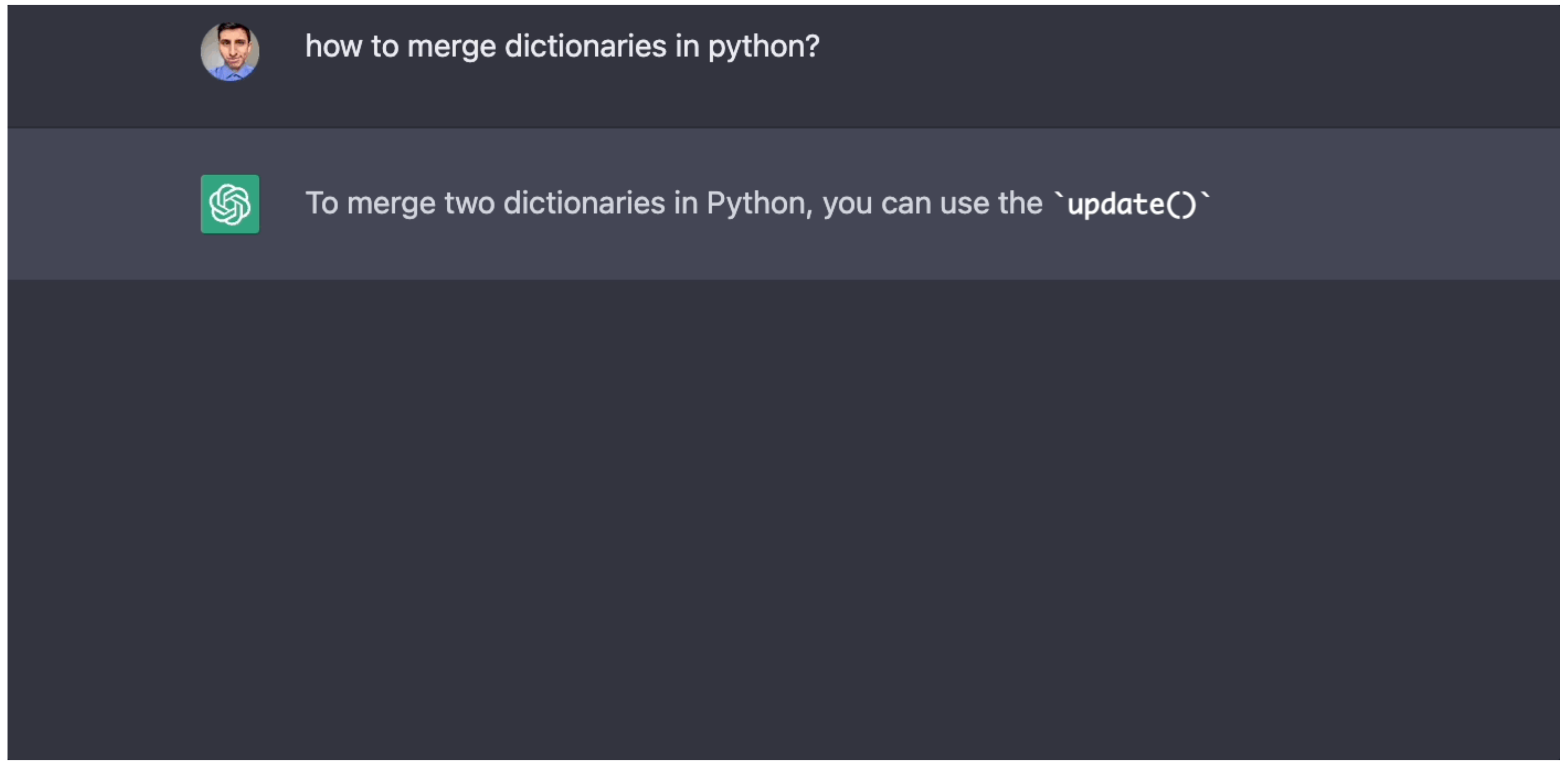
Deep learning Research (CSE 61301)
Special Topic: DL Research (AI72301)

UNIST AIGS, CSE
Jooyeon Kim, Ph.D.

Transformers

- Given a sequence of words, predict the next word
  - Given a sequence from n-1 to n, predict what word will appear next
  - Predict n+1 th word given 0~nth words
    - Based on the n th word !
    - Do this recursively!



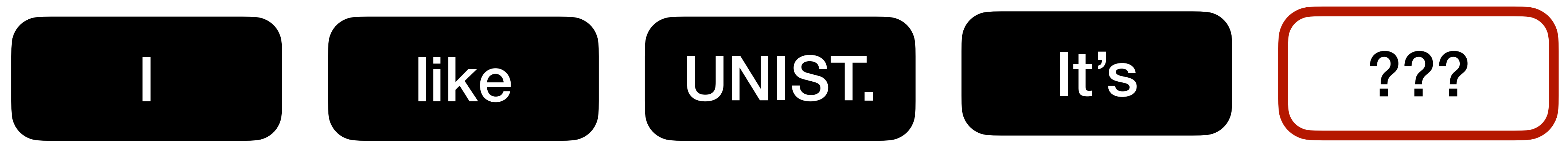
how to merge dictionaries in python?

