# 9. Rețea neuronală Transformer

Rețelele convoluționale
Convoluția
Operația de pooling
Rețeaua neurală convoluțională

Rețelele neuronale bazate pe arhitecturi de tip Transformer [6], popularizate prin aplicații precum ChatGPT, sunt folosite cu precădere în procesarea limbajului natural (NLP - Natural Language Processing), fiind bazate pe un mecanism de atenție capabil să extragă dependințe globale dintre intrări și ieșiri. Transformer este un model auto-regresiv ce generează un element de ieșire $y_i$ dintr-o secvență de intrare $(x_1, ..., x_n)$, unde elementul $y_i$, odata generat, devine parte a secvenței de intrare utilizată în generarea următorului element.

## 9.1 Modelarea Vocabularului

Datele de intrare sunt mai întâi prelucrate de coduri ce transformă limbajul natural în secvențe numerice ce pot fi interpretate de o rețea neuronală. Fiecare element unic din cod poartă denumirea de "token" și este folosit pentru a descrie elementele vocabularului folosit în aplicație (e.g. vocabularul limbii engleze sau vocabularul limbii române). În funcțe de modul în care sunt definite token-urile, aceste coduri sunt împărțite în:

1. coduri ce atribuie o valoare numerică fiecărui caracter din vocabular, așa cum este ilustrat în Figura 9.1(a),
2. coduri ce descriu printr-o valoare unică subgrupuri de caractere (v. Figura 9.1(b)),
3. coduri în care fiecare cuvânt este reprezentat printr-o valoare numerică, așa cum a fost definit și vocabularul din cursul 4 (v. Figura 9.1(c)).

În continuare, pentru claritatea expunerii, vom folosi primul cod, unde fiecare caracter va primi un element identificator unic. Vocabularul în acest caz este definit de următoarele 73 de litere, cifre și caractere speciale:

!'+,-.0123456789:;?ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz

Secvența de caractere de mai sus este codificată prin 73 de numere întregi $(x_1, ..., x_{73})$.

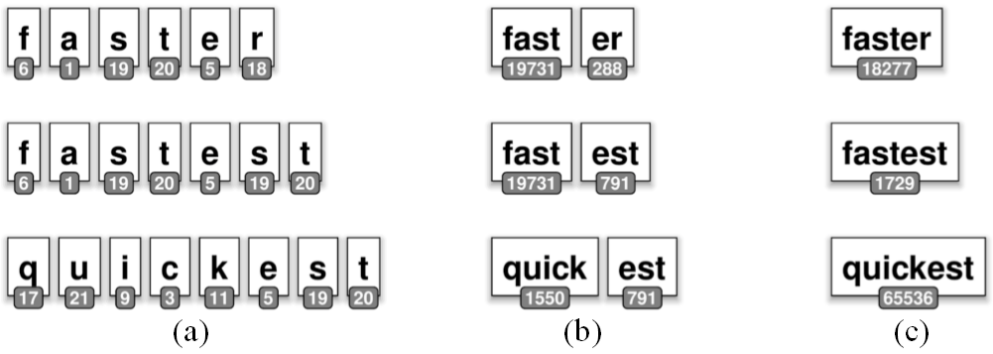


Fig. 9.1 Metode de codare a vocabularului în tokens.

## 9.2 Model Bigram

Modelul Bigram este unul dintre cele mai simple modele, ce învață să prezică următorul caracter (output token), doar dintr-un singur caracter de intrare (input token). Diagrama bloc a modelului este ilustrată în Figura 9.2. În acest model, vom prezice următorul caracter probabil dându-se un singur caracter de intrare.

