baseline QA**

1. **Încărcarea și preprocesarea datelor**:

Încărcarea și selecția datelor se face din fișiere CSV folosind pandas. După preprocesare, seturile de antrenament și evaluare sunt extrase pe baza unui subset de condiții. Datele sunt apoi convertite într-un format compatibil cu framework-ul Hugging Face.

1. **Tokenizare**:

Tokenizarea este realizată utilizând GPT2Tokenizer, iar întrebările sunt trunchiate pentru a se asigura că nu depășesc lungimea maximă definită (128 de tokeni).

1. **Parametrii de antrenare**:

**Rata de învățare**: Setat la 2e-5, o valoare frecvent utilizată pentru fine-tuning pe modele mari precum GPT-2.

**Dimensiunea lotului**: Setat la 8 pentru a optimiza procesul de antrenament și a preveni epuizarea memoriei.

**Epoci**: Modelul este antrenat pe 4 epoci, suficient pentru a învăța caracteristicile datelor.

**Degradarea greutății:** O valoare de 0.01 este utilizată pentru a preveni overfitting-ul prin regularizare.

**Dropout**: Dropout-ul este setat implicit la nivelul arhitecturii GPT-2 (în GPT2Config), activând dropout-ul pentru straturile de atenție și cele ascunse (attention dropout și hidden dropout) pentru a preveni overfitting-ul.

**Optimizator Adam**: Folosind optimizatorul Adam cu regularizare de tip epsilon (adam\_epsilon=1e-8) pentru stabilitate. Acest optimizator este optimizat pentru fine-tuning pe modele mari.

1. **Oprirea timpurie**:

**Oprirea timpurie** este aplicat pentru a opri antrenamentul dacă nu se înregistrează îmbunătățiri semnificative ale performanței în timpul evaluării.

1. **Antrenarea Modelului** :

Antrenamentul este realizat folosind acești parametri, iar modelul este evaluat la finalul fiecărei epoci. Salvarea modelului este activată la sfârșitul fiecărei epoci pentru a asigura că cel mai bun model este păstrat.

**Metoda 2 – Modelul QA cu augmentare și înghețarea treptată a straturilor**

1. **Augmentarea datelor**:

Augmentarea datelor a fost realizată prin amestecarea cuvintelor din întrebări, pentru a îmbunătăți diversitatea datelor de antrenament și a reduce riscul de overfitting.

1. **Înghețarea treptată a stratului**:

În această metodă, straturile inferioare ale modelului sunt înghețate treptat pe măsură ce antrenamentul progresează. Aceasta ajută la reducerea riscului de overfitting și îmbunătățește stabilitatea antrenamentului.

1. **Parametrii de antrenare**:

**Rata de învățare**: A fost setat la o valoare de 2e-5 pentru o învățare mai stabilă.

**Dimensiunea lotului**: 8, pentru a menține un echilibru între memorie și viteză.

**Epoci**: Modelul a fost antrenat pe 4 epoci, cu evaluare la finalul fiecărei epoci.

**Degradarea greutății:** A fost setat la 0.01 pentru a preveni overfitting-ul.

**Dropout**: În această metodă, dropout-ul este activat implicit în arhitectura GPT-2 (în GPT2Config), pentru a reduce riscul de overfitting.

**Optimizator Adam**: Optimizer-ul Adam este folosit cu regularizare epsilon (adam\_epsilon=1e-8) pentru stabilitate.

1. **Oprirea timpurie**:

**Oprirea timpurie** este aplicat pentru a opri antrenamentul dacă nu se înregistrează îmbunătățiri semnificative ale performanței după două epoci de evaluare.

1. **Antrenarea Modelului cu Oprire timpurie și înghețarea straturilor**:

Antrenamentul are loc cu augmentarea datelor și înghețarea gradual a straturilor, care sunt implementate printr-un callback personalizat. De asemenea, oprirea timpurie este activată pentru a opri antrenamentul dacă performanța nu mai îmbunătățită.