To merge two dictionaries in Python, you can use the `update()`

https://medium.com/@tanyamarleytsui/coding-with-chatgpt-b50ab3fcb45f

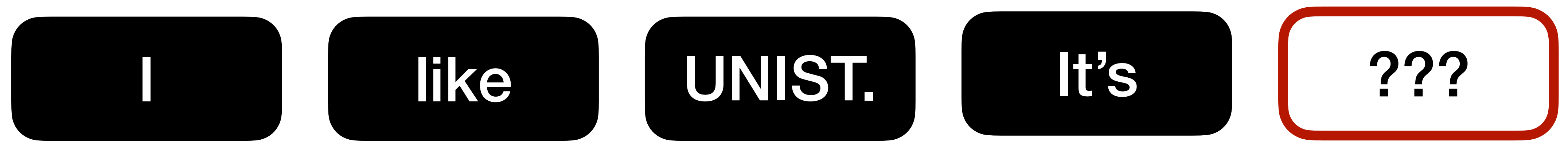
Transformers

- Given a sequence of words, predict the next word
  - Given a sequence from n-1 to n, predict what word will appear next
  - Predict n+1 th word given 0~nth words
    - Based on the n th word !
    - Do this recursively!
    - "How" can we do this?
      - Consider not only the words that have appeared right before, but (as much as) **all** the previous words
        - SOTA LLM models like ChatGPT, GPT4 consider up to thousands, or up to MILLIONS of previous words!



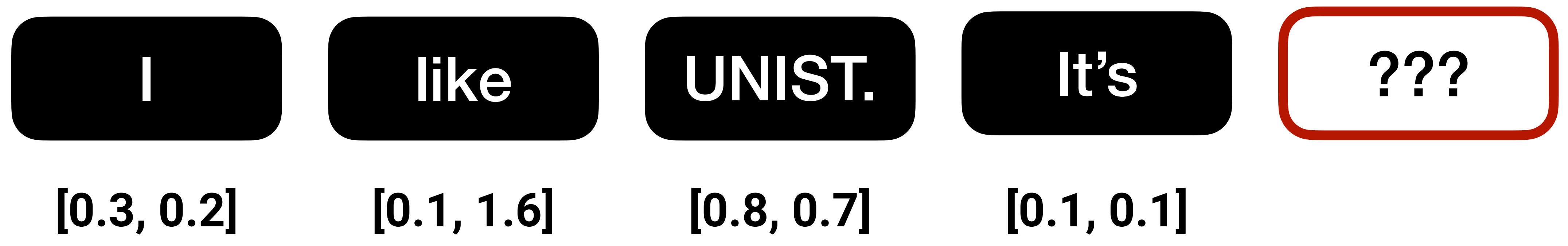
Transformers

- Given a sequence of words, predict the next word
  - Given a sequence from n-1 to n, predict what word will appear next
  - Predict n+1 th word given 0~nth words
    - Based on the n th word !
    - Do this recursively!
    - "How" can we do this?
      - Consider not only the words that have appeared right before, but (as much as) **all** the previous words
        - SOTA LLM models like ChatGPT, GPT4 consider up to thousands, or up to MILLIONS of previous words!
      - Also, we need to consider the orders of the previous words



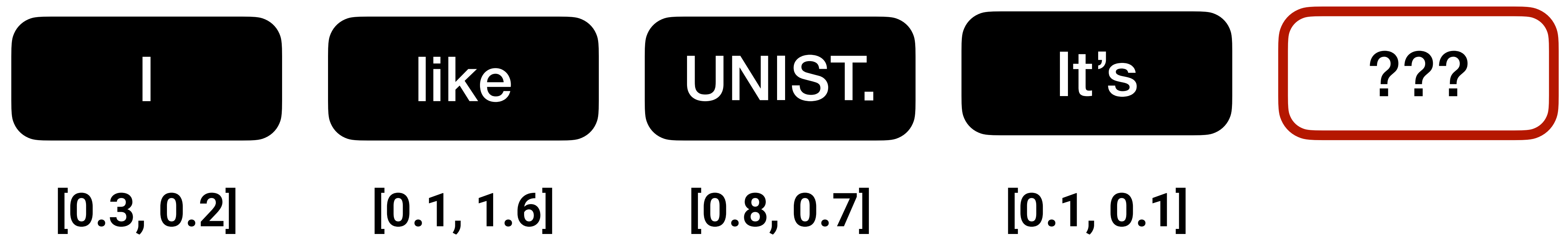
A tool that we can use: word embeddings

- Word embeddings (WE)
  - A fixed-sized vector that *represents* each word
  - Here, for simplicity, we assume that it's two-dimensional
  - WE (somehow) contains relevant info about the word
    - Semantic, syntactic information
  - Vec(King) - Vec(Man) + Vec(Woman) ≈ Vec(Queen)
  - However, it doesn't consider context within which the word has been used
    - Vec(배:boat) == Vec(배:belly)
  - Recent LLMs like ChatGPT learns WEs internally (torch.nn.Embedding)
  - However, we assume that WEs are given