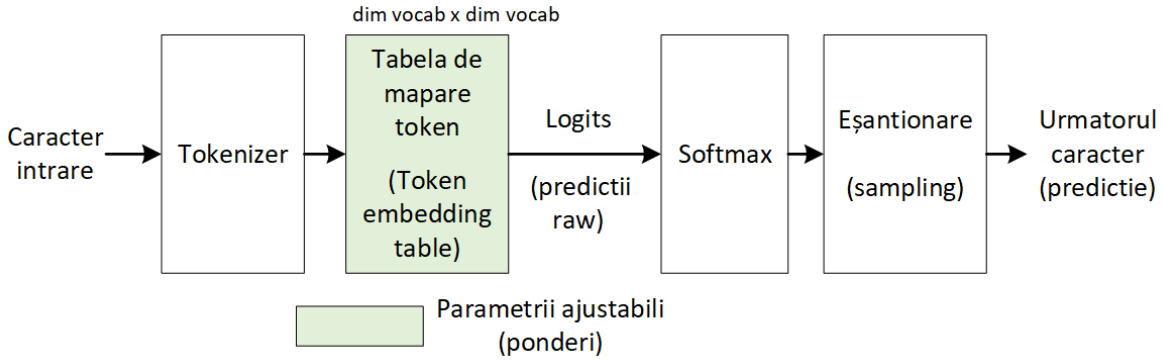


Fig. 9.2 Arhitectura modelului Bigram. Blocurile reprezentate cu verde conțin parametrii antrenabili (ponderi).

Caracterul de intrare este codificat utilizând modulul Tokenizer. Predicția următorului caracter este facută pe baza unei tabele de mapare (Embeddings Table) cu un număr egal de linii și coloane, unde liniile reprezintă caracterul de intrare, iar coloanele probabilitatea de apariție a urmatorului caracter din secvență, ce urmează a fi prezis. Pentru a putea face predicția, valorile din tabela de mapare sunt scalate în așa fel încât suma valorilor fiecărei linii să fie egală cu 1. Acest lucru este obținut prin aplicarea funcției Softmax pe fiecare linie a tabelei. Figura 9.3 reprezintă o tabelă de mapare pentru caractere minuscule, unde valorea fiecărei celule a fost scalată utilizându-se funcția Softmax.

Singurii parametrii antrenabili ai modelului din Figura 9.2 sunt valorile celulelor din tabela de mapare. Ajustarea parametrilor se face prin prelucrarea perechilor de caractere din datele de antrenare, folosind funcția de cost introdusă in cursul despre Regresie Logistică.

Odată antrenat, predicția următorului caracter se face prin eșantionarea liniei din tabela de mapare crespunzătoare caracterului de intrare. Această eșantionare se face pe baza

