

# INTELIGENȚĂ ARTIFICIALĂ



**Sisteme inteligente**

Sisteme care învață singure

– generative AI–

Laura Dioșan

# Sumar

---

## A. Scurtă introducere în Inteligența Artificială (IA)

## C. Sisteme inteligente

### ■ Sisteme care învață singure

- Arbori de decizie
- **Rețele neuronale artificiale**
- kNN
- Algoritmi evolutivi
- Mașini cu suport vectorial

### ■ Sisteme bazate pe reguli

### ■ Sisteme hibride

## B. Rezolvarea problemelor prin căutare

### ■ Definirea problemelor de căutare

### ■ Strategii de căutare

- Strategii de căutare neinformate
- Strategii de căutare informate
- Strategii de căutare locale (Hill Climbing, Simulated Annealing, Tabu Search, Algoritmi evolutivi, PSO, ACO)
- Strategii de căutare adversială

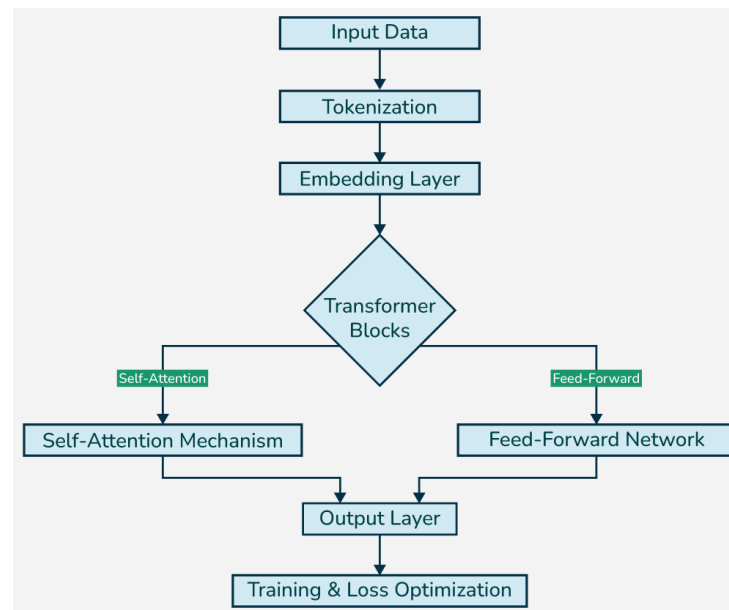
# Rețele neuronale artificiale

---

- Rețele neuronale dense (fully-connected)
- Rețele neuronale convolutive
- Transformers

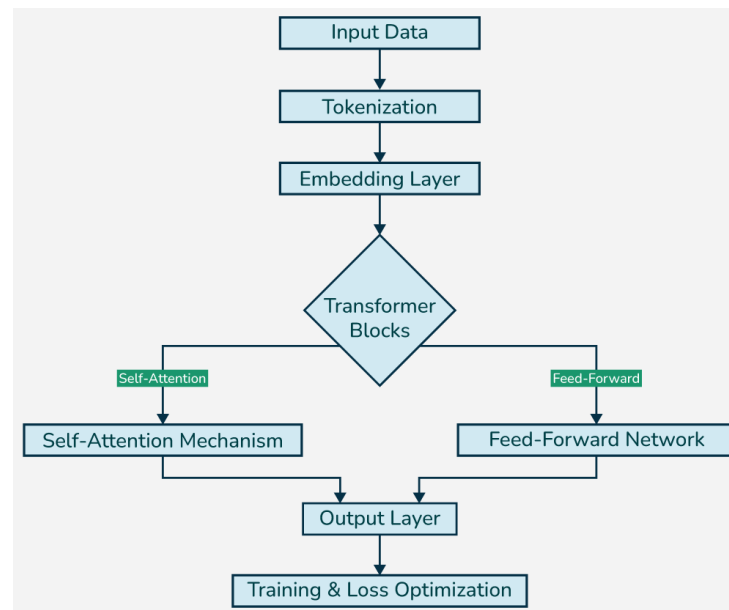
# Procesarea textelor/limbajelor

- ❑ Large language Models (LLMs)
  - Tokenisation
  - Embeddings
  - Transformer & Self-attention mechanism
  - Feed-forward network

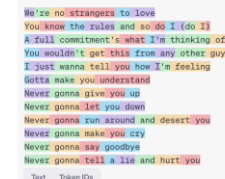


# Procesarea textelor/limbajelor

- ❑ Large language Models (LLMs)
  - **Tokenisation**
  - Embeddings
  - Transformer & Self-attention mechanism
  - Feed-forward network



# Large language Models (LLMs)



We're no strangers to love  
You know the rules and so do I (do I)  
A full commitment's what I'm thinking of  
You wouldn't get this from any other guy  
I just wanna tell you how I'm feeling  
Gotta make you understand  
Never gonna give you up  
Never gonna let you down  
Never gonna run around and desert you  
Never gonna make you cry  
Never gonna say goodbye  
Never gonna tell a lie and hurt you

Text    Token IDs

## □ Tokenisation

### ■ Token-based models

- White Space: Splits on spaces (simple but limited)
- Word: Breaks into words (common for English)
- Sentence: Divides text into sentences
- Character: Splits into individual characters
- N-gram: Creates sequences of n items/elements
  - fastText (2016)
- subword: Breaks words into smaller parts
  - Byte Pair Encoding (2016): Merges common character pairs → GPT, GPT2, GPT3
  - WordPiece (2012): Google's method for balancing words and subwords → BERT
  - Unigram (2018) → T5, AIBERT

### ■ Token-free models

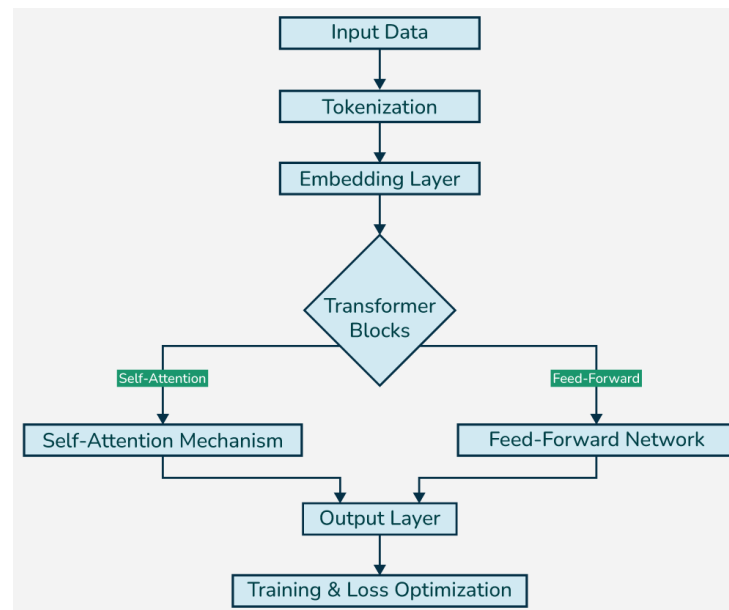
- CharFormer
- ByT5
- MegaByte (2023)
- Byte-latent transformer (2024)

### ■ Useful resources

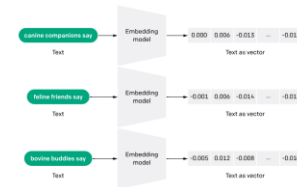
- 07\_tokenizers.ipynb
- Karpathy's tutorial <https://www.youtube.com/watch?v=zduSFxRajkE>
- Hugging Face tutorial <https://huggingface.co/learn/llm-course/en/chapter2/4>

# Procesarea textelor/limbajelor

- ❑ Large language Models (LLMs)
  - Tokenisation
  - **Embeddings**
  - Transformer & Self-attention mechanism
  - Feed-forward network



# Large language Models (LLMs)



## □ Word Embeddings

### ■ Frequency-based embeddings

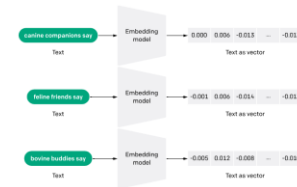
- One-hot encoding
- TF-IDF
- Co-occurrence matrix

### ■ Prediction based embeddings → Word / subword embeddings

- De ce?
  - Token → reprezentari dense
- Cum?
  - Word2vec (Skip-gram or Continuous Bag of Words (CBOW), FastText, GloVe, altele



# Large language Models (LLMs)

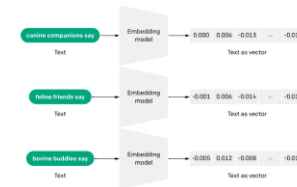


## □ Word Embeddings

### ■ Vectori rari vs. vectori densi

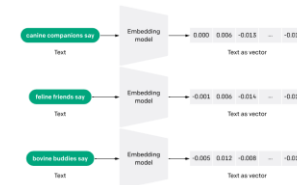
- Vectorii clasici → matricea de aparitie/co-ocurenta a termenilor
  - lungi (length  $|V| = 20,000 \rightarrow 50,000$ )
  - rari (f multe elemente sunt 0)
- Alternativa: vectori invatati (prin AI/ML)
  - scurti (length 200-1000)
  - densi (multe elemente nu sunt 0)
- De ce vectori densi?
  - Vectorii scurti -> folositi mai usor ca si features in algoritmii de invatare (mai putini coeficienti de invatat)
  - Vectorii densi pot generaliza mai bine, captand sinonimia termenilor
    - Bike – scooter
    - Car – automobile
    - House – apartment

# Large language Models (LLMs)



- Word Embeddings
  - Learnt embeddings
    - Static (context-free)

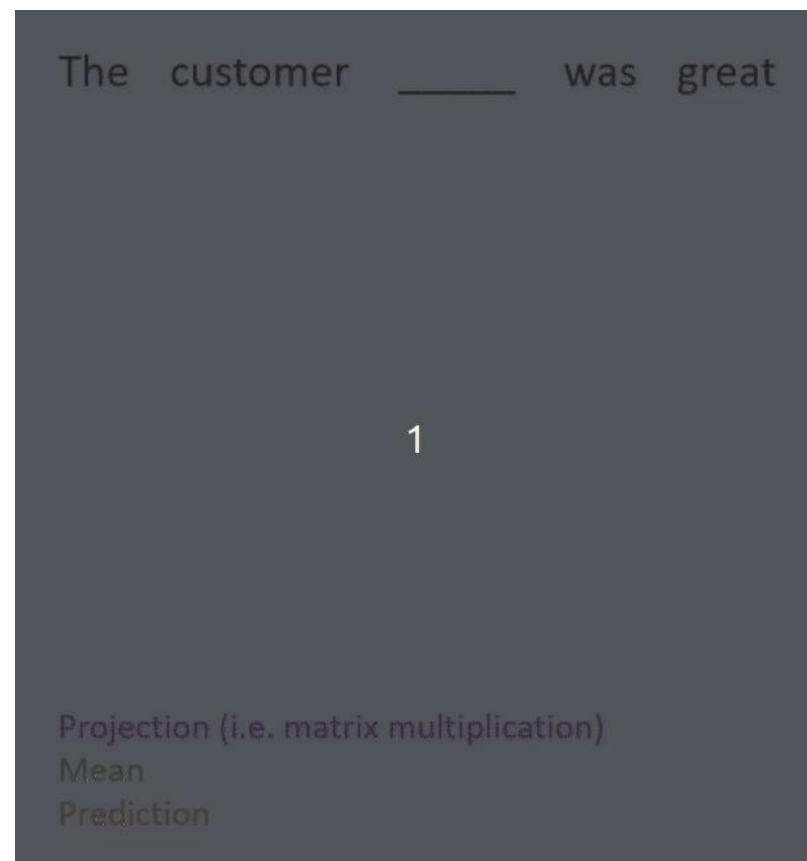
# Large language Models (LLMs)



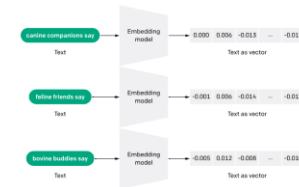
## □ Word Embeddings

### ■ Learnt embeddings

- Static (context-free)
  - Word2vec (2013)
  - GLoVe (2014)



# Large language Models (LLMs)



## □ Word Embeddings

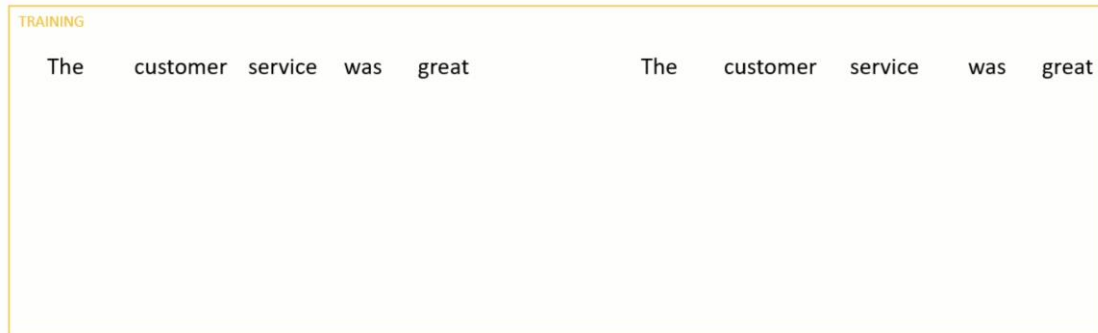
### ■ Learnt embeddings

#### □ Static (context-free)

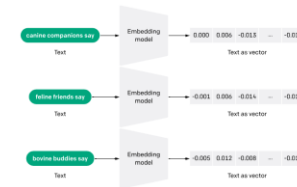
- Word2vec (2013)
- GLoVe (2014)

#### □ Dinamic (context-based)

- ELMo (2018)



# Large language Models (LLMs)



## □ Word Embeddings

### ■ Learnt embeddings

#### □ Static (context-free)

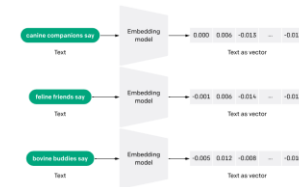
- Word2vec (2013)
- GLoVe (2014)

#### □ Dinamic (context-based)

- ELMo (2018)
- BERT (2019)

[CLS] The \_\_\_\_ service was great [SEP] They \_\_\_\_ very help ##ful [SEP]

# Large language Models (LLMs)



## □ Word Embeddings

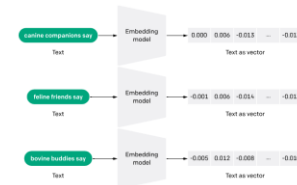
### ■ Learnt embeddings

- Static (context-free)
  - Word2vec (2013)
  - GLoVe (2014)
- Dinamic (context-based)
  - ELMo (2018)
  - BERT (2019)

### ■ Great resources

- Word Embeddings [https://lens-voita.github.io/nlp\\_course/word\\_embeddings.html](https://lens-voita.github.io/nlp_course/word_embeddings.html)
- The Illustrated Word2vec – Jay Alammam – Visualizing machine learning one concept at a time <https://jalammar.github.io/illustrated-word2vec/>

# Large language Models (LLMs)



## □ Word Embeddings

### ■ Frequency-based embeddings

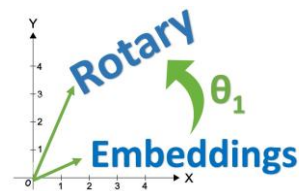
- One-hot encoding
- TF-IDF
- Co-occurrence matrix

### ■ Prediction based embeddings → Word / subword embeddings

- De ce?
  - Token → reprezentari dense
- Cum?
  - Word2vec (Skip-gram or Continuous Bag of Words (CBOW), FastText, GloVe, altele

### ■ Positional embeddings

# Large language Models (LLMs)



## □ Word Embeddings

### ■ Positional embeddings

#### □ Ce valori conține PE?

##### ■ Principii

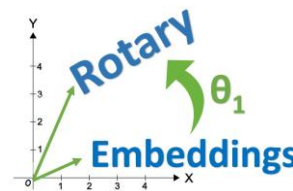
- Embedding-urile trebuie să păstreze distanța originală între cuvinte
  - $\text{dist}(\text{Emb}(\text{câine}), \text{Emb}(\text{pisică})) < \text{dist}(\text{Emb}(\text{câine}), \text{Emb}(\text{fereastră}))$
  - $\text{Dist}(\text{Emb}(\text{word}_k), \text{Emb}(\text{word}_{k+1})) = \text{Dist}(\text{Emb}(\text{word}_p), \text{Emb}(\text{word}_{p+1}))$
- Ortogonalitate - Embedding-urile cuvintelor ne-conectate să fie perpendiculare
  - $\text{Emb}(\text{ușă}) \perp \text{Emb}(\text{pește})$
  - $\text{Similaritate}(\text{ușă}, \text{pește}) = 0$
- Embedding-uri independente de lungimea propoziției
  - → putere de generalizare (train vs. test)
  - → *bounded*

##### ■ Reprezentări / valori posibile

- Poziția efectivă (1,2,3,4)
- Poziția în reprezentare 1-hot encoding
- Poziția în reprezentare binară
- Poziția în reprezentare polară



# Pozitia in propozitie



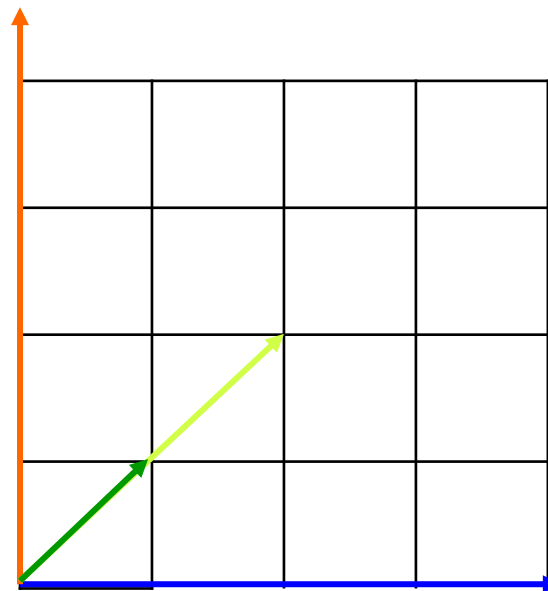
## □ Exemplu

- O propoziție cu 5 cuvinte și  $d = 1$
- $PE(cuv) = \text{poziția cuvântului}$

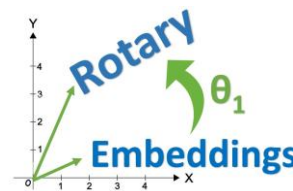
0
1
2
3
4

Dist	w0	w1	w2	w3	w4
w0	0	1	2	3	4
w1	1	0	1	2	3
w2	2	1	0	1	2
w3	3	2	1	0	1
w4	4	3	2	1	0

**$\text{sim}(PE(w_i), PE(w_j)) \neq 0$ , orice  $i \neq j$**



# Pozitia normalizată în propoziție



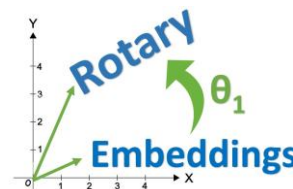
## □ Exemplu

- O propoziție cu 5 cuvinte și  $d = 1$
- $PE(cuv) = \text{poziția cuvântului}$

0/5
1/5
2/5
3/5
4/5

- Problema: al k-lea cuvânt într-o propoziției de N cuvinte are alt embedding decât al k-lea cuvânt într-o propoziție cu M cuvinte

# 1-hot encoding



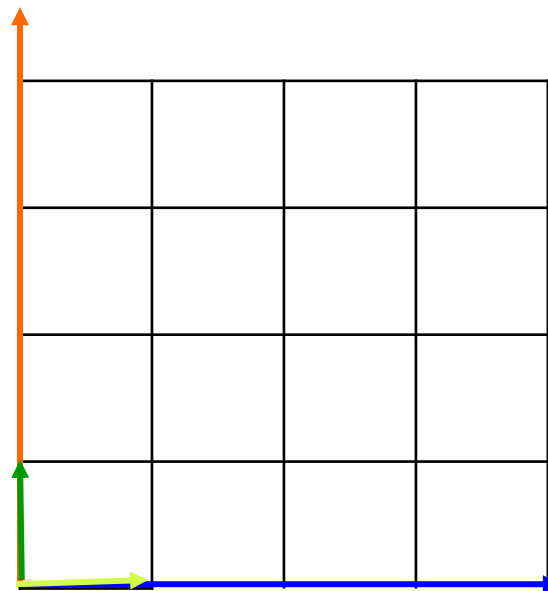
## □ Exemplu

- O propoziție cu 5 cuvinte și  $d = 6$
- $PE(cuv) =$  reprezentarea 1-hot a poziției cuvântului

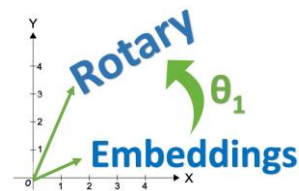
0	0	0	0	0	1
0	0	0	0	1	0
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	0	0	0

Dist	w0	w1	w2	w3	w4
w0	0	1.4	1.4	1.4	1.4
w1	1.4	0	1.4	1.4	1.4
w2	1.4	1.4	0	1.4	1.4
w3	1.4	1.4	1.4	0	1.4
w4	1.4	1.4	1.4	1.4	0

$\text{sim}(PE(w_i), PE(w_j)) = 1.4$ , orice  $i \neq j$



# Binary encoding



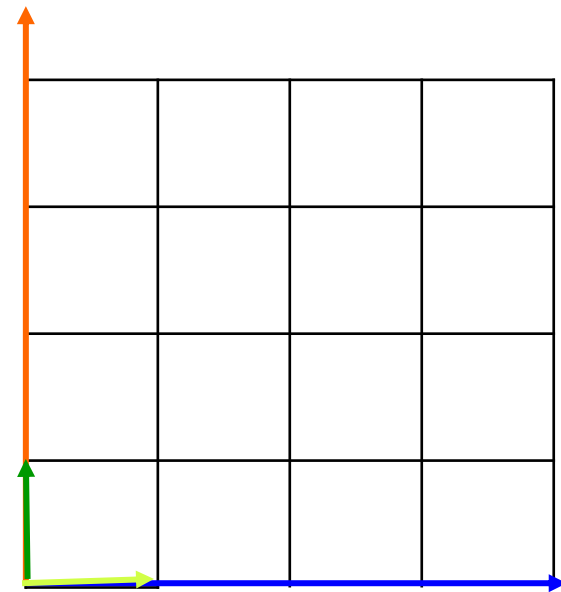
## □ Exemplu

- O propoziție cu 5 cuvinte și  $d = 6$
- $PE(cuv) =$  reprezentarea binară a poziției cuvântului

0	0	0	0	0	0
0	0	0	0	0	1
0	0	0	0	1	0
0	0	0	0	1	1
0	0	0	1	0	0

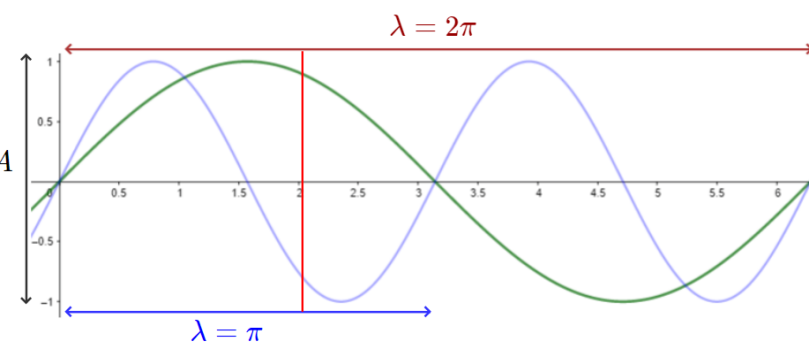
Dist	w0	w1	w2	w3	w4
w0	0	1	1	$\sqrt{2}$	1
w1	1	0	$\sqrt{2}$	1	$\sqrt{2}$
w2	1	$\sqrt{2}$	0	1	$\sqrt{2}$
w3	$\sqrt{2}$	1	1	0	$\sqrt{3}$
w4	1	$\sqrt{2}$	$\sqrt{2}$	$\sqrt{3}$	0

exista  $i, j$  a.î.  $\text{sim}(PE(w_i), PE(w_j))=0$



- Problema: dist nu e functie monotona!