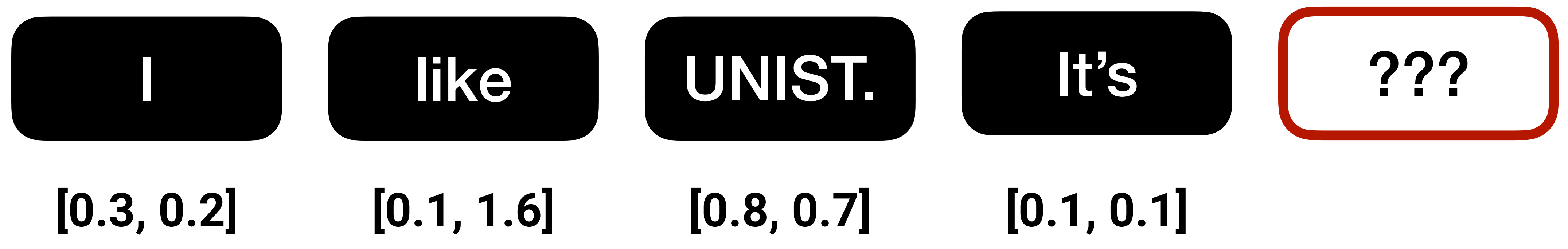
Method 1: Concatenation

- Concatenation
  - To predict the following word, gather (concatenate) all the previous word embeddings
  - [0.3, 0.2, 0.1, 1.6, 0.8, 0.7, 0.1, 0.1] -> MLP -> Softmax -> predict the next word
  - Problem 1: The curse of dimensionality
    - Typically, dimensionality of each WE vector goes up to thousands
    - Thousands, tens of thousands previous words (contexts)
    - -> Million (1e6) or even billion (1e8) dimensions -> infeasible
  - Problem 2: Input dim varies
    - Predicting the word “Like”: 2-dimensional
    - Predicting the word “UNIST”: 4-dimensional…



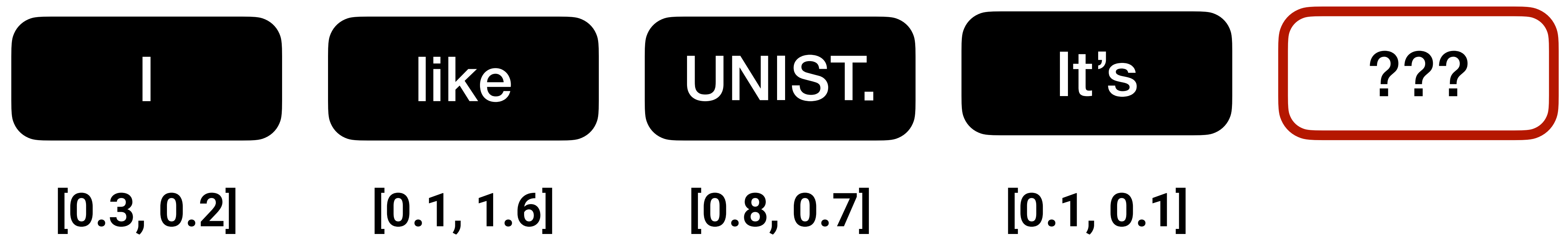
Method 2: Summation, averaging

- Summation, Averaging
  - To predict the following word, the summation or the average of all the previous word embeddings
  - sum([0.3, 0.2], [0.1, 1.6], [0.8, 0.7], [0.1, 0.1])
  - mean([0.3, 0.2], [0.1, 1.6], [0.8, 0.7], [0.1, 0.1])
  - -> MLP -> Softmax() -> predict the next word
  - Problem: We cannot keep the word ordering info!
    - We inevitably have to go through the loss of (certain) information as we do averaging or summation
      - I like you == You like me



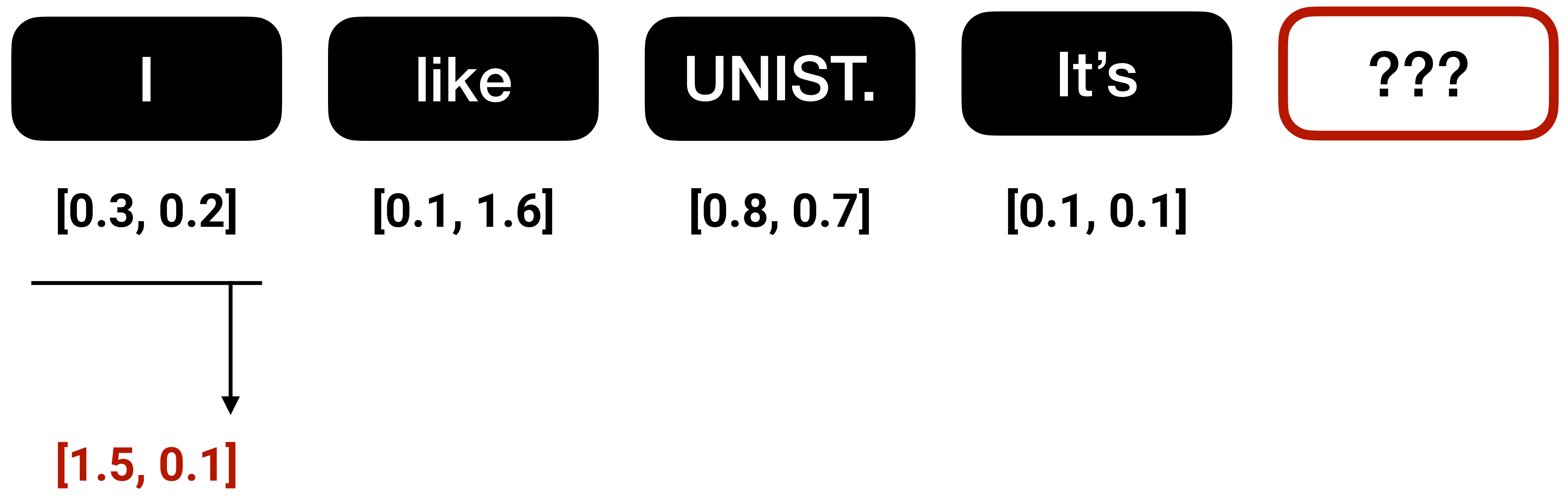
Method 3: Recurrent neural networks

- LSTM, GRU RNNs
  - Similar to transformers, fixed-sized hidden units encompass info about previous word embeddings
  - At least theoretically, unlike transformers, the hidden units can contain information about infinitely distant (past) word embeddings
  - However, in practice, as the data and model size increase, yields inferior performance to transformers that handles a set number of past contexts
  - Codes are much more complicated (LSTMs)