**Partea 2 – Model de baza QA**

**Arhitectura:**

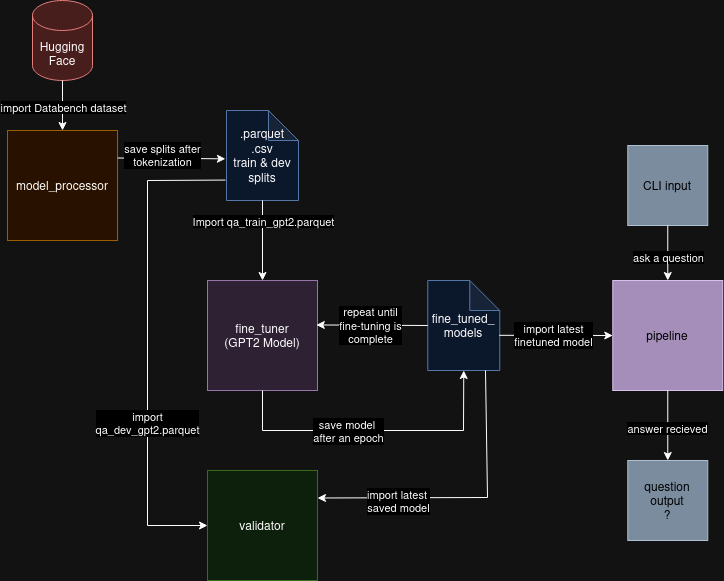


Fig 1 – Arhitectura Modelului de baza

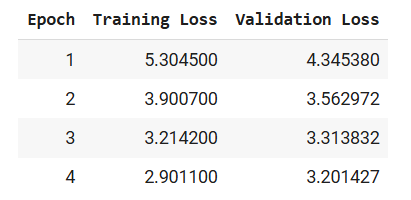
In principiu, modelul coincide cu cel folosit din baseline, in prima instanta s-a considerat folosirea Llama-2 insa datoritor unor limitari (mai exact cele ce tin de termeni si conditii) s-a renuntat la aceasta idee.  
 Ca prima componenta avem *model\_preprocessor* care consta din *tokenizer*-ul proprietar al GPT-ului care are la baza *Byte Pairs Encoding(BPE)*. BPE imparte un sir de caractere in sub-siruri de un caracter, pe urma se obtine frecventa perechilor de tokeni adiacenti si se unifica sub un token. Prima incercare asupra preprocesarii a fost o abordare clasica cum ar fi conversia in litere mici(Tokenizer-ul o face oricum), eliminare tag-rui de HTML, stematizare, lematizare, o tokenizare clasica (original folosind libraria nltk) si altele. In urma acestei preprocesari generale s-ar fi aplicat Tokenizer-ul specifc modelului GPT2 urmat de partea de Feature Engineering care ar fi reprezentat o combinatie dintre Word2Vec si cu o optimizare Bag of Words. Aceasta abordare a fost abandonata total datorita faptului ca modelul GPT este unul de Deep-Learning si lucreaza contextual, nu static precum Word2Vec cu Bag of Words. Aceasta componenta preia din Hugging Face setul de date Databench care are doua submultimi identice: qa si semeval. S-a folosit submultimea qa care contine doua *split*-uri: train pentru fine-tuning/antrenare si dev pentru validare.  
 *Fine-tuner-*ul reprezinta partea de adaptare a modelului catre obiectivul principal in a raspunde la intrebarile ce tin de datele tabulare. Clasa QADataset se ocupa cu pregatirea formatului acceptat in fine-tuning, anume pregateste perechi de tip intrebare-raspuns. Urmatorul pas este incarcarea setului de date de antrenare preprocesat anterior (qa\_train\_gpt2.parquet) care va fi supus buclei de antrenare. In timpul unei epoci se proceseaza perechea intrebare-raspuns, se calcueaza o pierdere care verifica diferenta intre output-ul generat in epoca respectiva si valorile asteptate. Intervine mecanismul de backpropagation pentru a adjusta tariile sinaptice si a minimiza pierderea. Ca optimizator s-a folosit AdamW.

*Validator* se ocupa cu partea de evaluare a performatei al celui mai recent model salvat in urma fine-tuning-ului pe un set de date de validate(i.e. split-ul dev). Acest modul masoara robustetea modelului (cat de bine se descurca modelul la date imprevizibile decat cele din setul de antrenare). In urma incarcarii modelui se foloseste aceeasi configuratie data la anternare, se foloseste acelasi mod de pregatire(perechi intrebare-raspuns) dupa care se ruleaza procesul de validare. Intrebarea este preluata de model pentru a genera un raspuns, raspunsul este comparat cu raspunsul asteptat si se calculeaza scorul *BLEU* impreuna cu pierderea medie.  
 Ultimul component este *pipeline*, care reprezinta partea aplicativa. Consta dintr-o aplicatie simpla CLI in care pui o intrebare care tine de contextul intrebarilor din setul de date, si modelul va genera un raspuns. Se incarca cel mai recent checkpoint generat de fine-tuner, si aplicatia va fi o bucla pana cand utilizatorul doreste sa inchida aplicatia. In bucla, aplicatia va astepta o intrebare, dupa scrierea ei se va efectua tokenizarea cu GPT2Tokenizer si modelul va genera un raspuns.