# Sin-based encoding (Rotary PE)



## Exemplu

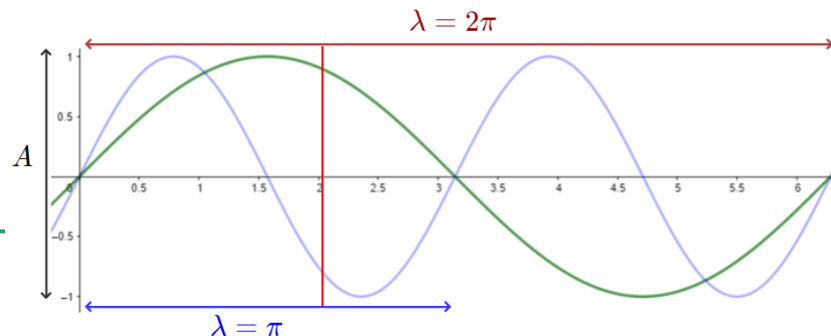
- O propoziție cu 5 cuvinte și  $d = 6$
- $PE(cuv) =$  valorile funcției sin pentru diferite argumente (frecvențe sau lungimi de undă)
  - E.g.  $\sin(2 \pi \text{ pos} / \lambda_i), i = 0, 1, 2, \dots, d-1$

pos		$\lambda = \pi$	$\lambda = 2\pi$	$\lambda = 3\pi$	$\lambda = 4\pi$	$\lambda = 5\pi$	$\lambda = 6\pi$
0	→	$\sin(2*0)$	$\sin(0)$	$\sin(2/3*0)$	$\sin(2/4*0)$	$\sin(2/5*0)$	$\sin(2/6*0)$
1	→	$\sin(2*1)$	$\sin(1)$	$\sin(2/3*1)$	$\sin(2/4*1)$	$\sin(2/5*1)$	$\sin(2/6*1)$
2	→	$\sin(2*2)$	$\sin(2)$	$\sin(2/3*2)$	$\sin(2/4*2)$	$\sin(2/5*2)$	$\sin(2/6*2)$
3	→	$\sin(2*3)$	$\sin(3)$	$\sin(2/3*3)$	$\sin(2/4*3)$	$\sin(2/5*3)$	$\sin(2/6*3)$
4	→	$\sin(2*4)$	$\sin(4)$	$\sin(2/3*4)$	$\sin(2/4*4)$	$\sin(2/5*4)$	$\sin(2/6*4)$

# Sin-based encoding

## Exemplu

- O propoziție cu 5 cuvinte și  $d = 6$
- $PE(cuv) =$  valorile funcției sin pentru diferite argumente (frecvențe sau lungimi de undă)
  - E.g.  $\sin(2 \pi \text{ pos} / \lambda_i), i = 0, 1, 2, \dots, d-1$



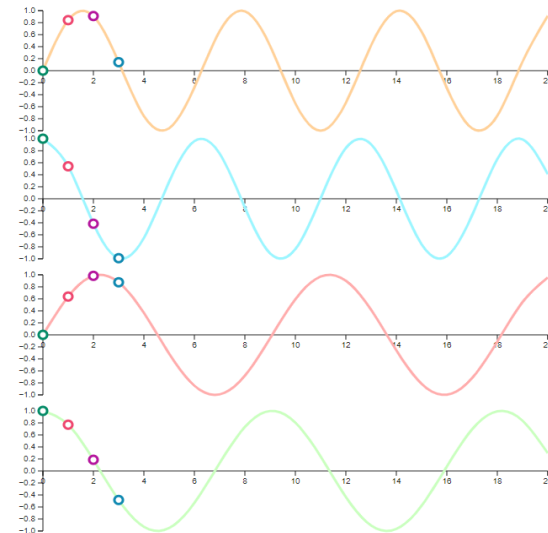
pos		$\lambda = \pi$	$\lambda = 2\pi$	$\lambda = 3\pi$	$\lambda = 4\pi$	$\lambda = 5\pi$	$\lambda = 6\pi$
0	→	$\sin(0)$	$\sin(0)$	$\sin(0)$	$\sin(0)$	$\sin(0)$	$\sin(0)$
1	→	$\sin(2*1)$	$\sin(1)$	$\sin(2/3*1)$	$\sin(2/4*1)$	$\sin(2/5*1)$	$\sin(2/6*1)$
2	→	$\sin(2*2)$	$\sin(2)$	$\sin(2/3*2)$	$\sin(2/4*2)$	$\sin(2/5*2)$	$\sin(2/6*2)$
3	→	$\sin(2*3)$	$\sin(3)$	$\sin(2/3*3)$	$\sin(2/4*3)$	$\sin(2/5*3)$	$\sin(2/6*3)$
4	→	$\sin(2*4)$	$\sin(4)$	$\sin(2/3*4)$	$\sin(2/4*4)$	$\sin(2/5*4)$	$\sin(2/6*4)$

$$\text{sim}(PE(\text{word}_0), PE(\text{word}_k)) = 0!!!$$

# Sin&cos-based encoding

## □ Exemplu

- O propoziție cu 5 cuvinte și  $d = 6$
- $PE(cuv) =$  perechi (sin, cos) pentru diferite argumente (frecvențe sau lungimi de undă)
  - E.g.  $(\sin(pos / 10000^{2i/d}), \cos(pos / 10000^{2i/d}))$ ,  $i = 0, 1, 2, \dots, d-1$

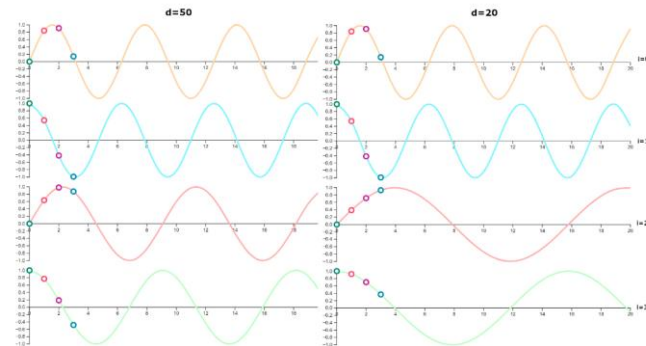


pos		i=0	i=1	i=2	i=3	i=4	i=5
0	→	sin(0)	cos(0)	sin(0)	cos(0)	sin(0)	cos(0)
1	→	sin(1)	cos(1)	$\sin(1/10000^{4/6})$	$\cos(1/10000^{4/6})$	$\sin(1/10000^{8/6})$	$\cos(1/10000^{8/6})$
2	→	sin(2)	cos(2)	$\sin(2/10000^{4/6})$	$\cos(2/10000^{4/6})$	$\sin(2/10000^{8/6})$	$\cos(2/10000^{8/6})$
3	→	sin(3)	cos(3)	$\sin(3/10000^{4/6})$	$\cos(3/10000^{4/6})$	$\sin(3/10000^{8/6})$	$\cos(3/10000^{8/6})$
4	→	sin(4)	cos(4)	$\sin(4/10000^{4/6})$	$\cos(4/10000^{4/6})$	$\sin(4/10000^{8/6})$	$\cos(4/10000^{8/6})$

# Sin&cos-based encoding

## □ Resources

- <https://towardsdatascience.com/why-and-how-to-achieve-longer-context-windows-for-lms-5f76f8656ea9/>
- <https://aiexpjourney.substack.com/p/an-in-depth-exploration-of-rotary-position-embedding-rope-ac351a45c794>
- [https://kazemnejad.com/blog/transformer\\_architecture\\_positional\\_encoding/](https://kazemnejad.com/blog/transformer_architecture_positional_encoding/)





# Positional embedding

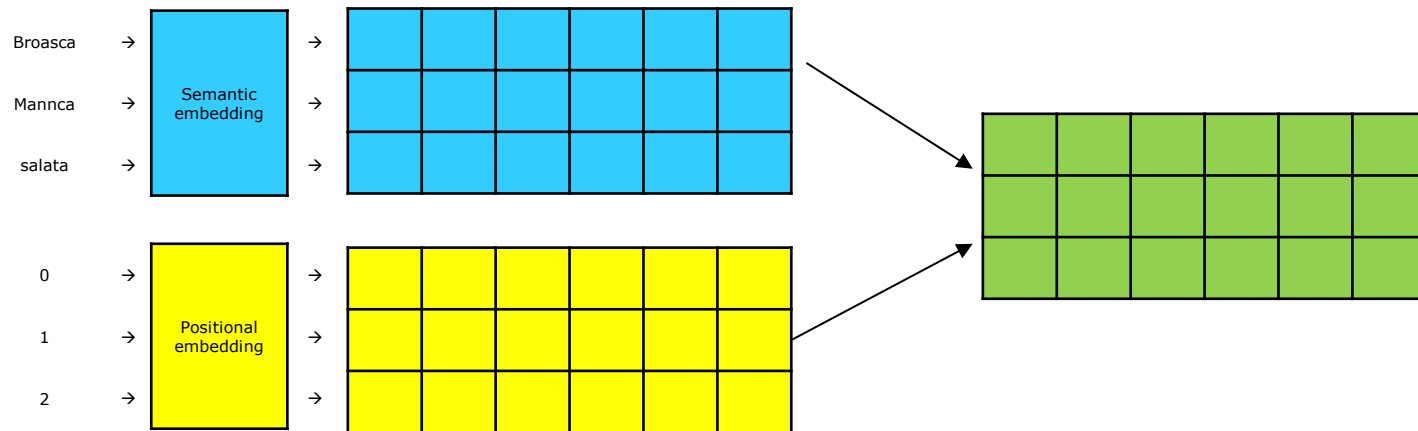
---

- Cuvinte → Embeddings
- Pentru o propoziție cu  $n$  cuvinte, se obțin  $n$  embedding-uri (de lungime  $d$ )
  - Care țin cont de semantica cuvintelor
    - Word Embeddings
      - descoperite / învățate
      - prin ML (Word2Vec, GloVe, etc.)
  - Care țin cont de poziția (absolută sau relativă) a cuvintelor în propoziției
    - Positional Embeddings
      - calculate

# Positional embedding

## □ Exemplu

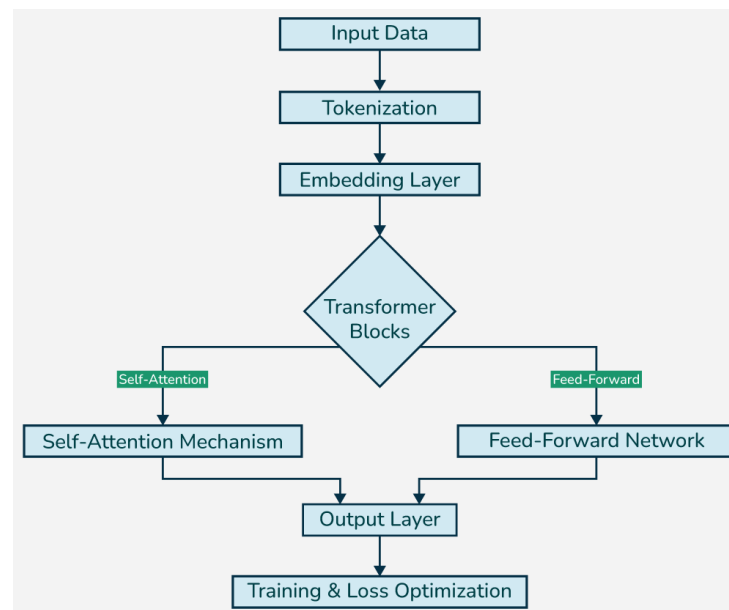
- O propoziție cu  $n$  cuvinte
- Fiecărui cuvânt  $i$  se asociază
  - Un embedding semantic SE de lungime  $d$
  - Un embedding pozițional PE de lungime  $d$
  - Embedding-ul final corespunzător cuvântului va fi suma celor 2 (WE + PE)  $\rightarrow n \times d$  valori



# Procesarea textelor/limbajelor

## ❑ Large language Models (LLMs)

- Tokenisation
- Embeddings
- **Transformer & Self-attention mechanism**
- Feed-forward network



# LLMs - Mecanismul de “atenție” (attention)

---

Broasca a mancat salata pentru ca ea era ...

flamanda

delicioasa

# LLMs - Mecanismul de “atenție” (attention)

---

Broasca a mancat salata pentru ca ea era ...

flamanda

delicioasa

# Mecanismul de “atenție” (attention)

---

Broasca a mancat salata pentru ca ea era ...

flamanda

delicioasa

În acvariu era o **broască** și un **pește**

S-a stricat **broasca** de la **ușă**

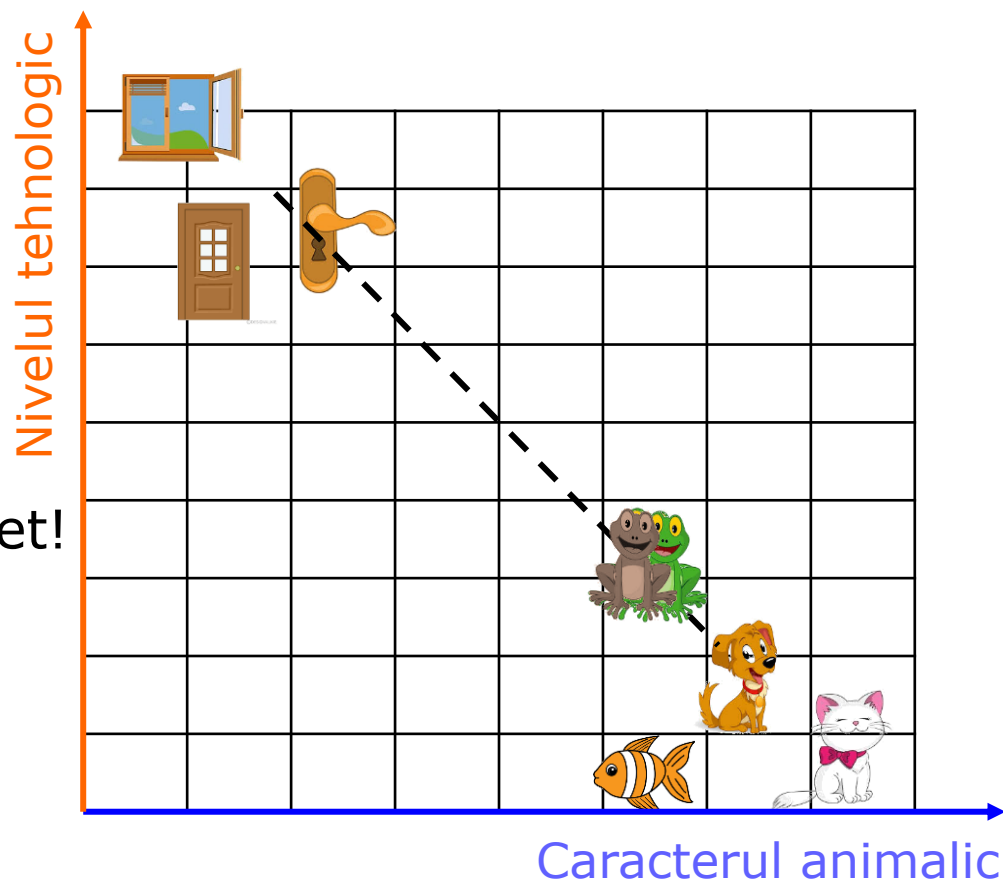
# Mecanismul de “atenție” (attention)

## □ Remember – embeddings

- cuvinte izolate precum: *geam*, *ușă*, *clanță*, *pisică*, *pește*, *câine*, *broască*
- Presupunem 2 attribute:
  - *caracterul animalic*
  - *nivelul tehnologic*

## □ Attention

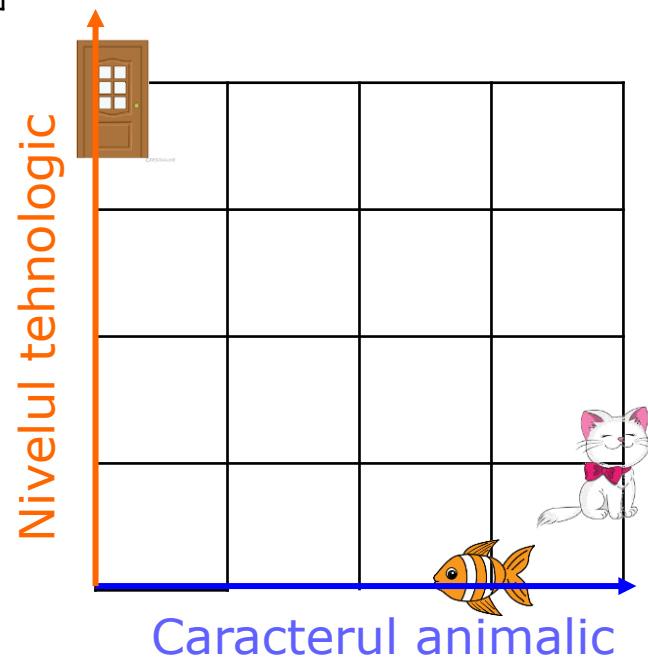
- Contextul e ca un magnet!
- Atrage cuvintele care se potrivesc!



# Mecanismul de “atenție” (attention)

## □ Similaritatea între cuvinte

Cuvântul	<i>caracterul animalic</i>	<i>nivelul tehnologic</i>
Pisică	4	1
Pește	3	0
Ușă	0	4



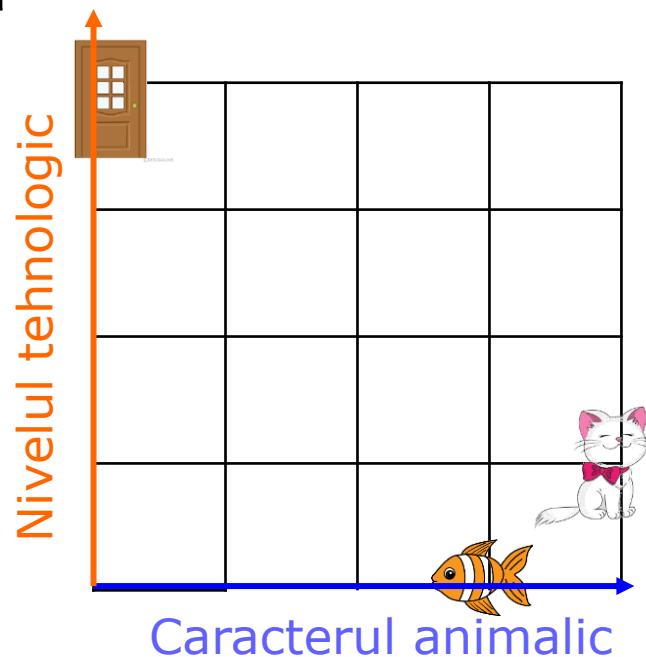


# Mecanismul de “atenție” (attention)

## □ Similaritatea între cuvinte

Cuvântul	<i>caracterul animalic</i>	<i>nivelul tehnologic</i>
Pisică	4	1
Pește	3	0
Ușă	0	4

□  $\text{sim}(\text{pisică}, \text{pește}) =$

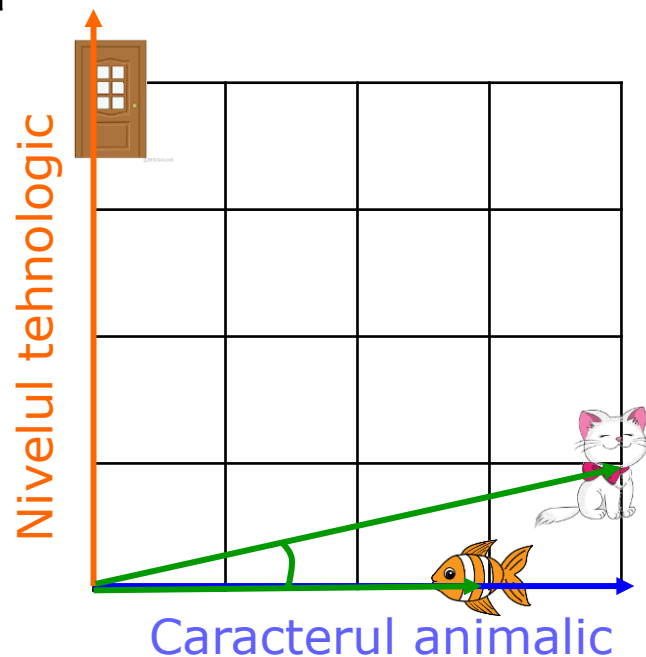


# Mecanismul de “atenție” (attention)

## □ Similaritatea între cuvinte

Cuvântul	<i>caracterul animalic</i>	<i>nivelul tehnologic</i>
Pisică	4	1
Pește	3	0
Ușă	0	4

□  $\text{sim}(\text{pisică}, \text{pește}) =$

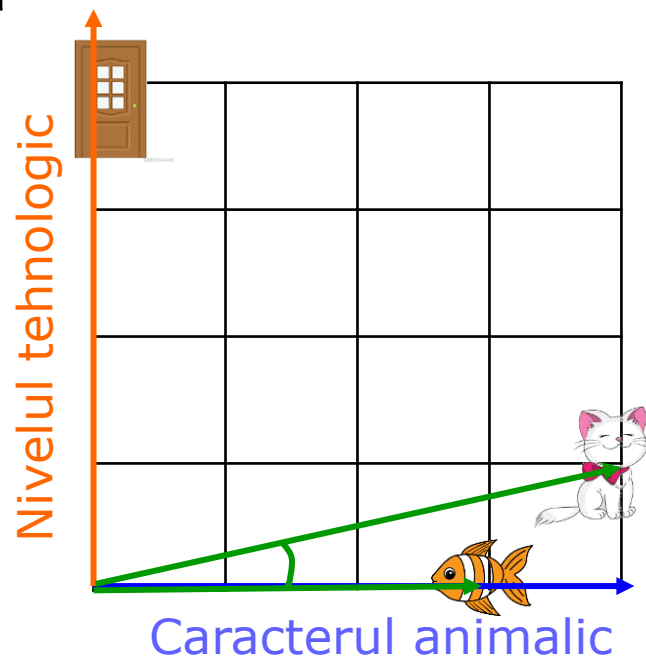


# Mecanismul de “atenție” (attention)

## □ Similaritatea între cuvinte

Cuvântul	<i>caracterul animalic</i>	<i>nivelul tehnologic</i>
Pisică	4	1
Pește	3	0
Ușă	0	4

□  $\text{sim}(\text{pisică}, \text{pește}) =$   
 $\text{sim}([4,1], [3,0]) = (4*3 + 1*0) / (\sqrt{17} * \sqrt{9})$   
 $= 12 / (4.12 * 3) = 0.97$