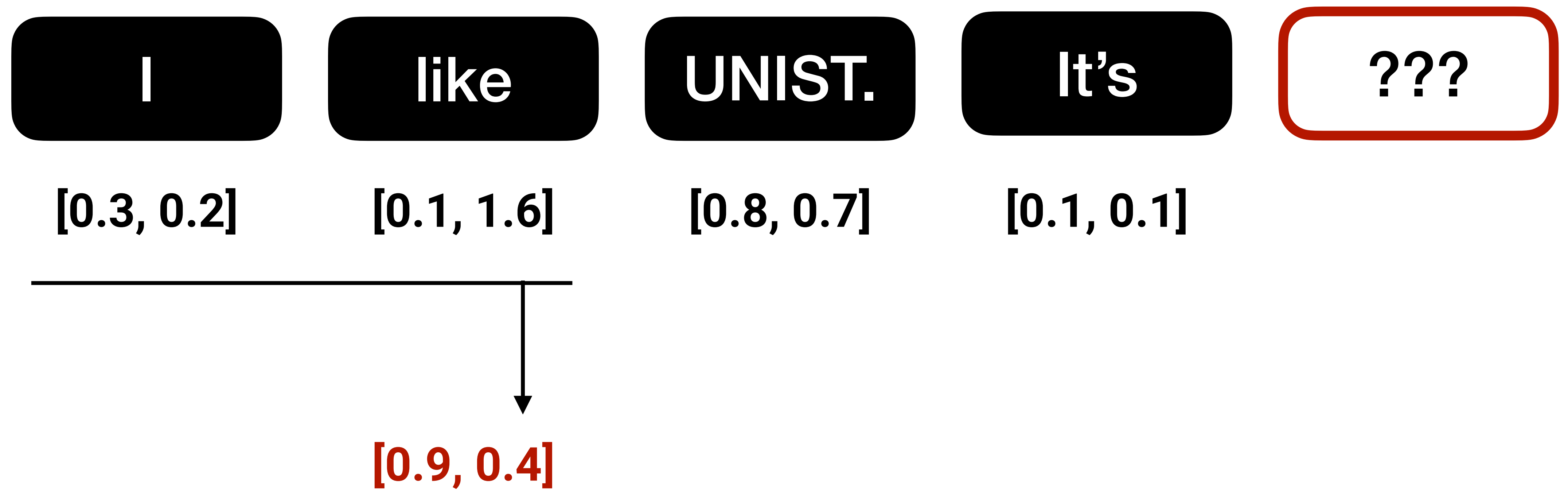
Context Vectors

- Context Vectors
  - Made by a certain combination of current and the previous WEs
  - A vector that as the same dimensionality as WEs



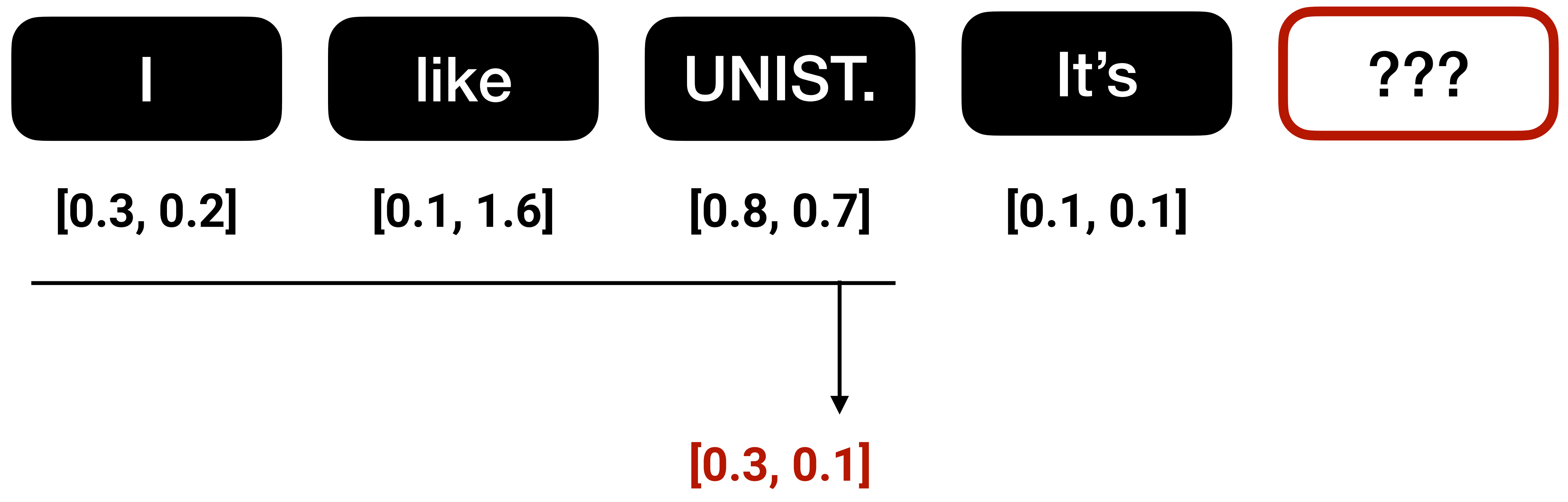
Context Vectors

- Context Vectors
  - Made by a certain combination of current and the previous WEs
  - A vector that as the same dimensionality as WEs