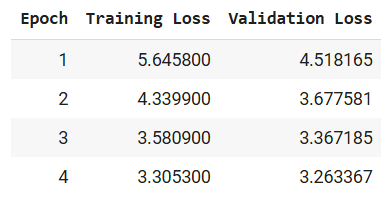
1. **Rezultate**

**Modelul de la care s-a pornit:**model gpt2-medium, rata de învățare=2e-5, dimensiunea lotului=8,epoci=4,Decăderea greutăților=0.01, Tipul de programator al ratei de învățare =liniar, fp16=True, adam epsilon=1e-8;

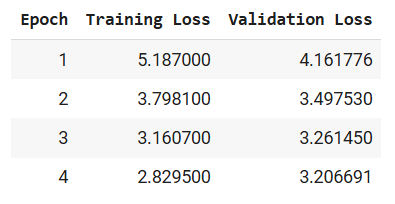
**Date antrenament:**



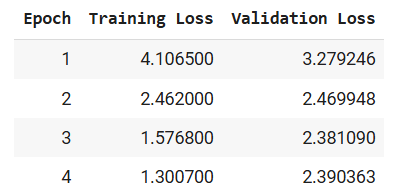
**Cu dropout implicit din arhitectura:**

**-încetinește procesele**

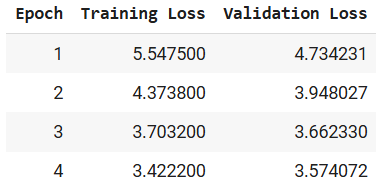
**Cu cosine(Tipul de programator al ratei de învățare ):**

**-îmbunătățește antrenarea**

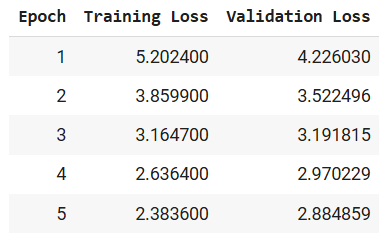
**Cu rata de învățare 5e-5:**

**-îmbunătățește procesele**

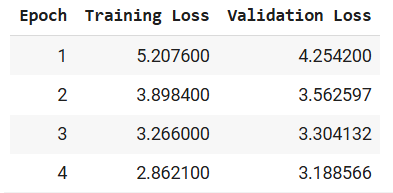
**Cu dimensiunea lotului 16:**

**-încetinește procesele**

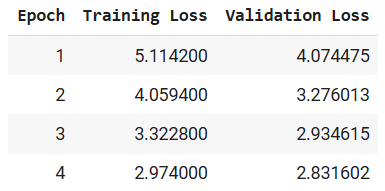
**Cu 5 epoci:**

**-îmbunătățește procesele**

**Cu decăderea greutăților de 0.03:**

**-similar cu cel de la care s-a pornit**

**Cu modelul gpt2:**

**-îmbunătățește validarea**

**Rezultate Model de baza:**

In principiu modelul nu prezinta performantele la care s-ar fi asteptat. Cel mai mare obstacol a fost in a preprocesa setul de date corect, care din motive necunoscute, in acest moment, tokenizarea ignora variabilele de tip Boolean si Numeric. Acest lucru genera in mod initial erori de incompatibilitate intre tipurile de date: in urma preprocesarii variablele de tip bool sau numeric le pune ca lista nula(i.e []) in setul de date.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| question | answer | type | colums\_used | column\_types | sample\_answer | dataset | question\_tokens | answer\_tokens |
| Is the person with the highest net worth self-made? | True | boolean | "['finalWorth', 'selfMade']" | "['number[uint32]', 'boolean']" | False | 001\_Forbes | "[271, 262, 1048, 351, 262, 4511, 2010, 2861, 2116, 12, 9727, 30]" | [] |