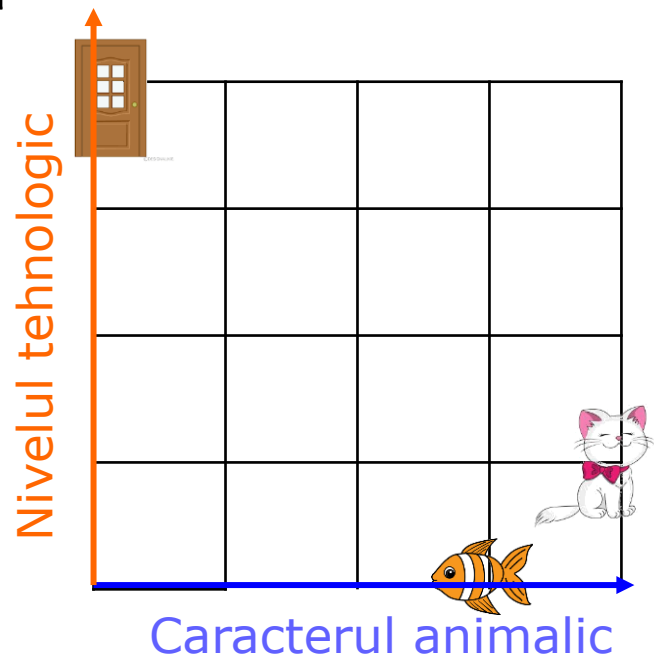
# Mecanismul de “atenție” (attention)

## ❑ Similaritatea între cuvinte

Cuvântul	<i>caracterul animalic</i>	<i>nivelul tehnologic</i>
Pisică	4	1
Pește	3	0
Ușă	0	4

❑  $\text{sim}(\text{pisică}, \text{pește}) =$   
 $\text{sim}([4,1], [3,0]) = (4*3 + 1*0) / (\sqrt{17} * \sqrt{9})$   
 $= 12 / (4.12 * 3) = 0.97$

❑  $\text{sim}(\text{pisică}, \text{ușă}) =$



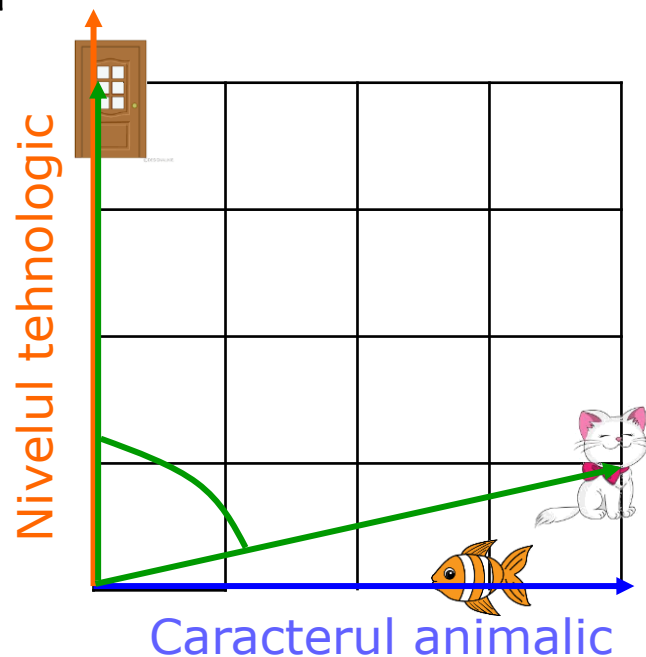
# Mecanismul de “atenție” (attention)

## □ Similaritatea între cuvinte

Cuvântul	<i>caracterul animalic</i>	<i>nivelul tehnologic</i>
Pisică	4	1
Pește	3	0
Ușă	0	4

□  $\text{sim}(\text{pisică}, \text{pește}) =$   
 $\text{sim}([4,1], [3,0]) = (4*3 + 1*0) / (\sqrt{17} * \sqrt{9})$   
 $= 12 / (4.12 * 3) = 0.97$

□  $\text{sim}(\text{pisică}, \text{ușă}) =$



# Mecanismul de “atenție” (attention)

## ❑ Similaritatea între cuvinte

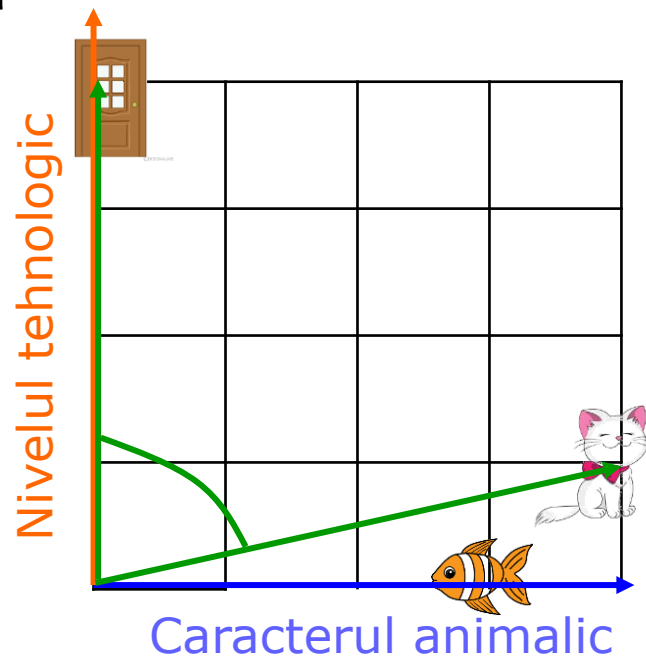
Cuvântul	<i>caracterul animalic</i>	<i>nivelul tehnologic</i>
Pisică	4	1
Pește	3	0
Ușă	0	4

## ❑ $\text{sim}(\text{pisică}, \text{pește}) =$

$$\begin{aligned}\text{sim}([4,1], [3,0]) &= (4*3 + 1*0) / (\sqrt{17} * \sqrt{9}) \\ &= 12 / (4.12 * 3) = 0.97\end{aligned}$$

## ❑ $\text{sim}(\text{pisică}, \text{ușă}) =$

$$\begin{aligned}\text{sim}([4,1], [0,4]) &= (4*0 + 1*4) / (\sqrt{17} * \sqrt{16}) \\ &= 4 / (4.12 * 4) = 0.24\end{aligned}$$



# Mecanismul de “atenție” (attention)

## □ Similaritatea între cuvinte

Cuvântul	<i>caracterul animalic</i>	<i>nivelul tehnologic</i>
Pisică	4	1
Pește	3	0
Ușă	0	4

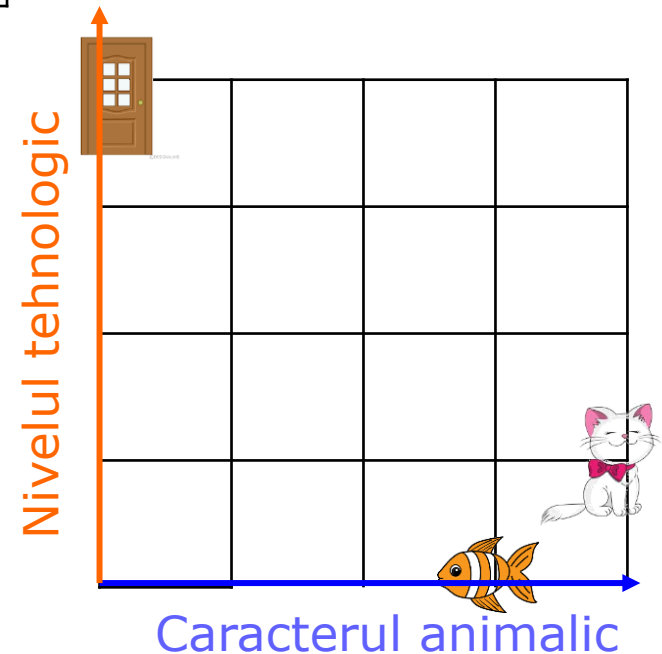
### □ $\text{sim}(\text{pisică}, \text{pește}) =$

$$\begin{aligned}\text{sim}([4,1], [3,0]) &= (4*3 + 1*0) / (\sqrt{17} * \sqrt{9}) \\ &= 12 / (4.12 * 3) = 0.97\end{aligned}$$

### □ $\text{sim}(\text{pisică}, \text{ușă}) =$

$$\begin{aligned}\text{sim}([4,1], [0,4]) &= (4*0 + 1*4) / (\sqrt{17} * \sqrt{16}) \\ &= 4 / (4.12 * 4) = 0.24\end{aligned}$$

### □ $\text{sim}(\text{pește}, \text{ușă}) =$



# Mecanismul de “atenție” (attention)

## ❑ Similaritatea între cuvinte

Cuvântul	<i>caracterul animalic</i>	<i>nivelul tehnologic</i>
Pisică	4	1
Pește	3	0
Ușă	0	4

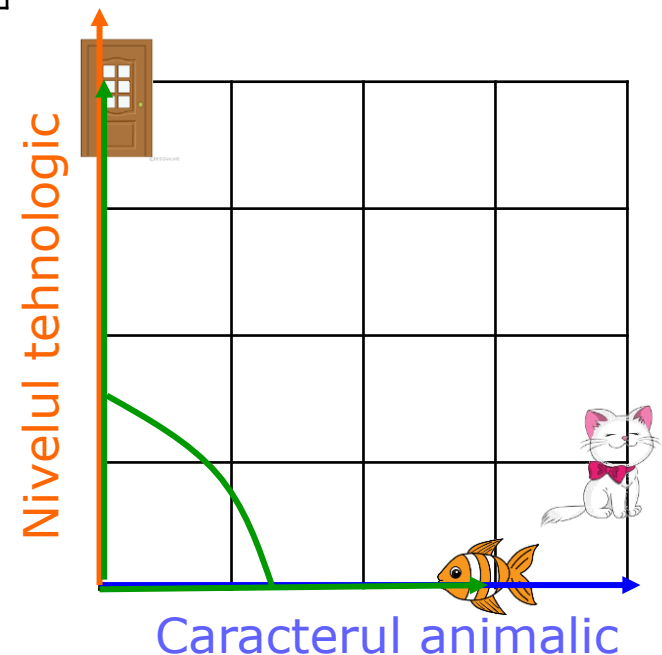
### ❑ $\text{sim}(\text{pisică}, \text{pește}) =$

$$\begin{aligned}\text{sim}([4,1], [3,0]) &= (4*3 + 1*0) / (\sqrt{17} * \sqrt{9}) \\ &= 12 / (4.12 * 3) = 0.97\end{aligned}$$

### ❑ $\text{sim}(\text{pisică}, \text{ușă}) =$

$$\begin{aligned}\text{sim}([4,1], [0,4]) &= (4*0 + 1*4) / (\sqrt{17} * \sqrt{16}) \\ &= 4 / (4.12 * 4) = 0.24\end{aligned}$$

### ❑ $\text{sim}(\text{pește}, \text{ușă}) =$





# Mecanismul de “atenție” (attention)

## ❑ Similaritatea între cuvinte

Cuvântul	<i>caracterul animalic</i>	<i>nivelul tehnologic</i>
Pisică	4	1
Pește	3	0
Ușă	0	4

### ❑ $\text{sim}(\text{pisică}, \text{pește}) =$

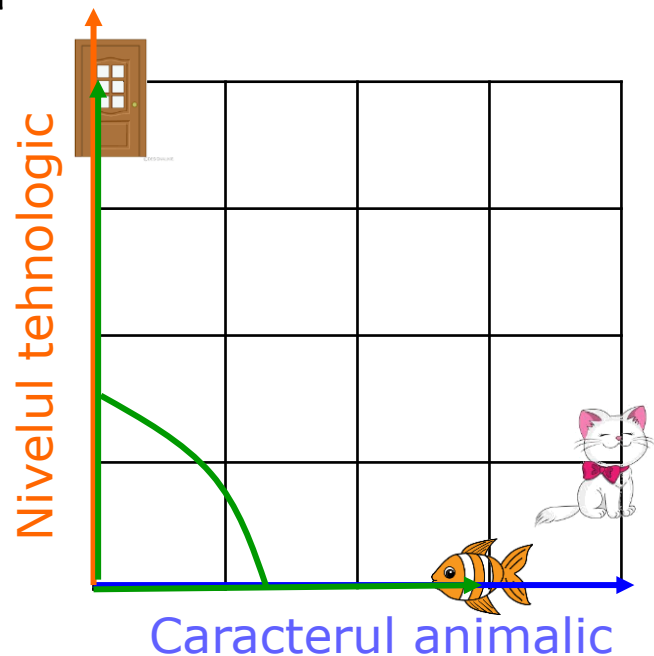
$$\begin{aligned}\text{sim}([4,1], [3,0]) &= (4*3 + 1*0) / (\sqrt{17} * \sqrt{9}) \\ &= 12 / (4.12 * 3) = 0.97\end{aligned}$$

### ❑ $\text{sim}(\text{pisică}, \text{ușă}) =$

$$\begin{aligned}\text{sim}([4,1], [0,4]) &= (4*0 + 1*4) / (\sqrt{17} * \sqrt{16}) \\ &= 4 / (4.12 * 4) = 0.24\end{aligned}$$

### ❑ $\text{sim}(\text{pește}, \text{ușă}) =$

$$\begin{aligned}\text{sim}([3,0], [0,4]) &= (3*0 + 0*4) / (\sqrt{9} * \sqrt{16}) \\ &= 0\end{aligned}$$

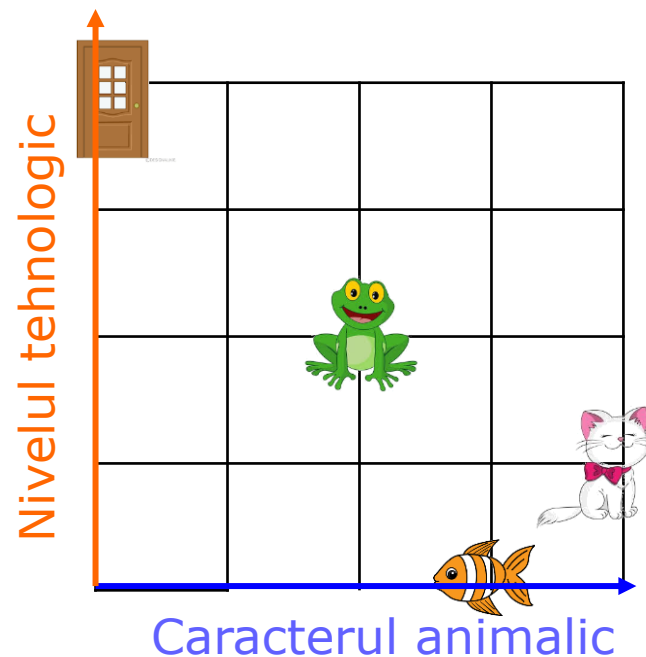


# Mecanismul de “atenție” (attention)

## □ Contextul cuvintelor – matricea de afinitate

■ *S-a stricat **broasca** de la **ușă***

	broască	ușă
broască		
ușă		



# Mecanismul de “atenție” (attention)

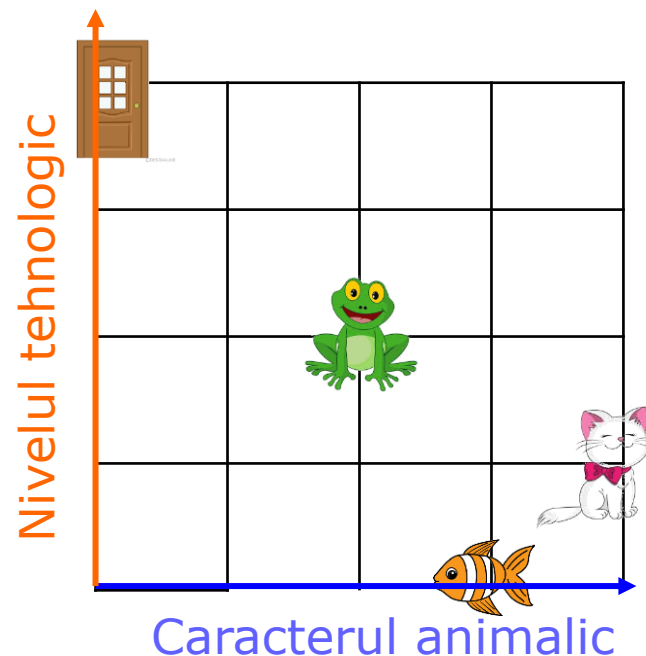
## Contextul cuvintelor – matricea de afinitate

■ S-a stricat **broasca** de la **ușă**

	broască	ușă
broască		
ușă		

■  $\text{sim}(\text{broască}, \text{broască}) = 1$

■  $\text{sim}(\text{broască}, \text{ușă}) =$   
 $\text{sim}([2,2], [0,4]) =$   
 $(2*0 + 2*4) / (\sqrt{8} * \sqrt{16}) =$   
 $0.7$



# Mecanismul de “atenție” (attention)

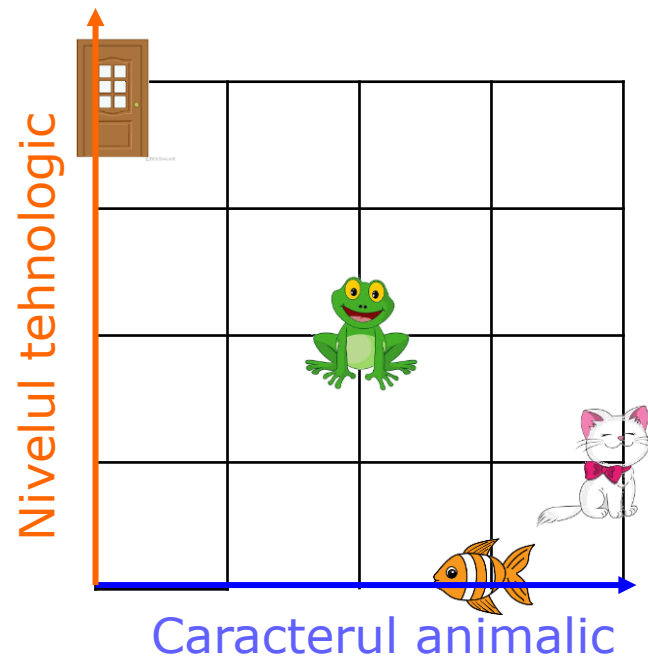
## Contextul cuvintelor – matricea de afinitate

■ S-a stricat **broasca** de la **ușă**

	broască	ușă
broască	1.0	0.7
ușă	0.7	1.0

■  $\text{sim}(\text{broască}, \text{broască}) = 1$

■  $\text{sim}(\text{broască}, \text{ușă}) =$   
 $\text{sim}([2,2], [0,4]) =$   
 $(2*0 + 2*4) / (\sqrt{8} * \sqrt{16}) =$   
 $0.7$



# Mecanismul de “atenție” (attention)

---

## □ Contextul cuvintelor – matricea de afinitate

■ *S-a stricat **broasca** de la **ușă***

	broască	ușă
broască	1.0	0.7
ușă	0.7	1.0

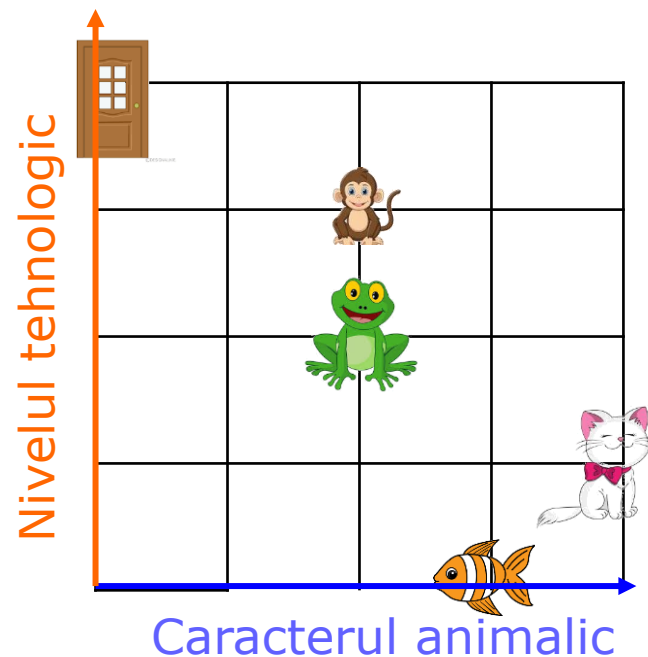
□  $\text{broască} = 1.0 * \text{broască} + 0.7 * \text{ușă}$

# Mecanismul de “atenție” (attention)

## □ Contextul cuvintelor – matricea de afinitate

- O **broască** s-a întâlnit cu o **maimuță**

	Broască	Maimuță
Broască		
Maimuță		



# Mecanismul de “atenție” (attention)

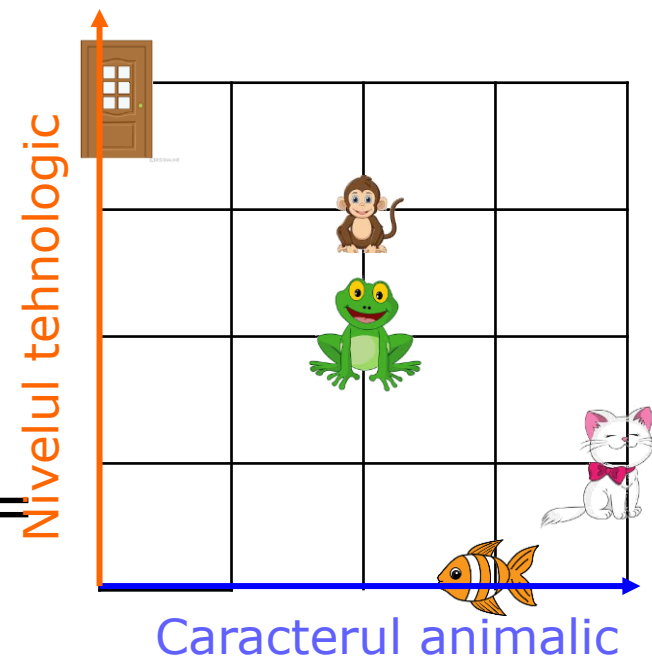
## Contextul cuvintelor – matricea de afinitate

- O **broască** s-a întâlnit cu o **maimuță**

	Broască	Maimuță
Broască		
Maimuță		

- $\text{sim}(\text{broască}, \text{broască}) = 1$

- $\text{sim}(\text{broască}, \text{măimuță}) =$   
 $\text{sim}([2,2], [2,3]) =$   
 $(2*2 + 2*3) / (\sqrt{8} * \sqrt{13}) =$   
 $12 / \sqrt{2} = 0.9$



# Mecanismul de “atenție” (attention)

## Contextul cuvintelor – matricea de afinitate

- O **broască** s-a întâlnit cu o **maimuță**

	Broască	Maimuță
Broască	1.0	0.9
Maimuță	0.9	1.0

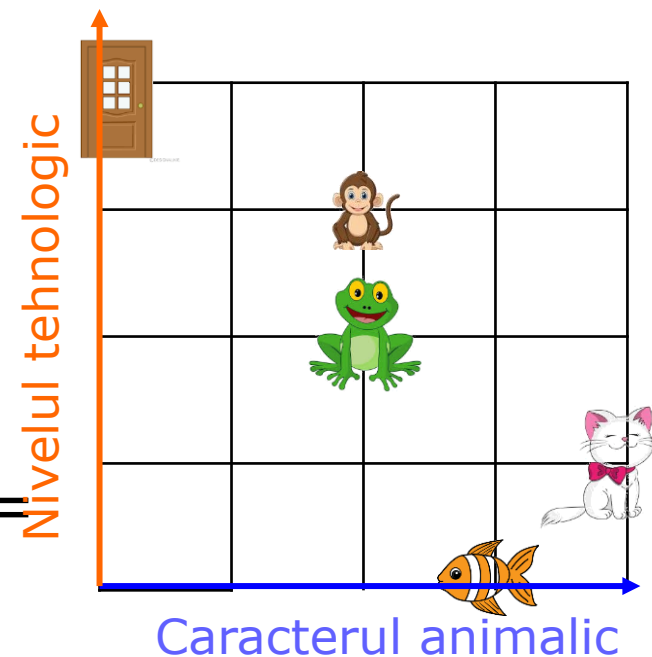
- $\text{sim}(\text{broască}, \text{broască}) = 1$