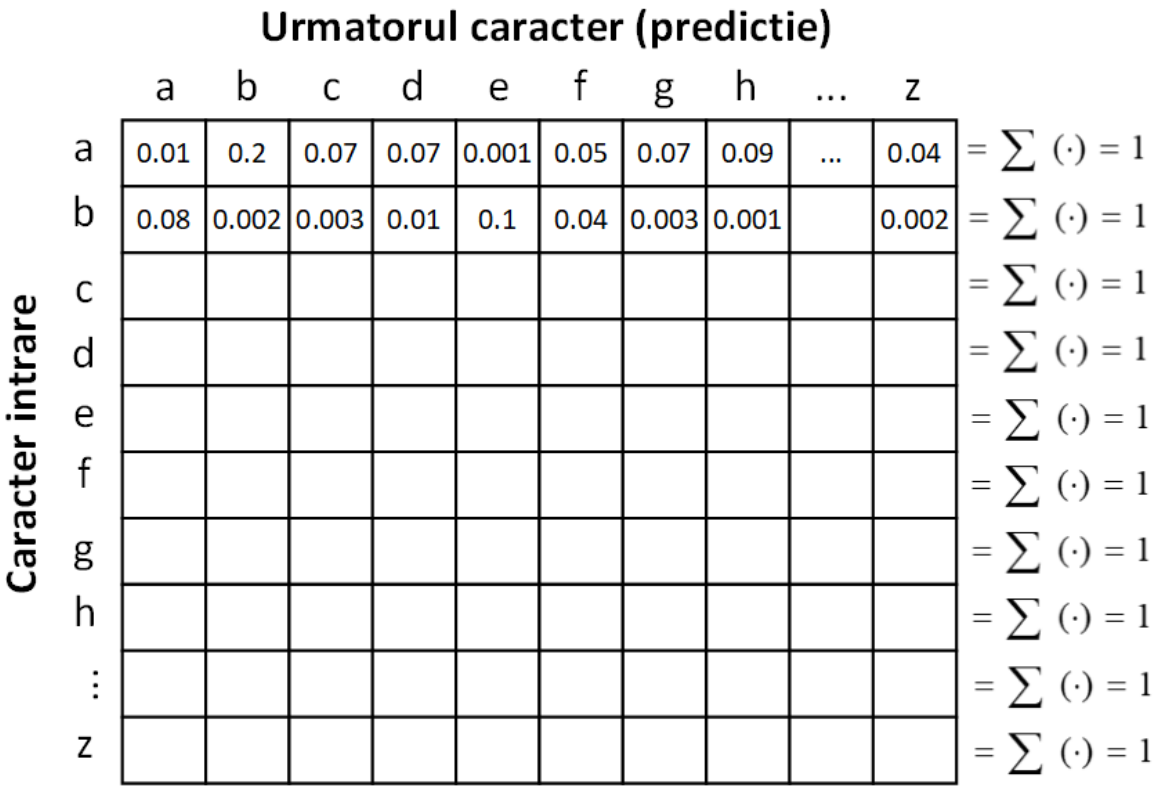


Fig. 9.3 Tabelă de mapare a caracterelor în modelul Bigram.

distribuției de probabilitate a liniei din tabelă.

## 9.3 Perceptron Multistrat

Așa cum ne-am aștepta, performanțele modelului Bigram sunt reduse, în special luând în considerare cantitatea redusă de informații pe care o procesează (perechi de caractere), cât și numărul redus de parametrii în care poate stoca informația învățată (ponderile din tabela de mapare).

O îmbunătățire semnificativă a modelului se poate aduce prin adăugarea de noi unități antrenabile ce procesează datele din tabela de mapare. Astfel, coloanele matricei nu mai descriu probabilitatea directă a următorului caracter, ci un spațiu latent $(z_1, z_2, ..., z_q)$, ce reprezintă informații de intrare în noi unități neuronale de calcul. Figura 9.4 ilustrează structura tabelei de mapare a token-urilor (caracterelor) într-un spațiu latent de dimeniune dim vocabular $\times$ $q$.

Un astfel de model, ilustrat în diagrama bloc din Figura 9.5, are aceeași structură cu modelul Bigram, la care însă a fost adăugat un strat intermediar de calcul bazat pe arhitectura unui perceptron multistrat (Multi Layer Perceptron). Spațiul latent din tabela de mapare reprezintă intrarea într-un strat liniar format din mai multe unități de calcul (neuroni). Ieșirile acestora sunt procesate utilizând o funcție de activare, ca de exemplu sigmoid sau tangentă hiperbolică. În final, un al doilea strat liniar este utilizat pentru procesarea activărilor, înaintea operațiilor de Softmax și eșantionare.