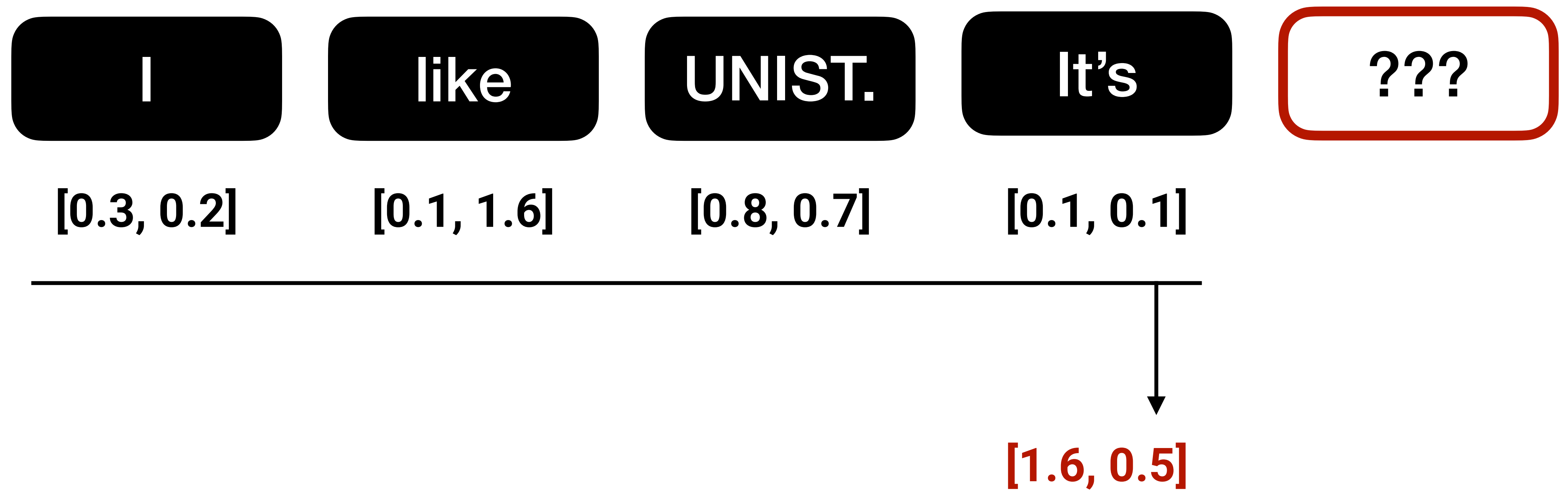
Context Vectors

- Context Vectors
  - Made by a certain combination of current and the previous WEs
  - A vector that as the same dimensionality as WEs



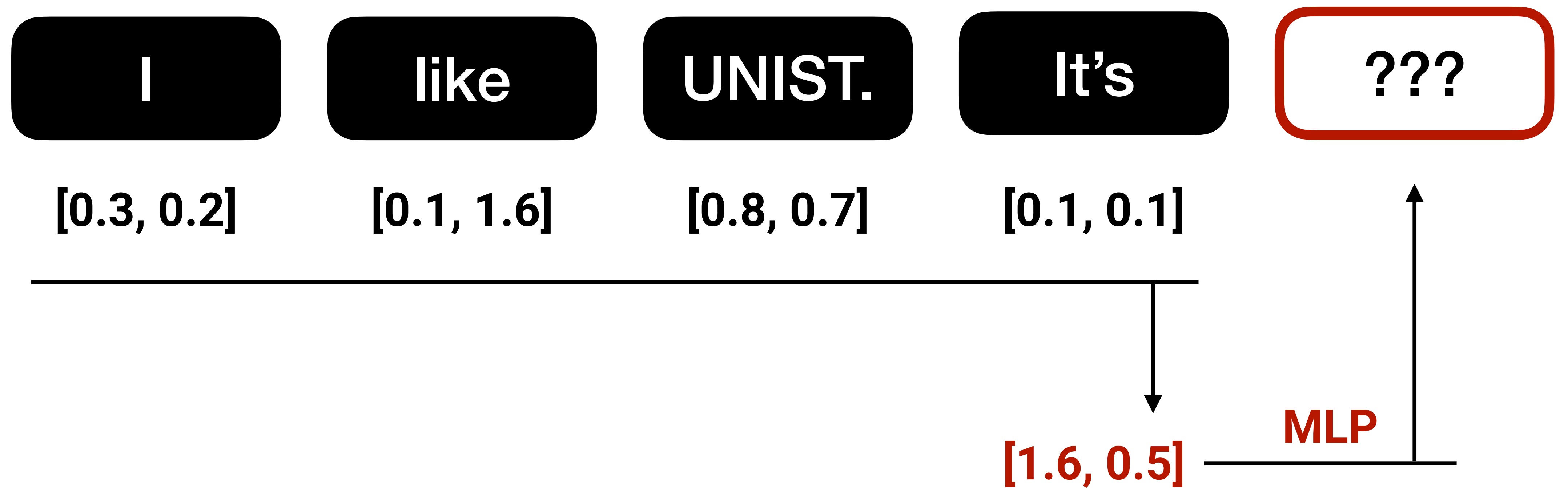
Context Vectors

- Context Vectors
  - Made by a certain combination of current and the previous WEs
  - A vector that as the same dimensionality as WEs



