Търсене и извличане на информация. Приложение на дълбоко машинно обучение

Стоян Михов

Лекция 14+15: Генерация на текст с езиков модел. Условен езиков модел. Модел “Последователност към последователност” (Sequence to sequence). Архитектури с “внимание”.

План на лекцията

1. **Формалности за курса (5 мин)**
2. Условни езикови модел (15 мин)
3. Методи за генерация на текст / декодиране (20 мин)
4. Приложения на генерация на текст с езиков модел (5 мин)
5. Архитектура енкодер-декодер за невронен машинен превод (20 мин)
6. Архитектура енкодер-декодер с “внимание” (Attention) (20 мин)
7. Transformer архитектура (30 мин)
8. Оценяване на резултат от машинен превод (10 мин)

Формалности

- Това е последната лекция на курса за този семестър
- Очаквам решенията на домашно задание №3 да бъдат предадени до края на 24.01.2021 г.
- Днес на упражнение ще бъде представено условието за курсовата работа.
- Формалното условие за курсовата работа ще бъде публикувано до края на седмицата.
- След договаряне може да проведем консултация за курсовите работи.
- Лекция 14 се базира на глава 17 от втория учебник.

План на лекцията

1. Формалности за курса (5 мин)
2. **Условни езикови модел (15 мин)**
3. Методи за генерация на текст / декодиране (20 мин)
4. Приложения на генерация на текст с езиков модел (5 мин)
5. Архитектура енкодер-декодер за невронен машинен превод (20 мин)
6. Архитектура енкодер-декодер с “внимание” (Attention) (20 мин)
7. Transformer архитектура (30 мин)
8. Оценяване на резултат от машинен превод (10 мин)

Езиков модел

- Под езиков модел разбираме система, която за всяка начална последователност $w_1 w_2 \ldots w_i$ ни връща вероятностно разпределение за следващия елемент от последователността $\Pr[w \mid w_1 w_2 \ldots w_i]$.

- От Верижното правило следва:
$$\Pr[w_1 w_2 \ldots w_n] = \Pr[w_1] \Pr[w_2 \mid w_1] \Pr[w_3 \mid w_1 w_2] \ldots \Pr[w_n \mid w_1 w_2 \ldots w_{n-1}]$$

- Ако нашата система е в състояние експлицитно да представи $\Pr[w \mid w_1 w_2 \ldots w_i]$, то тя може да се използва за **генерация на текст.**

- По-нататък в лекцията ще разгледаме различни методи за генерация на текст с езиков модел.

Обучение на езиков модел

- За обучението ни е необходим корпус от документи: $\mathbf{C} = \{\mathbf{w}^{(i)} \mid i = 1,2,\ldots,N\}$.

- Обучението извършваме като минимизираме крос-ентропията:

$$H_{\mathbf{C}} = -\frac{1}{\|\mathbf{C}\|}\sum_{k=1}^{N}\sum_{i=1}^{|\mathbf{w}^{(k)}|}\log \Pr[\mathbf{w}_i^{(k)} \mid \mathbf{w}_1^{(k)}, \mathbf{w}_2^{(k)}, \ldots, \mathbf{w}_{i-1}^{(k)}]$$

- За целта използваме спускане по стохастичен градиент.

РНМ за представяне на езиков модел

- При входен текст $w_1 w_2 \ldots w_n$ с произволна дължина n, моделираме вероятностното разпределение за следващата дума като:

- $\Pr[w \mid w_1 w_2 \ldots w_i] = \mathrm{softmax}(U\mathbf{h}_i)_w$, където $\mathbf{h}_j = g(\mathbf{h}_{j-1}, w_j)$, $\mathbf{h}_0$ фиксирано.

- Функцията g зависи от конкретното влагане и конкретната РНМ архитектура (LSTM, GRU, ...), брой слоеве и др.

Условен езиков модел

- Под условен езиков модел ще разбираме система, която за условие x и начална последователност $w_1 w_2 \dots w_i$ ни връща вероятностно разпределение за следващия елемент от последователността при условието x:

- $\Pr[w \mid x, w_1 w_2 \dots w_i]$.

- От Верижното правило следва:
  $\Pr[w_1 w_2 \dots w_n \mid x] = \Pr[w_1 \mid x] \Pr[w_2 \mid x, w_1] \Pr[w_3 \mid x, w_1 w_2] \dots \Pr[w_n \mid x, w_1 w_2 \dots w_{n-1}]$

- Ако нашата система е в състояние експлицитно да представи $\Pr[w \mid x, w_1 w_2 \dots w_i]$, то тя може да се използва за **условна генерация на текст.**

- Условното генериране на текст често се нарича **декодиране.**

Обучение на условен езиков модел

- За обучението ни е необходим корпус от двойки:
  $\mathbf{C} = \{(x^{(i)}, \mathbf{w}^{(i)}) \mid i = 1,2,\ldots,N\}.$

- Обучението ще извършим като минимизираме крос-ентропията:

$$H_{\mathbf{C}} = -\frac{1}{\|\mathbf{C}\|} \sum_{k=1}^{N} \sum_{i=1}^{|\mathbf{w}^{(k)}|} \log \Pr[\mathbf{w}_i^{(k)} \mid x^{(k)}, \mathbf{w}_1^{(k)}, \mathbf{w}_2^{(k)}, \ldots, \mathbf{w}_{i-1}^{(k)}]$$

- За целта отново ще използваме спускане по стохастичен градиент.

Реализиране на условен езиков модел с PHM

- Съществуват различни подходи за реализиране на условен езиков модел с PHM:

1. $\Pr[w \mid x, w_1 w_2 \ldots w_i] = \mathrm{softmax}(U\mathbf{h}_i)_w$, където $\mathbf{h}_j = g(\mathbf{h}_{j-1}, w_j)$, и $\mathbf{h}_0 = f(x)$ (initial binding)

2. $\Pr[w \mid x, w_1 w_2 \ldots w_i] = \mathrm{softmax}(U\mathbf{h}_i)_w$, където $\mathbf{h}_j = g(\mathbf{h}_{j-1}, f(w_j, x))$ (early binding)

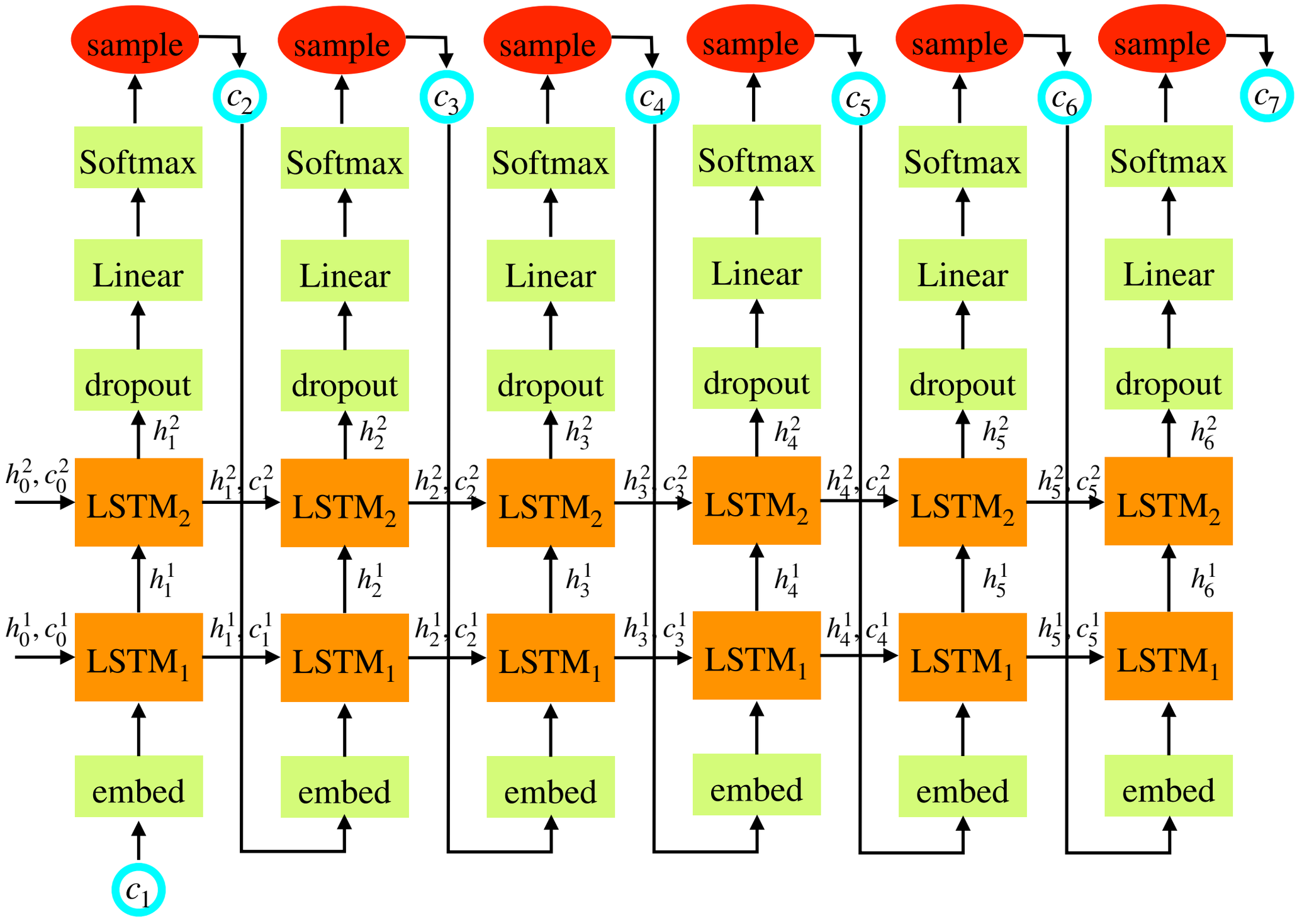
3. $\Pr[w \mid x, w_1 w_2 \ldots w_i] = \mathrm{softmax}(Uf(\mathbf{h}_i, x))_w$, където $\mathbf{h}_j = g(\mathbf{h}_{j-1}, w_j)$ (late binding)

- Също така, различните подходи могат да се комбинират.

План на лекцията

1. Формалности за курса (5 мин)
2. Условни езикови модел (15 мин)
3. **Методи за генерация на текст / декодиране (20 мин)**
4. Приложения на генерация на текст с езиков модел (5 мин)
5. Архитектура енкодер-декодер за невронен машинен превод (20 мин)
6. Архитектура енкодер-декодер с “внимание” (Attention) (20 мин)
7. Transformer архитектура (30 мин)
8. Оценяване на резултат от машинен превод (10 мин)

Генерация на текст в домашно задание №3



Метод на семплиране за генерация на текст с езиков модел

1. Започваме с празната последователност $\mathbf{w}^{(1)} = \varepsilon$.

2. Нека сме получили последователността $\mathbf{w}^{(i)} = \hat{w}_1\hat{w}_2\ldots\hat{w}_{i-1}$. Намираме разпределението $\mathbf{Pr}[w \mid \hat{w}_1\hat{w}_2\ldots\hat{w}_{i-1}]$.

3. Избираме $\hat{w}_i$ като семплираме с разпределението $\mathbf{Pr}[w \mid \hat{w}_1\hat{w}_2\ldots\hat{w}_{i-1}]$. С така избрания елемент разширяваме последователността: $\mathbf{w}^{(i+1)} = \hat{w}_1\hat{w}_2\ldots\hat{w}_{i-1}\hat{w}_i$.

4. Ако $\hat{w}_i$ е символът за край на последователност, то извеждаме така получената последователност и прекратяваме процедурата.

5. В противен случай отиваме в точка 2.

Особености на метода на семплиране за генерация на текст с езиков модел

- Методът на семплиране за генерация на текст с езиков модел може да се използва и в случая на условен езиков модел.

- Този метод дава вариативност на резултата, която е желана при някои приложения, но е нежелана при други приложения.

- Няма изисквания за крос-ентропията на получената последователност.

$$H_{\mathbf{w}} = -\frac{1}{|\mathbf{w}|}\sum_{i=1}^{|\mathbf{w}|}\log \Pr[\mathbf{w}_i \,|\, \mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{i-1}]$$

Декодиране при условни езикови модели

- В приложенията, включващи условен езиков модел, целта е да минимизираме скоростта на крос-ентропия при дадено условие x.
- Т.е. задачата е да намерим последователността

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} -\frac{1}{|\mathbf{w}|}\sum_{i=1}^{|\mathbf{w}|}\log \Pr[\mathbf{w}_i \,|\, x, \mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{i-1}]$$

- Горната задача в общия случай е неразрешима (съществуват безкраен брой последователности).

Метод на “Алчно декодиране” (Greedy decoding)

1. При дадено условие x започваме с празната последователност $\mathbf{w}^{(1)} = \varepsilon$.

2. Нека сме получили последователността $\mathbf{w}^{(i)} = \hat{w}_1\hat{w}_2\ldots\hat{w}_{i-1}$. Намираме разпределението $\Pr[w \mid x, \hat{w}_1\hat{w}_2\ldots\hat{w}_{i-1}]$.

3. Избираме елементът $\hat{w}_i$, който е най-вероятен при даденото разпределение. Т.е. $\hat{w}_i = \arg\max_w \Pr[w \mid x, \hat{w}_1\hat{w}_2\ldots\hat{w}_{i-1}]$. С така избрания елемент разширяваме последователността: $\mathbf{w}^{(i+1)} = \hat{w}_1\hat{w}_2\ldots\hat{w}_{i-1}\hat{w}_i$.

4. Ако $\hat{w}_i$ е символът за край на последователност, то извеждаме така получената последователност и прекратяваме процедурата.

5. В противен случай отиваме в точка 2.

Особености на метода на алчно декодиране

- Целта ни е да получим последователност, която минимизира скоростта на крос-ентропия.

- Няма гаранция, че ще получим оптимален резултат.

- Често се случва да изберем по средата на последователността даден елемент, поради което от там нататък генерираме грешна последователност при даденото условие.

- Няма ли начин да се върнем назад — някакъв вид back-tracking?

Метод на търсене по лъча (Beam search)
Ширина на лъча β

---

1. При дадено услови x започваме с празната последователност $\mathbf{w}^{(1)} = \varepsilon$.

2. Нека сме получили не повече от β на брой последователности $\mathbf{w}^{(i,1)}, \mathbf{w}^{(i,2)}, \ldots, \mathbf{w}^{(i,\beta)}$, където $\mathbf{w}^{(i,j)} = \hat{w}_1^{(i,j)}\hat{w}_2^{(i,j)}\ldots\hat{w}_{i-1}^{(i,j)}$. За всяка от тях намираме разпределението $\Pr[w \mid x, \hat{\mathbf{w}}^{(i,j)}]$.

3. От разпределението $\Pr[w \mid x, \hat{\mathbf{w}}^{(i,j)}]$ намираме β на брой най-вероятни елемента $\hat{w}_1^{(i,j)}, \hat{w}_2^{(i,j)}, \ldots, \hat{w}_\beta^{(i,j)}$. С така избраните елементи разширяваме последователностите и получаваме β^2 кандидат последователности $\mathbf{w}^{(i,1)}\hat{w}_1^{(i,1)}, \ldots, \mathbf{w}^{(i,1)}\hat{w}_\beta^{(i,1)}, \ldots, \mathbf{w}^{(i,\beta)}\hat{w}_1^{(i,\beta)}, \ldots, \mathbf{w}^{(i,\beta)}\hat{w}_\beta^{(i,\beta)}$.

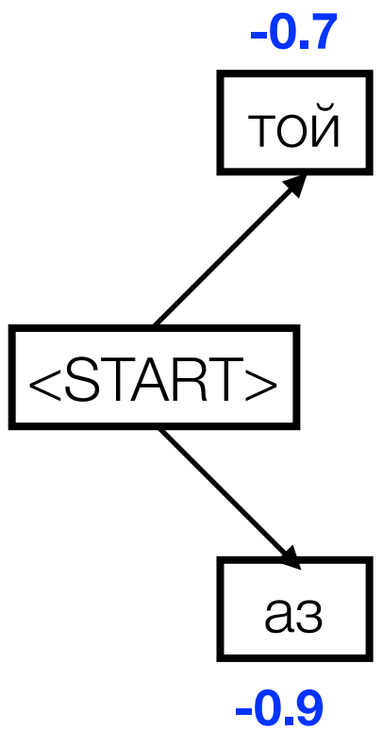
4. От получените β^2 кандидат последователности намираме β с най-висока скорост на крос-ентропия $H_{\mathbf{w}^{(i,j)}\hat{w}_k^{(i,j)}}$. Получените β на брой последователности означаваме като $\mathbf{w}^{(i+1,1)}, \mathbf{w}^{(i+1,2)}, \ldots, \mathbf{w}^{(i+1,\beta)}$.

5. От списъка с последователностите премахваме и извеждаме,завършващите със символът за край на последователност. Ако сме получили достатъчно изведени последователности, избираме най-добрия и прекратяваме процедурата.

6. В противен случай отиваме в точка 2.