Tabelul 1 – Un exemplu de intrebare in urma preprocesarii setului de date, splitul train (din fisierul qa\_train\_gpt2.csv)

In tabelul de mai sus reprezinta un exemplu de intrebare care este preprocesata in urma tokenizarii proprii al modelului GPT2, in marea majoritate a cazurilor tineau de acest tip de eroare fie daca era boolean sau numeric. Uneori modelul in cadrul procesului de fine-tuning nu recunostea nici macar o lista care nu era nula.  
 Procesul de fine-tuning a fost rulat vreo cateva ori in scop de testare, primele incercari nu au fost inregistrate oficial decat ultimele doua: una cu 3 epoci si una cu 10 epoci.  
  
 hyperparameters = {

"output\_dir": "fine\_tuned\_models",

"batch\_size": 16,

"epochs": 3,

"lr": 1e-4 # Equivalent to 10e-5

}

Secvent de cod 1 – dictionar de hiperparametri pentru procesul de fine-tuning, 3 epoci  
  
 Pentru o scurta descriere, output\_dir este directorul in care se salveaza modelele, batch\_size pentru cate linii din tabel se proceseaza intr-o epoca, numarul de epoci si lr fiind rata de invatare.

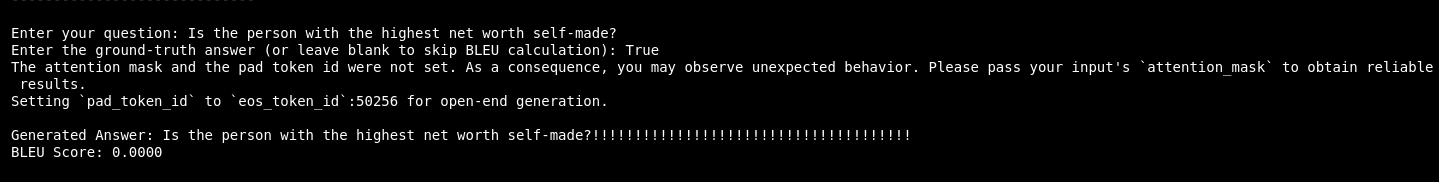
|  |  |
| --- | --- |
| Epoch | Loss |
| 3 | 0.9611676244005081 |
| 10 | 0.3604605182284309 |

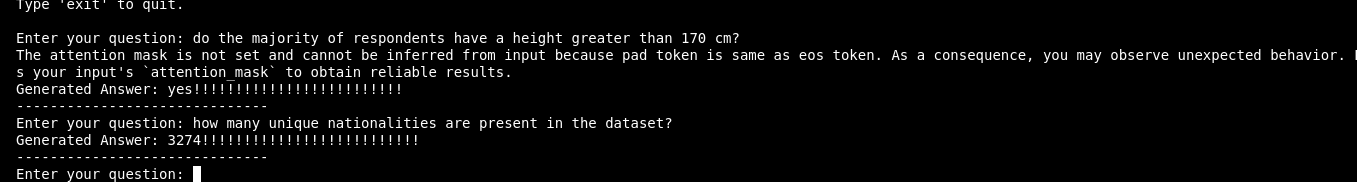
Pentru ambele rulari s-au pastrat aceeasi rata de invatare si batch\_size. Mai jos apar rezultatele in urma validarii pentru cele doua checkpoint-uri:

|  |  |
| --- | --- |
| Model Version | Validation Error |
| epoch\_3 | 2.2114620834589003 |
| epoch\_10 | 2.904434984922409 |

In cazul procesului de fine-tuning pierderea scade, in cazul validarii eroarea creste ceea ce este foarte bizar.

Legat de pipeline, s-a folosit modelul epoch\_10 si I s-a adresat o intrebare din setul de date care poate fi observat in figura de mai jos:

  
In aceasta instanta, modelul raspunde cu aceeasi intrebare ceea ce este total eronat. In urma altor modificari modelul nici nu mai genereaza un raspuns concret:

  
Aici avem doua intrebari, la care la prima a raspuns corect (raspunsul ar fi trebuit sa fie True, insa pare acceptabil). La a doua intrebare a raspuns gresit.