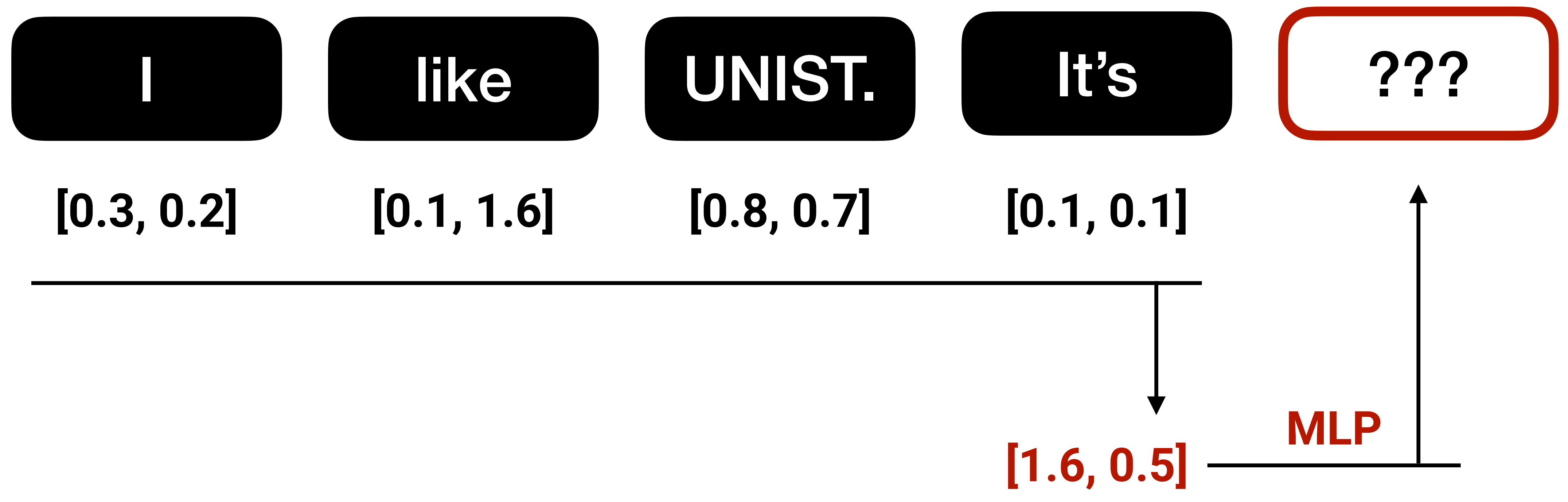
Context Vectors

- Context Vectors (CVs)
  - Made by a certain combination of current and the previous WEs
  - A vector that as the same dimensionality as WEs
  - Transformers use at most N past and current WEs to compute CV
    - GPT 2 -> N = 1024, Recent models up to a MILLION



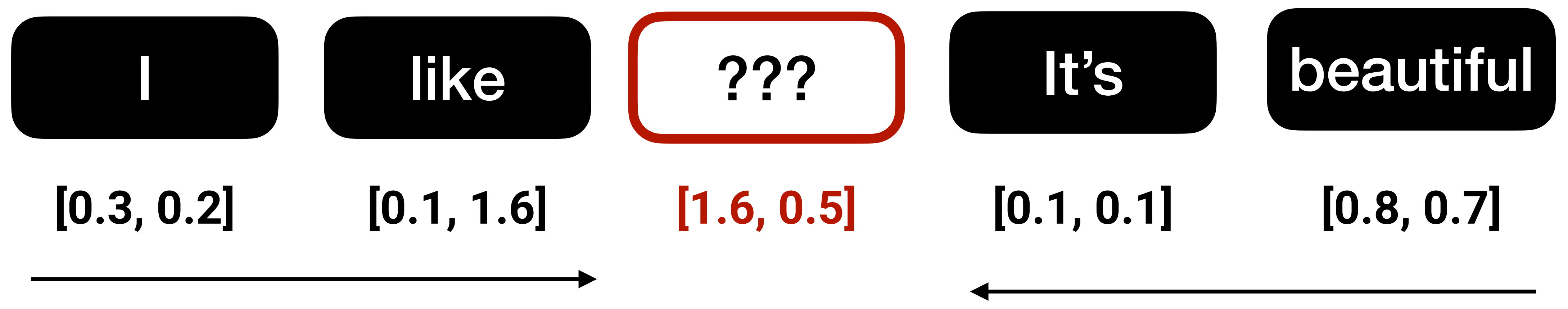
Context Vectors

- Context Vectors (CVs)
  - Made by a certain combination of current and the previous WEs
  - A vector that as the same dimensionality as WEs
  - Transformers use at most N past and current WEs to compute CV
    - GPT 2 -> N = 1024, Recent models up to a MILLION
  - Compute CV -> MLP -> classification -> Predict next word
    - Number of classes: ALL UNIQUE vocabularies in the dataset
    - Vocabulary (vocab.) size
    - Usually 50,000 ~ 110,000 (uint16 or uint32)