- $\text{sim}(\text{broască}, \text{mămuță}) =$

$$\text{sim}([2,2], [2,3]) =$$

$$(2*2 + 2*3) / (\sqrt{8} * \sqrt{13}) =$$

$$12 / \sqrt{2} = 0.9$$





# Mecanismul de “atenție” (attention)

---

## □ Contextul cuvintelor – matricea de afinitate

■ O **broască** s-a întâlnit cu o **maimuță**

	broască	Maimuță
Broască	1.0	0.9
Maimuță	0.9	1.0

□  $\text{broască} = 1.0 * \text{broască} + 0.9 * \text{maimuță}$

# Mecanismul de “atenție” (attention)

## □ Contextul cuvintelor – matricea de afinitate

- O **broască** s-a întâlnit cu o **maimuță**

- S-a stricat **broasca** de la **ușă**

broască = 1 \* **broască** + 0.9 \* maimuță

broască = 1 \* **broască** + 0.7 \* ușă



Normalizare – de care?

broască = ? \* **broască** + ? \* maimuță

broască = ? \* **broască** + ? \* ușă

# Mecanismul de “atenție” (attention)

## □ Contextul cuvintelor – matricea de afinitate

■ O **broască** s-a întâlnit cu o **maimuță**

■ S-a stricat **broasca** de la **ușă**

broască = 1 \* **broască** + 0.9 \* maimuță

broască = 1 \* **broască** + 0.7 \* ușă



Normalizare - softmax

broască = 0.52 \* **broască** + 0.48 \* maimuță

broască = 0.57 \* **broască** + 0.43 \* ușă

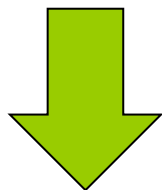
# Mecanismul de “atenție

## Contextul cuvintelor

- O **broască** s-a în
- S-a stricat **broască**

broască = 1 \* broas

broască = 1 \* broască

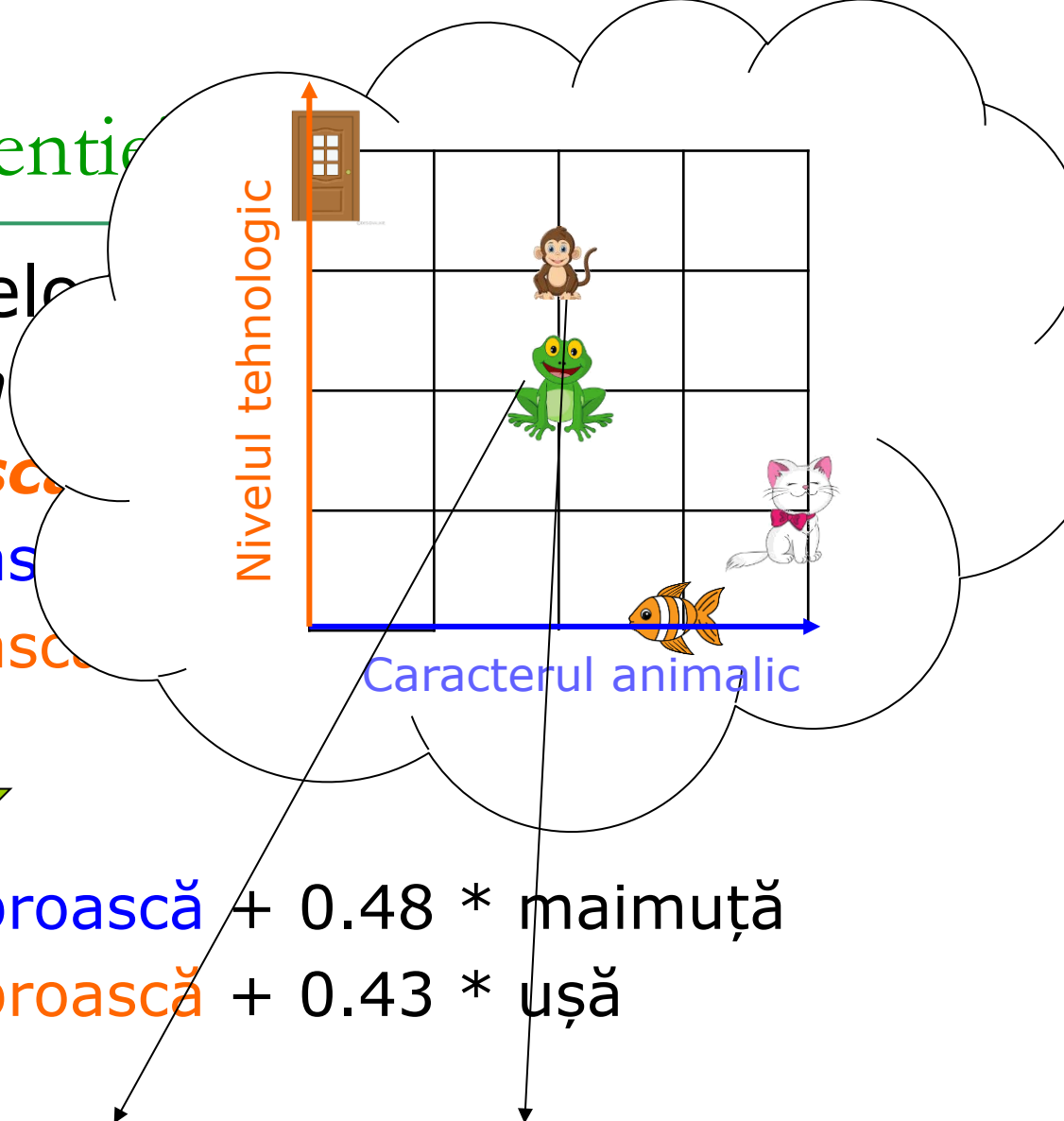


broască = 0.52 \* broască + 0.48 \* maimuță

broască = 0.57 \* broască + 0.43 \* ușă

broască = 0.52 \* [2, 2] + 0.48 \* [2, 3]

broască = 0.57 \* [2, 2] + 0.43 \* [0, 4]



# Mecanismul de “atenție

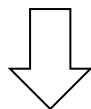
## Contextul cuvintelor – matrice

■ O **broască** s-a întâlnit

■ S-a stricat **broasca** de

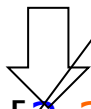
broască = 1 \* **broască** + 0.

broască = 1 \* **broască** + 0



broască = 0.52 \* **broască** + 0.48

broască = 0.57 \* **broască** + 0.43 \* ușă



broască = 0.52 \* [2,2] + 0.48 \* [2,3]

broască = 0.57 \* [2,2] + 0.43 \* [0,4]

broască = [2.00,2.48]

broască = [1.14,2.86]

Nivelul tehnologic

Caracterul animalic

V (Values)

	Broască	Maimuță
Broască	0.52	0.48
Maimuță	0.48	0.52

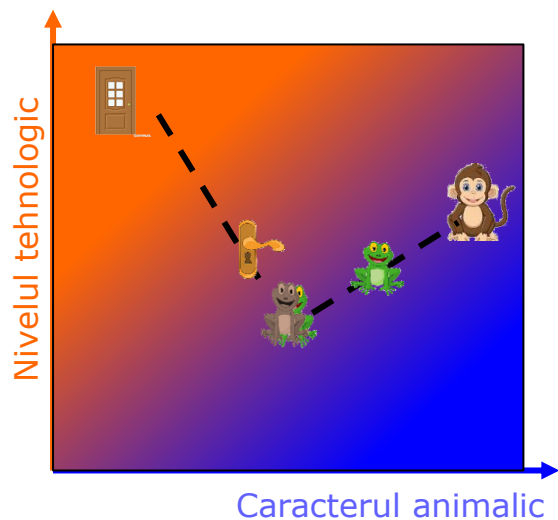
# Mecanismul de “atenție” (attention)

---

Un alt exemplu:

O **broască** s-a întâlnit cu o **maimuță**

S-a stricat **broasca** de la **ușă**

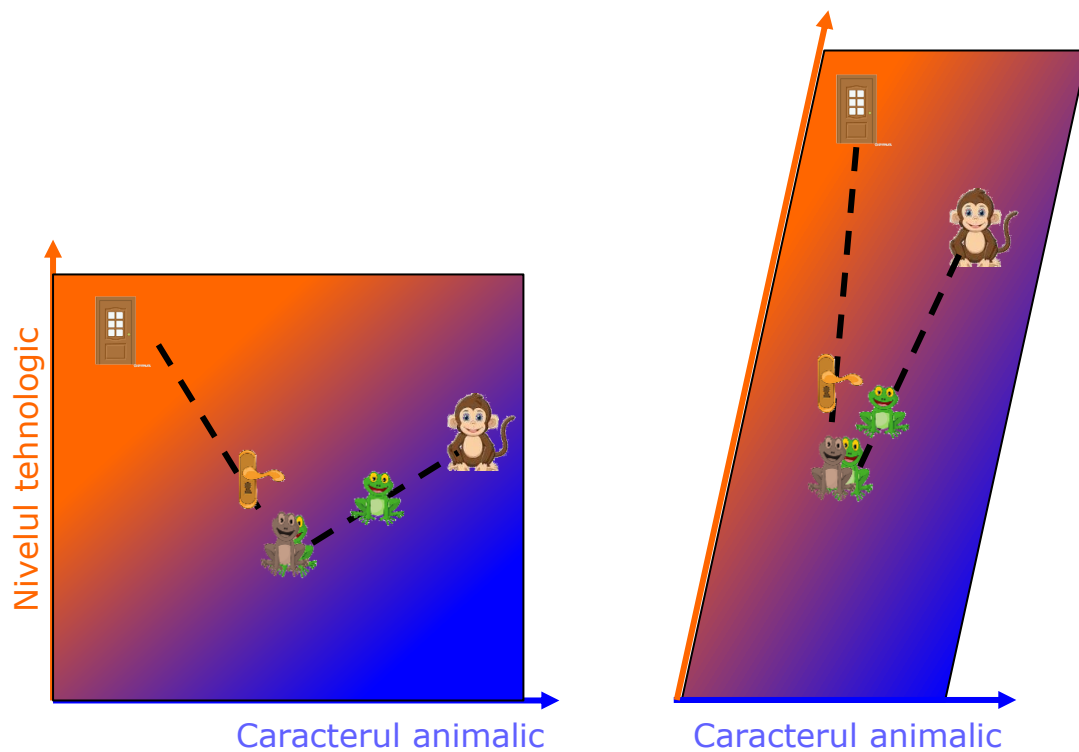


# Mecanismul de “atenție” (attention)

Un alt exemplu:

O **broască** s-a întâlnit cu o **maimuță**

S-a stricat **broasca** de la **ușă**

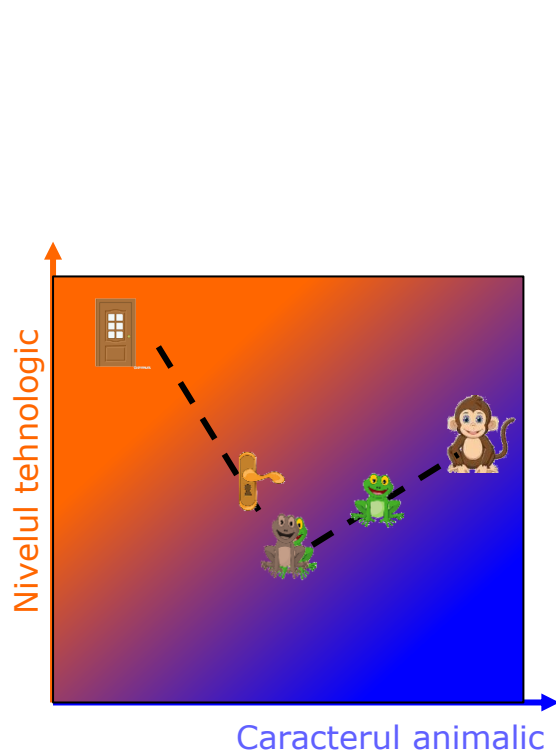


# Mecanismul de “atenție” (attention)

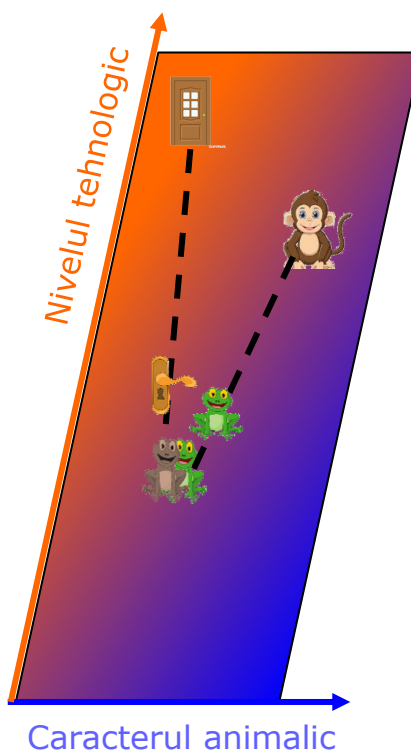
Un alt exemplu:

O **broască** s-a întâlnit cu o **maimuță**

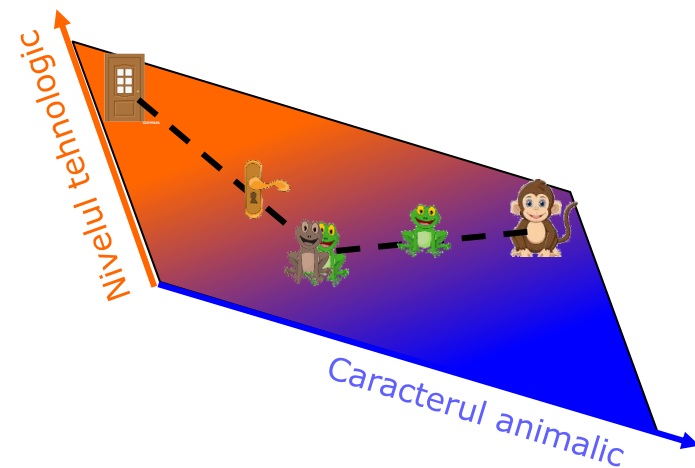
S-a stricat **broasca** de la **ușă**



a) reprez1



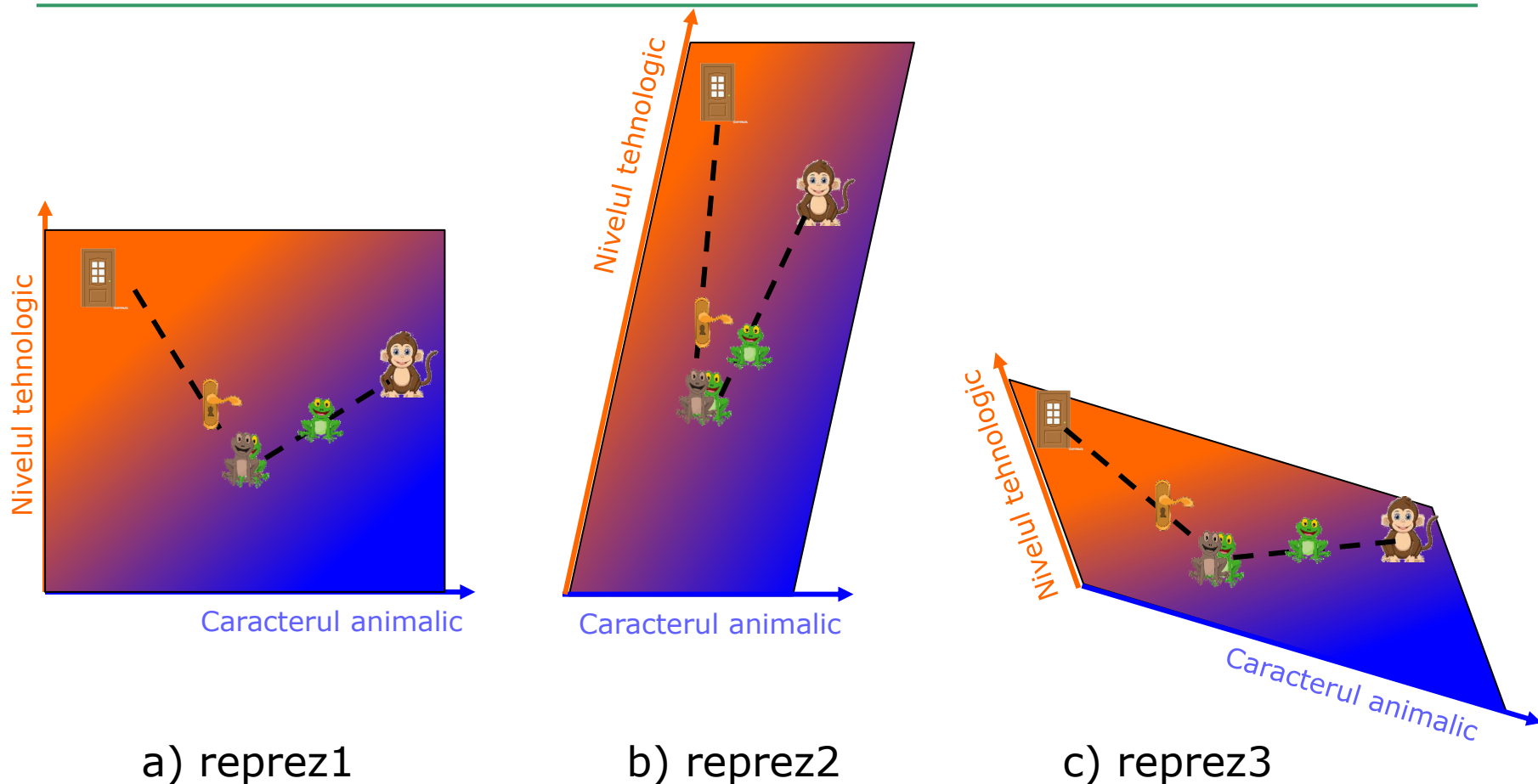
b) reprez2



c) reprez3

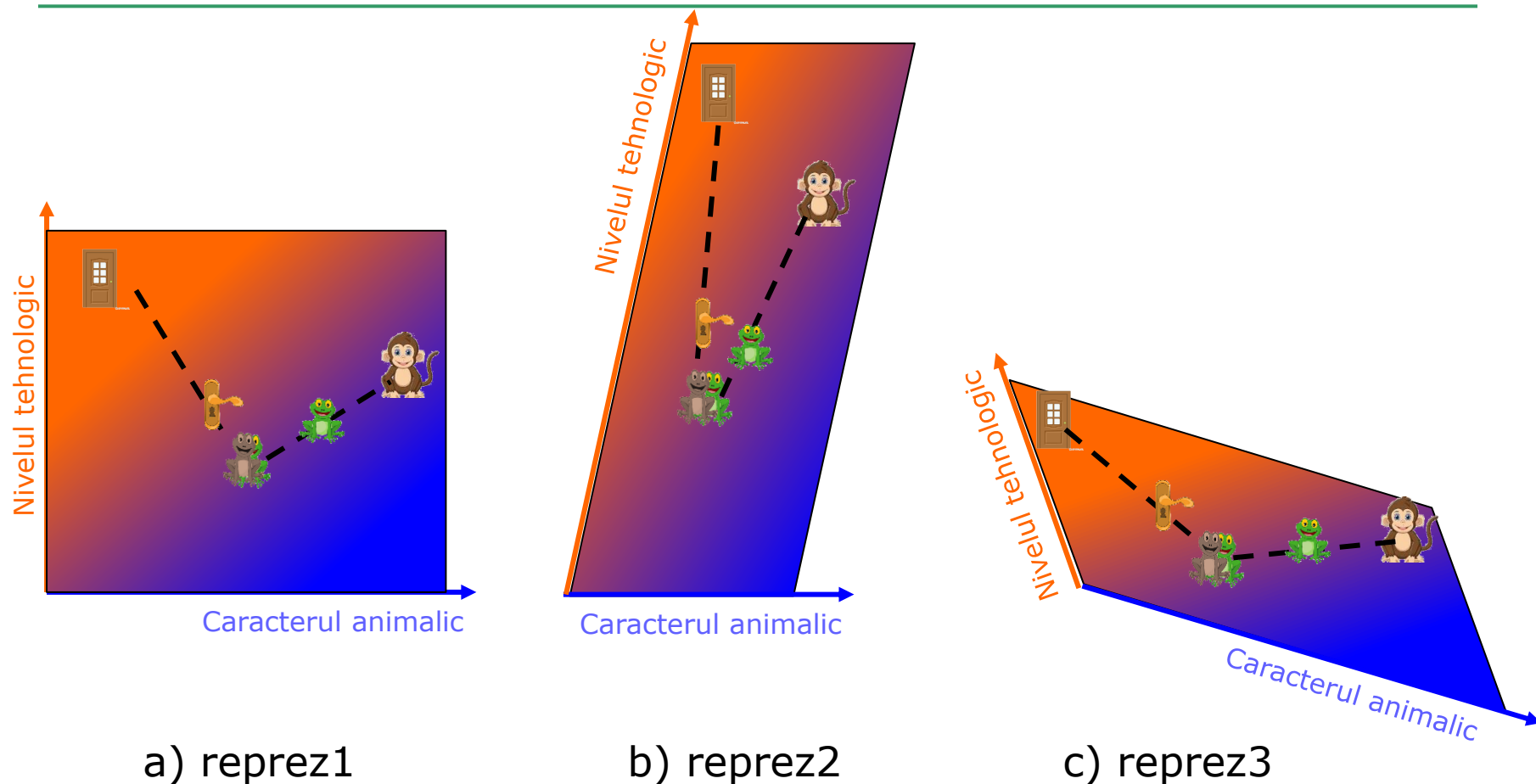


# Mecanismul de “atenție” (attention)



**Q1: reprez1 = f(reprez2) = g(reprez3) ???**

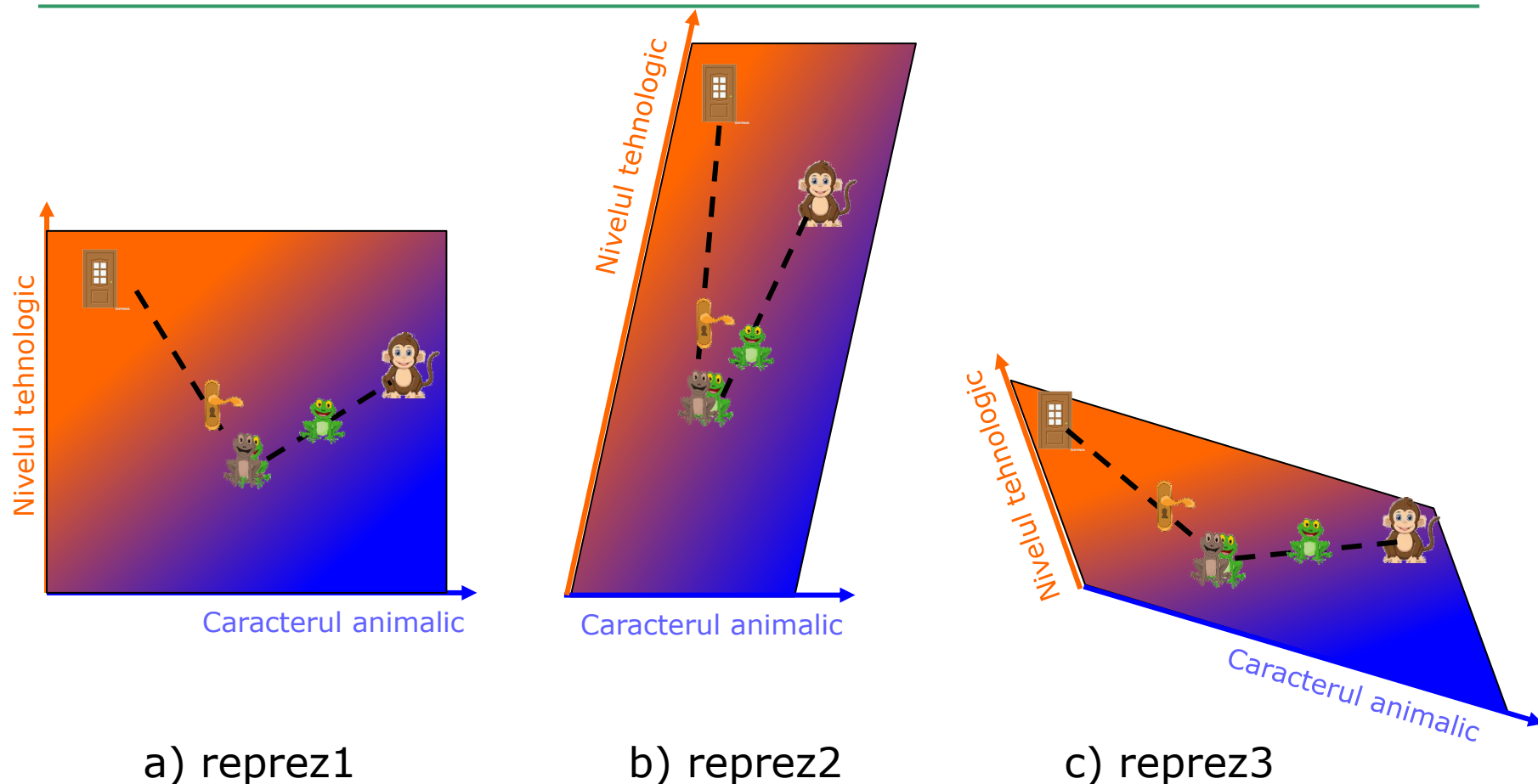
# Mecanismul de “atenție” (attention)