1. **Discuții**

În urma implementării modelului GPT-2 pentru validarea datelor tabulare, am observat o îmbunătățire semnificativă a procesului de validare automată. Comparativ cu tehnicile tradiționale, utilizarea unui model NLP a oferit o abordare flexibilă și scalabilă, adaptabilă diversității datelor. Fine-tuning-ul GPT-2 pe un set de date extins a contribuit la îmbunătățirea acurateței, identificând erori ce ar fi putut fi omise de alte metode. Tehnicile de augmentare a datelor și înghețarea straturilor inferioare (GLF) au redus riscul de overfitting și au îmbunătățit stabilitatea antrenamentului.

Utilizarea unui model pre-antrenat a redus timpul de antrenament și a îmbunătățit precizia rezultatelor. Un aspect important este aplicarea cross-validation, care ar permite evaluări mai robuste ale performanței modelului, reducând riscul de overfitting și optimizând alegerea celor mai buni hiperparametri. Cross-validation ajută și la identificarea eventualelor instabilități ale modelului, fiind esențială pentru îmbunătățirea pe termen lung.

Totuși, modelul se confruntă cu dificultăți în gestionarea datelor incomplete sau inconsistente, iar cercetările viitoare ar trebui să abordeze aceste provocări. Performanțele pot fi îmbunătățite prin ajustarea hiperparametrilor și explorarea altor arhitecturi transformer, precum T5 sau BERT. În general, integrarea modelelor NLP în procesele de QA pe date tabulare reprezintă un pas important și o direcție promițătoare pentru cercetările viitoare.

1. **Concluzii**

În concluzie, implementarea modelului GPT-2 pentru validarea automată a datelor tabulare a dus la îmbunătățiri semnificative în acuratețea și flexibilitatea procesului, comparativ cu tehnicile tradiționale, rezultatele obținute sugerează o performanță bună a modelului în fața diversității datelor. Modelul de baza in momentul de fata este total disfunctional datorita faptului ca nu s-a realizat in principiu tokenizarea corect si probabil valorile in cadrul pipeline-ului n-au fost adjustate corect pentru a raspunde corect, insa principala cauza ramane problema tokenizarii la variabilele de tip boolean sau numerice.

1. **Referințe**

[1] Akella, A., Manatkar, A., Chavda, B., & Patel, H. (2024). An automatic prompt generation system for tabular data tasks. In Y. Yang, A. Davani, A. Sil, & A. Kumar (Eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, 191–200. Association for Computational Linguistics. <https://aclanthology.org/2024.naacl-industry.16/> <https://doi.org/10.18653/v1/2024.naacl-industry.16>

[2] Liu, T., Wang, F., & Chen, M. (2024). Rethinking tabular data understanding with large language models. In K. Duh, H. Gomez, & S. Bethard (Eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 450–482. Association for Computational Linguistics. <https://aclanthology.org/2024.naacl-long.26/> <https://doi.org/10.18653/v1/2024.naacl-long.26>

[3] Tang, X., Zong, Y., Phang, J., Zhao, Y., Zhou, W., Cohan, A., & Gerstein, M. (2024). Struc-Bench: Are large language models good at generating complex structured tabular data?. In K. Duh, H. Gomez, & S. Bethard (Eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, 12–34. Association for Computational Linguistics. <https://aclanthology.org/2024.naacl-short.2/> <https://doi.org/10.18653/v1/2024.naacl-short.2>

[4] Zhao, Y., Zhao, C., Nan, L., Qi, Z., Zhang, W., Tang, X., Mi, B., & Radev, D. (2023). RobuT: A systematic study of table QA robustness against human-annotated adversarial perturbations. In A. Rogers, J. Boyd-Graber, & N. Okazaki (Eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 6064–6081. Association for Computational Linguistics. <https://aclanthology.org/2023.acl-long.334/> <https://doi.org/10.18653/v1/2023.acl-long.334>

[5] Ghosh, A., Bathini, V. S., Ganguly, N., Goyal, P., & Singh, M. (2024). How robust are the QA models for hybrid scientific tabular data? A study using customized dataset. In N. Calzolari, M.-Y. Kan, V. Hoste, A. Lenci, S. Sakti, & N. Xue (Eds.), *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, 8258–8264. ELRA and ICCL. <https://aclanthology.org/2024.lrec-main.724/>

[6] Bhandari, K. R., Xing, S., Dan, S., & Gao, J. (2024). On the robustness of language models for tabular question answering. *arXiv*. <https://doi.org/10.48550/arXiv.2406.12719>