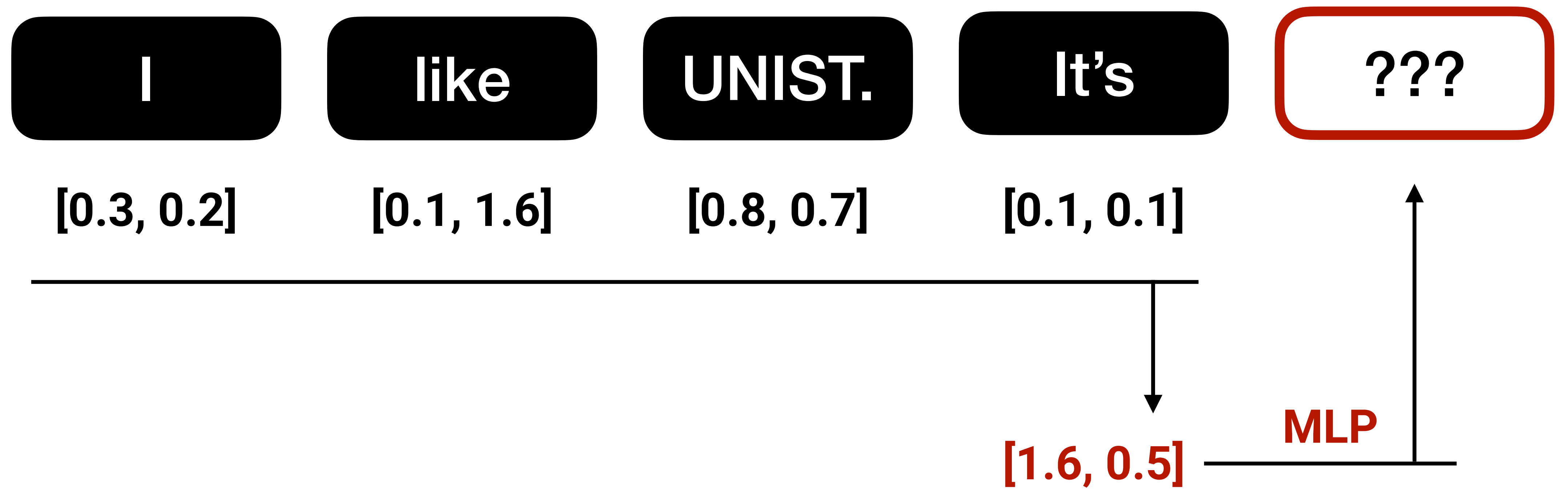
Context Vectors

- Context Vectors (CVs)
  - Some models compute CV bidirectionally
  - BERT (Bidirectional Encoder Representations from Transformers)



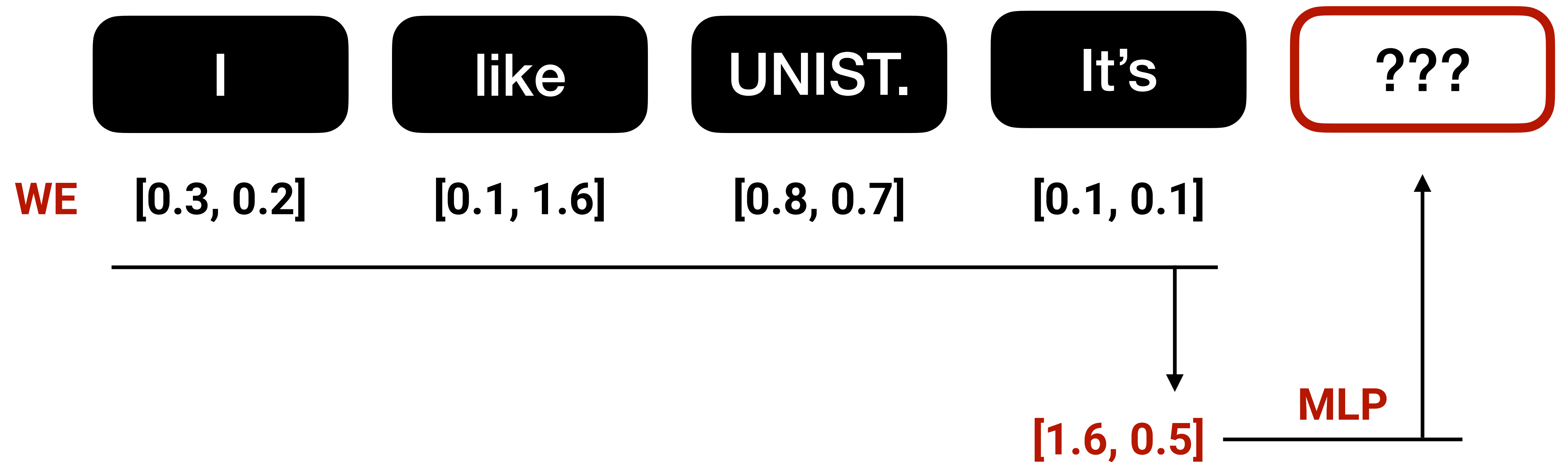
Context Vectors

- Context Vectors (CVs)
  - Made by a certain combination of current and the previous WEs
  - A vector that as the same dimensionality as WEs
  - Transformers use at most N past and current WEs to compute CV
    - GPT 2 -> N = 1024, Recent models up to a MILLION
  - Compute CV -> MLP -> classification -> Predict next word
    - Number of classes: ALL UNIQUE vocabularies in the dataset
    - Vocabulary (vocab.) size
    - Usually 50,000 ~ 110,000 (uint16 or uint32)