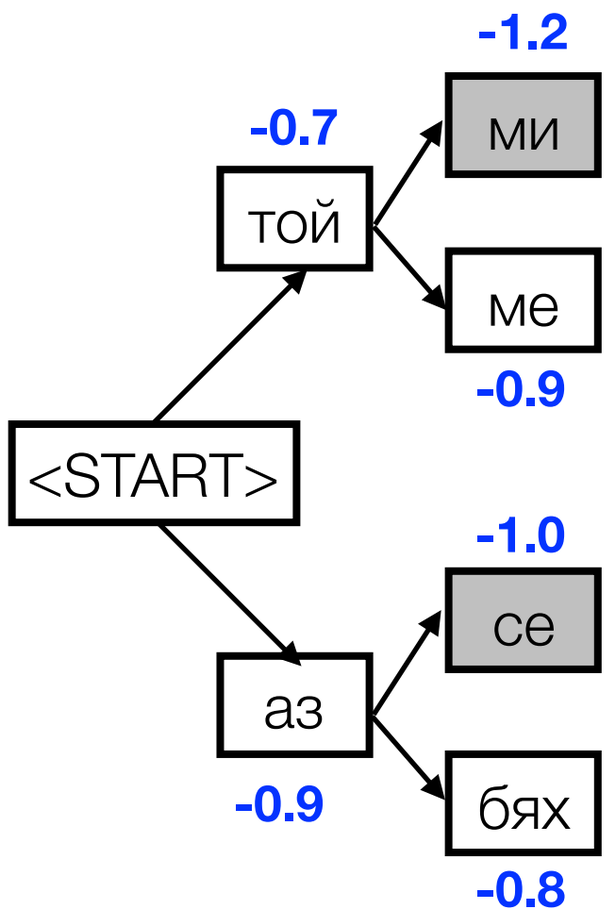
Пример за търсене по лъча ($\beta = 2$)

Пример за търсене по лъча ($\beta = 2$)



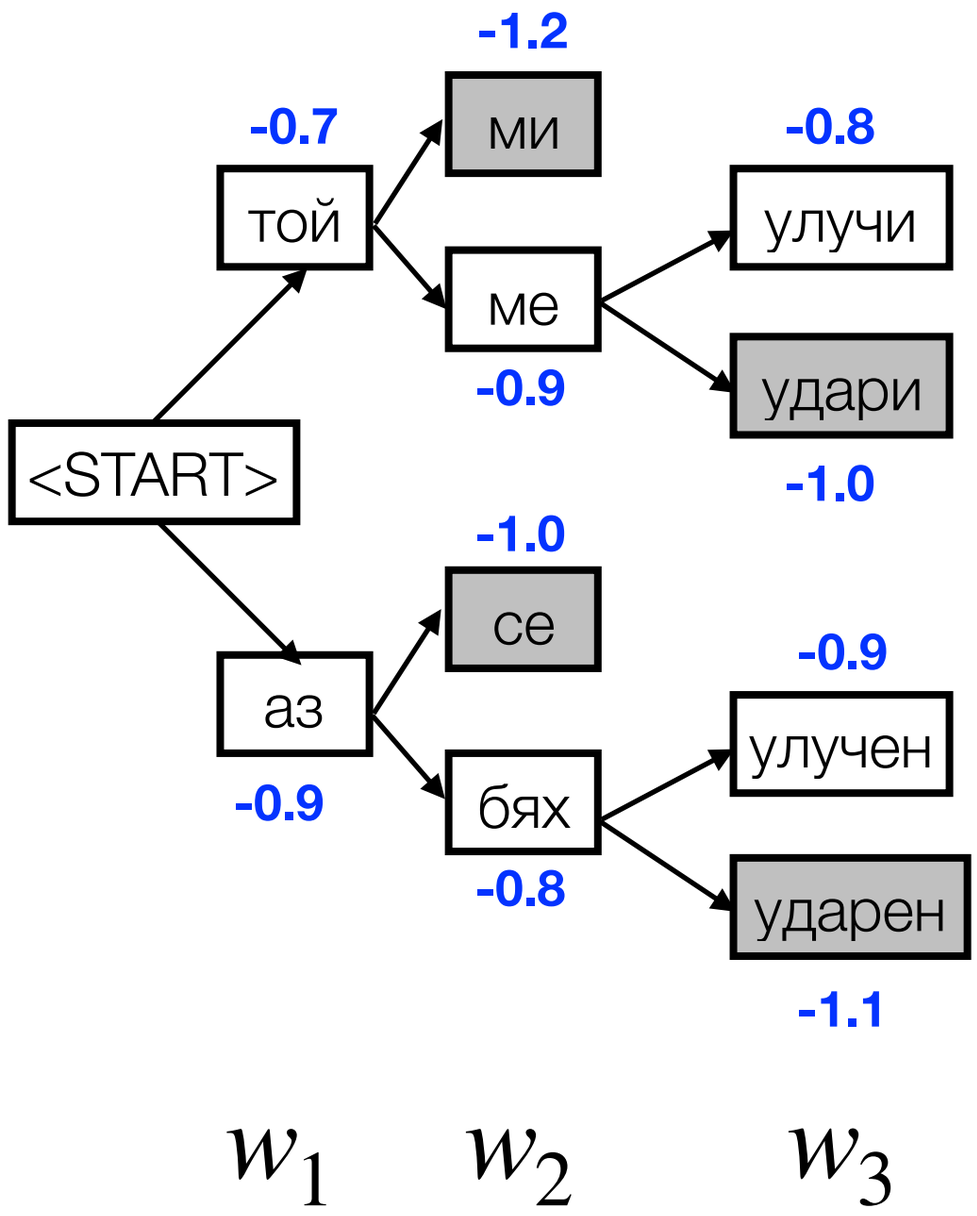
w_1

Пример за търсене по лъча ($\beta = 2$)

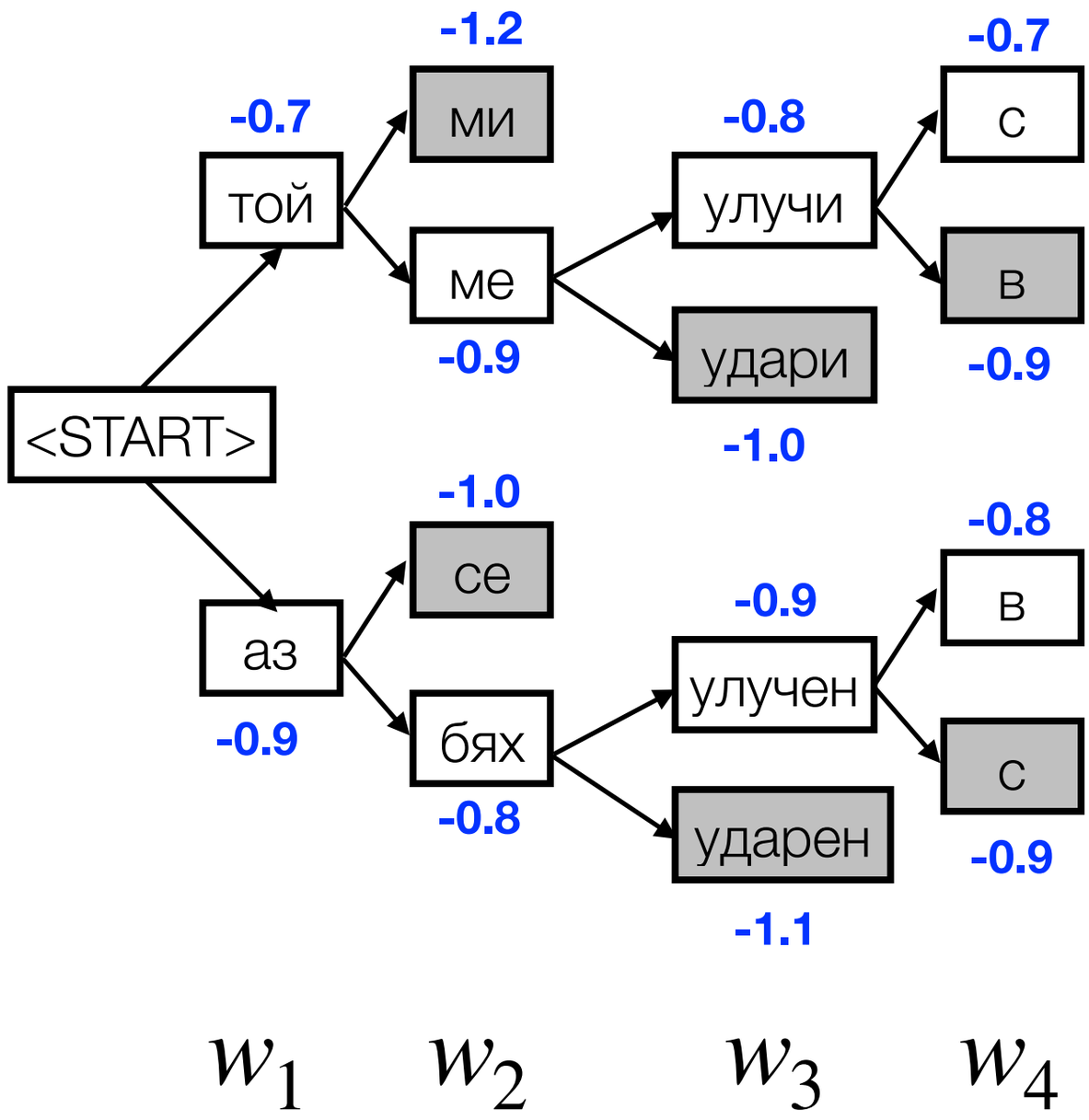


w_1 w_2

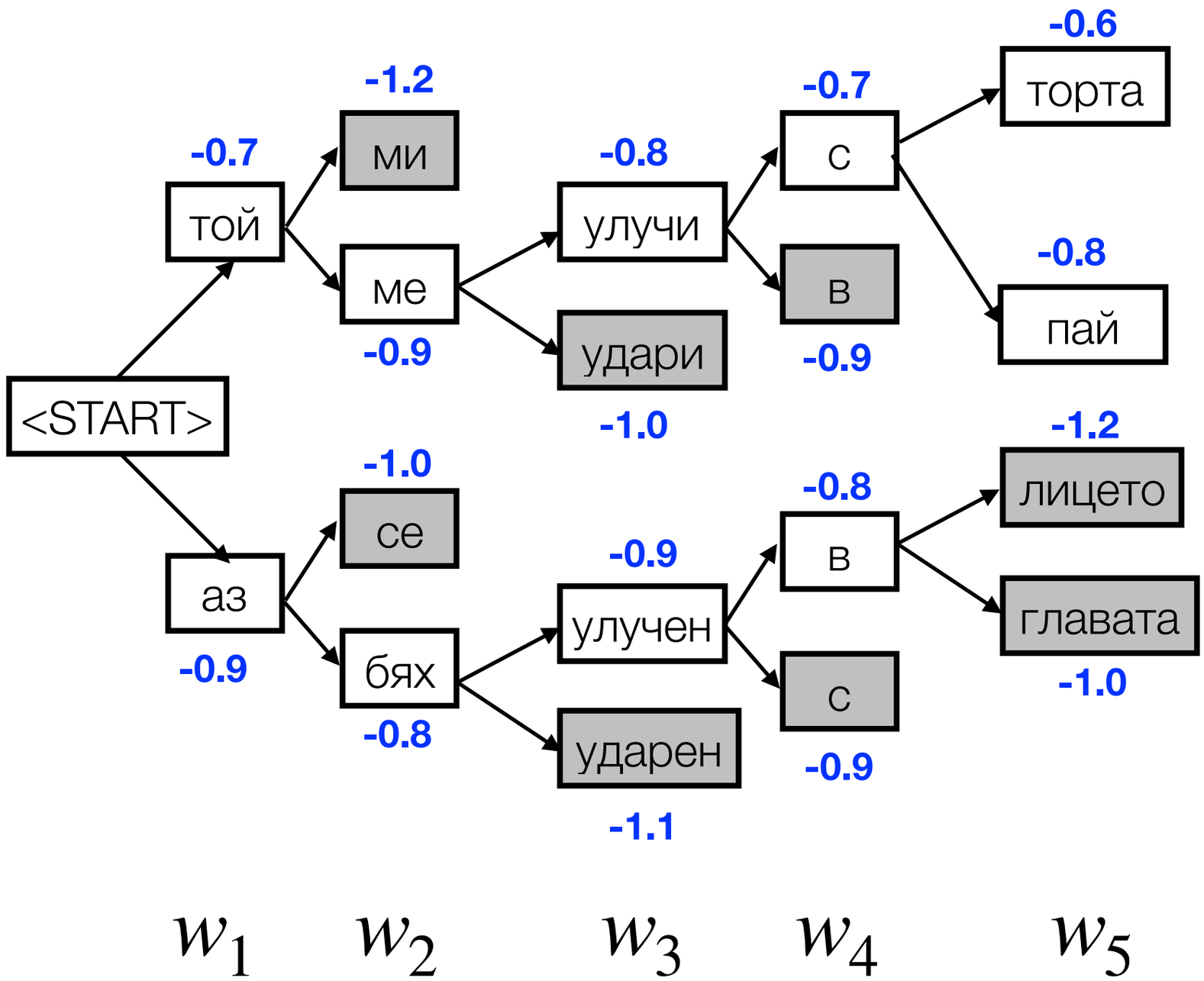
Пример за търсене по лъча ($\beta = 2$)



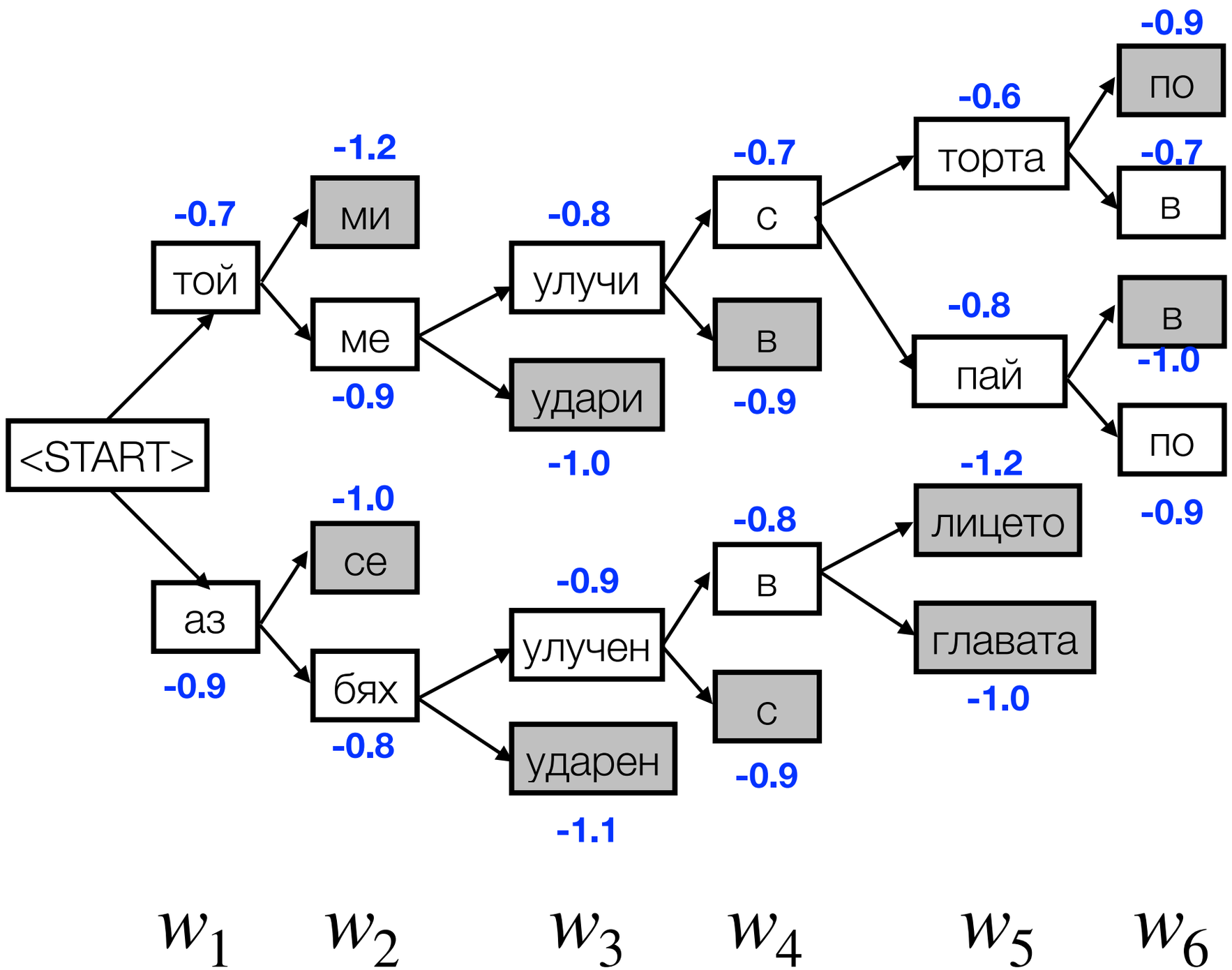
Пример за търсене по лъча ($\beta = 2$)