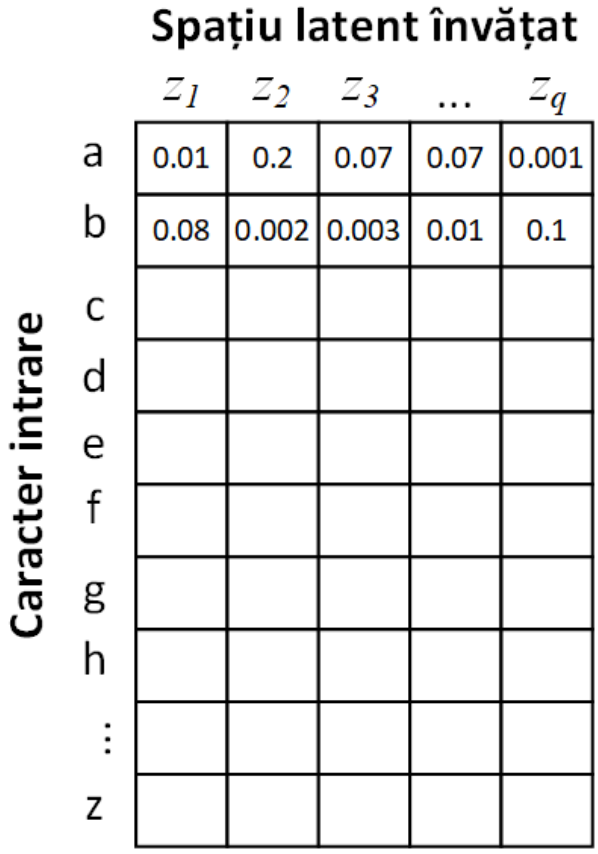


Fig. 9.4 Tabelă de mapare a caracterelor în modelul Bigram.

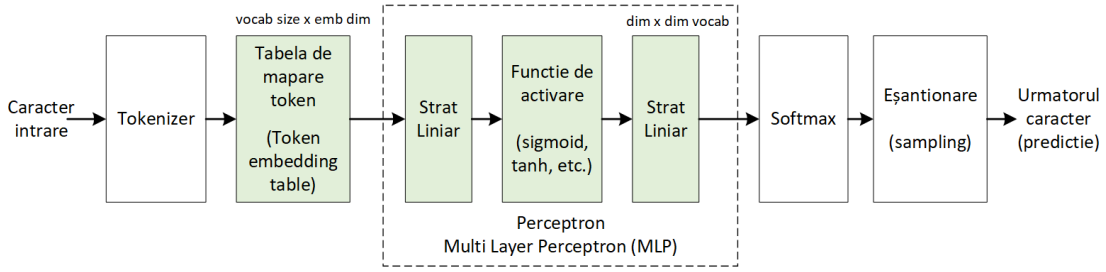


Fig. 9.5 Arhitectura modelului Bigram bazat pe MLP.

## 9.4 Mecanismul de Atenție

Mecanismul de atenție (self-attention) a devenit începând cu 2017 un model standard utilizat în rețelele neuronale pentru procesarea limbajului natural. Acesta crează relații între diferitele poziții ale token-urilor dintr-o secvență de text, în scopul calculului unei reprezentări latente a acestei secvențe.

O funcție ce implementează mecanismul de atenție poate fi descrisă ca o modalitate de mapare a unei întrebari (query) și a unei perechi cheie-valoare (key-value) la o ieșire, unde întrebarea, cheia, valoarea și ieșirea sunt vectori. Ieșirea este calculată ca o sumă ponderată a valorii, unde ponderea asociată fiecărei valori este calculată cu ajutorul unei funcții de compatibilitate a întrebării cu cheia corespondentă.

Mecanismul de atenție este ilustrat în "modulul atenție" din Figura 9.5. Un vector de

intrare **x**, este prelucrat în paralel de 3 straturi liniare de procesare: întrebare (query), cheie (key) și valoare (value). Ieșirea stratului întrebare este multiplicat cu ieșirea stratului cheie. Rezultatul multiplicării este trecut prin funcția Softmax, înainte de a fi multiplicat cu ieșirea stratului valoare.

## 9.5 Arhitectura Transformer

Arhitectura unei rețele neuronale Transformer este bazată pe o colecție de funcții ce implementează mecanismul de atenție descris mai sus.

În plus fața de modelele Bigram si Perceptron introduse anterior, un Transformer conține două tabele de mapare (Tokens Embedding). Prima dintre ele este tabela de mapare a token-urilor, identică cu tabela introdusă anterior. A doua tabela (Positional Embedding) este destinată învățării diferitelor poziții ale token-urilor dintr-o secvență de intrare. Vectorul de intrare **x** în funcția de atenție este suma spaților latente ale celor două tabele.

