

# Üzleti Intelligencia

## 11. Előadás: Transzformáló architektúrák

Kuknyó Dániel  
Budapesti Gazdasági Egyetem

2023/24  
1.félév

- 1 Bevezetés
- 2 Transzformáló architektúrák
- 3 Figyelmi mechanizmus

- 1 Bevezetés
- 2 Transzformáló architektúrák
- 3 Figyelmi mechanizmus

# Visszacsatolósos neurális hálózatok alapjai

## Alkalmazás

## Input

## Output

Beszéd felismerés



"Milyen szép időnk van ma!"

Szemantikai értelmezés

"Ez egy rossz film volt."



DNS szekvencia elemzés

AGCCCTGTACTAG

AGCCCTGTACTAG

Gépi fordítás

"Willst du mit mir tanzen?"

"Szeretnél velem táncolni?"

Videók elemzése



Futás

Nevek felismerése

Tegnap Józsi letörölte a  
termelési adatbázist.

Tegnap **Józsi** letörölte a  
termelési adatbázist.

# Szavak reprezentálása 1-hot vektorokkal

**Input:** A kedvenc sportom a foci.

**Reprezentáció:**  $X = [x_1, x_2, x_3, x_4, x_5,]$

**Szókincs:**  $\begin{bmatrix} a, & foci, & kedvenc, & sportom \\ 1 & 2 & 3 & 4 \end{bmatrix}$

**Reprezentáció:**  
 $foci = [0, 1, 0, 0]$   
 $sportom = [0, 0, 0, 1]$

## Problémák:

- Ha van egy 10.000 szóból álló szövegtörzs, minden szava egy 10.000 elemű vektorként lesz reprezentálva, aminek csak egyetlen eleme 1, a többi 0. **Ez nem egy skálázható megoldás.**
- **Nincs kapcsolat a szavak között.** A szavak külön-külön vannak kezelve, hasonló jelentésű szavak reprezentációja nagyban eltérhet.

# Szavak reprezentálása beágyazóvektorokkal

## Beágyazás

Egy szó beágyazása **egy magas dimenziójú vektortérben való numerikus reprezentáció**. Ezek a vektorok tartalmazzák a szavak **struktúráját, szemantikáját, és szintaktikai szerkezetét**.

Ezáltal képesek a mélytanuló modellek elsajátítani a szavak közötti hasonlóságokat és az egyes szavak jelentését.

	Férfi	Nő	Király	Királynő	Alma
Nem	−1	1	−0.95	0.97	0.0
Előkelő	0.01	0.02	0.93	0.95	−0.01
Kor	0.03	0.02	0.7	0.68	0.03
Étel	0.04	0.01	0.02	0.01	0.96

Tehát ebben az esetben például a férfi szó beágyazóvektora:

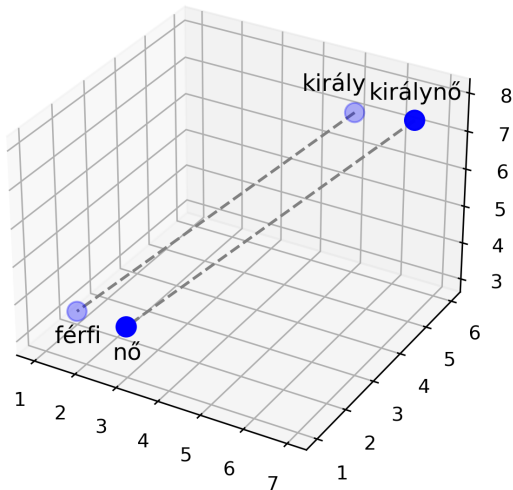
$$e_{\text{férfi}} = [-1, 0.01, 0.03, 0.04]$$

# Beágyazóvektorok reprezentálása

A beágyazóvektorok használatával lehetőség nyílik a szavak hasonlóságának kiszámítására.