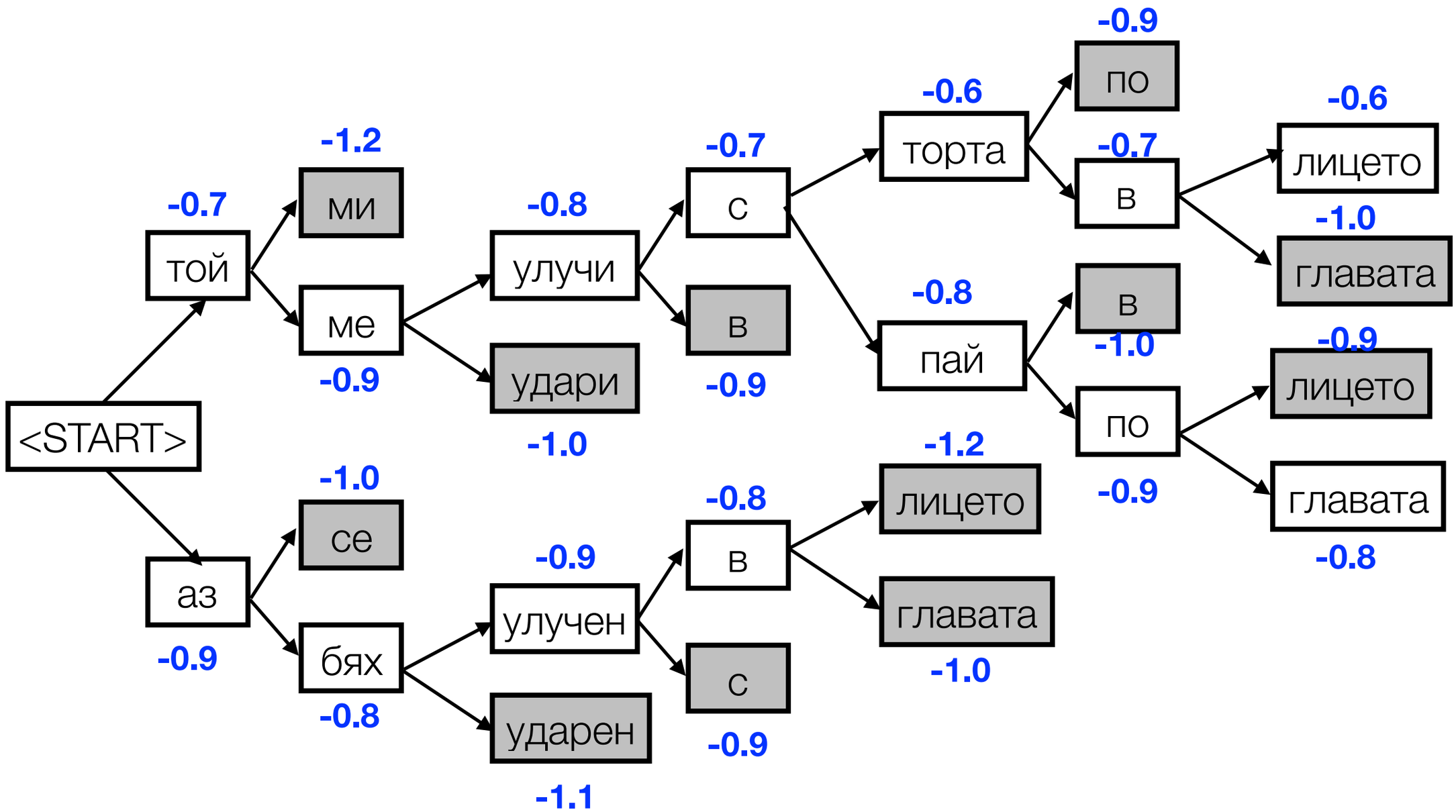
Пример за търсене по лъча ($\beta = 2$)



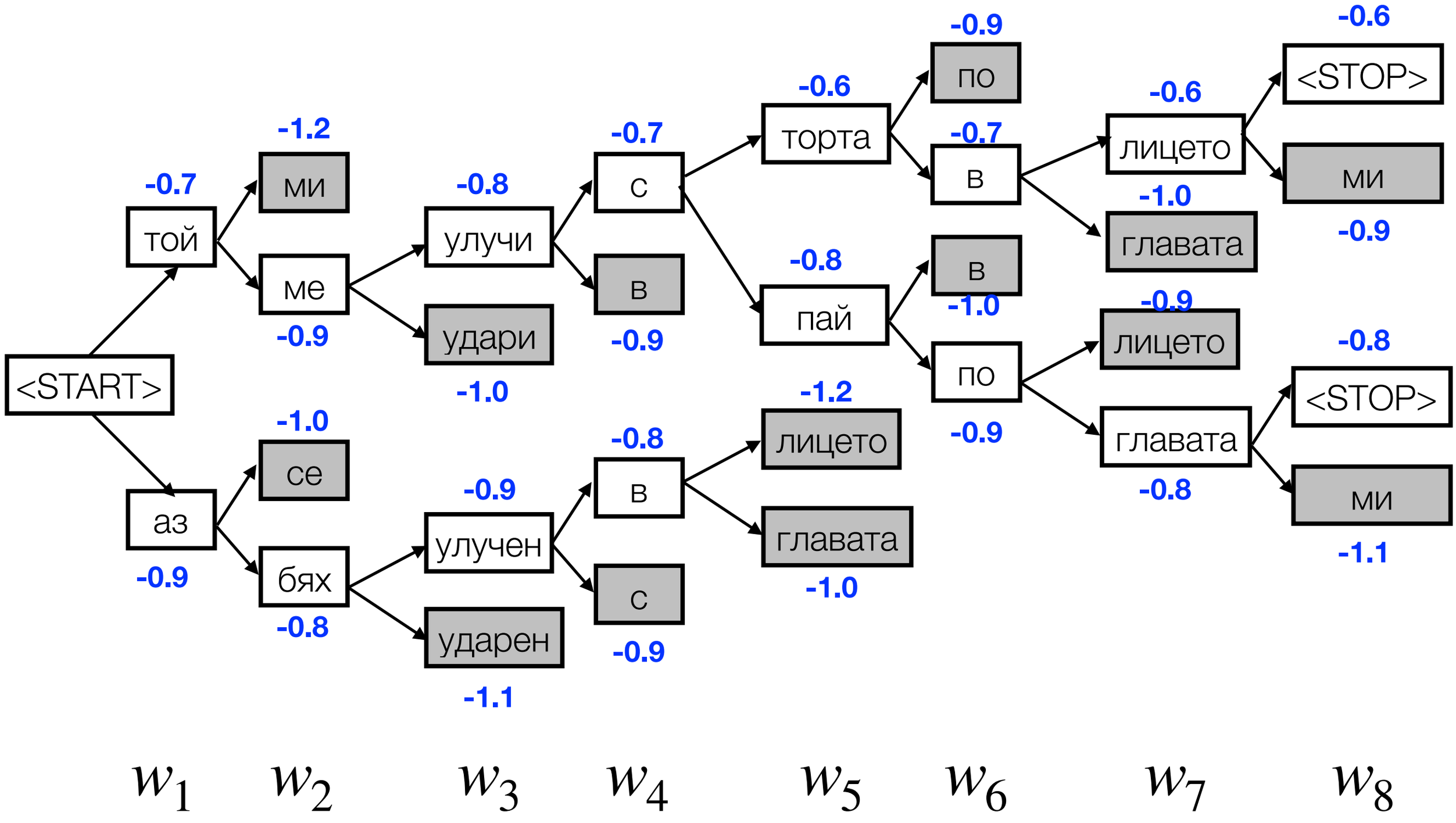
Пример за търсене по лъча ($\beta = 2$)



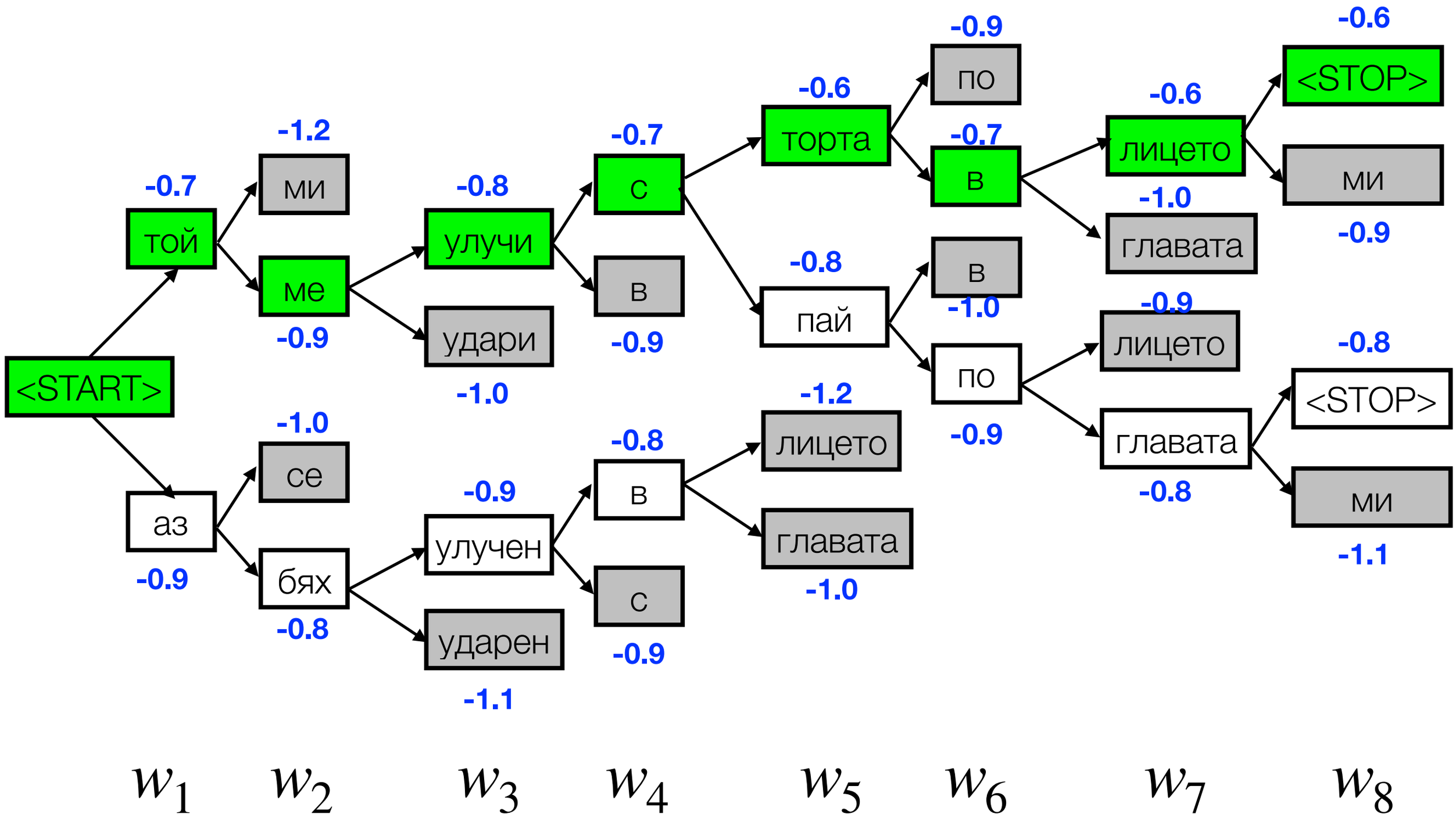
Пример за търсене по лъча ($\beta = 2$)



Пример за търсене по лъча ($\beta = 2$)



Пример за търсене по лъча ($\beta = 2$)



Особености при търсенето по лъча

- Не гарантира намирането на най-вероятната последователност.
- При добър избор на β резултатът е по-добър от алчния избор.
- Представя евристичен компромис между изчислителна ефективност и пълнота и коректност.
- На практика параметърът β се избира в порядък 3-500 с напасване.

План на лекцията

1. Формалности за курса (5 мин)
2. Условни езикови модел (15 мин)
3. Методи за генерация на текст / декодиране (20 мин)
4. **Приложения на генерация на текст с езиков модел (5 мин)**
5. Архитектура енкодер-декодер за невронен машинен превод (20 мин)
6. Архитектура енкодер-декодер с “внимание” (Attention) (20 мин)
7. Transformer архитектура (30 мин)
8. Оценяване на резултат от машинен превод (10 мин)

Приложения на условната генерация на текст

Условие x	**Изходен текст w**
Автор	Текст със стила на автора
Тема	Статия по темата
Изречение на английски	Превод на български
Снимка	Описание на снимката
Статия	Резюме
Запис на реч	Транскрипция
Въпрос + статия	Отговор

Модел “Последователност към последователност”

- При тези модели входна последователност от елементи се проебразува в изходна последователност.

- Примери:
  - Машинен превод от един език на друг
  - Автоматично разпознаване на реч
  - Генериране на резюме на статия

Модел “Последователност към последователност”

- Условието е последователност: $\mathbf{x} = x_1 x_2 \ldots x_l$.

- Търсим последователност $\mathbf{w} = w_1 w_2 \ldots w_k$, така че

$$\mathbf{w} = \arg\max_{\mathbf{w}} \Pr[\mathbf{w} \mid \mathbf{x}] = \arg\max_{\mathbf{w}} \prod_{i=1}^{k} \Pr[w_i \mid \mathbf{x}, w_1 w_2 \ldots w_{i-1}]$$

- Вместо да максимизираме правдоподобието ще минимизираме скоростта на крос-ентропията:

$$\mathbf{w} = \arg\min_{\mathbf{w}} -\frac{1}{|\mathbf{w}|} \sum_{i=1}^{|\mathbf{w}|} \log \Pr[w_i \mid \mathbf{x}, w_1, w_2, \ldots, w_{i-1}]$$

- При последователности с равни дължини решенията съвпадат. В случай, че дължините не са равни трябва да нормализираме спрямо дължината. Съществуват и други начини за отчитане на дължината.

  Wu et al. (2016): Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. https://arxiv.org/abs/1609.08144)

План на лекцията

1. Формалности за курса (5 мин)
2. Условни езикови модел (15 мин)
3. Методи за генерация на текст / декодиране (20 мин)
4. Приложения на генерация на текст с езиков модел (5 мин)
5. **Архитектура енкодер-декодер за невронен машинен превод (20 мин)**
6. Архитектура енкодер-декодер с “внимание” (Attention) (20 мин)
7. Transformer архитектура (30 мин)
8. Оценяване на резултат от машинен превод (10 мин)

Статистически (преди невронен) машинен превод

- $\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} \Pr[\mathbf{y} \mid \mathbf{x}] = \arg\max_{\mathbf{y}} \Pr[\mathbf{x} \mid \mathbf{y}] \Pr[\mathbf{y}]$
- $\Pr[\mathbf{y}]$ е езиков модел на целевия езикът
- $\Pr[\mathbf{x} \mid \mathbf{y}]$ е преводен модел — ще се стремим да разбием поелементно. За целта ни е необходимо подравняване (alignment) a.
- Подравняването може да разгледаме като функция, която на дадена позиция в изходната последователност съпоставя позиция в целевата последователност. (Има по-добри модели за подравняване.)
- $\Pr[\mathbf{x} \mid \mathbf{y}] = \sum_a \Pr[\mathbf{x}, a \mid \mathbf{y}] = \sum_a \prod_i \Pr[a(i) \mid \mathbf{x}, \mathbf{y}] \Pr[\mathbf{x}_i \mid \mathbf{y}_{a(i)}]$
- Търсим $\hat{\mathbf{y}} = \arg\max_{\mathbf{y},a} \prod_i \Pr[a(i) \mid \mathbf{x}, \mathbf{y}] \Pr[\mathbf{x}_i \mid \mathbf{y}_{a(i)}] \Pr[\mathbf{y}]$

Пример за подравняване

	Аз	бях	на	кино
I				
have				
been				
to				
the				
cinema				

Невронен машинен превод — пробив 2014 г.