**Q1: reprez1 = f(reprez2) = g(reprez3) ???**

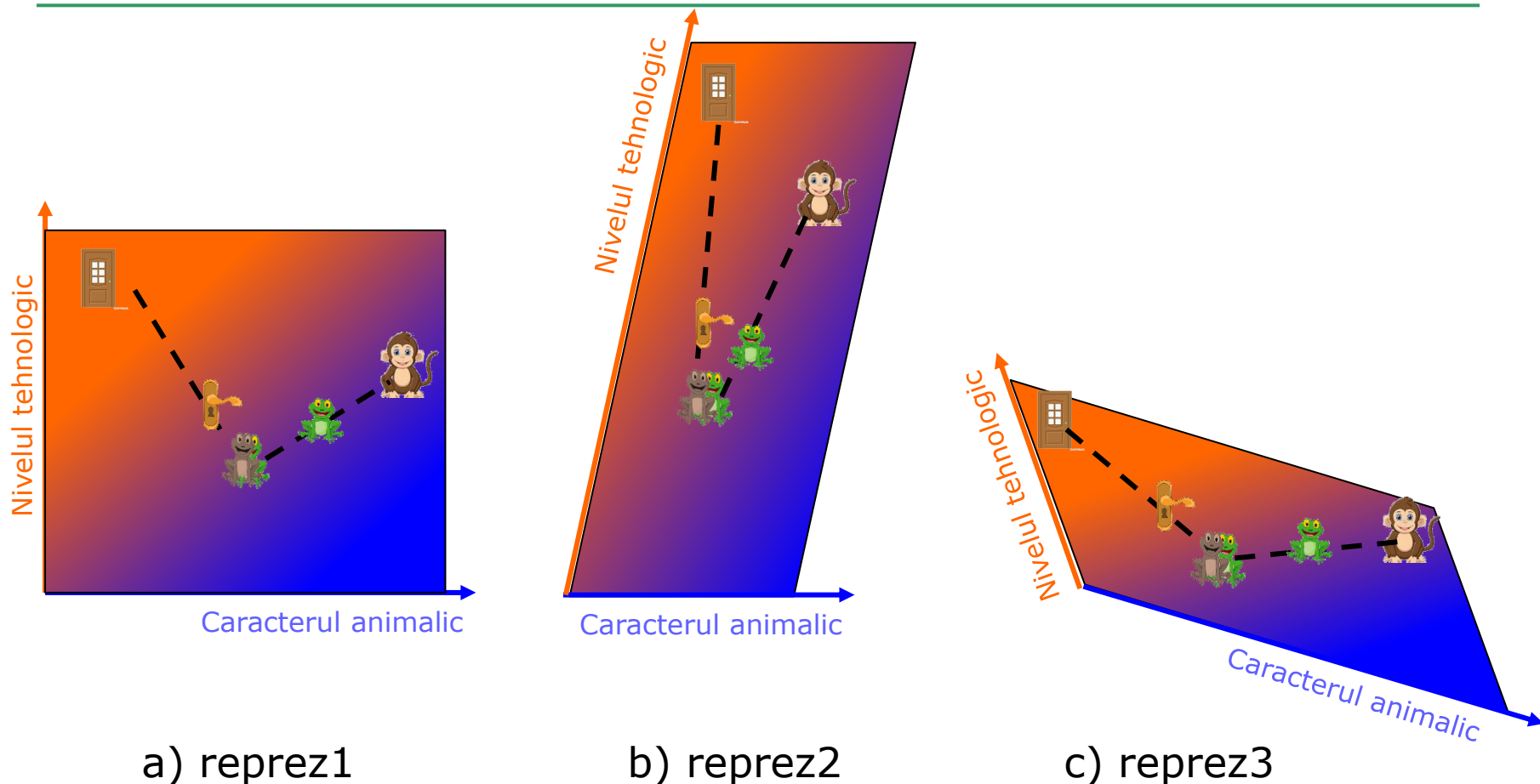
**Da!!! Transformare liniară!!!**

# Mecanismul de “atenție” (attention)



**Q2: Care reprezentare e mai bună pentru a stabili similaritatea (deci a face diferența între sensurile lui *broască*)?**

# Mecanismul de “atenție” (attention)



a) reprez1

b) reprez2

c) reprez3

**Q2: Care reprezentare e mai bună pentru a stabili similaritatea (deci a face diferența între sensurile lui *broască*)?**

**Reprezentarea 1 sau 3!**

# Mecanismul de “atenție” (attention)

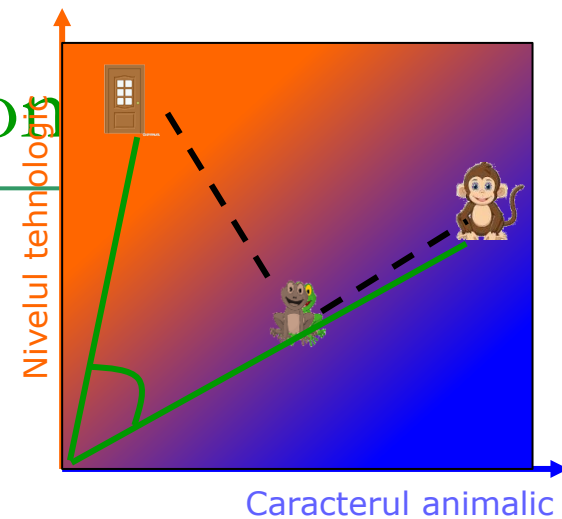
□  $\text{Sim}(\text{ușă}, \text{mămuță}) =$

$$\text{sim}(\begin{bmatrix} \text{purple} & \text{orange} \end{bmatrix}, \begin{bmatrix} \text{blue} & \text{orange} \end{bmatrix}) = \begin{bmatrix} \text{purple} & \text{orange} \\ & \text{blue} \end{bmatrix}$$

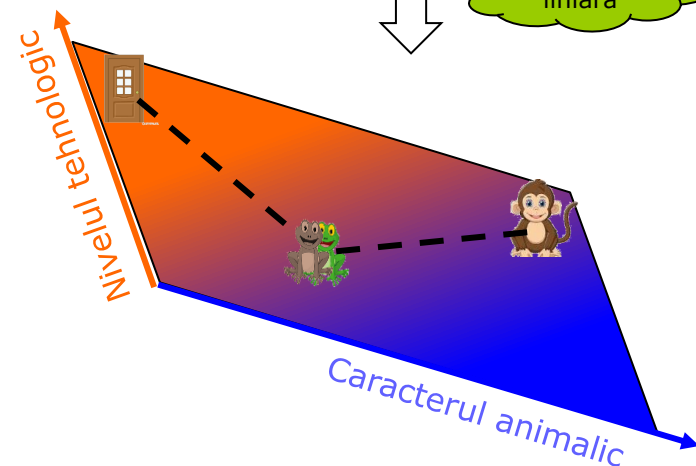
□  $\text{Sim}(\text{ușă}, \text{mămuță}) =$

$$\text{sim}(\begin{bmatrix} \text{purple} & \text{orange} \end{bmatrix}, \begin{bmatrix} \text{blue} & \text{orange} \end{bmatrix}) = \begin{bmatrix} \text{purple} & \text{orange} & \begin{bmatrix} \text{green} & \text{green} & \text{green} & \text{green} \end{bmatrix} & \begin{bmatrix} \text{pink} & \text{pink} \\ \text{pink} & \text{pink} \\ \text{pink} & \text{pink} \end{bmatrix} & \begin{bmatrix} \text{blue} \\ \text{orange} \end{bmatrix} \end{bmatrix}$$

Transformare liniară



Transformare liniară



# Mecanismul de “atenție” (attention)

□  $\text{Sim}(\text{ușă}, \text{mămuță}) =$

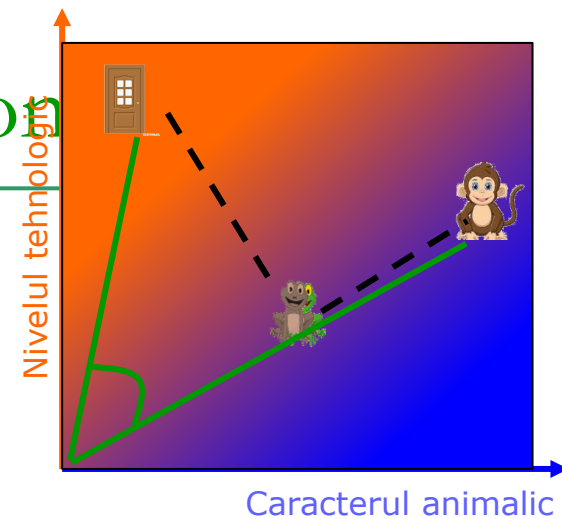
$$\text{sim}(\begin{bmatrix} \text{purple} & \text{orange} \end{bmatrix}, \begin{bmatrix} \text{blue} & \text{orange} \end{bmatrix}) = \begin{bmatrix} \text{purple} & \text{orange} \\ \text{blue} & \text{orange} \end{bmatrix}$$

□  $\text{Sim}(\text{ușă}, \text{mămuță}) =$

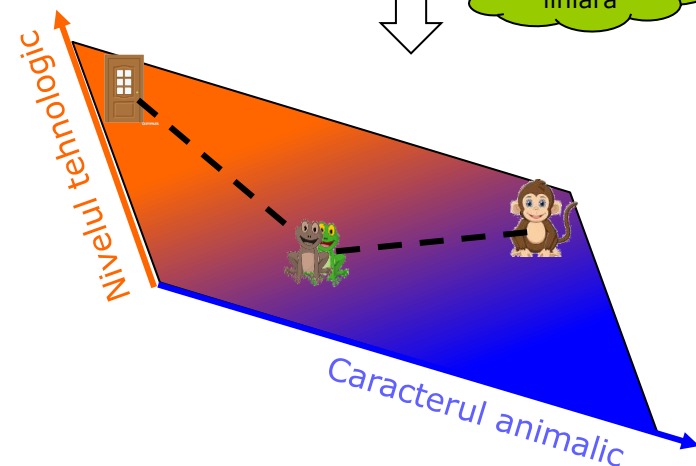
$$\text{sim}(\begin{bmatrix} \text{purple} & \text{orange} \end{bmatrix}, \begin{bmatrix} \text{blue} & \text{orange} \end{bmatrix}) = \begin{bmatrix} \text{purple} & \text{orange} \end{bmatrix} \begin{bmatrix} \text{green} & \text{green} & \text{green} & \text{green} \\ \text{green} & \text{green} & \text{green} & \text{green} \end{bmatrix} \begin{bmatrix} \text{pink} & \text{pink} \\ \text{pink} & \text{pink} \\ \text{pink} & \text{pink} \\ \text{pink} & \text{pink} \end{bmatrix} \begin{bmatrix} \text{blue} \\ \text{orange} \end{bmatrix}$$

K (Key) = 

Q (Queries) =



Transformare liniară



Valorile optime - ANN

# Mecanismul de “atenție

## Contextul cuvintelor

- O **broască** s-a în
- S-a stricat **broască**.

broască = 1 \* broas

broască = 1 \* broască

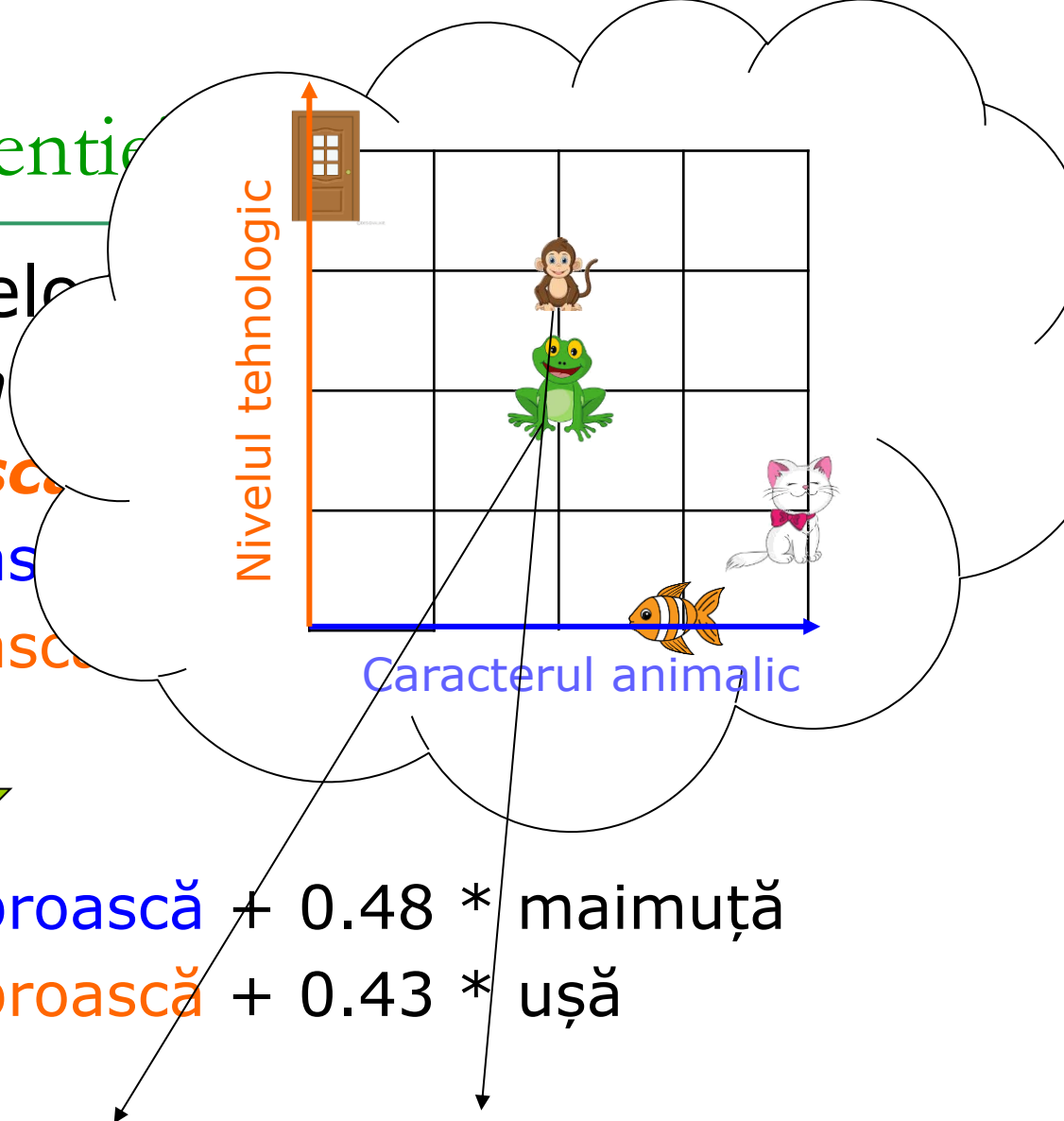


broască = 0.52 \* broască + 0.48 \* maimuță

broască = 0.57 \* broască + 0.43 \* ușă

broască = 0.52 \* [2, 2] + 0.48 \* [2, 3]

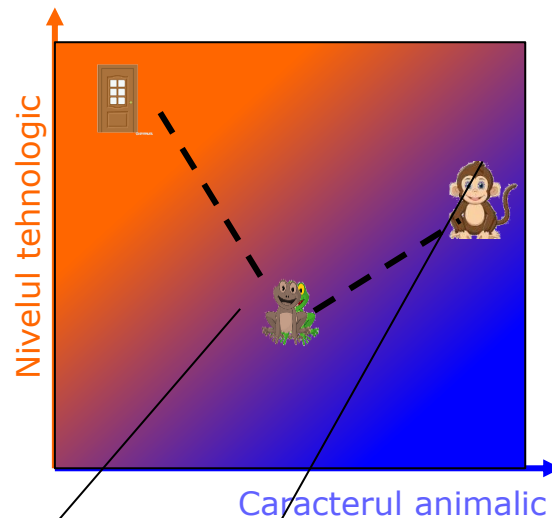
broască = 0.57 \* [2, 2] + 0.43 \* [0, 4]



# Mecanismul de “atenție

## Contextul cuvintelor

- O **broască** s-a în
- S-a stricat **broască**



broască = 1 \* broască + 0.9 \* maimuță

broască = 1 \* broască + 0.7 \* ușă

broască = 0.52 \* broască + 0.48 \* maimuță

broască = 0.57 \* broască + 0.43 \* ușă

broască = 0.52 \* [2,2] + 0.48 \* [2,3]

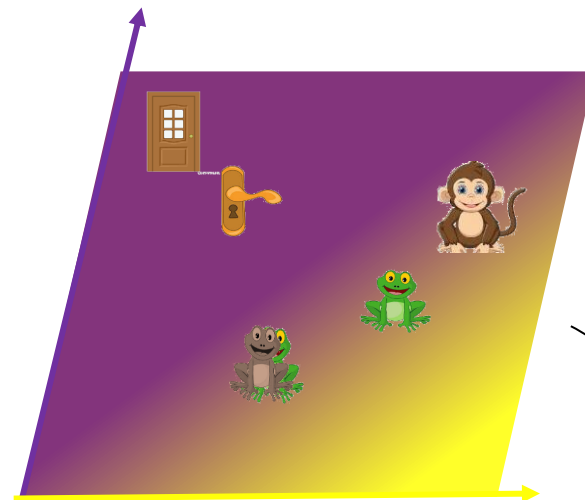
broască = 0.57 \* [2,2] + 0.43 \* [0,4]



# Mecanismul de “atenție

## □ Contextul cuvintelor

- O **broască** s-a în
- S-a stricat **broască**.



broască = 0.52 \* **broască** + 0.48 \* maimuță

broască = 0.57 \* **broască** + 0.43 \* ușă

broască = 0.52 \* [2,2] + 0.48 \* [2,3]

broască = 0.57 \* [2,2] + 0.43 \* [0,4]

broască = [2.00,2.48]

broască = [1.14,2.86]

# Mecanismul de “atenție” (2)

## Contextul cuvintelor –

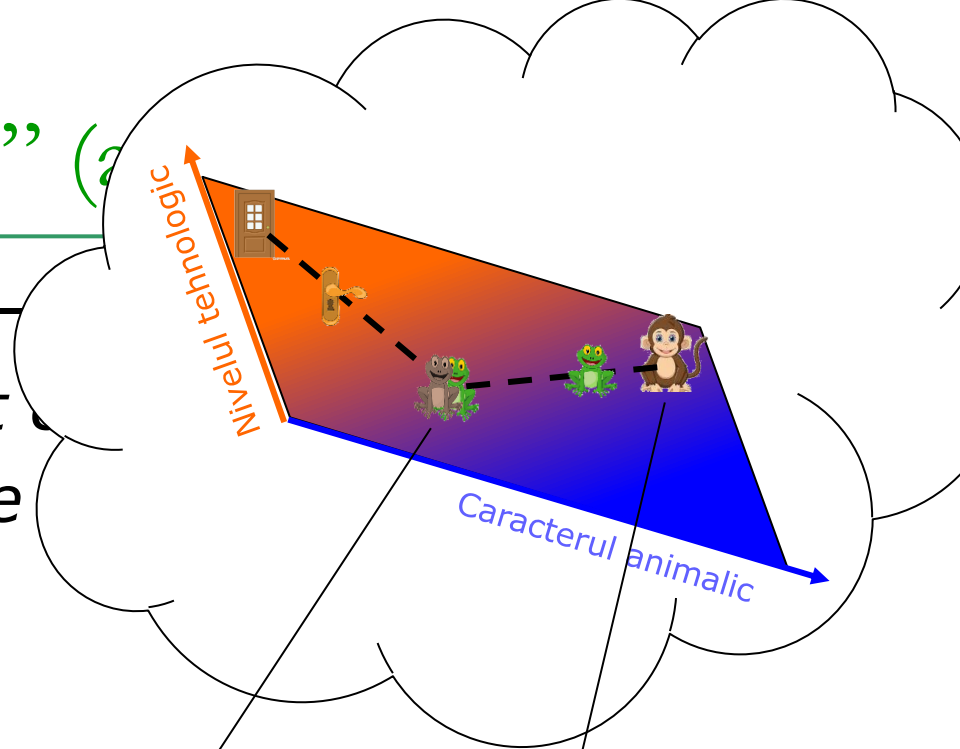
- O **broască** s-a întâlnit
- S-a stricat **broasca** de

broască = 0.52 \* **broască** + 0.48 \* maimuță

broască = 0.57 \* **broască** + 0.43 \* ușă

broască = 0.52 \* [broască(**KQ**,**KQ**)] + 0.48 \* [maimuță(**KQ**,**KQ**)]