Az egymáshoz jelentés tartalmilag **közelebb álló szavak beágyazóvektorainak matematikai távolsága alacsonyabb lesz, mint az egymástól távolabb eső szavaké.**

Ezáltal továbbá lehetséges analógiák kiszámítása is. *A férfi és a király olyanok egymásnak, mint a nő és a királynő.*

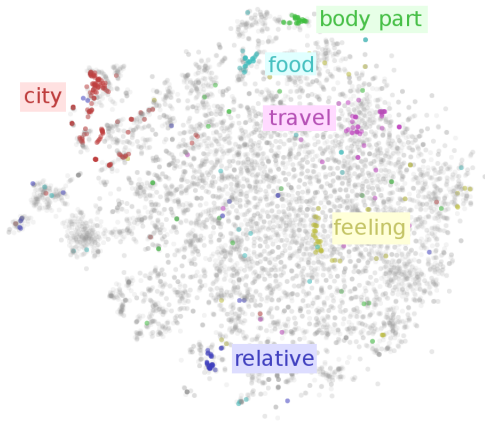


# Beágyazások vizualizálása

Dimenziócsökkentő algoritmusok segítségével lehetőség nyílik a **magasabb dimenziós vektorok alacsonyabb térben való reprezentációjára**. Az egyik ilyen algoritmus a T-SNE, ami jól használható komplex input adatok esetén.

Ez hasznos a következő problémák esetén:

- Vizualizáció
- Klaszterezés
- Adatminőség mérése
- Szemantikai kapcsolatok elemzése
- Hiperparaméter hangolás



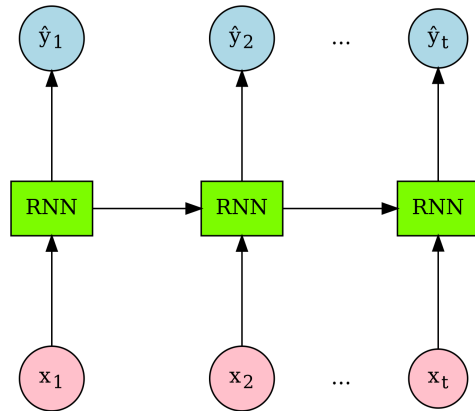


- 1 Bevezetés
- 2 Transzformáló architektúrák
- 3 Figyelmi mechanizmus

# Hagyományos visszacsatolós architektúrák

A visszacsatolós neurális hálózatok (RNN) olyan mesterséges neurális hálózatok, amelyek képesek kezelni **időbeli szekvenciákat és más időfüggő adatokat**.

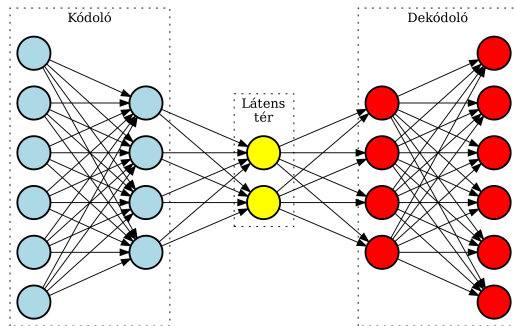
Ezek a hálózatok olyan struktúrával rendelkeznek, amely lehetővé teszi a **korábbi lépések eredményeinek visszacsatolását az aktuális lépésbe**. Ennek eredményeként képesek tartani az emlékezetüket korábbi állapotokról, és ezáltal kezelni a szekvenciális adatokat



# Önkódoló architektúrák

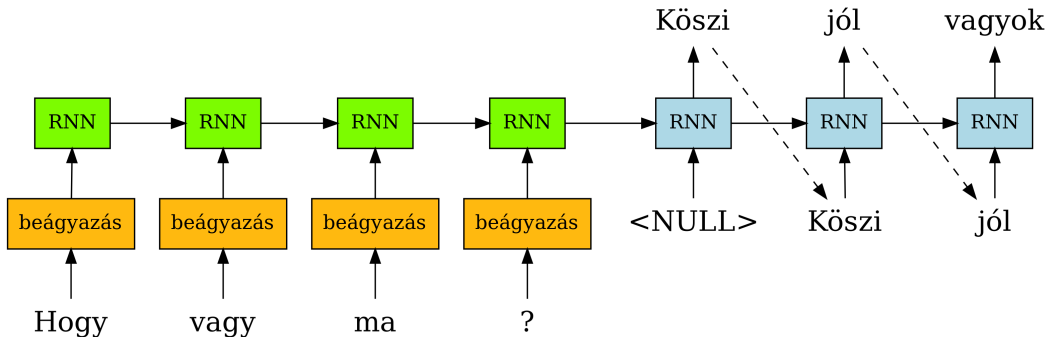
Az önkódoló neurális hálózatok feladata az inputot átmásolni az outputba úgy, hogy közben megismeri az adatok alacsony szintű struktúráját:

- **Kódoló:** A bemeneti adatokat tömöríti egy rövidebb, alacsony dimenziójú reprezentációba.
- **Látens tér:** Az az alacsony dimenziójú tér, amelyben a kódoló reprezentálja a bemeneti adatokat. Ez a tér tartalmazza az információkat a bemenetről kompakt formában.
- **Dekódoló:** Feladata a látens térben lévő reprezentációt visszaalakítani eredeti vagy közelítőleges formájára.



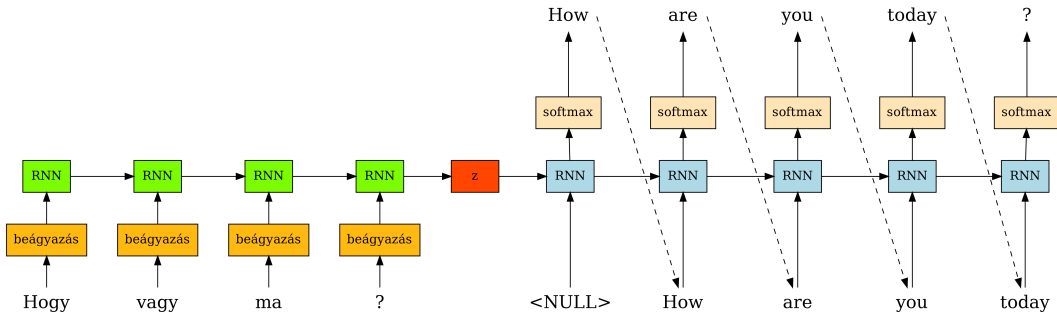
# Transzformáló architektúrák

A transzformáló architektúrák rendkívül sokoldalúak és hatékonyak a mesterséges mélytanulásban. A transzformálók feladata **két szekvencia közötti leképezés megtanulása**. Rendkívül jól teljesítenek olyan területeken mint a **természetes nyelvfeldolgozás, képfelismerés, hangfeldolgozás, megerősítéses tanulás**.



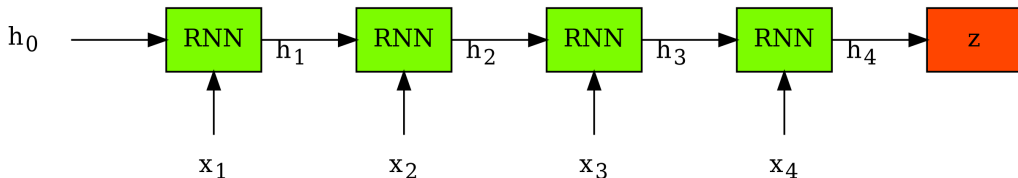
# Gépi fordítás

A transzformáló architektúrák **jól képesek teljesíteni a gépi fordítás területén**. Hasonlóan az önkódoló architektúrákhoz a fő részei a **kódoló** az input feldolgozására, a **látens** tér az input reprezentálására és a **dekódoló** az output előállítására.