Sutskever, Vinyals and Le (2014): Sequence to Sequence Learning with Neural Networks
https://arxiv.org/abs/1409.3215v3

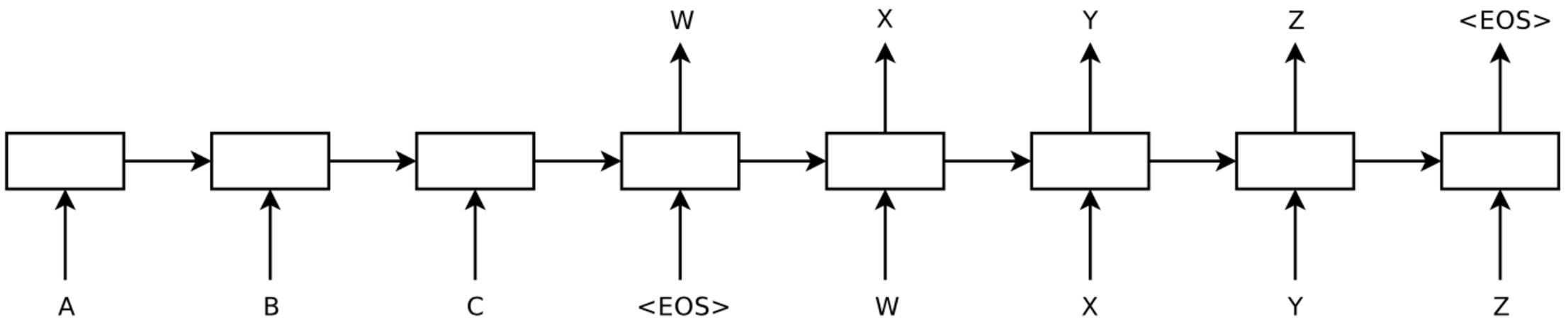
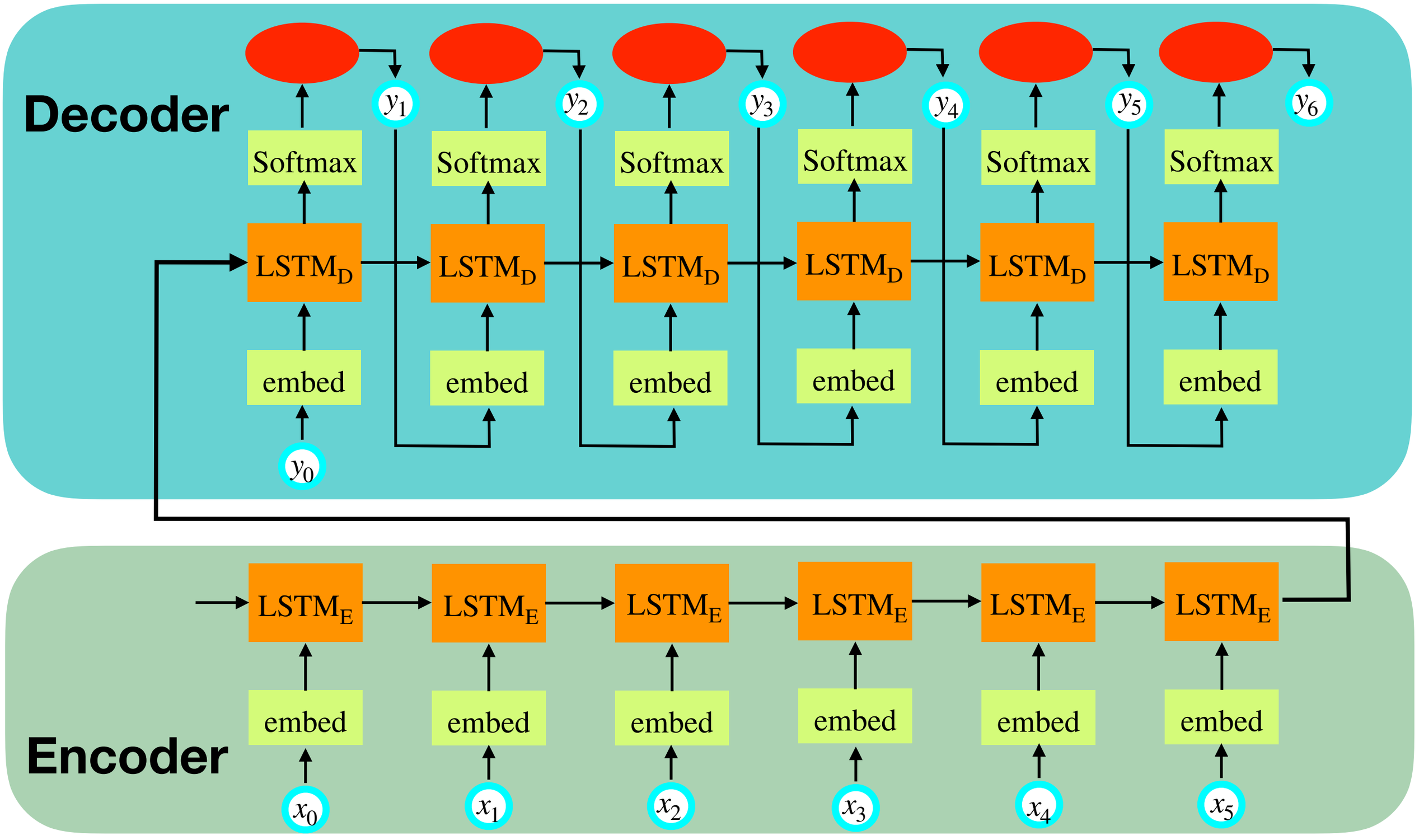
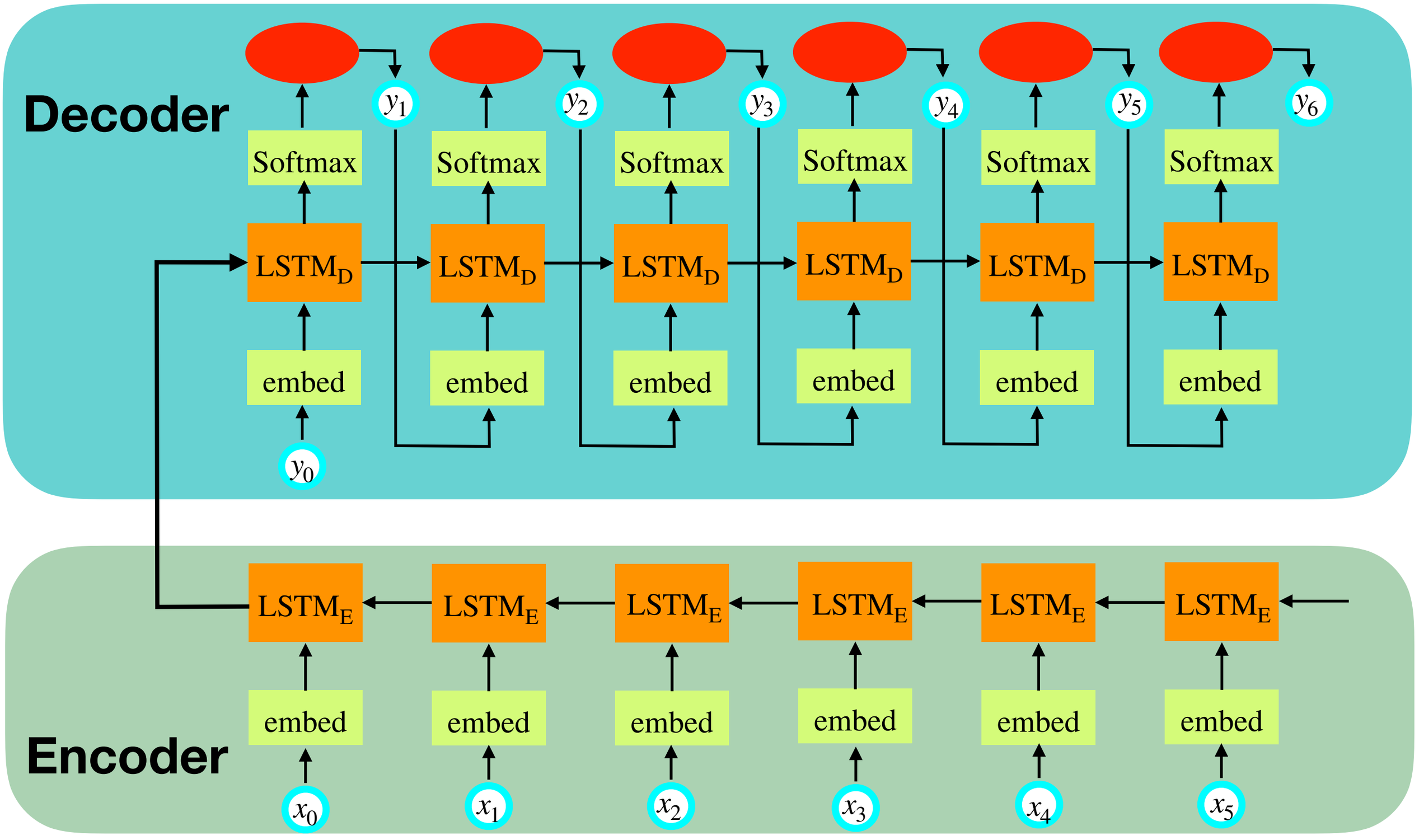


Figure 1: Our model reads an input sentence "ABC" and produces "WXYZ" as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that make the optimization problem much easier.

Невронен машинен превод — пробив 2014 г.



Невронен машинен превод — пробив 2014 г.



Невронен машинен превод — пробив 2014 г.

- В модела на Sutskever входният текст се кодира от LSTM рекурентна невронна мрежа.
- Последният скрит вектор и състояние от LSTM мрежата на енкодера се подава като начален скрит вектор и състояние на LSTM мрежата на декодера.
- Това съответства на първия подход за реализиране на условен езиков модел с PHM (initial binding).
- Експериментите описани в статията показват по-добри резултати при обръщане на входната последователност.
- **Проблем**: Размерността на скрития вектора, кодиращ входната последователност не зависи от дължината му.

Prof. Ray Mooney: “You can't cram the meaning of a whole %&!$# sentence into a single $&!# vector!”*

План на лекцията

1. Формалности за курса (5 мин)
2. Условни езикови модел (15 мин)
3. Методи за генерация на текст / декодиране (20 мин)
4. Приложения на генерация на текст с езиков модел (5 мин)
5. Архитектура енкодер-декодер за невронен машинен превод (20 мин)
6. **Архитектура енкодер-декодер с “внимание” (Attention) (20 мин)**
7. Transformer архитектура (30 мин)
8. Оценяване на резултат от машинен превод (10 мин)

Реализиране на архитектура за “внимание” — Attention

- Резултатът от кодирането на входната последователност не е само последния скрит вектор, а последователността от всички скрити вектори по пътя — $\mathbf{h}_1^E, \mathbf{h}_2^E, \ldots, \mathbf{h}_l^E$. Размерът зависи от дължината на последователността.

- Архитектурата на декодера очаква вход с фиксиран размер.

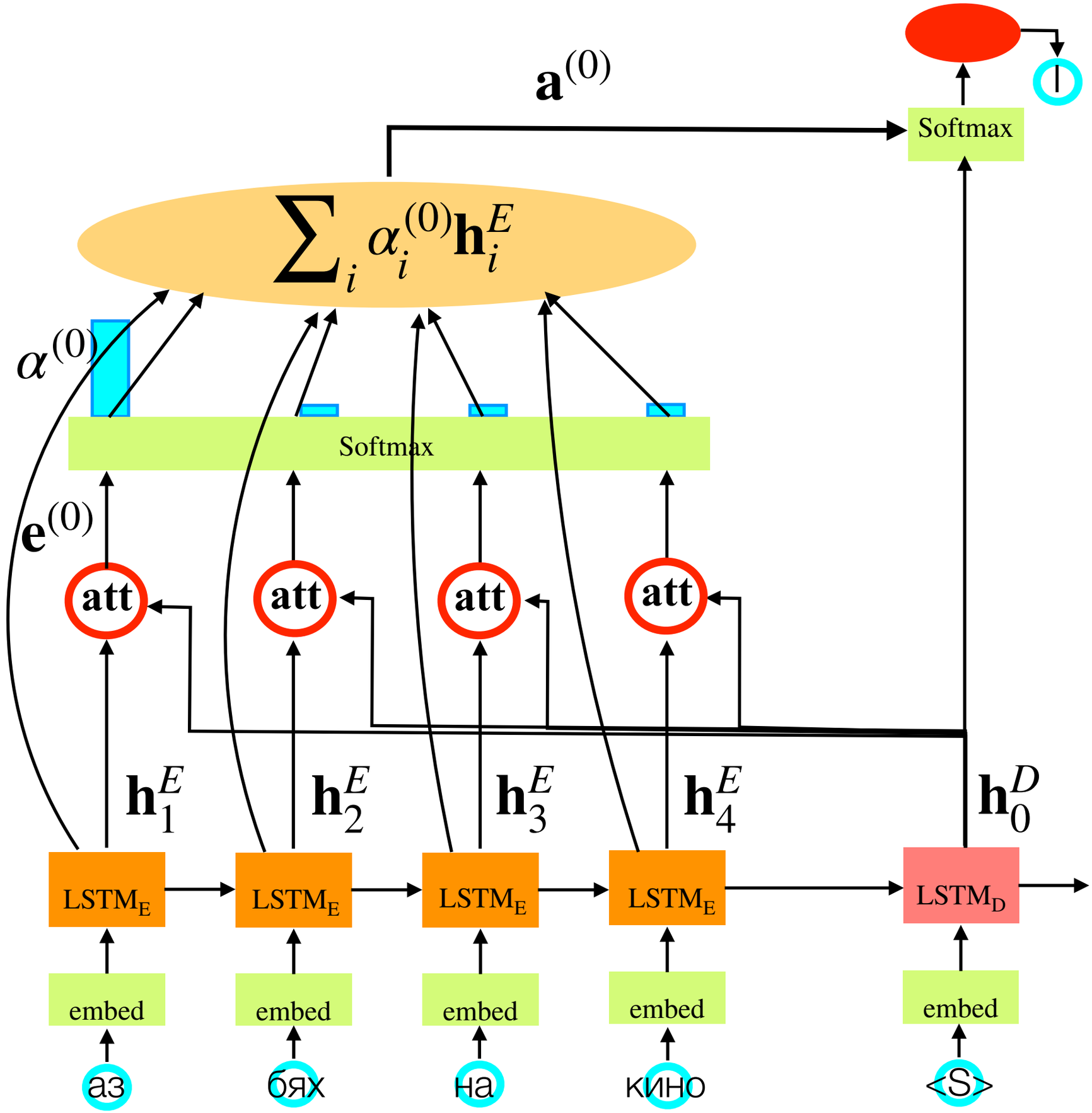
- В дадена позиция резултатът от декодирането следва да зависи повече от скритията вектор, който е около съответната подравнена позиция при кодирането на входната последователност.

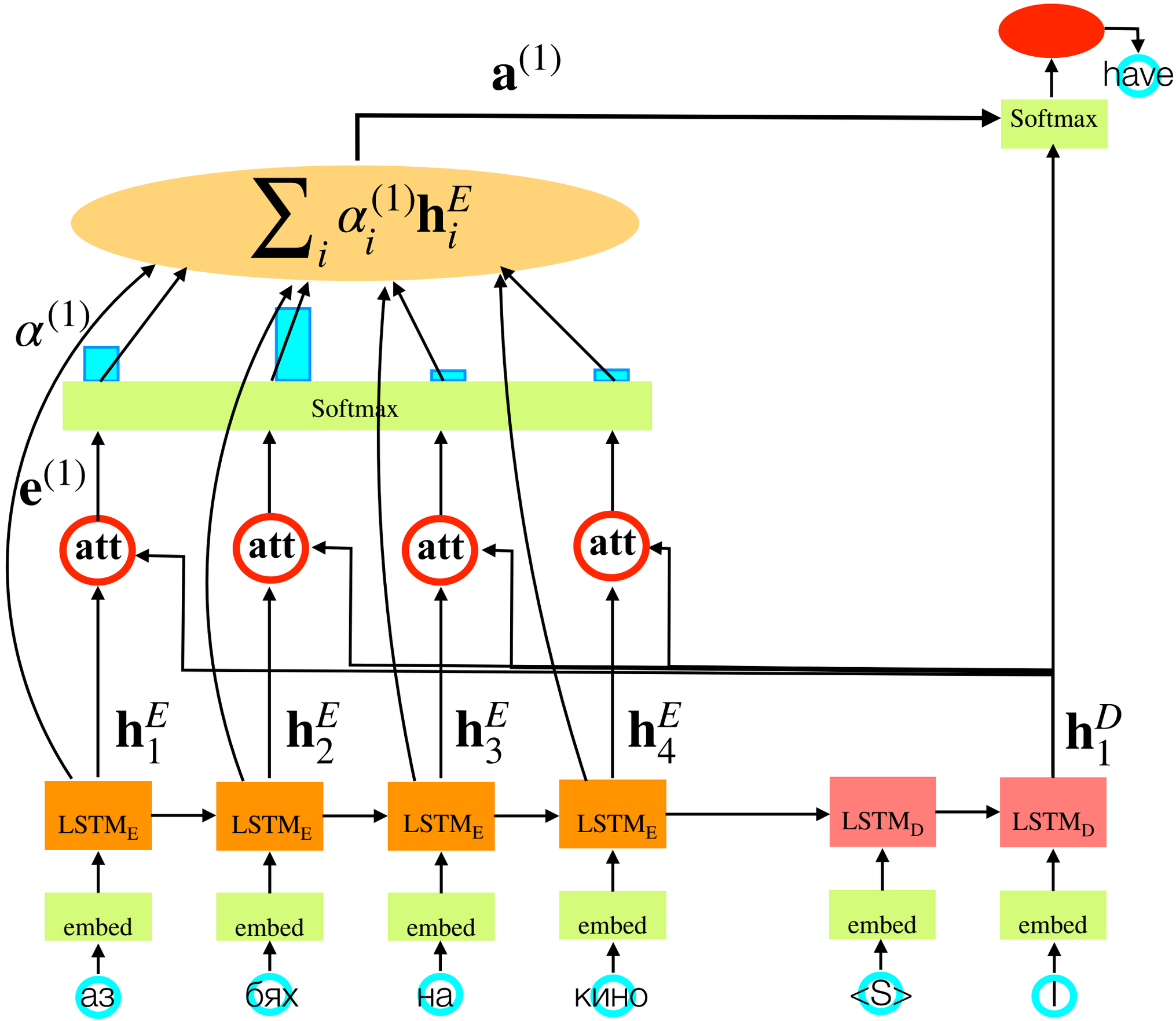
- Ще реализираме архитектура на внимание за моделирането на подравняване.