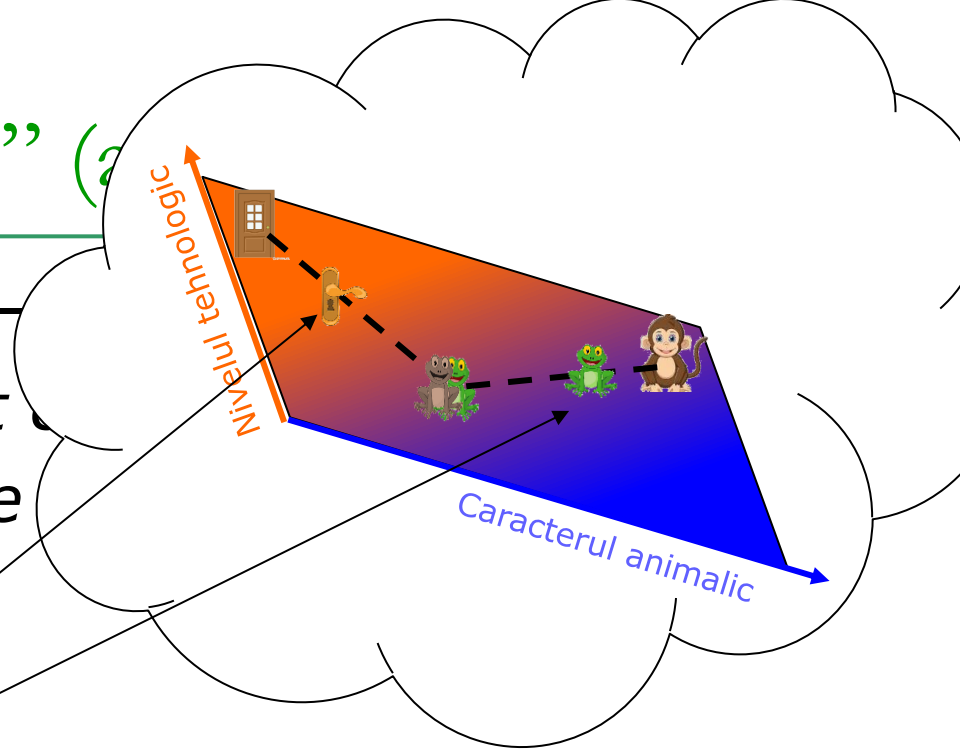
broască = 0.57 \* [broască(**KQ**,**KQ**)] + 0.43 \* [ușă(**KQ**,**KQ**)]



# Mecanismul de “atenție” (2)

## □ Contextul cuvintelor –

- O **broască** s-a întâlnit
- S-a stricat **broasca** de



broască = 0.52 \* **broască** + 0.48 \* maimuță

broască = 0.57 \* **broască** + 0.43 \* ușă

broască = 0.52 \* [broască(**KQ**,**KQ**)] + 0.48 \* [maimuță(**KQ**,**KQ**)]

broască = 0.57 \* [broască(**KQ**,**KQ**)] + 0.43 \* [ușă(**KQ**,**KQ**)]

# Mecanismul de “atenție” (a)

## Contextul cuvintelor – m

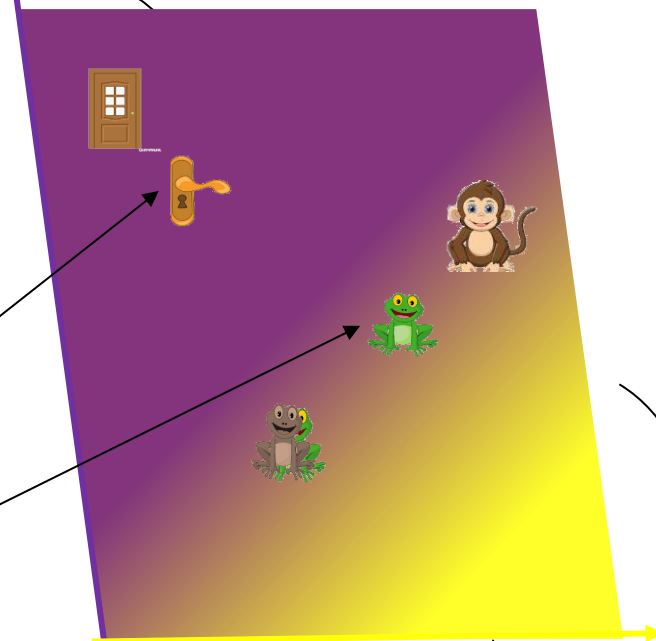
- O **broască** s-a întâlnit
- S-a stricat **broasca** de

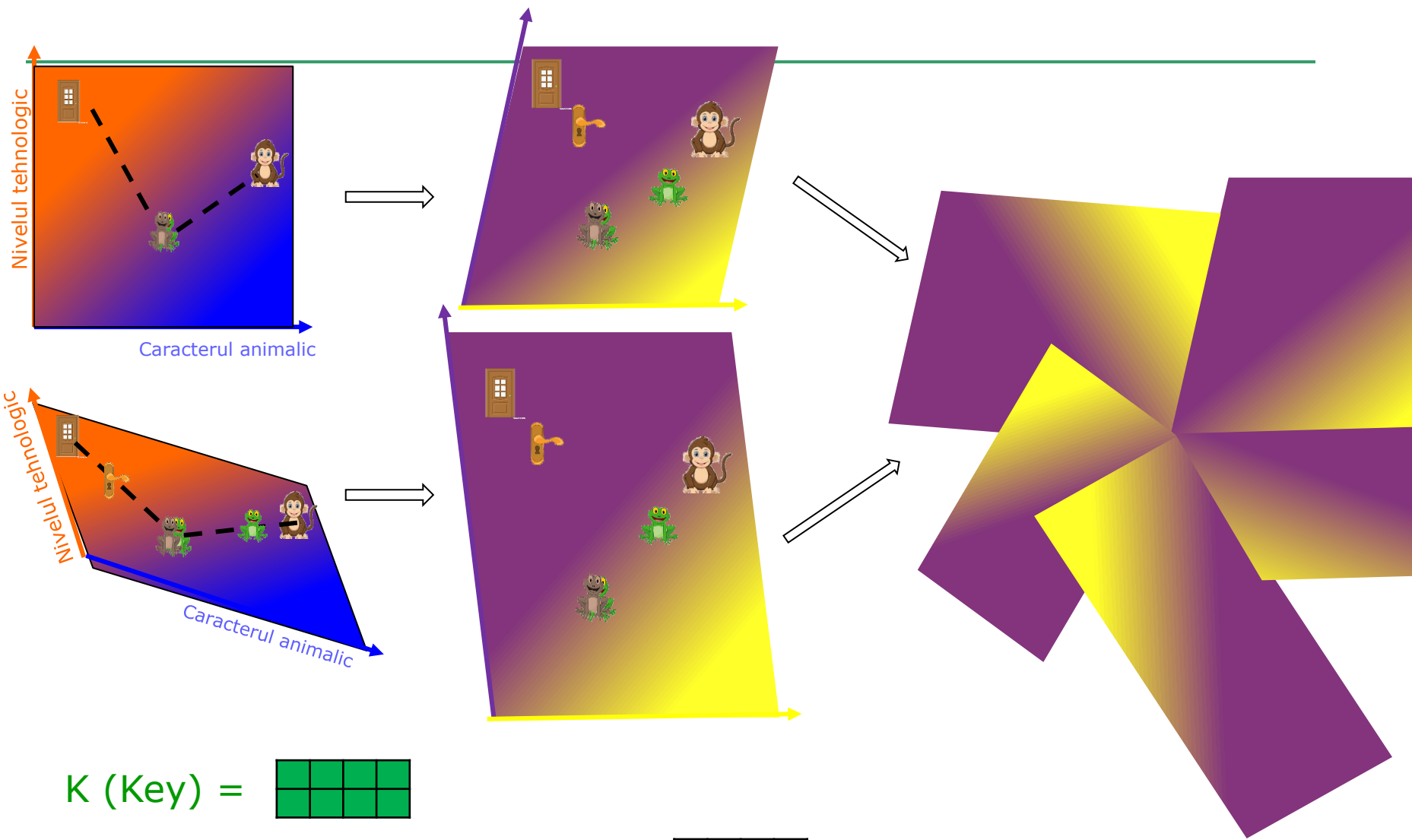
broască = 0.52 \* **broască** + 0.48 \* maimuță

broască = 0.57 \* **broască** + 0.43 \* ușă

broască = 0.52 \* [broască(**KQ**,**KQ**)] + 0.48 \* [maimuță(**KQ**,**KQ**)]

broască = 0.57 \* [broască(**KQ**,**KQ**)] + 0.43 \* [ușă(**KQ**,**KQ**)]

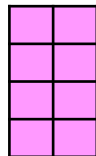




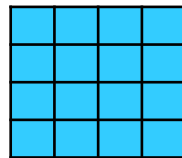
K (Key) =



Q (Queries) =



V (Values)



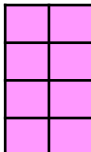
Inteligență artificială - generative AI

# Mecanismul de “atenție” (attention)

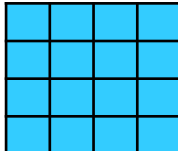
---

- Identificarea celor mai bune embedding-uri pentru a stabili similaritatea între elemente
  - Se bazează pe feature-urile elementelor (e.g. Caracterul animalic, nivelul tehnologic)

K (Key) = 

Q (Queries) = 

- Identificarea celor mai bune embedding-uri pentru a prezice următorul cuvânt
  - Se bazează pe co-apariția elementelor în același context

V (Values) = 

## Mecanismul de “atenție” (attention)

[illegible]

# Mecanismul de “atentie” (attention)

---

## ❑ Interogari intr-o BD

### ■ Input:

- ❑ O multime de recorduri  $\{ (key_i, value_i), i = 1, 2, \dots, m \}$
- ❑ Un query  $q$

### ■ Output

- ❑  $value_j$  a.i.  $q$  e similar cu  $key_j$

### ■ Exemplu

- ❑  $DB = \{ (Apostol, Mihai), (Baciu, Alina), (Cretu, Gabriel), (Pop, Ionica), (Popa, Madalina), (Popescu, Cosmina) \}$
- ❑ Query = “Pop”
- ❑ Answer:



# Mecanismul de “atenție” (attention)

## ❑ Interogari intr-o BD

### ■ Input:

- ❑ O multime de recorduri  $\{ (key_i, value_i), i = 1, 2, \dots, m \}$
- ❑ Un query  $q$

### ■ Output

- ❑  $value_j$  a.i.  $q$  e similar cu  $key_j$

### ■ Exemplu

- ❑  $DB = \{(Apostol, Mihai), (Baciu, Alina), (Cretu, Gabriel), (Pop, Ionica), (Popa, Madalina), (Popescu, Cosmina)\}$
- ❑ Query = “Pop”
- ❑ Answer:
  - Ionica – perfect match (e.g. Hamming Distance)
  - Ionica, Madalina, Cosmina – partial match (e.g. Levenshtein Distance), top-k raspunsuri

Q=Pop	Apostol	Baciu	Cretu	Pop	Popa	Popescu
Hamming	6	5	5	0	4	7
Levenshtein	6	5	5	0	1	4

# Mecanismul de “atentie” (attention)

## ❑ Interogari intr-o baza de date

### ■ Exemplu - text

- ❑ DB = {(Apostol, Mihai), (Baciu, Alina), (Cretu, Gabriel), (Pop, Ionica), (Popa, Madalina), (Popescu, Cosmina)}

- ❑ Q = “Pop”

- ❑ Answer:

Q=Pop	Apostol	Baciu	Cretu	Pop	Popa	Popescu
Hamming	6	5	5	0	4	7
Levenshtein	6	5	5	0	1	4

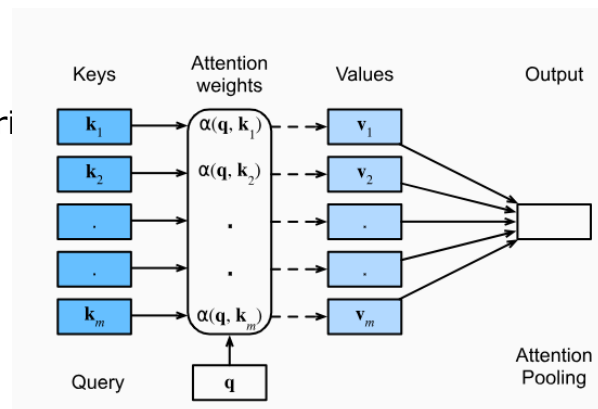
- {Ionica} – perfect match (e.g. Hamming Distance)
- {Ionica, Madalina, Cosmina} – partial match (e.g. Levenshtein Distance), top-k raspunsuri

### ■ Mecanismul de atentie

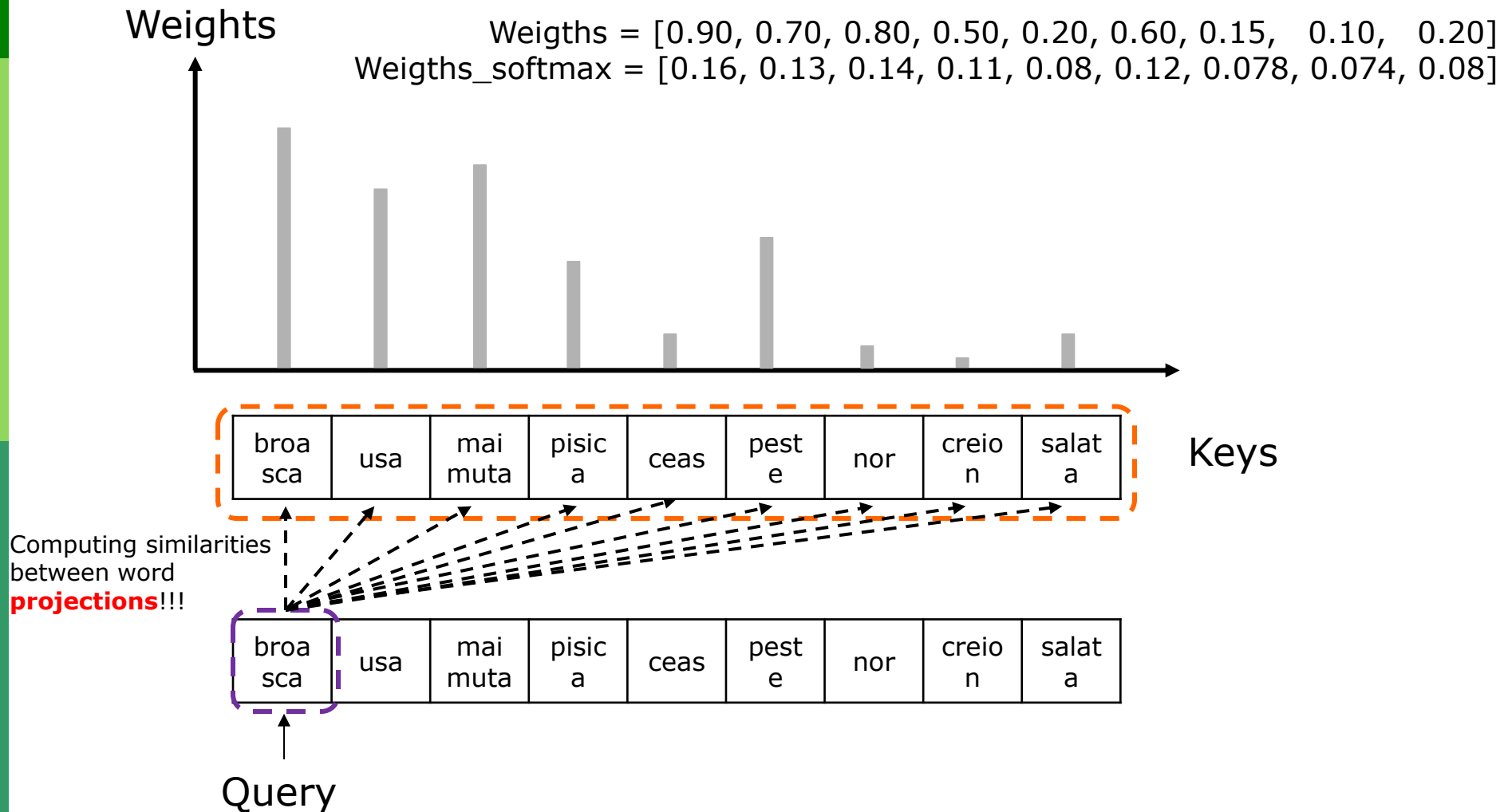
- ❑  $\text{Attention}(q, \text{DB}) = \sum \alpha(q, \text{key}_i) * \text{value}_i$ 
  - Query – focusul current (query-ul pe care modelul il adreseaza in legatura cu un cuvant/token)
  - Key – label-ul sau referinta unui cuvant
  - Value - Informatia actuala codata de un cuvant

- ❑  $\alpha(q, \text{key}_i)$  – attention weights

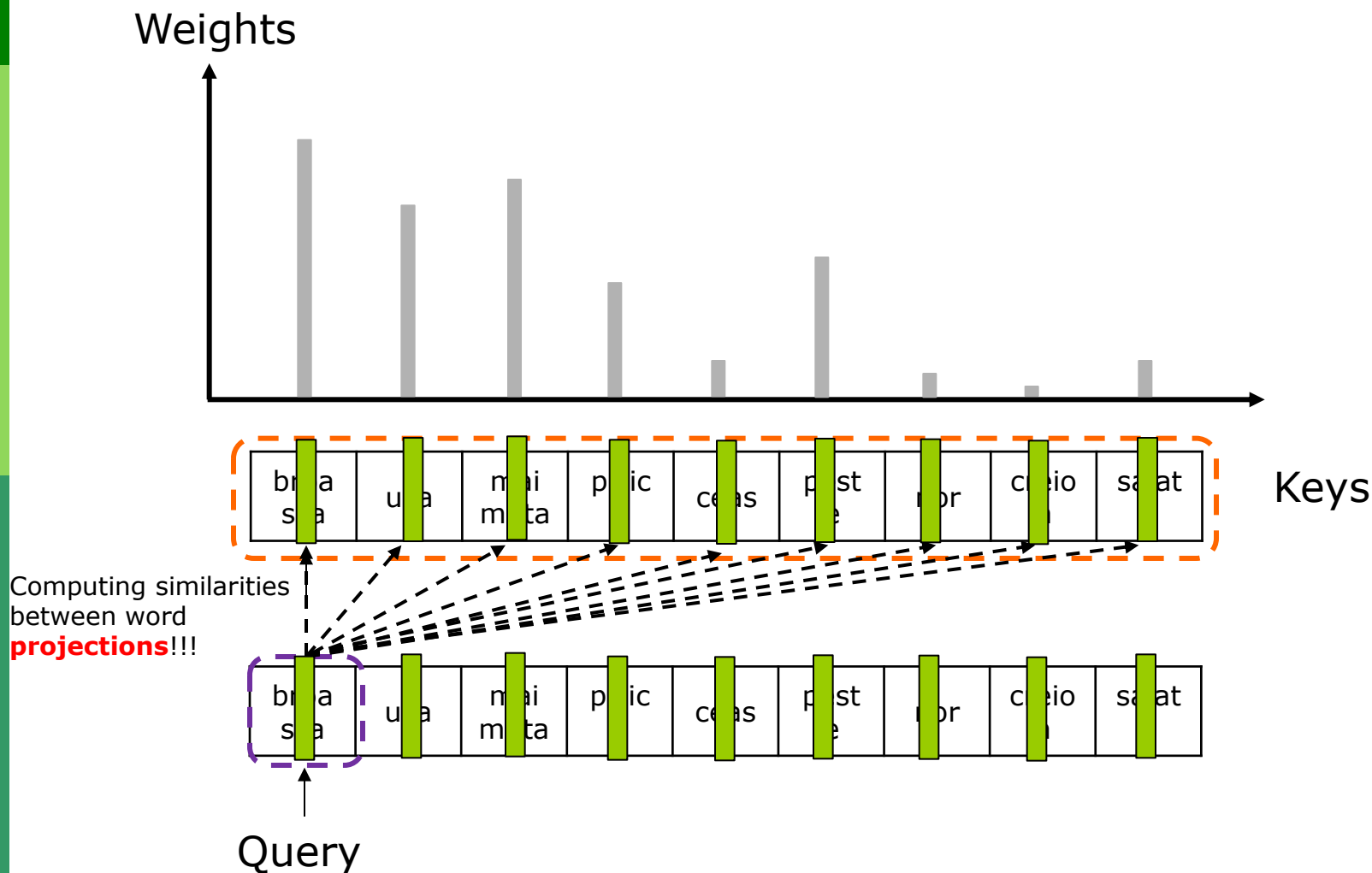
- Daca un singur coefficient e 1 si restul 0 → interogari
- Daca toti coefficientii  $\alpha = 1/m \rightarrow$  Average Pooling
- Pentru a avea coeficienti  $\alpha$  ca ponderi (weights)
  - $\alpha'(q, \text{key}_i) = \alpha(q, \text{key}_i) / \sum \alpha(q, \text{key}_j)$
  - $\alpha''(q, \text{key}_i) = \exp(\alpha(q, \text{key}_i)) / \sum \exp(\alpha(q, \text{key}_j))$



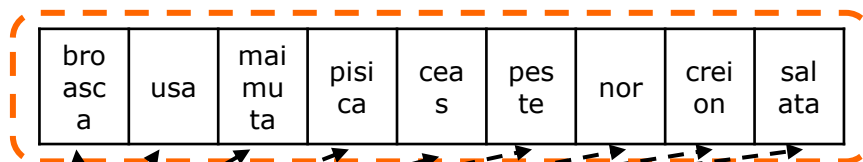
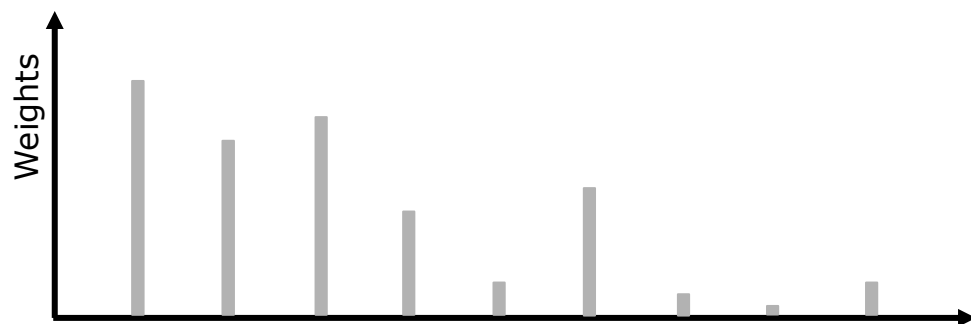
# Mecanismul de “atenție” (attention)



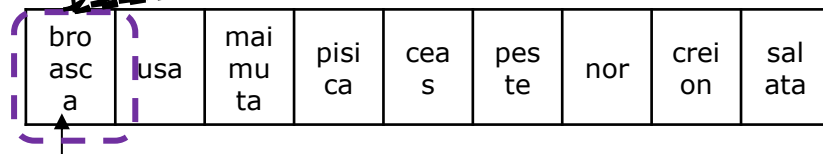
# Mecanismul de “atenție” (attention)



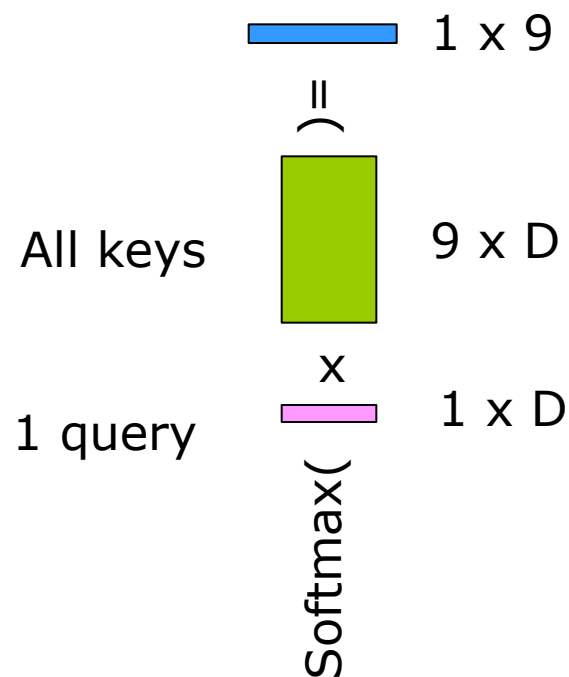
# Mecanismul de “atenție” (attention)