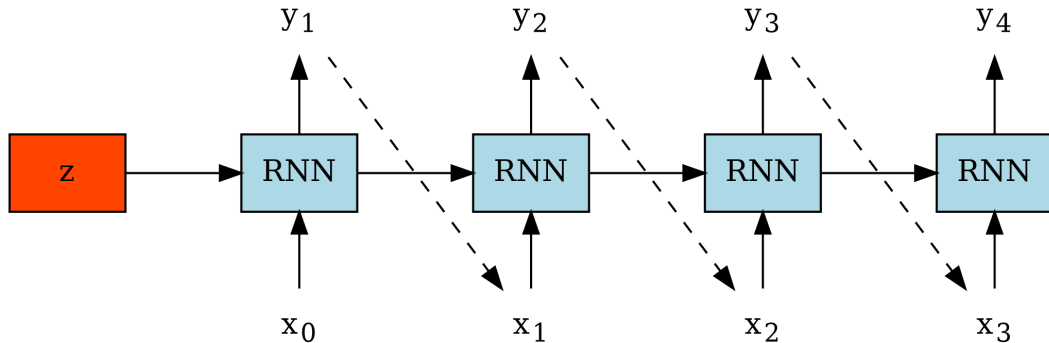
# Kódoló

A transzformáló architektúrákban a kódoló feladata az input adatok feldolgozása és egy **értelmes, kontextusban gazdag reprezentáció létrehozása**. Az kódolónak alapvető szerepe van az input sorozat megértésében és az alatta rejlő információk megragadásában. Az általa feldolgozott információ a  $z$  kontextus vektorban kerül átadásra a dekódolónak, ami megfelel az utolsó cella rejtett állapotának.



# Dekódoló

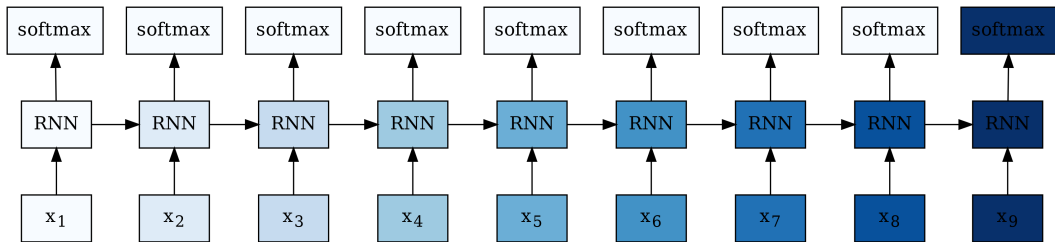
A dekódoló feladata a transzformáló architektúrákban az **output sorozat létrehozása az input sorozat kontextualizált reprezentációjának felhasználásával**. Dekódolás a legelső cella null inputot kap, és utána minden cella az előző cella outputját kapja meg inputként:  $x_0 = null$ ,  $x_i = y_i$ ,  $i > 0$ .



# A transzformálók problémája

Ha hosszú szekvenciákat kell generálniuk, a transzformálók gradiensei nagyon alacsonyak lesznek hiba visszaáramoltatás közben. Ez az **eltűnő gradiensek problémája**, és ahhoz vezet, hogy a hálózat elfelejti a korábbi információkat.

Továbbá a modellnek **minden fontos információt egyetlen kontextus vektorba kell besűrítenie**. Ezzel a  $z$  vektor lesz a tanulás szűk keresztmetszete.



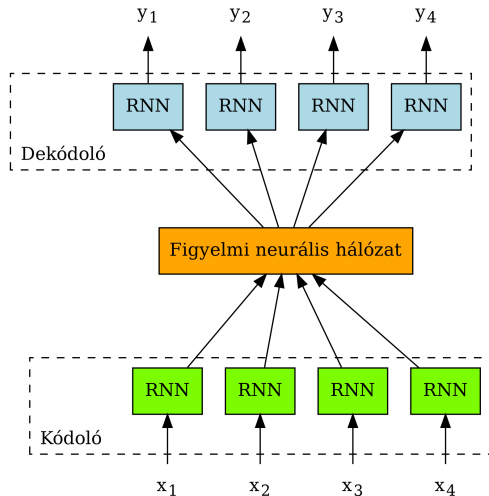


- 1 Bevezetés
- 2 Transzformáló architektúrák
- 3 Figyelmi mechanizmus

# Figyelem alapjai

Az alapvető megfontolás, hogy a  $z$  kontextusvektornak legyen **közvetlen kapcsolata nem csak az input szekvencia utolsó eleméhez, hanem mindegyikhez.**