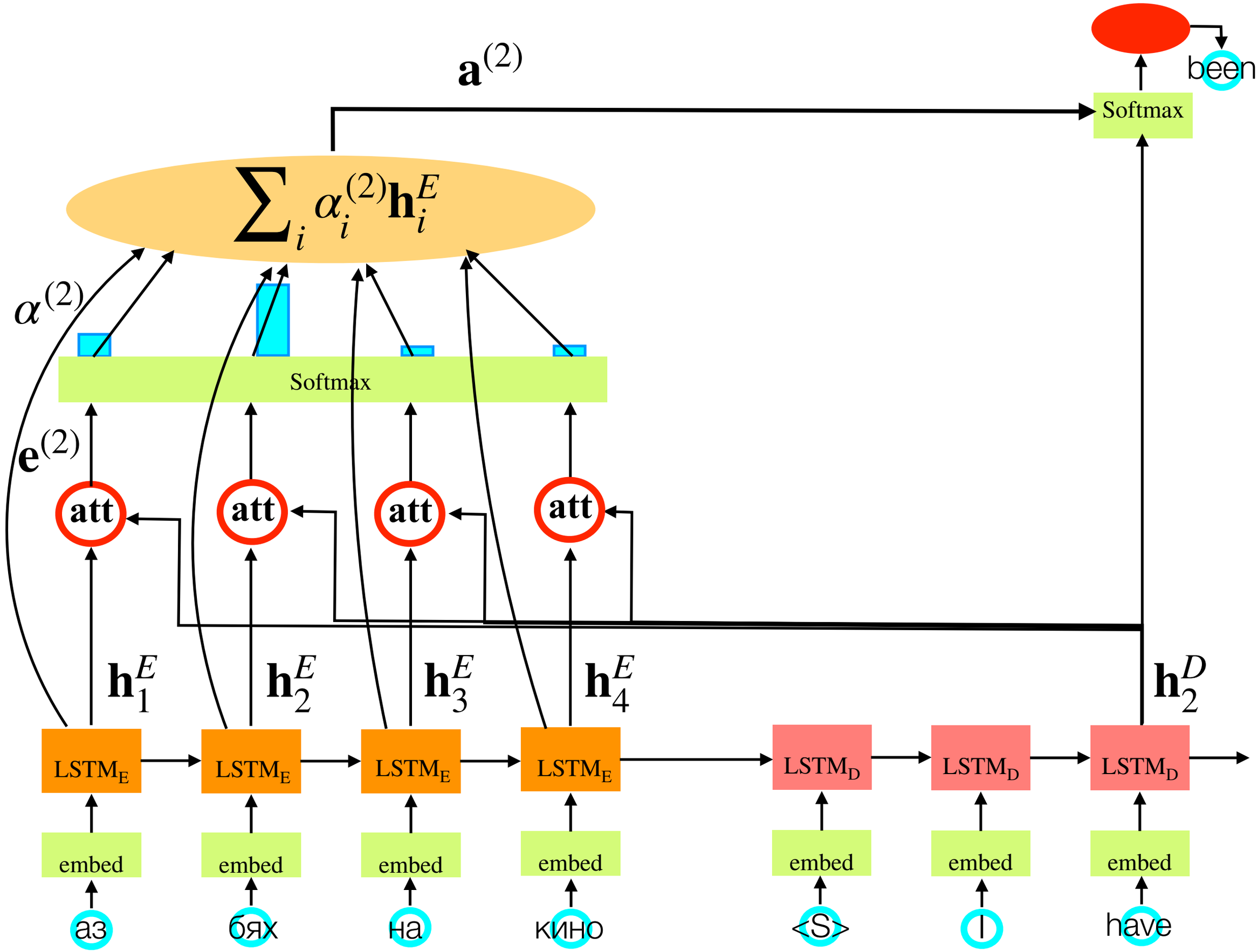
Bahdanau, Cho and Bengio (2014): Neural Machine Translation by Jointly Learning to Align and Translate,
https://arxiv.org/abs/1409.0473

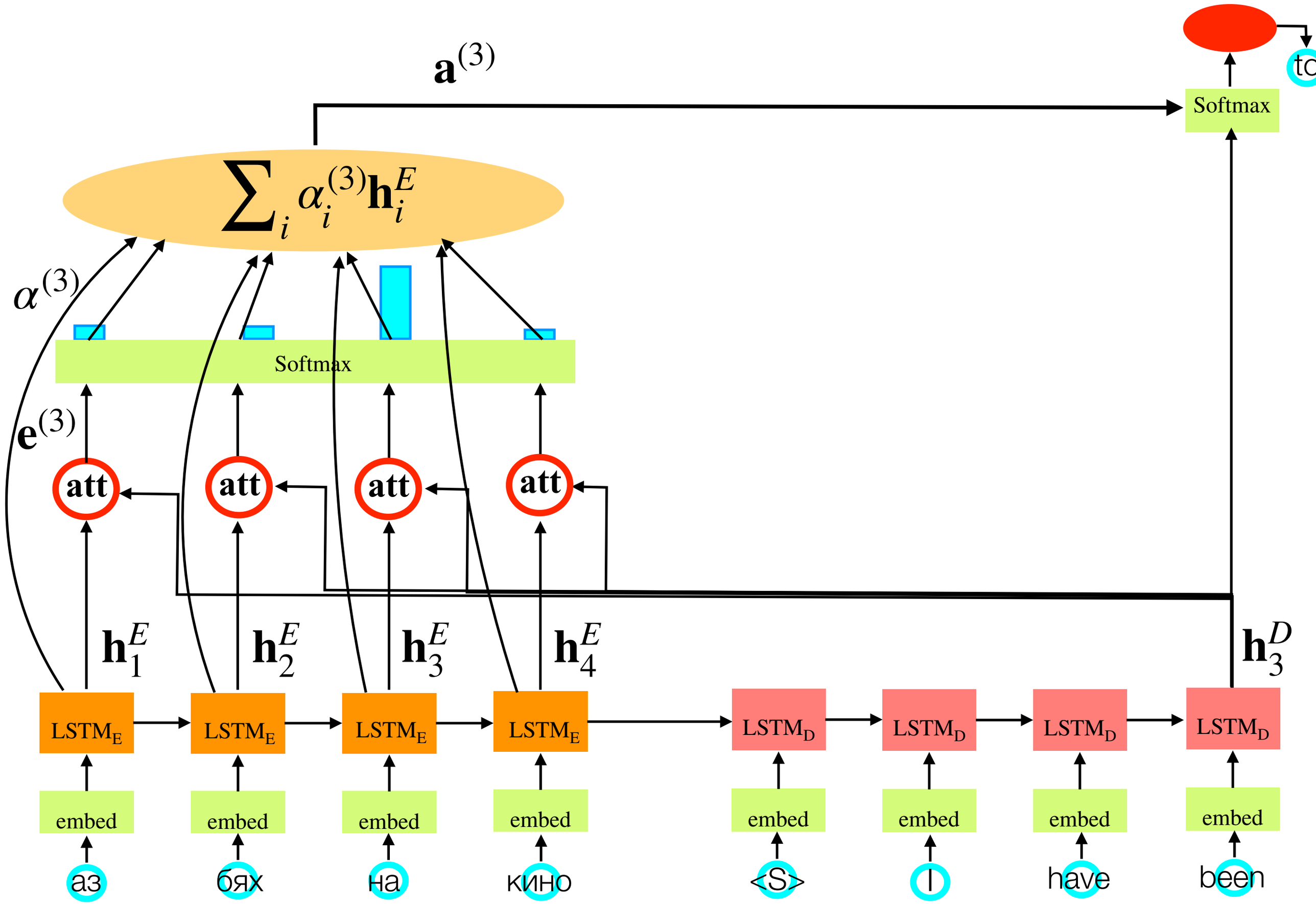
Реализиране на внимание

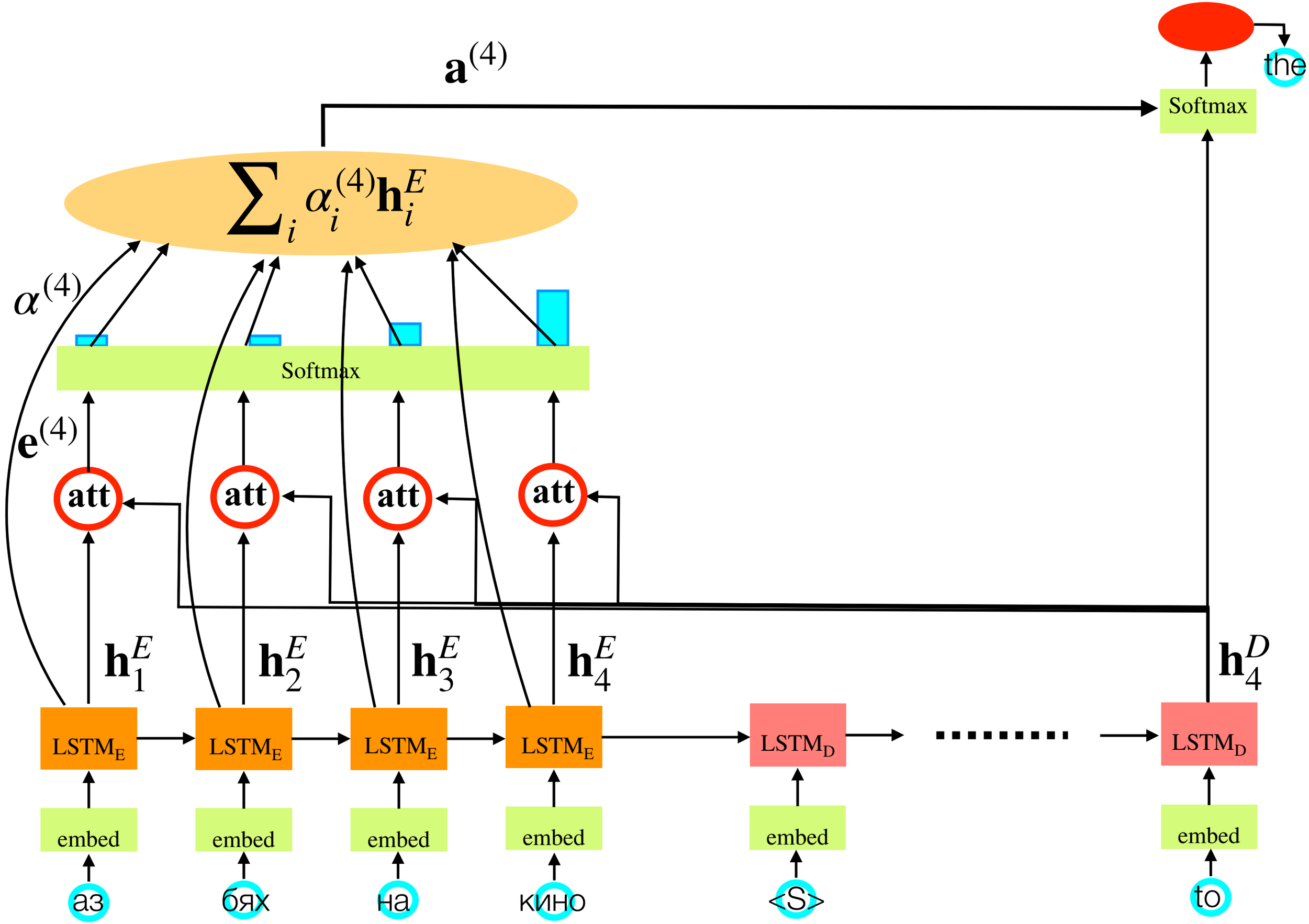
- Нека входната последователност $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_l$ е кодирана в последователността от скрити вектори $\mathbf{h}_1^E, \mathbf{h}_2^E, \ldots, \mathbf{h}_l^E$.
- Нека сме стигнали при декодиране до позиция j, на която сме получили скрит вектор $\mathbf{h}_j^D$.
- Създаваме вектора “размер на внимание” (attention score) в позиция j:
  $\mathbf{e}^{(j)} = [\text{att}(\mathbf{h}_j^D, \mathbf{h}_1^E), \text{att}(\mathbf{h}_j^D, \mathbf{h}_2^E), \ldots, \text{att}(\mathbf{h}_j^D, \mathbf{h}_l^E)] \in \mathbb{R}^l$
- Намираме теглата в позиция j: $\alpha^{(j)} = \text{softmax}(\mathbf{e}^{(j)}) \in \mathbb{R}^l$
- Претегляйки скритите вектори при декодиране с теглата получаваме вектора на внимание в позиция j: $\mathbf{a}^{(j)} = \sum_{i=1}^{l} \alpha_i^{(j)} \mathbf{h}_i^E \in \mathbb{R}^h$
- Накрая конкатенираме вектора на внимание с скрития вектор при декодиране за намирането на следващата дума: $[\mathbf{a}^{(j)}, \mathbf{h}_j^D] \in \mathbb{R}^{2h}$

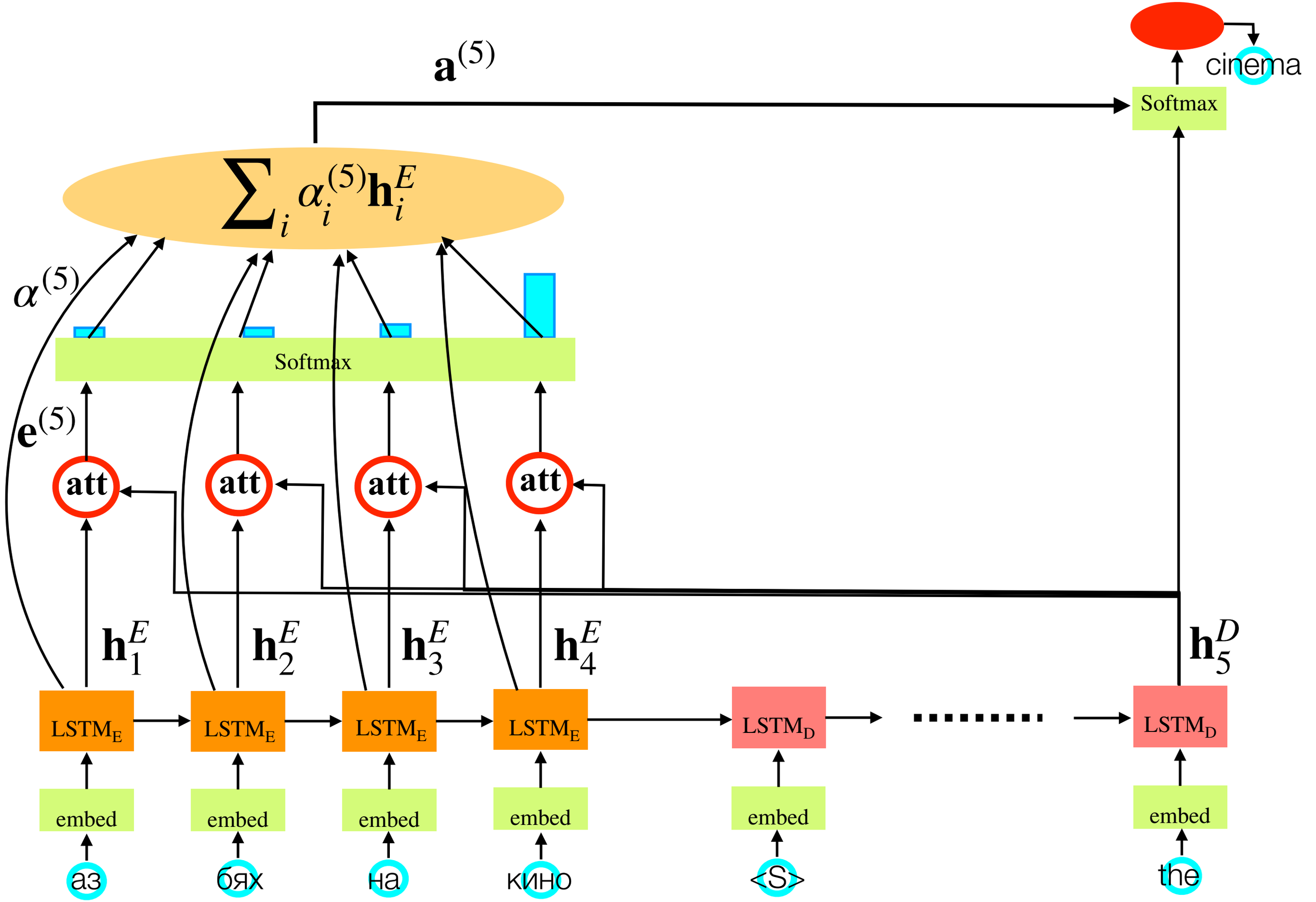

$\mathbf{a}^{(0)}$
$\sum_i \alpha_i^{(0)} \mathbf{h}_i^E$
$\alpha^{(0)}$
Softmax
Softmax
$\mathbf{e}^{(0)}$
att
att
att
att
$\mathbf{h}_1^E$
$\mathbf{h}_2^E$
$\mathbf{h}_3^E$
$\mathbf{h}_4^E$
$\mathbf{h}_0^D$
$LSTM_E$
$LSTM_E$
$LSTM_E$
$LSTM_E$
$LSTM_D$
embed
embed
embed
embed
embed
аз
бях
на
кино
<S>