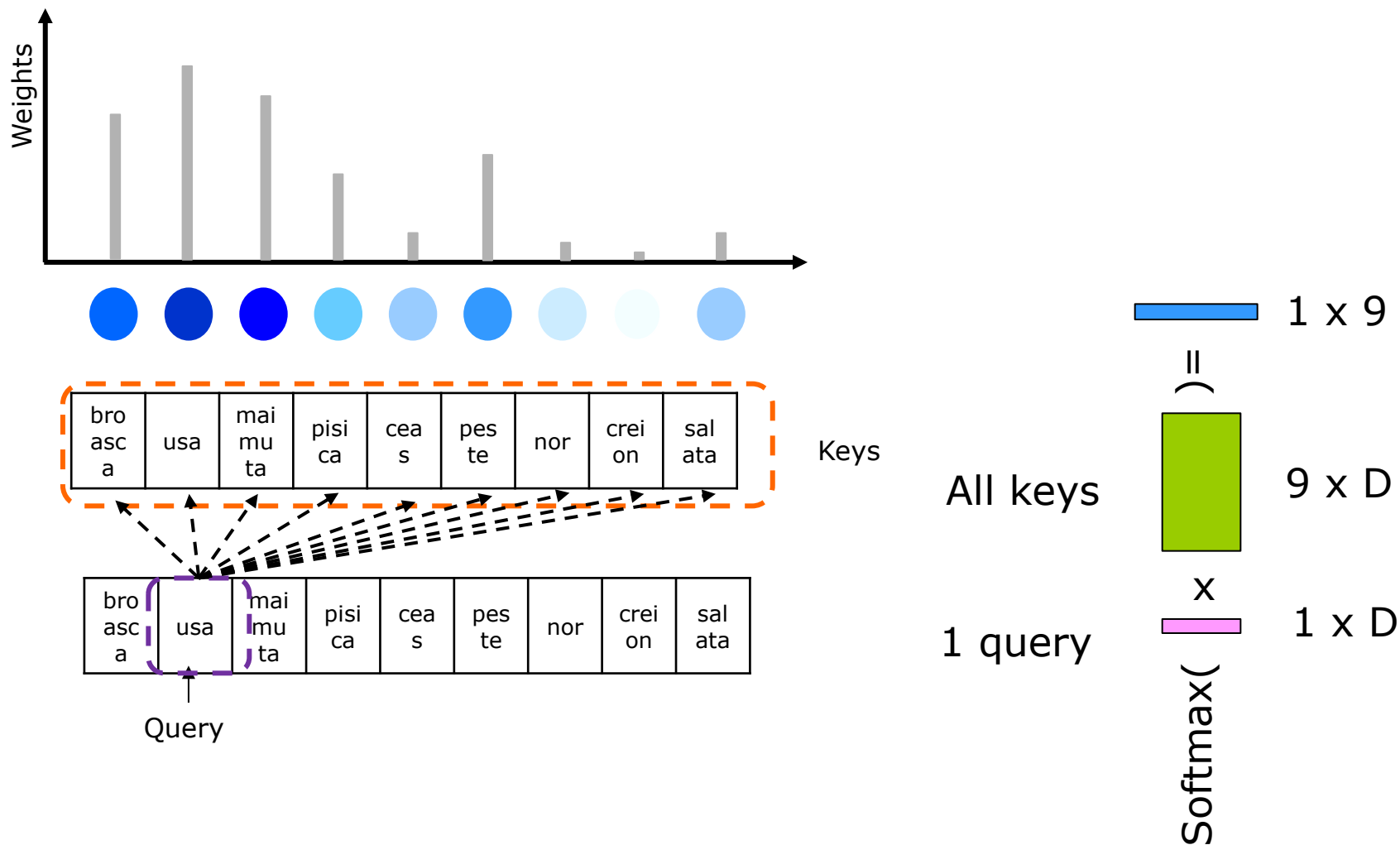
Keys



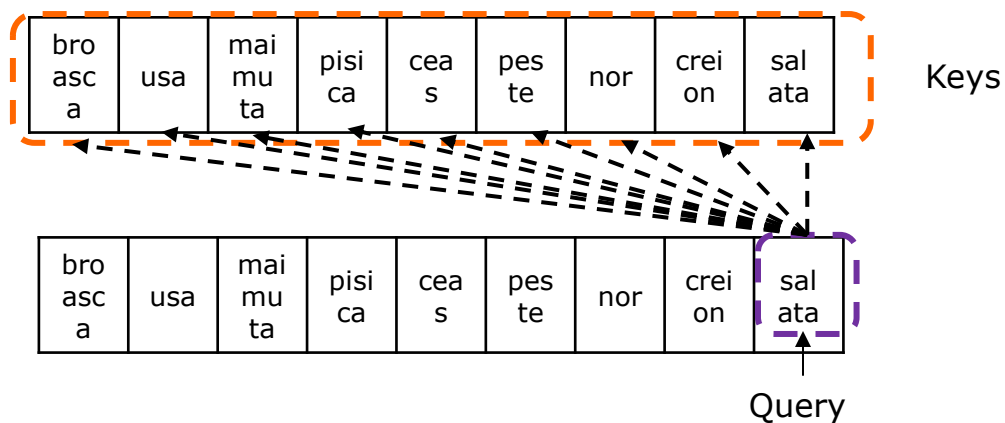
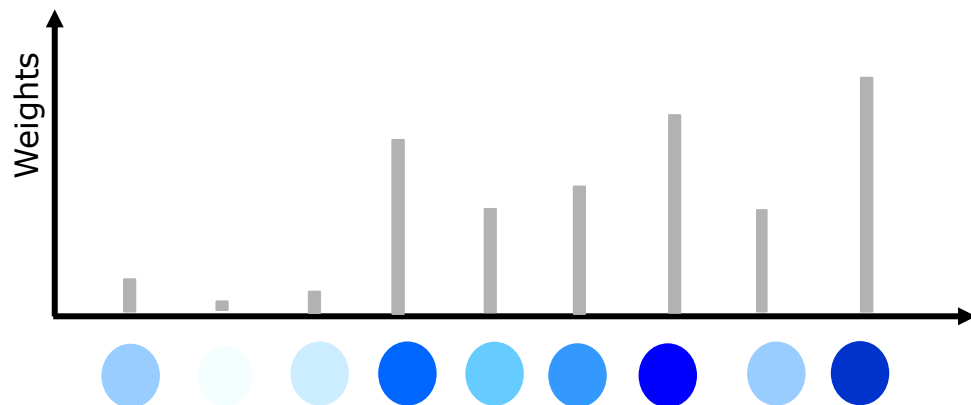
Query



# Mecanismul de “atenție” (attention)



# Mecanismul de “atenție” (attention)

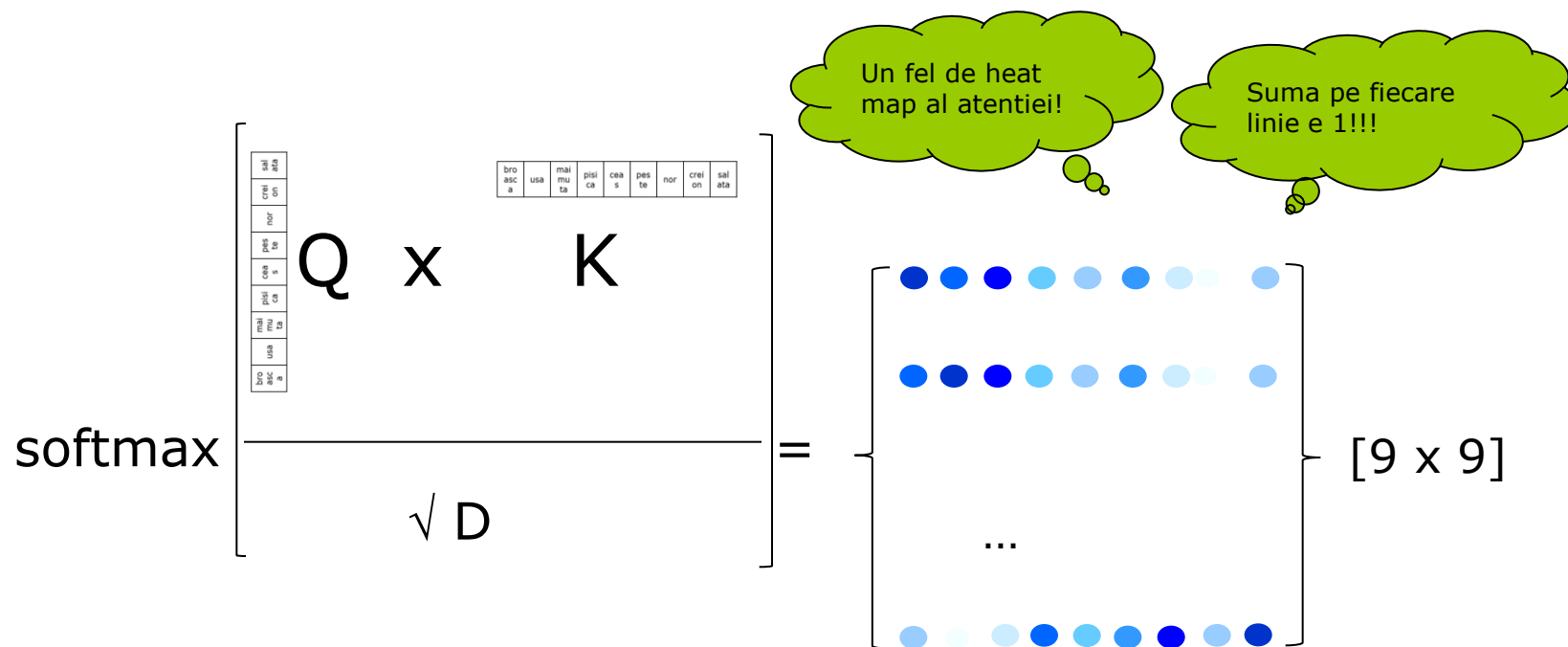


$$\text{Softmax}\left(\frac{\text{All keys} \times \text{1 query}}{1 \times D}\right)$$

Diagram illustrating the attention mechanism components:

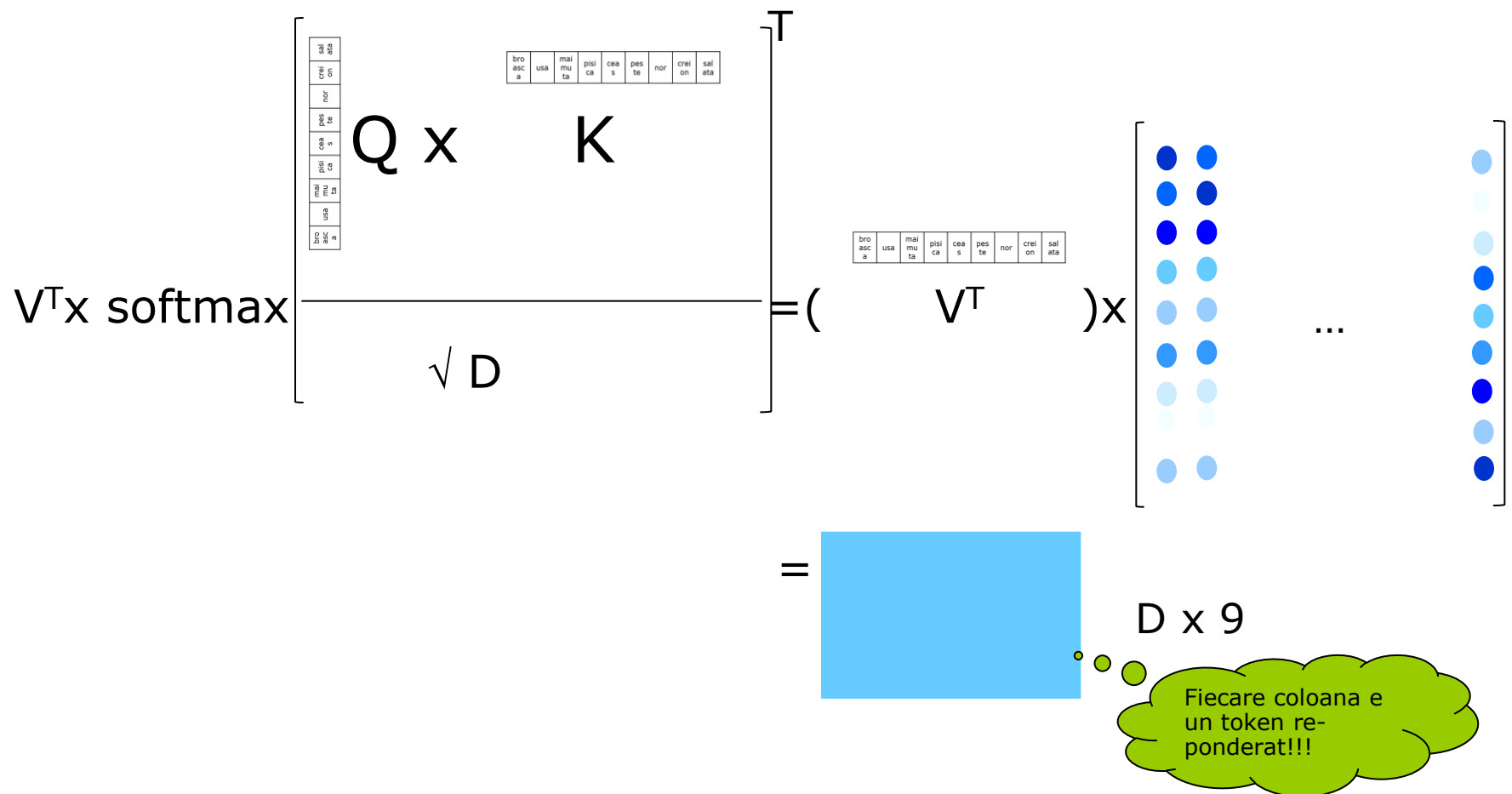
- 1 x 9**: A blue horizontal bar representing the query vector.
- 9 x D**: A green vertical bar representing the keys matrix.
- 1 x D**: A pink horizontal bar representing the query vector.

# Mecanismul de “atenție” (attention)

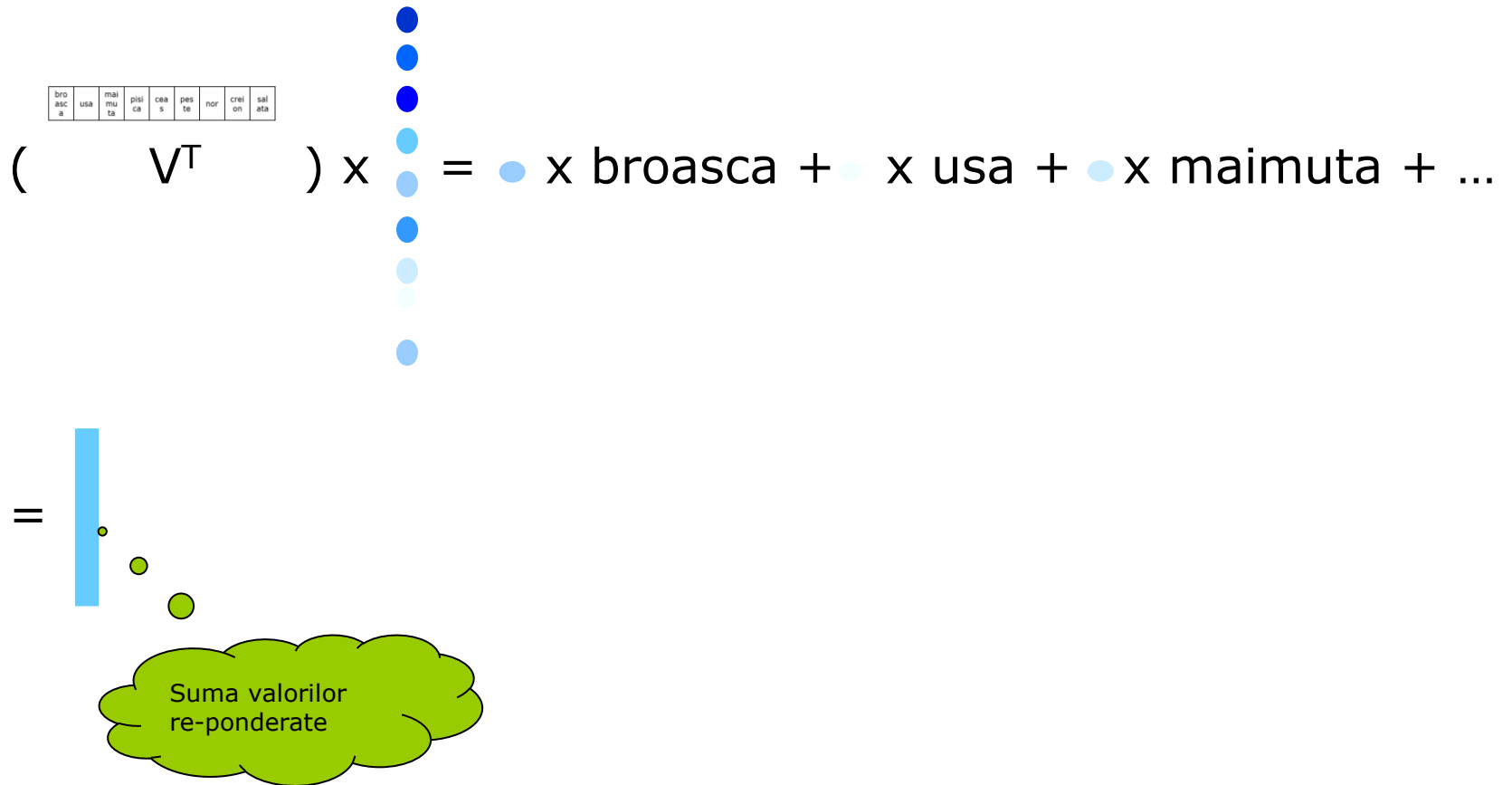




# Mecanismul de “atentie” (attention)



# Mecanismul de “atenție” (attention)



# Mecanismul de “atenție” (attention)

- Input = embedding-uri (de lungime  $n$ ) pt  $v$  cuvinte – matrice  $v \times n$
- $V$  – matrice  $v \times v$                        $v$  – nr de cuvinte
- $K$  – matrice  $n \times d$                        $n$  – nr de features (lungimea unui embedding)
- $Q$  – matrice  $n \times d$                        $d$  – nr de features abstracte

Features

$x_1$	$x_2$	$x_3$	$x_4$
2	0	0	2
0	1	0	0
0	2	1	0
0	0	1	1
2	0	0	0
1	0	1	1

# Mecanismul de “atenție” (attention)

- Input = embedding-uri (de lungime n) pt v cuvinte – matrice  $v \times n$
- V – matrice  $v \times v$                       v – nr de cuvinte
- K – matrice  $n \times d$                       n – lungimea reprezentarii dense (embedding)
- Q – matrice  $n \times d$                       d – nr de features abstracte

Features	$x_1$	$x_2$	$x_3$	$x_4$
	2	0	0	2
	0	1	0	0
	0	2	1	0
	0	0	1	1
	2	0	0	0
	1	0	1	1

## Transformare

input  $\rightarrow$  (input  $\times$  K)  $\times$  ( $Q^T \times$  input $^T$ )  $\times$  (V  $\times$  input)  $\rightarrow$  features

(v,n)  $\rightarrow$  ((v,n)  $\times$  (n,d))  $\times$  ((d,n)  $\times$  (n,v))  $\times$  ((v,v)  $\times$  (v,n))

(v,n)  $\rightarrow$             (v,d)             $\times$             (d,v)             $\times$             (v,n)

Diagram illustrating the relationship between Features,  $W_k$ , and  $k_j$ .