Ezáltal a modellnek minden időlépésben minden vektor elemhez lesz hozzáférése, így lehetséges lesz megtanítani arra, **melyik állapot esetén melyik input elemre kell figyelmet fordítania.**



# Figyelem alapjai

A dekódoló hálózat figyelem értéke  $i$  időlépésben:

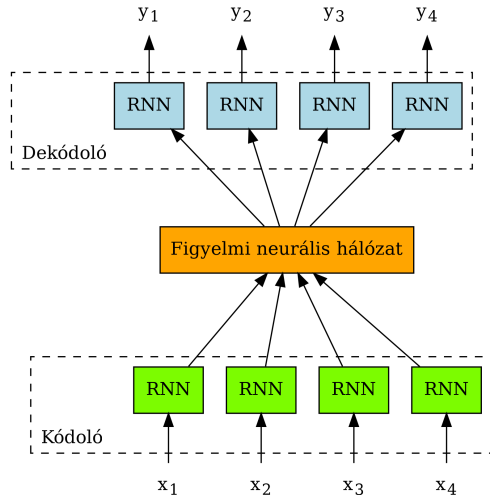
$$e_i = f(y_{i-1}, h_i) \in \mathbb{R}^n$$

A figyelmi értékek valószínűséggé alakítva és normalizálva:

$$\alpha_{ij} = \frac{e_{ij}}{\sum_{k=1}^{T_x} e_{ik}}$$

Ezáltal kontextus vektor  $i$  időlépésben:

$$z_i = \sum_{j=1}^T \alpha_{ij} h_j$$

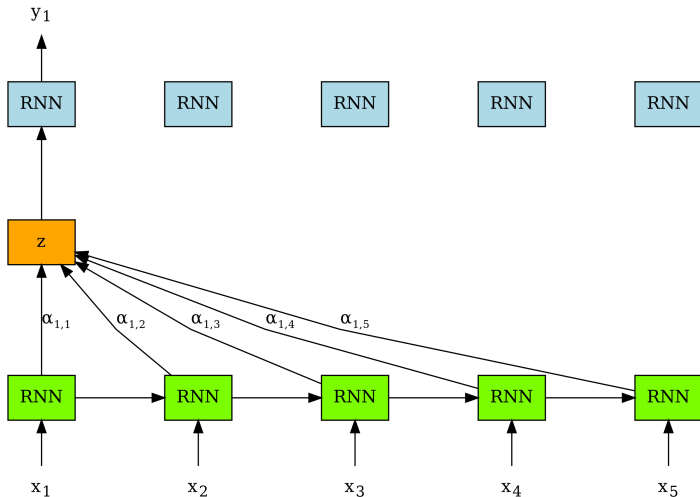


# Figyelmi modell

$$e_i = f(y_{i-1}, h_i) \in \mathbb{R}^n$$

$$\alpha_{ij} = \frac{e_{ij}}{\sum_{k=1}^{T_x} e_{ik}}$$

$$z_i = \sum_{j=1}^T \alpha_{ij} h_j$$

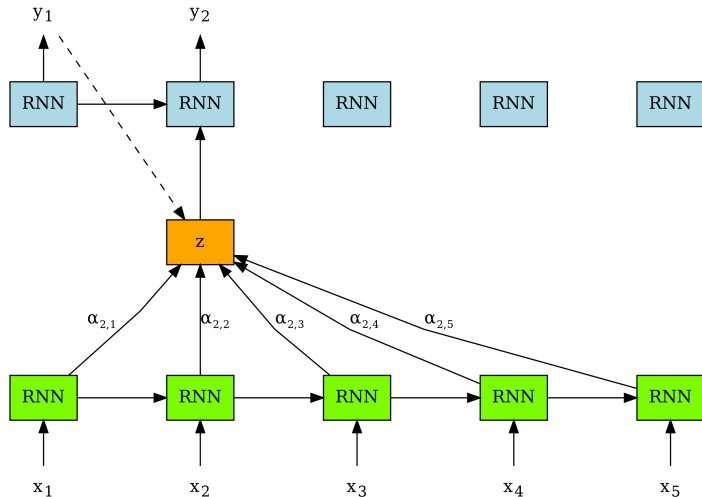


# Figyelmi modell

$$e_i = f(y_{i-1}, h_i) \in \mathbb{R}^n$$

$$\alpha_{ij} = \frac{e_{ij}}{\sum_{k=1}^{T_x} e_{ik}}$$

$$z_i = \sum_{j=1}^T \alpha_{ij} h_j$$

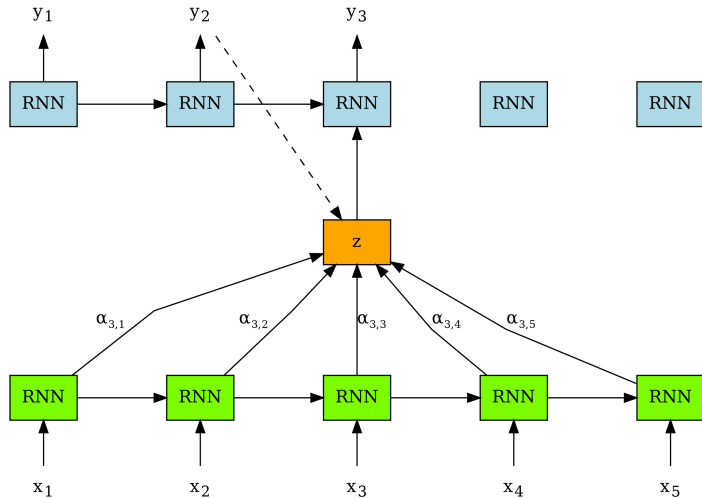


# Figyelmi modell

$$e_i = f(y_{i-1}, h_i) \in \mathbb{R}^n$$

$$\alpha_{ij} = \frac{e_{ij}}{\sum_{k=1}^{T_x} e_{ik}}$$

$$z_i = \sum_{j=1}^T \alpha_{ij} h_j$$

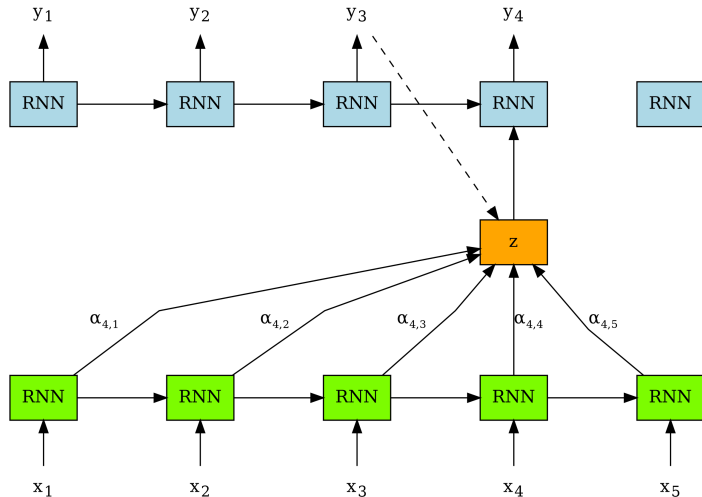


# Figyelmi modell

$$e_i = f(y_{i-1}, h_i) \in \mathbb{R}^n$$

$$\alpha_{ij} = \frac{e_{ij}}{\sum_{k=1}^{T_x} e_{ik}}$$

$$z_i = \sum_{j=1}^T \alpha_{ij} h_j$$

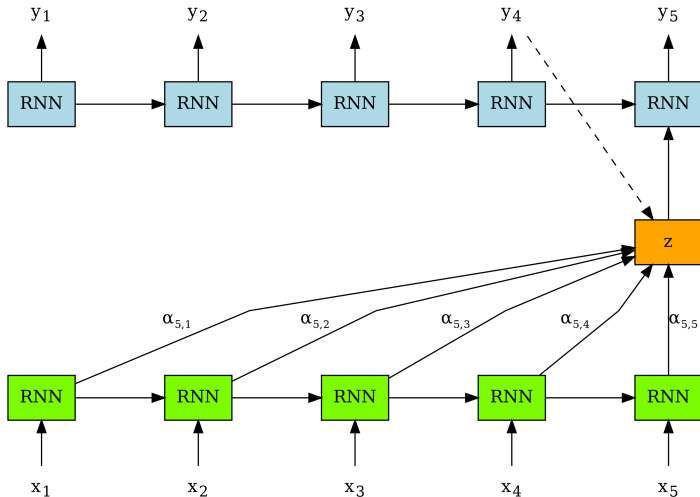


# Figyelmi modell

$$e_i = f(y_{i-1}, h_i) \in \mathbb{R}^n$$

$$\alpha_{ij} = \frac{e_{ij}}{\sum_{k=1}^{T_x} e_{ik}}$$

$$z_i = \sum_{j=1}^T \alpha_{ij} h_j$$





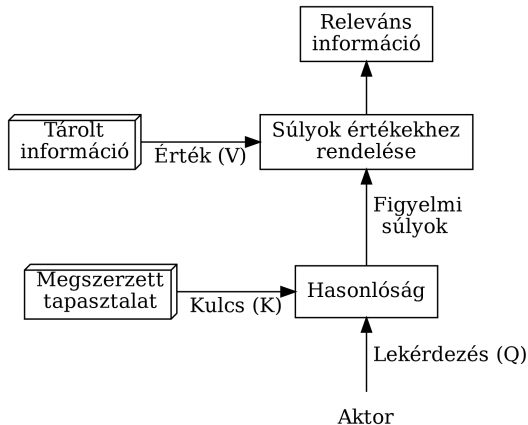


# Jellemzőalapú figyelem

A kulcs-érték-lekérdezés struktúra a transzformáló architektúrák **összetett információlekérdezési rendszerének alapjait képezik.**

Az eljárás mögötti intuíció:

- 1 A modell lekérdezést (**Q**) intéz a tárolóhoz.
- 2 A keresőmotor a lekérdezést kulcsokhoz (**K**) rendeli, amik megfelelően leírják azt.
- 3 Az algoritmus megkeresi a kulcsokra legjobban illeszkedő értékeket (**V**).



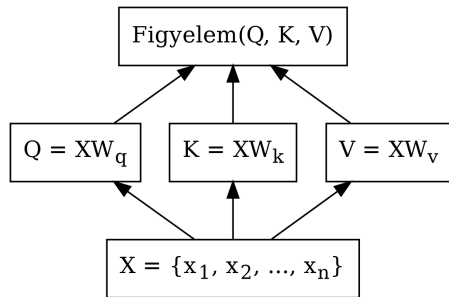
## Kulcs-érték-lekérdezés eljárása

A gyakorlatban a transzformáló 3 különböző reprezentációt használ a lekérdezéseknek, kulcsoknak és értékeknek. A reprezentáció az  $X$  beágyazómátrix és  $W_q, W_k, W_v$  súlymátrixok szorzataként számolódik ki. Az eredő dimenziószám kevesebb lesz az eredetinél.

Ezáltal előáll a transzformáló architektúra **önfigyelem** rétege:

$$\text{Figyelem}(Q, K, V) = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) V$$

Ahol  $d_k$  a kulcsvektor dimenziószáma.



# Önfigyelem

A figyelem nem csak két szekvencia között definiálható, hanem abban az esetben is, **ha az input és output szekvencia megegyezik.**

Ez teszi lehetővé a modellnek, hogy a szekvencián belüli elemekhez **fontossági súlyokat rendeljen.** Így sajátítja el a modell a különböző elemek **struktúráját és a közöttük lévő kapcsolatokat.**

A figyelmi reprezentáció minden szóra kiszámolódik:

$$A(Q, K, V) = A_1, A_2, \dots, A_n$$

Szia,	0.62	0.25	0.12	0
hogy	0.1	0.7	0.2	0
vagy	0	0.3	0.6	0.1
ma?	0	0	0	1
	Szia,	hogy	vagy	ma?