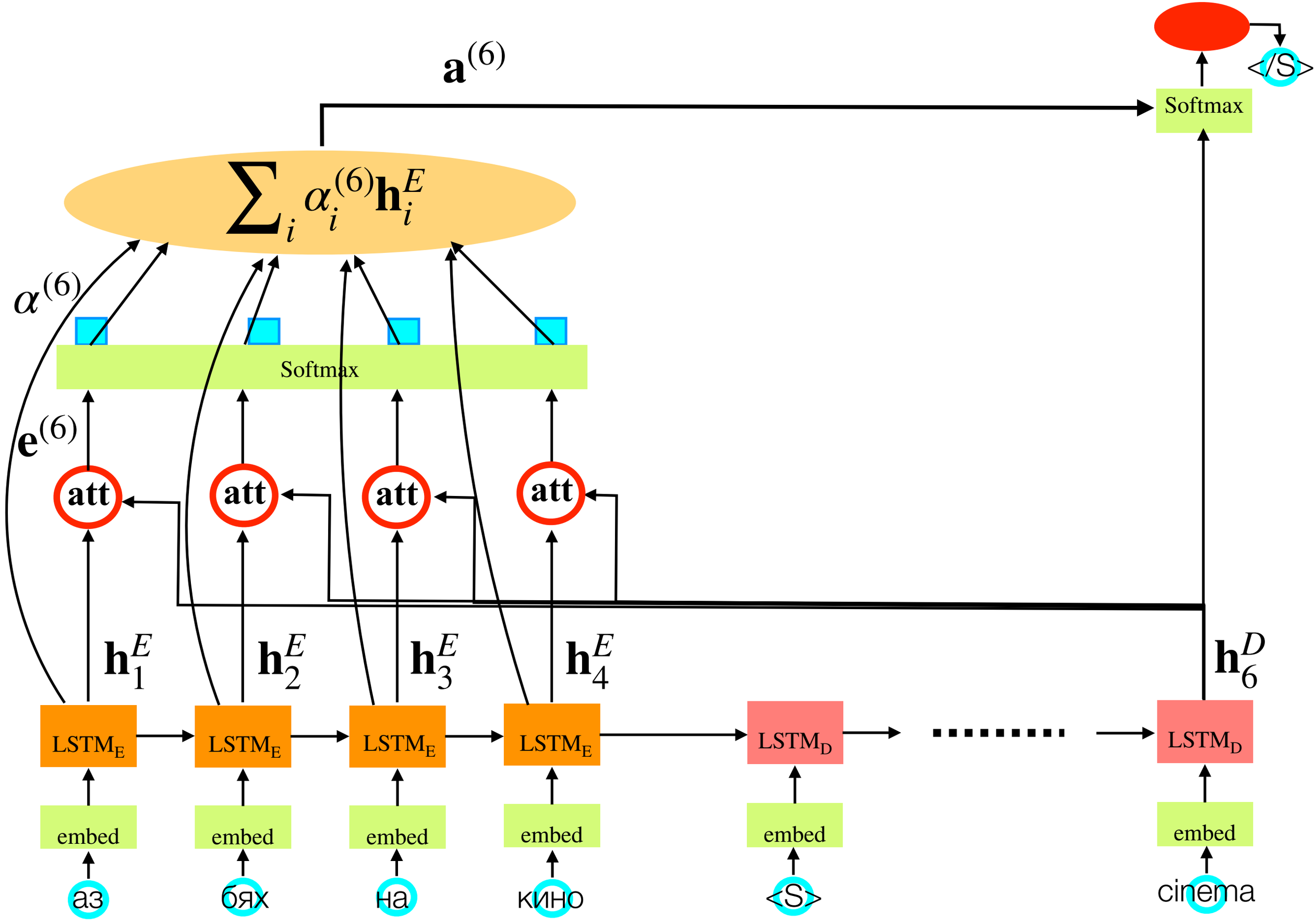

$\mathbf{a}^{(1)}$
$\sum_i \alpha_i^{(1)} \mathbf{h}_i^E$
$\alpha^{(1)}$
Softmax
$\mathbf{e}^{(1)}$
att
att
att
att
Softmax
have
$\mathbf{h}_1^E$
$\mathbf{h}_2^E$
$\mathbf{h}_3^E$
$\mathbf{h}_4^E$
$\mathbf{h}_1^D$
$LSTM_E$
$LSTM_E$
$LSTM_E$
$LSTM_E$
$LSTM_D$
$LSTM_D$
embed
embed
embed
embed
embed
embed
аз
бях
на
кино
<S>
I


$\mathbf{a}^{(2)}$
$\sum_i \alpha_i^{(2)} \mathbf{h}_i^E$
$\alpha^{(2)}$
Softmax
$\mathbf{e}^{(2)}$
att
att
att
att
Softmax
been
$\mathbf{h}_1^E$
$\mathbf{h}_2^E$
$\mathbf{h}_3^E$
$\mathbf{h}_4^E$
$\mathbf{h}_2^D$
$LSTM_E$
$LSTM_E$
$LSTM_E$
$LSTM_E$
$LSTM_D$
$LSTM_D$
$LSTM_D$
embed
embed
embed
embed
embed
embed
embed
аз
бях
на
кино
<S>
I
have


$\mathbf{a}^{(3)}$
$\sum_i \alpha_i^{(3)} \mathbf{h}_i^E$
$\alpha^{(3)}$
Softmax
Softmax
$\mathbf{e}^{(3)}$
att
att
att
att
$\mathbf{h}_1^E$
$\mathbf{h}_2^E$
$\mathbf{h}_3^E$
$\mathbf{h}_4^E$
$\mathbf{h}_3^D$
$LSTM_E$
$LSTM_E$
$LSTM_E$
$LSTM_E$
$LSTM_D$
$LSTM_D$
$LSTM_D$
$LSTM_D$
embed
embed
embed
embed
embed
embed
embed
embed
аз
бях
на
кино
<S>
I
have
been


$\mathbf{a}^{(4)}$
$\sum_i \alpha_i^{(4)} \mathbf{h}_i^E$
$\alpha^{(4)}$
Softmax
$\mathbf{e}^{(4)}$
att
att
att
att
$\mathbf{h}_1^E$
$\mathbf{h}_2^E$
$\mathbf{h}_3^E$
$\mathbf{h}_4^E$
$\mathbf{h}_4^D$
$LSTM_E$
$LSTM_E$
$LSTM_E$
$LSTM_E$
$LSTM_D$
$LSTM_D$
embed
embed
embed
embed
embed
embed
аз
бях
на
кино
<S>
to
Softmax
the


$\mathbf{a}^{(5)}$
cinema
Softmax
$\sum_i \alpha_i^{(5)} \mathbf{h}_i^E$
$\alpha^{(5)}$
Softmax
$\mathbf{e}^{(5)}$
att
att
att
att
$\mathbf{h}_1^E$
$\mathbf{h}_2^E$
$\mathbf{h}_3^E$
$\mathbf{h}_4^E$
$\mathbf{h}_5^D$
$LSTM_E$
$LSTM_E$
$LSTM_E$
$LSTM_E$
$LSTM_D$
$LSTM_D$
embed
embed
embed
embed
embed
embed
аз
бях
на
кино
<S>
the


$\mathbf{a}^{(6)}$
$\sum_i \alpha_i^{(6)} \mathbf{h}_i^E$
$\alpha^{(6)}$
Softmax
$\mathbf{e}^{(6)}$
att
att
att
att
$\mathbf{h}_1^E$
$\mathbf{h}_2^E$
$\mathbf{h}_3^E$
$\mathbf{h}_4^E$
$\mathbf{h}_6^D$
Softmax
</S>
LSTM$_E$
LSTM$_E$
LSTM$_E$
LSTM$_E$
LSTM$_D$
LSTM$_D$
embed
embed
embed
embed
embed
embed
аз
бях
на
кино
<S>
cinema

Варианти за функцията на внимание

$$\mathbf{e}^{(j)} = [\text{att}(\mathbf{h}_j^D, \mathbf{h}_1^E), \text{att}(\mathbf{h}_j^D, \mathbf{h}_2^E), \ldots, \text{att}(\mathbf{h}_j^D, \mathbf{h}_l^E)] \in \mathbb{R}^l$$

1. Най-прост вариант — скаларно произведение: $\text{att}(\mathbf{h}_j^D, \mathbf{h}_i^E) = (\mathbf{h}_j^D)^\top \mathbf{h}_i^E$. В този случай е необходимо размерностите на скритите вектори на енкодера и декодера да съвпадат.

2. Мултипликативно внимание:
   $\text{att}(\mathbf{h}_j^D, \mathbf{h}_i^E) = (\mathbf{h}_j^D)^\top W \mathbf{h}_i^E$, където $W \in \mathbb{R}^{h_D \times h_E}$ е матрица с тегла.

3. Адитивно внимание:
   $\text{att}(\mathbf{h}_j^D, \mathbf{h}_i^E) = \mathbf{v}^\top \tanh(W_1 \mathbf{h}_j^D + W_2 \mathbf{h}_i^E)$, където $W_1 \in \mathbb{R}^{d \times h_D}$, $W_2 \in \mathbb{R}^{d \times h_E}$ са матрици с тегла, $\mathbf{v} \in \mathbb{R}^d$ е вектор с тегла и d е метапараметър.

Експериментите показват, че адитивният вариант 3 дава най-добри резултати, но е изчислително най-тежък. Виж:
Britz et al, 2017: Massive Exploration of Neural Machine Translation Architectures,
https://arxiv.org/abs/1703.03906

План на лекцията

1. Формалности за курса (5 мин)
2. Условни езикови модел (15 мин)
3. Методи за генерация на текст / декодиране (20 мин)
4. Приложения на генерация на текст с езиков модел (5 мин)
5. Архитектура енкодер-декодер за невронен машинен превод (20 мин)
6. Архитектура енкодер-декодер с “внимание” (Attention) (20 мин)
7. **Transformer архитектура (30 мин)**
8. Оценяване на резултат от машинен превод (10 мин)

Проблеми при рекурентните невронни мрежи

- Последователната обработка на входната последователност води до неефективно използване на паралелните изчислителни архитектури

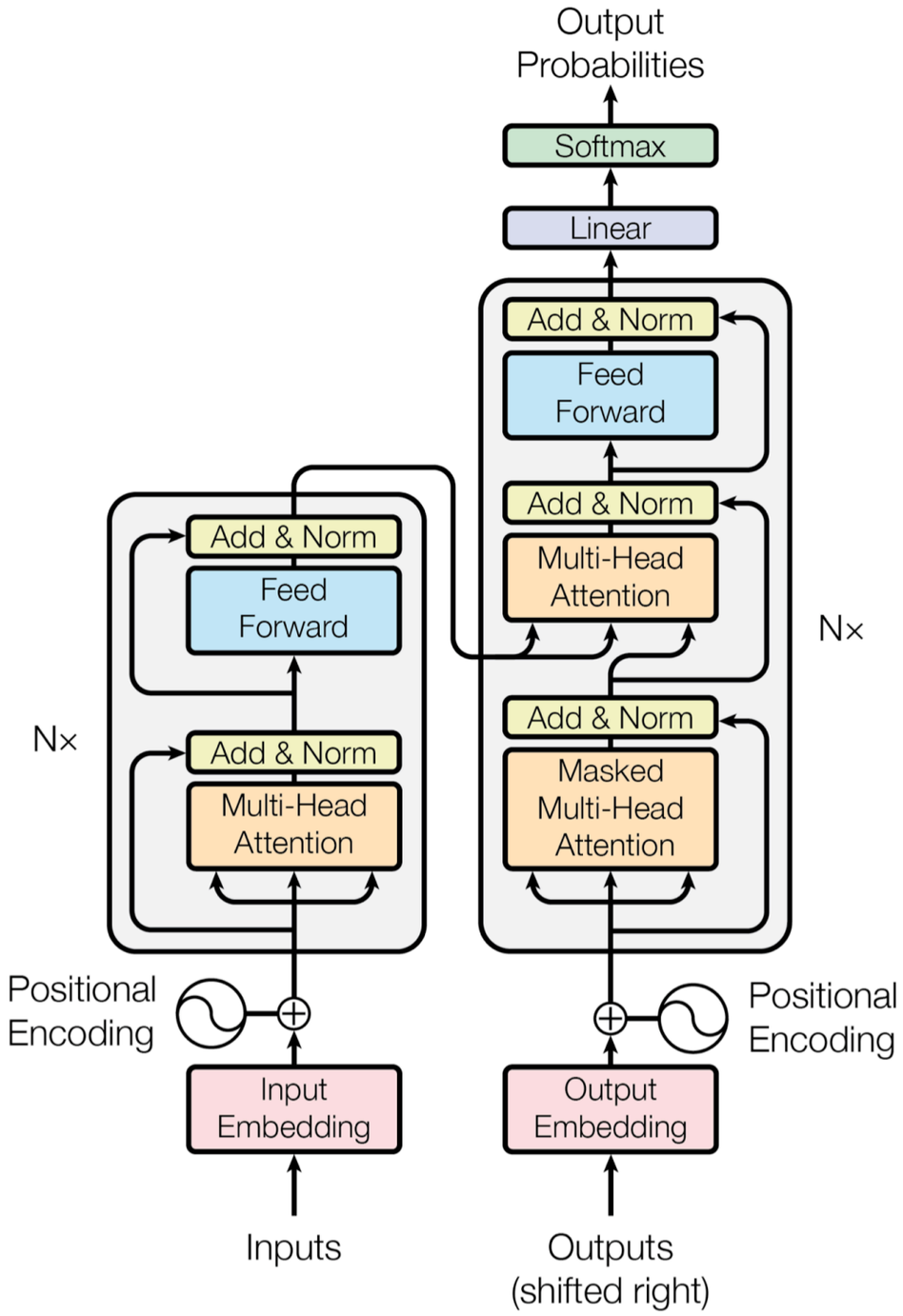
- Информацията се предава само в едната посока (или от ляво на дясно или от дясно на ляво)

- Информацията “избледнява” с дължината на последователността

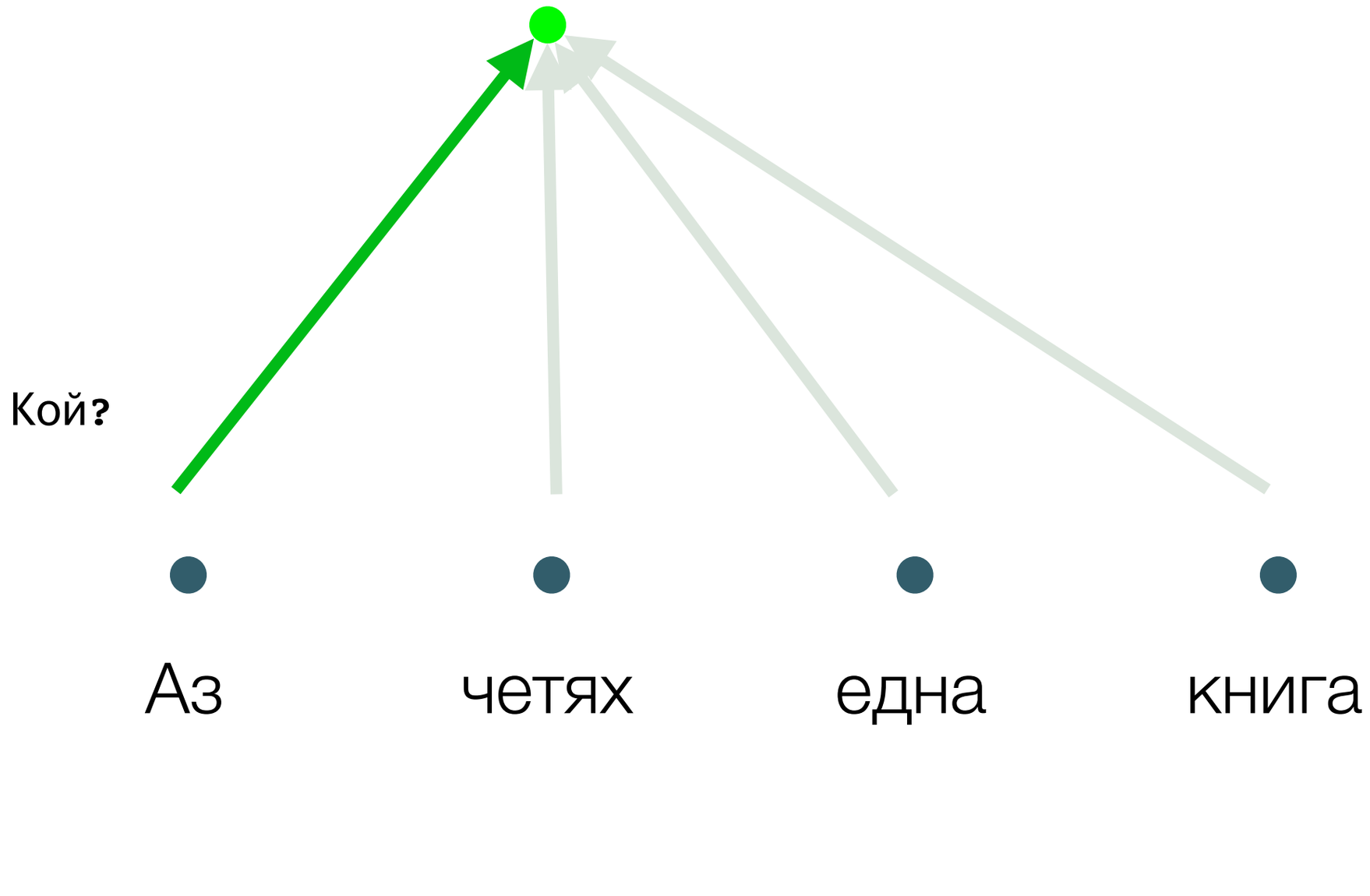
- **Не може ли да използваме “внимание” за да моделираме влиянието на контекста върху дадена позиция в последователността?**