**Features**

$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$
2	0	0	0	2	0
0	1	0	0	0	0
0	2	1	0	0	0
0	0	1	1	0	0
2	0	0	0	0	0
1	0	1	1	1	0

**$W_k$**

0	0	1	0	0	0
0	1	0	0	0	0
1	0	0	0	0	-1

**$k_j$**

$k_1$	$k_2$	$k_3$	$k_4$

Features						$x_1$	$x_2$	$x_3$	$x_4$
						2	0	0	2
						0	1	0	0
						0	2	1	0
						0	0	1	1
						2	0	0	0
						1	0	1	1

$W_Q$

1	1	0	0	0	0
0	1	0	1	0	0
0	0	1	0	1	1

$q_1$	$q_2$	$q_3$	$q_4$

# Mecanismul de “atenție” (attention)

- Input = embedding-uri (de lungime  $n$ ) pt  $v$  cuvinte – matrice  $v \times n$
- $V$  – matrice  $v \times v$   $v$  – nr de cuvinte
- $K$  – matrice  $n \times d$   $n$  – nr de features (lungimea unui embedding)
- $Q$  – matrice  $n \times d$   $d$  – nr de features abstracte

Features	$x_1$	$x_2$	$x_3$	$x_4$
	2	0	0	2
	0	1	0	0
	0	2	1	0
	0	0	1	1
	2	0	0	0
	1	0	1	1

## Transformare

input  $\rightarrow$  input  $\times K \times Q^T \times \text{input}^T \times V \times \text{input} \rightarrow$  features

$(v,n) \rightarrow ((v,n) \times (n,d)) \times ((d,n) \times (n,v)) \times ((v,v) \times (v,n))$   
 $(v,n) \rightarrow (v,d) \times (d,v) \times (v,n)$   
 $(v,n) \rightarrow (v,v) \times (v,n)$

	$q_1$	$q_2$	$q_3$	$q_4$
	2	1	0	2
	0	1	1	1
	3	2	2	1

$k_1^T$	0	0	1	
$k_2^T$	2	1	0	
$k_3^T$	1	0	-1	
$k_4^T$	0	0	1	

MatMul  
( $K^T Q$ )

# Mecanismul de “atenție” (attention)

- Input = embedding-uri (de lungime n) pt v cuvinte – matrice  $v \times n$
- V – matrice  $v \times v$                        $v$  – nr de cuvinte
- K – matrice  $n \times d$                        $n$  – nr de features (lungimea unui embedding)
- Q – matrice  $n \times d$                        $d$  – nr de features abstracte

Features	$x_1$	$x_2$	$x_3$	$x_4$
	2	0	0	2
	0	1	0	0
	0	2	1	0
	0	0	1	1
	2	0	0	0
	1	0	1	1

## Transformare

input -> input  $\times$  K  $\times$   $Q^T$   $\times$  input $^T$   $\times$  V  $\times$  input -> features  
 $(v,n) \rightarrow ((v,n) \times (n,d)) \times ((d,n) \times (n,v)) \times ((v,v) \times (v,n))$   
 $(v,n) \rightarrow (v,d) \times (d,v) \times (v,n)$   
 $(v,n) \rightarrow (v,v) \times (v,n)$

Scale	1	1	1	0
$\frac{\square}{\sqrt{dk}}$	2	1	0	2
$\approx \begin{bmatrix} 0 \\ 2 \end{bmatrix}$	0	0	-1	0
	1	1	1	0

## Scalare

input -> input  $\times$  K  $\times$   $Q^T$   $\times$  input $^T$   $\times$  V  $\times$  input -> features  
 $(v,n) \rightarrow ((v,n) \times (n,d)) \times ((d,n) \times (n,v)) \times ((v,v) \times (v,n))$   
 $(v,n) \rightarrow (v,d) \times (d,v) \times (v,n)$   
 $(v,n) \rightarrow (v,v) / \sqrt{d} \times (v,n)$

## Normalizare

input -> input  $\times$  K  $\times$   $Q^T$   $\times$  input $^T$   $\times$  V  $\times$  input -> features  
 $(v,n) \rightarrow ((v,n) \times (n,d)) \times ((d,n) \times (n,v)) \times ((v,v) \times (v,n))$   
 $(v,n) \rightarrow (v,d) \times (d,v) \times (v,n)$   
 $(v,n) \rightarrow \text{softmax}((v,v) / \sqrt{d}) \times (v,n)$

# Mecanismul de “atenție” (attention)

- Input = embedding-uri (de lungime n) pt v cuvinte – matrice  $v \times n$
- V – matrice  $v \times v$                        $v$  – nr de cuvinte
- K – matrice  $n \times d$                        $n$  – nr de features (lungimea unui embedding)
- Q – matrice  $n \times d$                        $d$  – nr de features abstracte

Features	$x_1$	$x_2$	$x_3$	$x_4$
	2	0	0	2
	0	1	0	0
	0	2	1	0
	0	0	1	1
	2	0	0	0
	1	0	1	1

## Transformare

input  $\rightarrow$  input  $\times K \times Q^T \times$  input $^T \times V \times$  input  $\rightarrow$  features

$(v,n) \rightarrow ((v,n) \times (n,d)) \times ((d,n) \times (n,v)) \times ((v,v)$   
 $(v,n) \rightarrow (v,d) \times (d,v) \times$   
 $(v,n) \rightarrow (v,v) \times$   
 $(v,n) \rightarrow (v,n)'$

MatMul

.3	.3	.4	.1
.9	.3	.2	.7
.1	.1	0	.1
.3	.3	.4	.1

Attention Weight Matrix (A)

## Scalare

input  $\rightarrow$  input  $\times K \times Q^T \times$  input $^T \times V \times$  input  $\rightarrow$  fe  
 $(v,n) \rightarrow ((v,n) \times (n,d)) \times ((d,n) \times (n,v)) \times ((v,v)$   
 $(v,n) \rightarrow (v,d) \times (d,v) \times$   
 $(v,n) \rightarrow (v,v) / \sqrt{v} \times$   
 $(v,n) \rightarrow (v,n)'$

$W_V$					
10	0	0	0	0	0
0	0	0	10	0	0
0	0	0	10	0	0
0	10	0	0	0	0

$v_1$	$v_2$	$v_3$	$v_4$
20	0	0	20
0	0	10	10
0	10	0	0

$z_1$	$z_2$	$z_3$	$z_4$

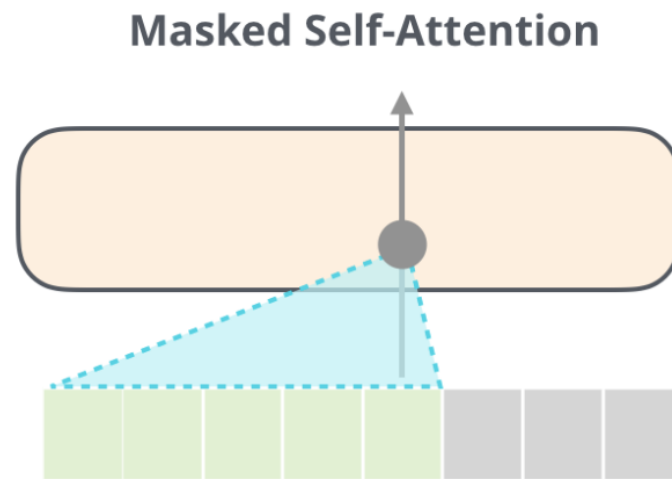
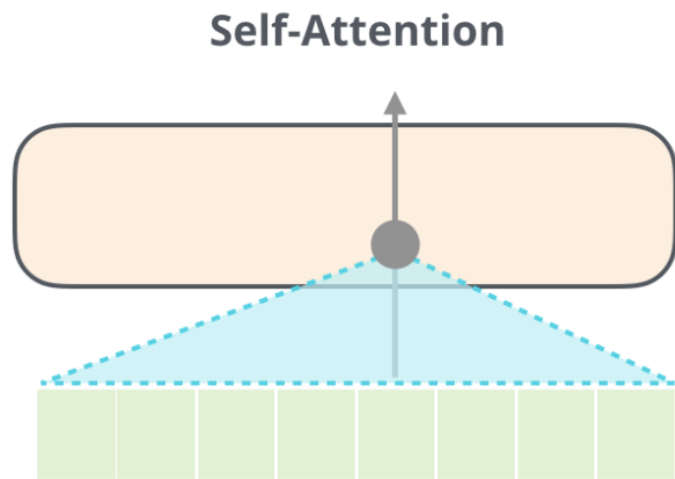
Attention Weighted Features

## Normalizare

input  $\rightarrow$  input  $\times K \times Q^T \times$  input $^T \times V \times$  input  $\rightarrow$  features  
 $(v,n) \rightarrow ((v,n) \times (n,d)) \times ((d,n) \times (n,v)) \times ((v,v) \times (v,n))$   
 $(v,n) \rightarrow (v,d) \times (d,v) \times (v,n)$   
 $(v,n) \rightarrow \text{softmax}((v,v) / \sqrt{v}) \times (v,n)$   
 $(v,n) \rightarrow (v,n)$

# Mecanismul de “atenție” (attention)

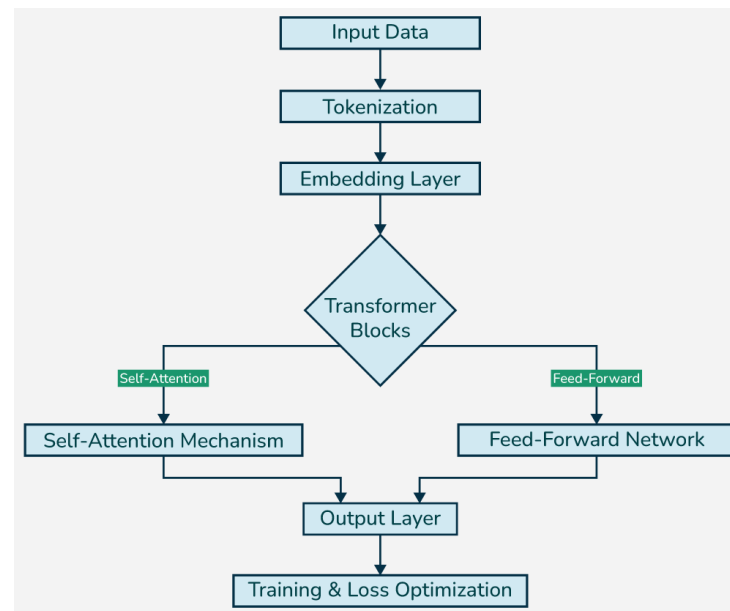
---





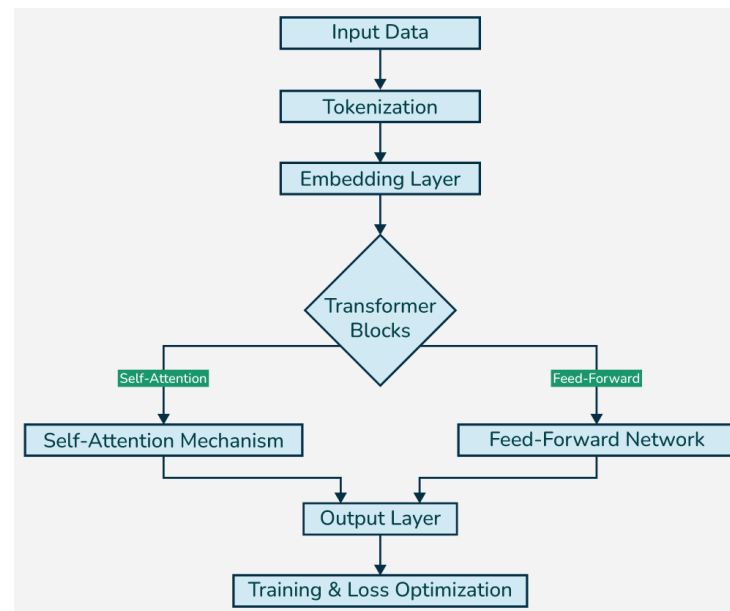
# Procesarea textelor/limbajelor

- ❑ Large language Models (LLMs)
  - Tokenisation
  - Embeddings
  - Transformer & Self-attention mechanism
  - **Feed-forward network**



# Procesarea textelor/limbajelor

- ❑ Large language Models (LLMs)
  - Tokenisation
  - Embeddings
  - Transformer & Self-attention mechanism
  - **Feed-forward network**
    - ❑ Normalisation layers
    - ❑ Softmax layers
    - ❑ Loss functions



# Procesarea textelor/limbajelor

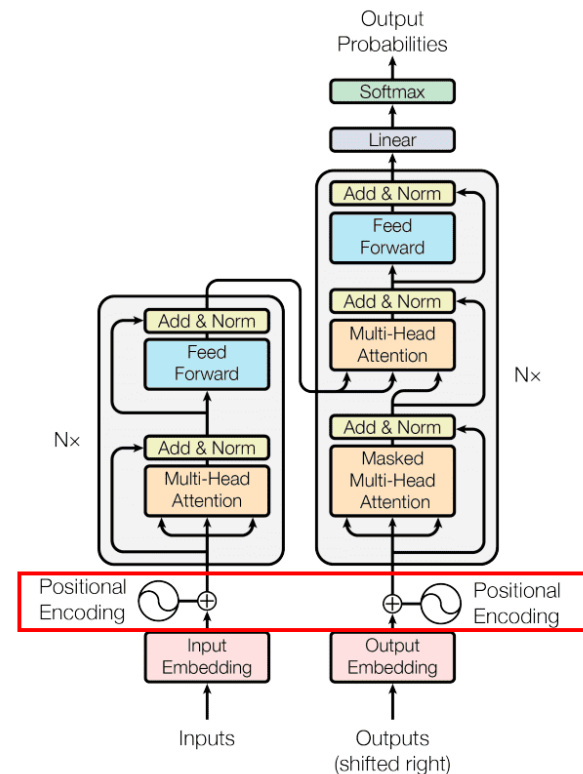
## Large language Models (LLMs)

### How large an LLM is?

- |Vocabulary| = V
  - Words/tokens
- |Embedding| = E (model dimension)
- |input sequence| = L (context window)
- No of blocks NB
- Size of MLP's hidden layers H

$$y = \frac{x - E[x]}{\sqrt{\text{Var}[x] + \epsilon}} * \gamma + \beta$$

GPT2Model(  
 TokenEmbedding,  $\rightarrow V \times E$  param  
 PositionalEmbedding,  $\rightarrow L \times E$  param  
 Dropout,  $\rightarrow 0$   
 NB x TransformerBlock(  
     Normalisation,  $\rightarrow 2 \times E$   
     MSA,  $\rightarrow 3 \times (E \times E + E) + E \times E + E$   
     Normalisation,  $\rightarrow 2E$   
     MLP(  
         Linear,  $\rightarrow E \times H + H$   
         Linear,  $\rightarrow H \times E + E$   
     )  
     Normalisation,  $\rightarrow 2E$   
     Linear  $\rightarrow E \times V$   
 )  
 )



$$\begin{aligned}
 \#param &= param(embeddings) + param(transformer) + param(others) \\
 &= (W_{TextEmb} + W_{PosEmb}) + NB * (Norm + Attn + Norm + MLP) + Norm + Linear \\
 &= (V \times E + L \times E) + NB * (2E + (3 * (E \times E + E) + (E \times E + E)) + 2E + (E \times H + H + H \times E + E)) + 2E + E \times V \\
 &= E \times (V + L) + NB \times (4E^2 + 9E + 2EH + H) + 2E + E \times V
 \end{aligned}$$

# Procesarea textelor/limbajelor

## □ Large language Models (LLMs)

### ■ How large an LLM is?

- |Vocabulary| = V
  - Words/tokens
- |Embedding| = E (model dimension)
- |input sequence| = L (context window)
  
- No of blocks NB
- Size of MLP's hidden layers H

$$\text{\#param} = E \times (V + L) + NB \times (4E^2 + 9E + 2EH + H) + 2E + E \times V$$

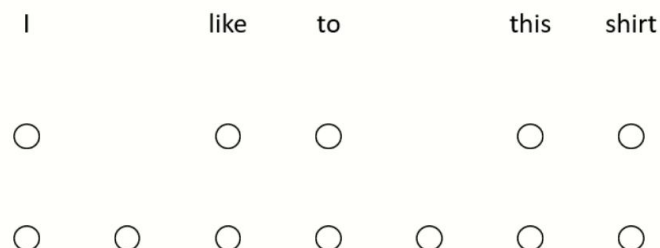
	V	E	L	NB	H	#param
GPT2	50 257	768	1 024	12	3 072	124 439 808
GPT3	50 257	12 288	2 048	96	12 288	87 627 632 640

# Tranformers

## □ Architecture

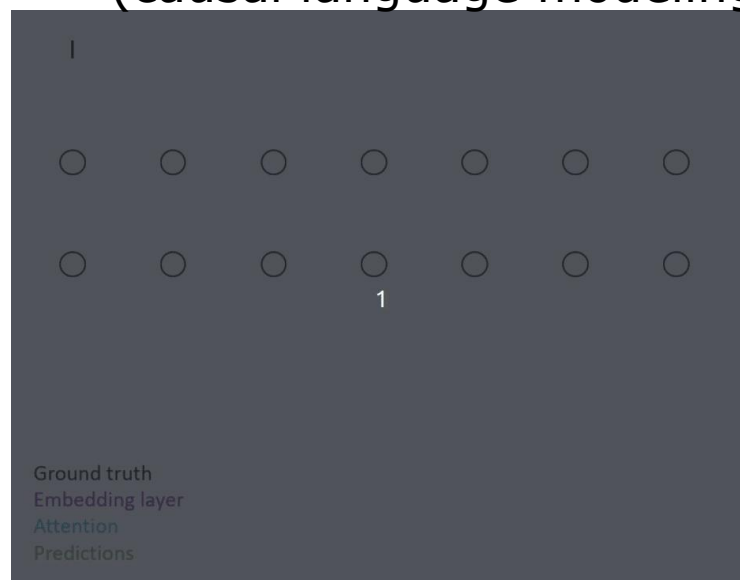
## □ Pre-training – unsupervised

### ■ Masked language modelling



Ground truth  
Embedding layer  
Attention  
Predictions

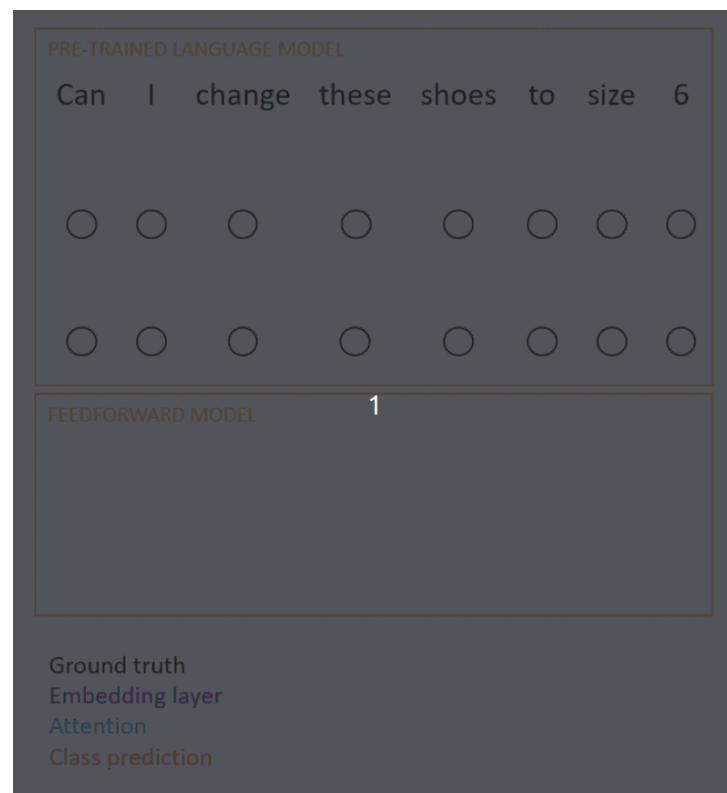
### ■ Next word prediction (causal language modeling)



Ground truth  
Embedding layer  
Attention  
Predictions

# Tranformers

- ❑ Architecture
- ❑ Pre-training – unsupervised
- ❑ Fine-tuning
  - Specific task
    - ❑ Another model -> Additional layers (params)



# Transformers

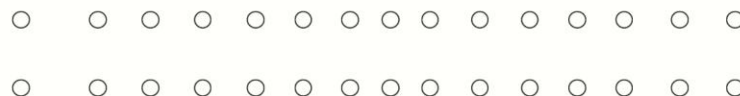
- Architecture
- Pre-training – unsupervised
- Fine-tuning
- Prompt-based learning
  - Simple
  - With examples (k-shot learning)

Identify the intent : Can I book a flight to London =>



Ground truth  
Embedding layer  
Attention  
Predictions

Translate French to English : chien => dog , chaise => chair , pomme =>



Ground truth  
Embedding layer  
Attention  
Predictions

# Transformers - roadmap

---

## ■ Resources

- <https://jalammar.github.io/illustrated-transformer/>
- Transformers (how LLMs work) explained visually | DL5 – YouTube <https://www.youtube.com/watch?v=wjZofJX0v4M>
- LLM Visualization <https://bbycroft.net/llm>
- nanoGPT <https://www.youtube.com/watch?v=kCc8FmEb1nY>
- Attention <https://lilianweng.github.io/posts/2018-06-24-attention/>
- <https://arxiv.org/pdf/1706.03762>
- <https://github.com/karpathy/nanoGPT>
- <https://github.com/karpathy/ng-video-lecture>
- [https://colab.research.google.com/drive/1JMLa53HDuA-i7ZBmqV7ZnA3c\\_fvtXnx-?usp=sharing](https://colab.research.google.com/drive/1JMLa53HDuA-i7ZBmqV7ZnA3c_fvtXnx-?usp=sharing)
- <https://paperswithcode.com/paper/attention-is-all-you-need>
- <https://www.youtube.com/watch?v=kCc8FmEb1nY>



# Cursul următor

---

## A. Scurtă introducere în Inteligența Artificială (IA)

## B. Sisteme inteligente

### ■ **Sisteme care învață singure**

- Arbori de decizie
- Rețele neuronale artificiale - transformers
- Mașini cu suport vectorial
- Algoritmi evolutivi

### ■ Sisteme bazate pe reguli

### ■ Sisteme hibride

## C. Rezolvarea problemelor prin căutare

### ■ Definirea problemelor de căutare

### ■ Strategii de căutare

- Strategii de căutare neinformate
- Strategii de căutare informate
- Strategii de căutare locale (Hill Climbing, Simulated Annealing, Tabu Search, Algoritmi evolutivi, PSO, ACO)
- Strategii de căutare adversială

---

❑ Informațiile prezentate au fost colectate din diferite surse de pe internet, precum și din cursurile de inteligență artificială ținute în anii anteriori de către:

- Conf. Dr. Mihai Oltean – [www.cs.ubbcluj.ro/~moltean](http://www.cs.ubbcluj.ro/~moltean)
- Lect. Dr. Crina Groșan – [www.cs.ubbcluj.ro/~cgrosan](http://www.cs.ubbcluj.ro/~cgrosan)
- Prof. Dr. Horia F. Pop – [www.cs.ubbcluj.ro/~hfpop](http://www.cs.ubbcluj.ro/~hfpop)
- Prof. Dr. Radu Ionescu – <https://raduionescu.herokuapp.com/>

# Transformers

---

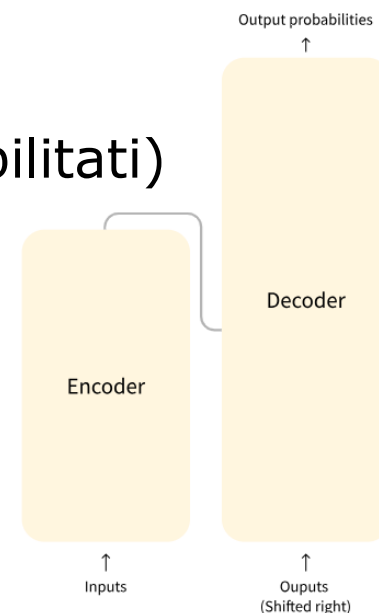
## □ Architecture

### ■ Encoder

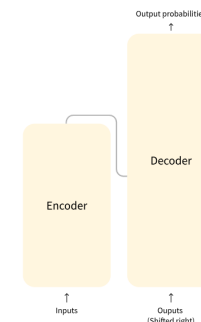
- Primește cuvinte
- Construiește reprezentari (features) ale cuvintelor

### ■ Decoder

- Primește
  - reprezentari (features) ale cuvintelor
  - Alte inputuri
- Generează secvențe de cuvinte (ca probabilitati)



# Transformers



## □ Arhitectura

### ■ Modele bazate doar pe encoder

- Proiectate pentru a învăța să prezică output-uri specifice unui task
  - In: texte
  - Out: labels sau tokens
- Task-uri pe care le pot rezolva
  - Clasificare de propoziții
  - Identificarea în texte a unor sentimente/emotii
- Ex. BERT, RoBERTa

### ■ Modele bazate doar pe decoder

- Proiectate pentru a genera noi texte (decoding realizat în stil auto-regresiv = generare de secvențe, *token by token*, fiecare token fiind condiționat de token-urile anterior generate)
  - In: embeddings
  - Out: texte
- Task-uri pe care le pot rezolva
  - Raspuns la diferite queries
- Ex. GPT

### ■ Modele bazate pe encoder-decoder

- Proiectate pentru a genera noi texte
  - In: texte
  - Out: texte
- Task-uri pe care le pot rezolva
  - rezumat de texte
  - Traducere de texte
- Ex. BART, T5

---

## ❑ Encoder-only

- Proiectata sa inteleaga textul de intrare si sa genereze reprezentari fixe ale textului dat ca input
- Mecanism de atentie bidirectional (un token e "atent" la toti ceilalti tokeni sin textul de intrare)

## ❑ Decoder-only (causal decoder architecture)

- Proiectata sa genereze texte (predictia urmatorului token intr-o anumita secventa)
- Mecanism de atentie unidirectional (un token e "atent" doar la tokenii care il preced si la el insusi)

## ❑ Encoder-decoder

- Proiectata sa inteleaga textul de intrare si sa genereze text de output
- Mecanism de atentie
  - ❑ Bidirectional in encoder
  - ❑ Unidirectional in decoder

---

## □ Encoder-decoder

### ■ Encoder

- Map input sequence into latent representation by some multi-head self-attention layers

### ■ Decoder

- Performs cross-attention on latent representations to generate target sequence

- 
- ❑ GitHub - mlabonne/llm-course: Course to get into Large Language Models (LLMs) with roadmaps and Colab notebooks.
  - ❑ Attention? Attention! | Lil'Log