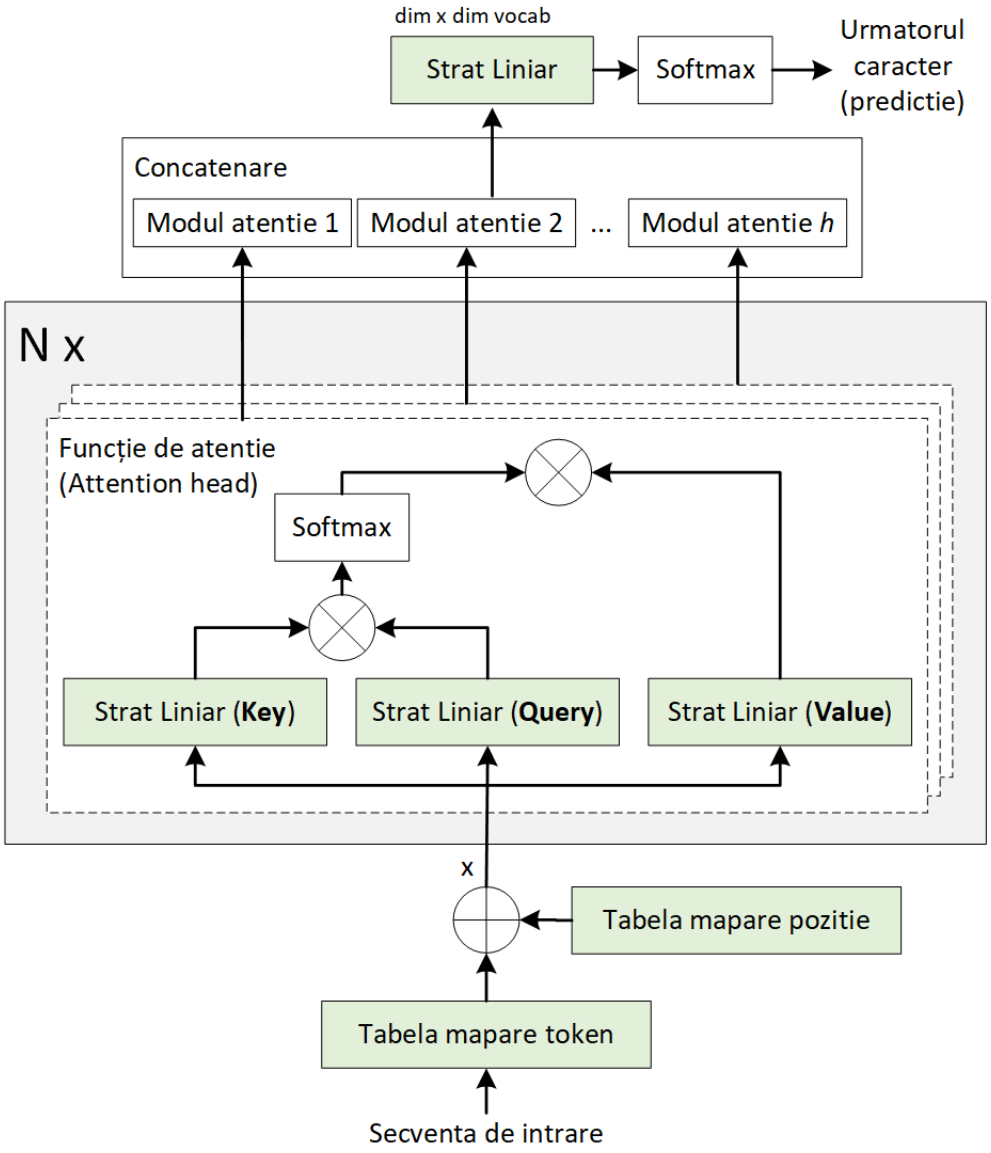


Fig. 9.6 Arhitectura modelului Transformer.

Un model Transformer contine $N$ funcții de atenție. Predicția unui nou token este făcută prin concatenarea ieșirilor celor $N$ funcții, respectiv procesarea vectorului concatenat printr-un strat liniar și funcția Softmax.

# Bibliografie


[1] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[2] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[3] M. Lutz, *Learning Python*, 2nd Ed. Sebastopol, CA, USA: O'Reilly & Associates, Inc., 2003.

[4] A. Ng, "Stanford cs229 - machine learning," 2008.

[5] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd Ed. Pearson Education, 2003.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, Vol. 30. Curran Associates, Inc., 2017.