Архитектурата TRANSFORMER: поглед отгоре

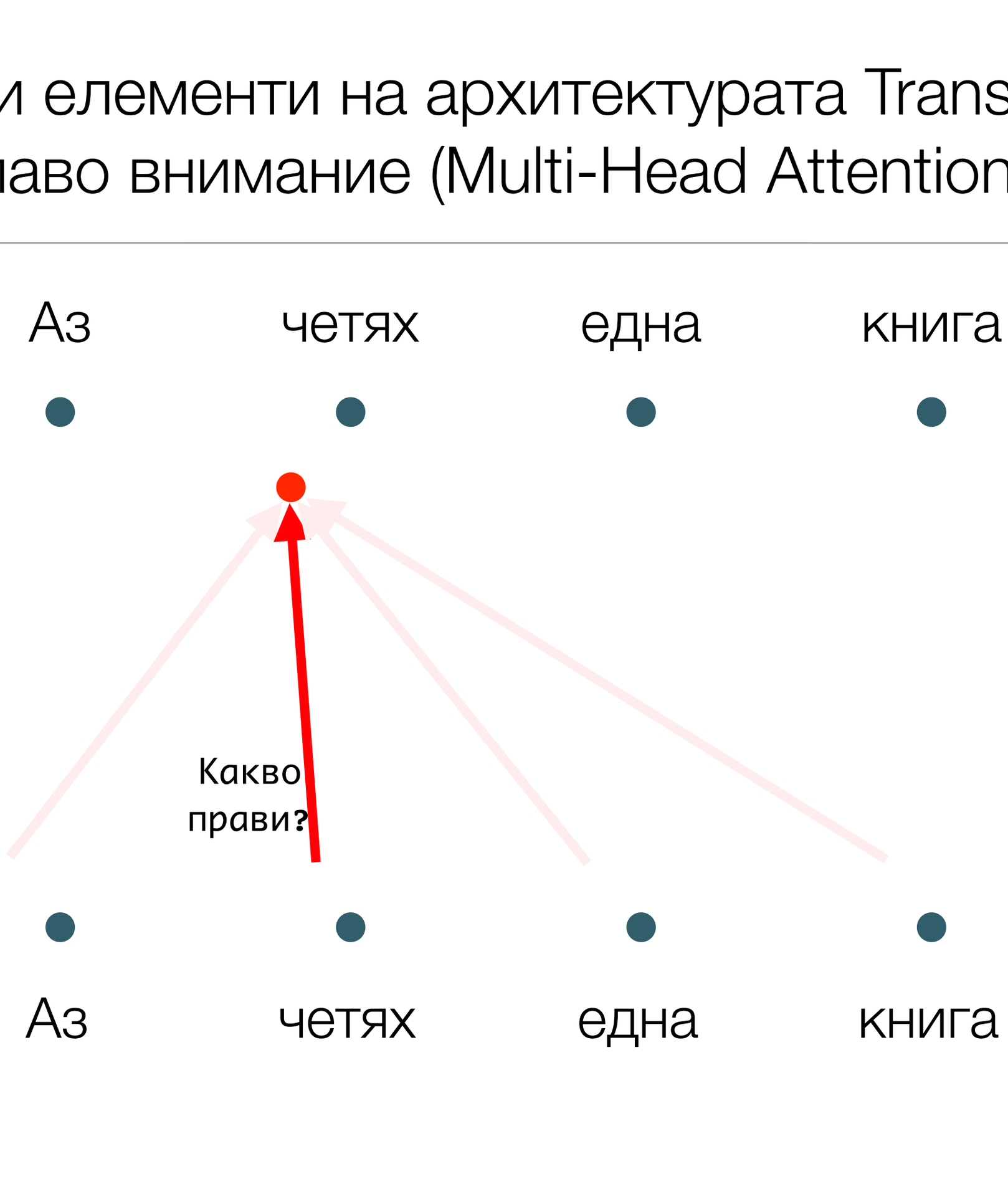
- Избягва използването на рекурентни и конволюционни блокове за анализ на контекста
- Изцяло се базира на екстензивно използване на внимание
- Използва се както внимание върху същата последователност (self-attention), така и внимание от изходната последователност към входната последователност (както преди)
- Използва още линейни блокове, преки връзки и нормализация на слоеве и dropout
- Vaswani et al., 2017,
  Attention Is All You Need
  https://arxiv.org/abs/1706.03762



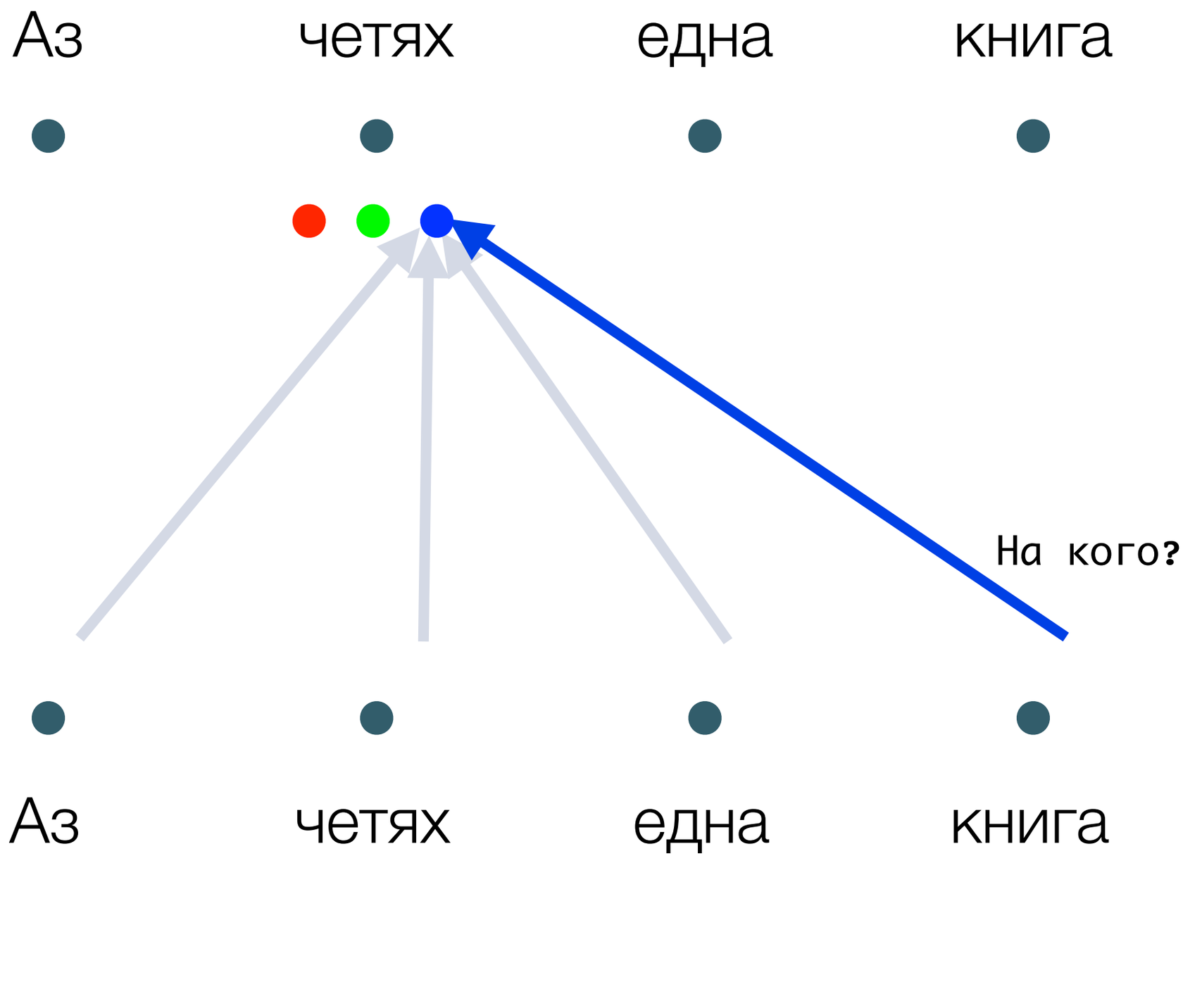
Основни елементи на архитектурата Transformer: многоглаво внимание (Multi-Head Attention)



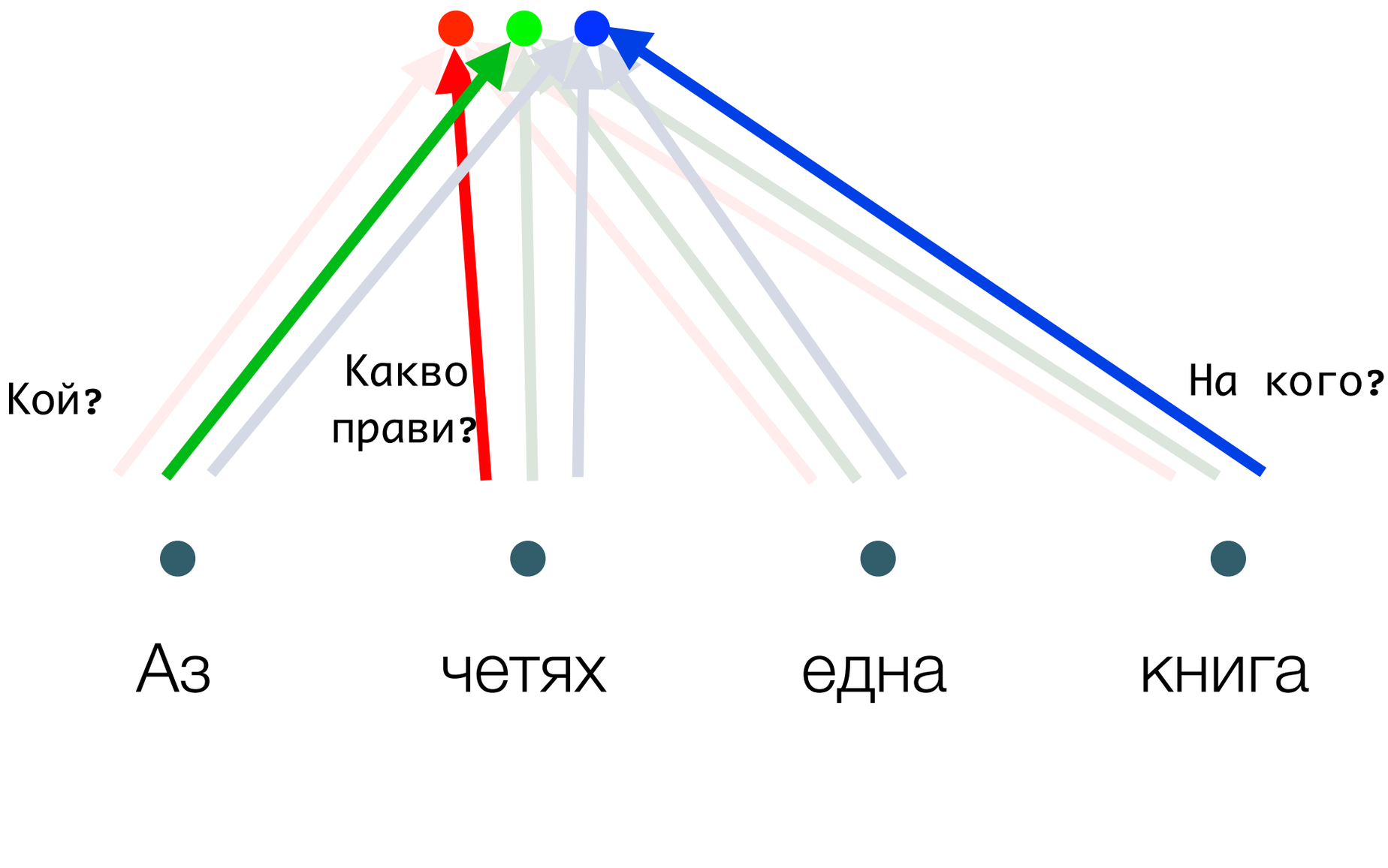
Основни елементи на архитектурата Transformer: многоглаво внимание (Multi-Head Attention)



Основни елементи на архитектурата Transformer: многоглаво внимание (Multi-Head Attention)

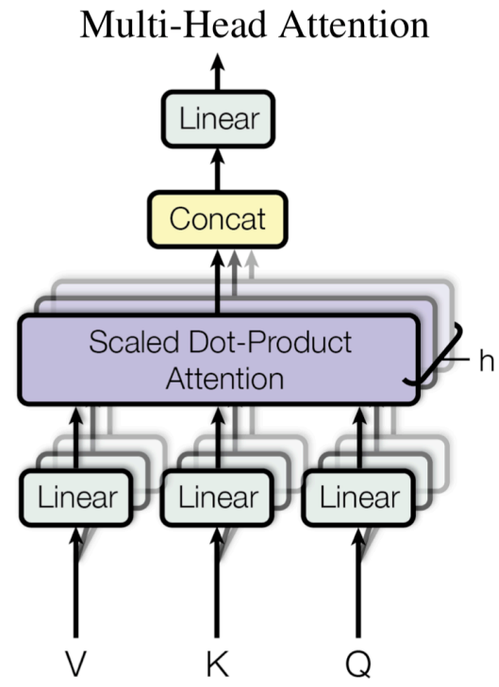


Основни елементи на архитектурата Transformer: многоглаво внимание (Multi-Head Attention)



Основни елементи на архитектурата Transformer: многоглаво внимание (Multi-Head Attention)

- Ще използваме внимание със скаларно произведение скалирано с $1/\sqrt{d_k}$, където d_k е размерността на векторите: $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$



- Вместо да използваме едно внимание, ще реализираме “многоглаво внимание”:

$$\text{MultiHead}(Q, K, V) = [\text{head}_1; \text{head}_2; \ldots; \text{head}_h]W^O$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

$$W_i^Q, W_i^K \in \mathbb{R}^{d_m \times d_k}; W_i^V \in \mathbb{R}^{d_m \times d_v}; W^O \in \mathbb{R}^{hd_v \times d_m}$$

- Q наричаме заявки, K наричаме ключове, а V стойности

Основни елементи на архитектурата Transformer: линеен блок (Feed-Forward Network FFN)

- $\mathbf{FFN}(x) = \mathbf{ReLU}(xW_1 + b_1)W_2 + b_2$
- $W_1 \in \mathbb{R}^{d_m \times d_{ff}}; b_1 \in \mathbb{R}^{d_{ff}}; W_2 \in \mathbb{R}^{d_{ff} \times d_m}; b_2 \in \mathbb{R}^{d_m}$
- обикновено $d_{ff} = 4d_m$

Основни елементи на архитектурата Transformer: трансформаторен блок (transformer block)

$$z_1 = \text{MultiHead}(x, x, x)$$

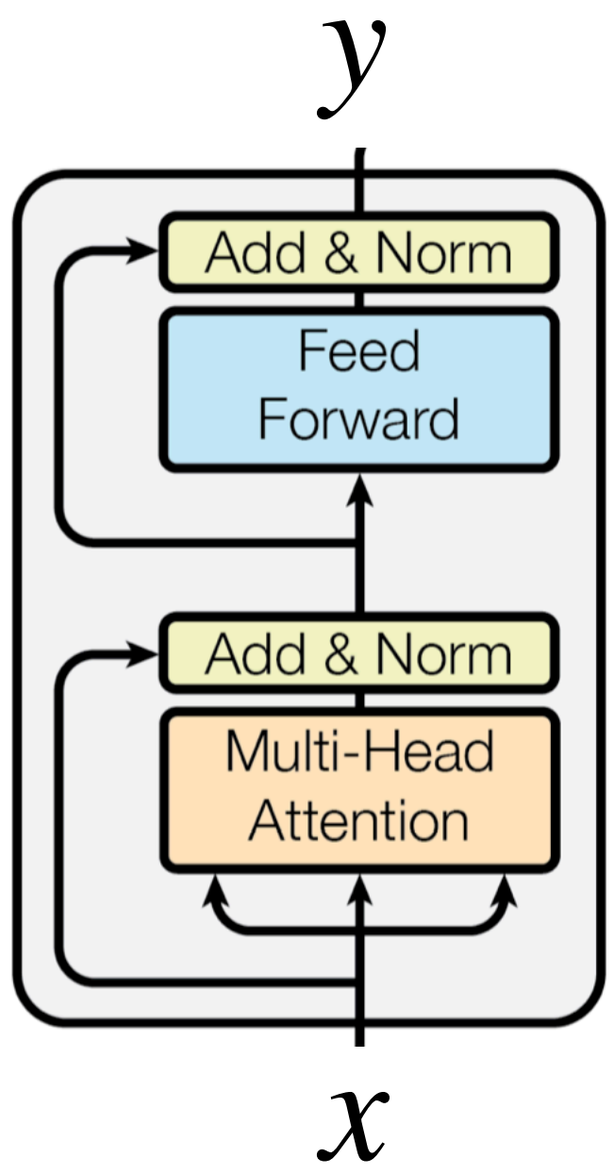
$$z_2 = \text{LayerNorm}(x + \text{dropout}(z_1))$$

$$z_3 = \text{FFN}(z_2)$$

$$y = \text{LayerNorm}(z_2 + \text{dropout}(z_3))$$

$$\text{LayerNorm}(x) = \frac{x - \text{E}[x]}{\text{Var}[x]} \odot \gamma + \beta$$

$$\gamma, \beta \in \mathbb{R}^{d_m}$$

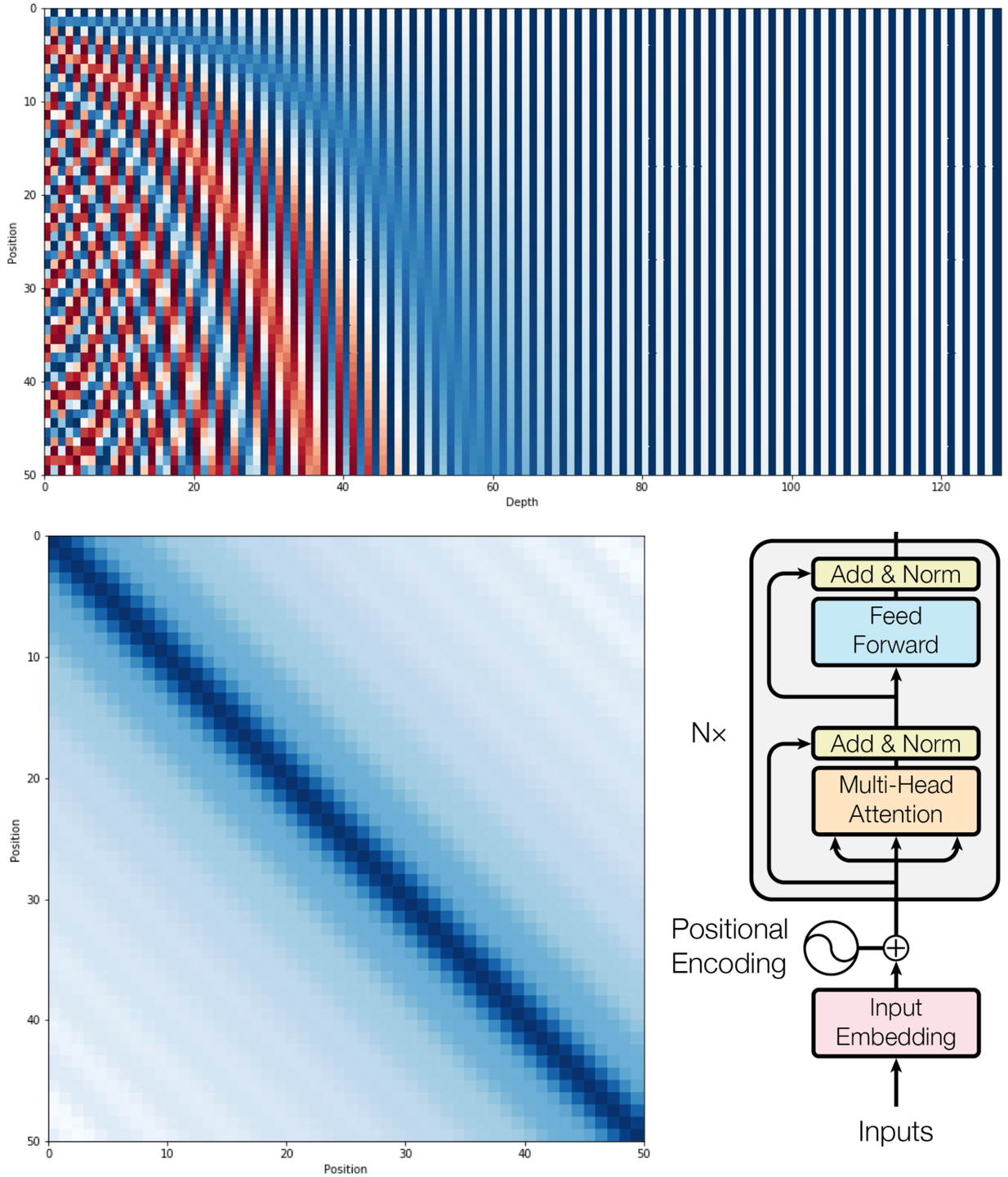


Основни елементи на архитектурата Transformer: позиционно влагане (positional embedding)

- За всяка входна позиция p генерираме вектор за позиционно влагане $\mathbf{PE}_p \in \mathbb{R}^{d_m}$

$$(PE_p)_{2i} = \sin\left(\frac{p}{10000^{2i/d_m}}\right)$$

$$(PE_p)_{2i+1} = \cos\left(\frac{p}{10000^{2i/d_m}}\right)$$

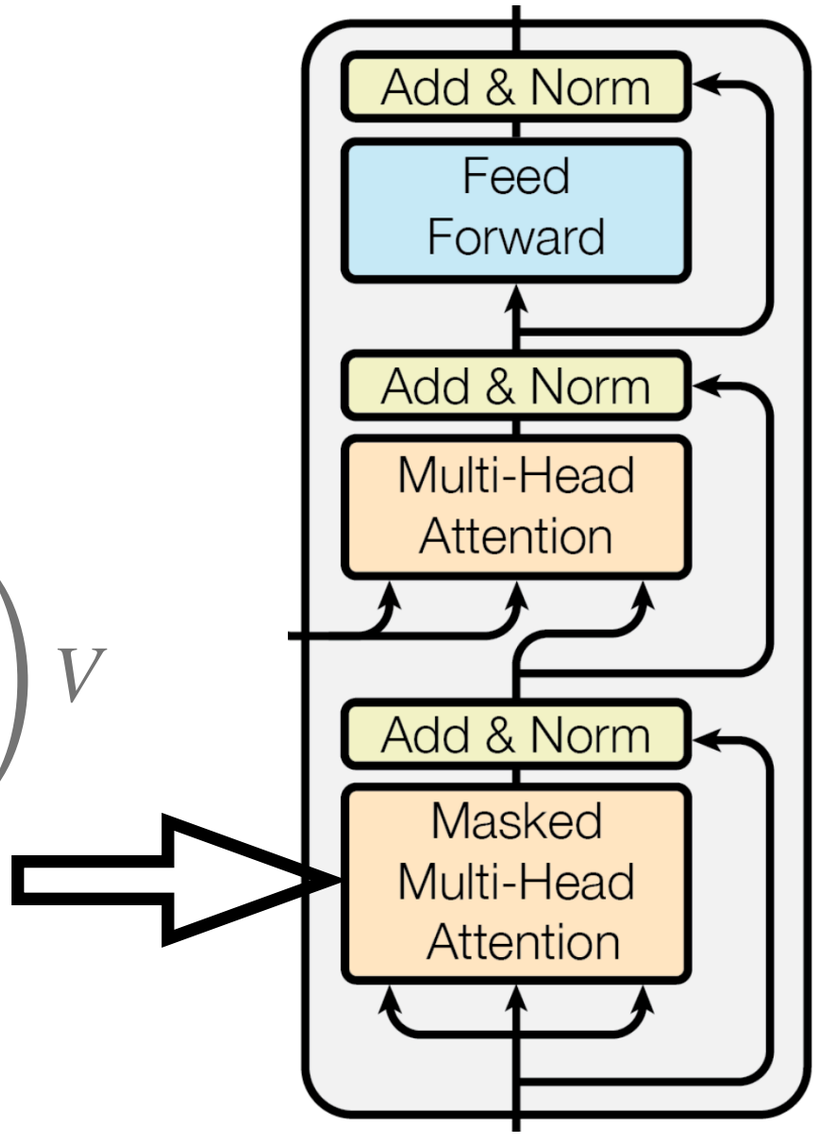


Основни елементи на архитектурата Transformer: многоглаво внимание с маскиране (Masked Multi-Head Attention)

- В декодера не може да се реализира внимание върху бъдещите елементи
- Стойностите в бъдещето се маскират с $-\infty$:

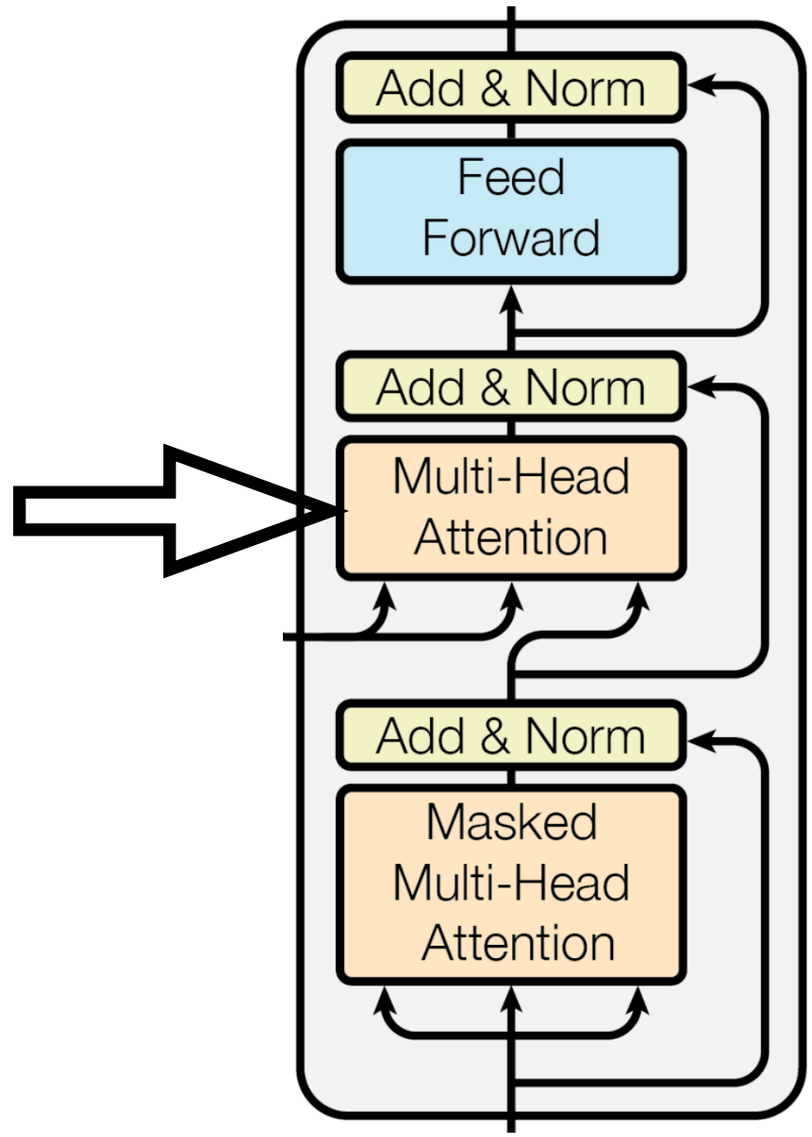
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{\text{mask}_{i>j}(QK^T)}{\sqrt{d_k}}\right)V$$

$$\Big(\text{mask}_{i>j}(A)\Big)_{i,j} = \begin{cases} A_{i,j} & \text{if } i \leq j \\ -\infty & \text{if } i > j \end{cases}$$



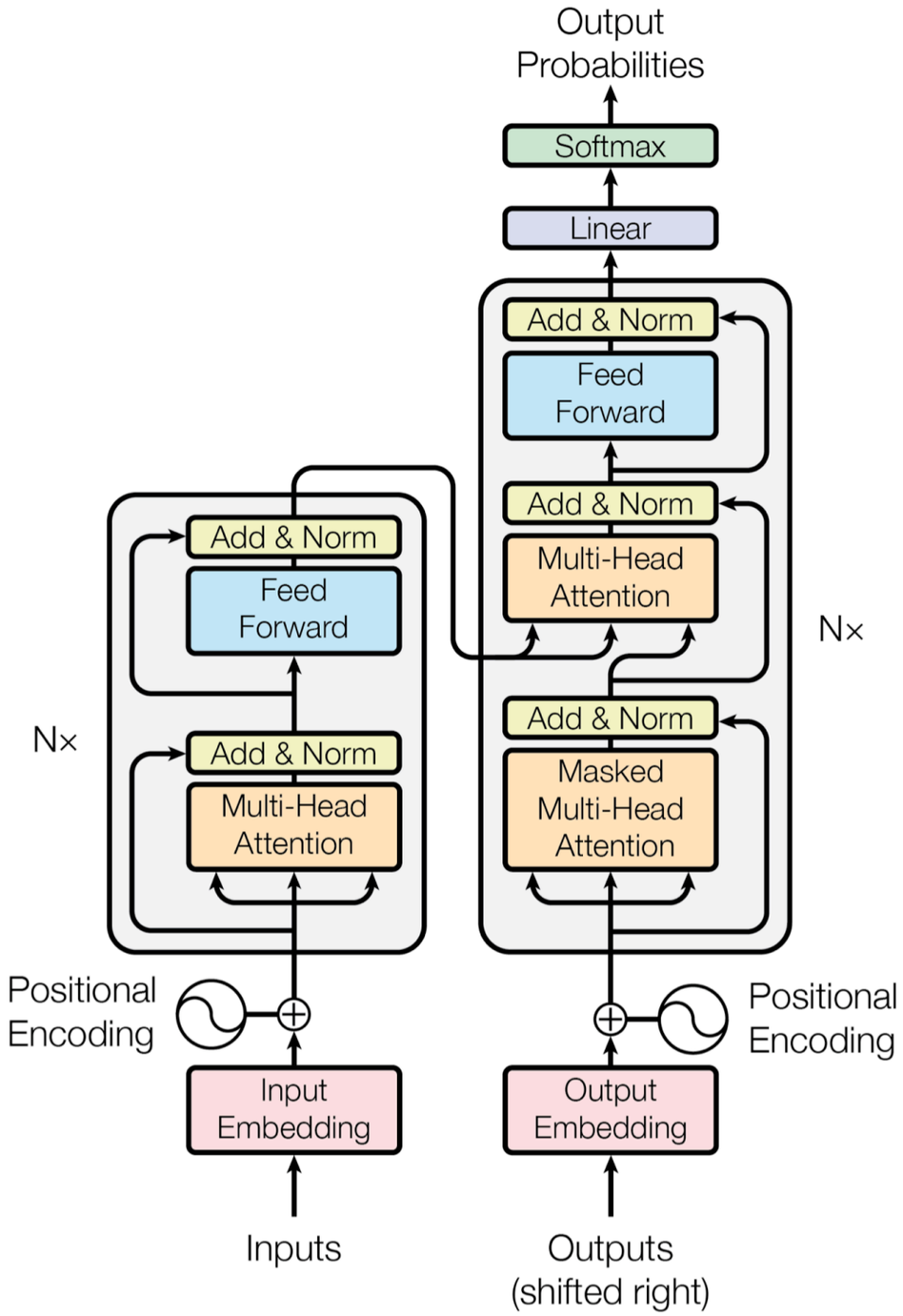
Основни елементи на архитектурата Transformer: многоглаво внимание в декодера върху енкодера

- Декодерът трябва да реализира внимание върху елементите на енкодера
- Ключовете K и стойностите V се получават от енкодера
- Заявките Q се получават от предходния слой на декодера



Архитектурата TRANSFORMER: цялата картина

- След позиционно влагане на входа енкодера прилага N трансформер блока на енкодера, с които се кодира входната последователност
- В режим на обучение декодера обработва изходната последователност с позиционно влагане и прилагане на N трансформер блока на декодера
- За всяка позиция се получава след проекция и softmax разпределение за следващата позиция, което се използва за обучението чрез минимизиране на емпиричната крос-ентропия
- В режим на генериране декодера от поредния изходен елемент получава разпределение за следващия, което се използва за получаване на следващия елемент
- За генерация може да се използва както метода на алчно търсене, така и метода за търсене по лъча



План на лекцията

1. Формалности за курса (5 мин)
2. Условни езикови модел (15 мин)
3. Методи за генерация на текст / декодиране (20 мин)
4. Приложения на генерация на текст с езиков модел (5 мин)
5. Архитектура енкодер-декодер за невронен машинен превод (20 мин)
6. Архитектура енкодер-декодер с “внимание” (Attention) (20 мин)
7. Transformer архитектура (30 мин)
8. **Оценяване на резултат от машинен превод (10 мин)**

Методи за оценяване на качеството на превода

1. Сравняване на крос-ентропията — лесно за изчисляване но не е пряко свързано с качеството на превода — трудно може да се интерпретира.

2. Оценяване от експерт — прецизно но много скъпо за реализиране.

3. Сравнение на общи k-грами с референтивен професионален превод — сравнително лесно за реализиране и дава до известна степен интерпретируема оценка за превода.

Papineni et al, ACL-2002: BLEU: a Method for Automatic Evaluation of Machine Translation,
https://www.aclweb.org/anthology/P02-1040.pdf

BLEU (Bilingual Evaluation Understudy)

BLEU е претеглено геометрично средно на прецизността на k-грамите с фактор за наказване на твърде къси преводи:

$$\text{Bleu}_4 = \exp(0.5 \log P_1 + 0.25 \log P_2 + 0.125 \log P_3 + 0.125 \log P_4 - \max(\frac{|\bar{\mathbf{y}}|}{|\mathbf{y}|} - 1,0)),$$

- P_1 е прецизността на 1-грамите — процентът на 1-грамите в оценявания превод, които се срещат в референтния превод.
- P_2 е прецизността на 2-грамите
- P_3 е прецизността на 3-грамите
- P_4 е прецизността на 4-грамите
- $|\bar{\mathbf{y}}|$ е дължината на референтния превод
- $|\mathbf{y}|$ е дължината на оценявания превод

Обобщение

- Архитектурите “последователност към последователност” с внимание дават чудесни резултати върху задачи като машинен превод, резюмиране на документ, разпознаване на реч и много други.

- Архитектурите базирани на Transformer в повечето случай дават по-добри резултати в сравнение с рекурентни невронни мрежи.

- Влагането на думите на входния език може да се реализира с конволюция на символи (виж лекция 13).

- За генерирането на непознати думи при извеждането на изходна дума за <UNK> символа може да се използва модел за генератор на символи. Виж: Luong and Manning, 2016: Achieving Open Vocabulary Neural Machine Translation with Hybrid Word-Character Models, https://arxiv.org/abs/1604.00788

- Кой вариант дава най-добри резултати зависи от конкретната задача, език, корпус и т.н.