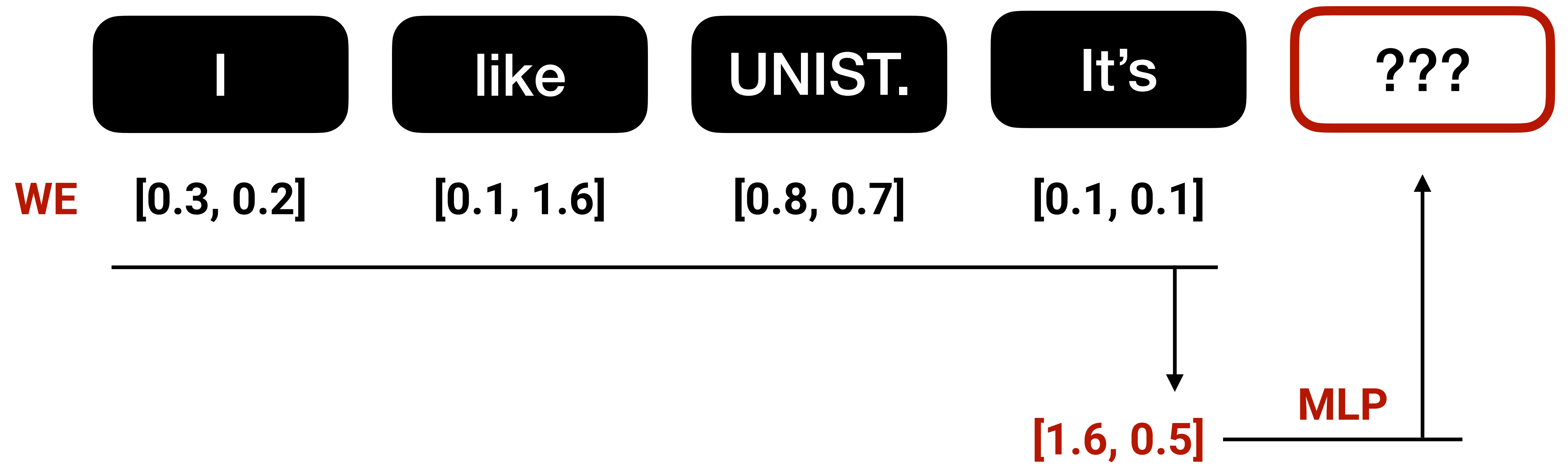
Context Vectors

- Context Vectors (CVs)
    - Made by a certain combination of current and the previous WEs



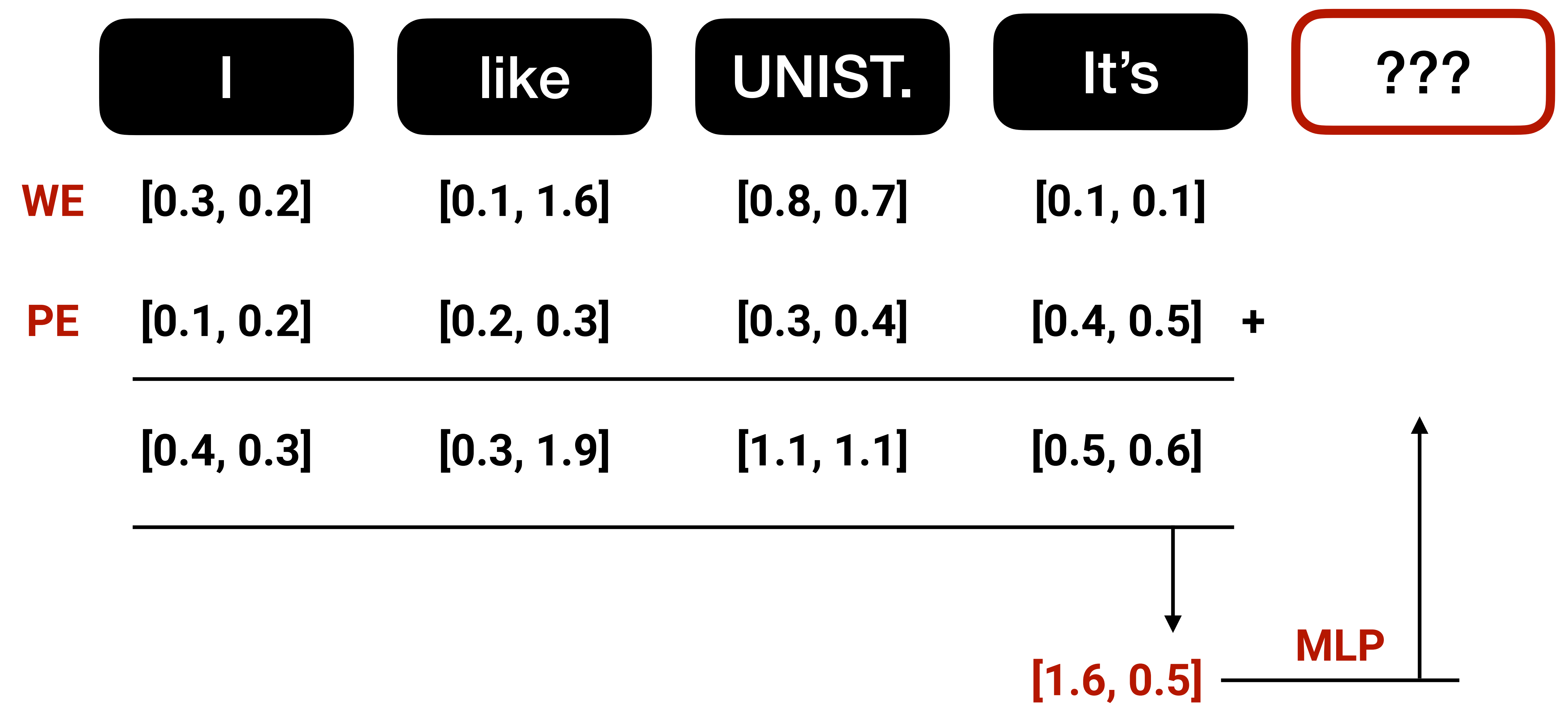
Positional Embeddings

- Context Vectors (CVs)
  - Made by a certain combination of current and the previous WEs
  - Develops from the “averaging” method
    - -> Still, “Positional infos” are lost!!
    - “I love you” == “You love me”



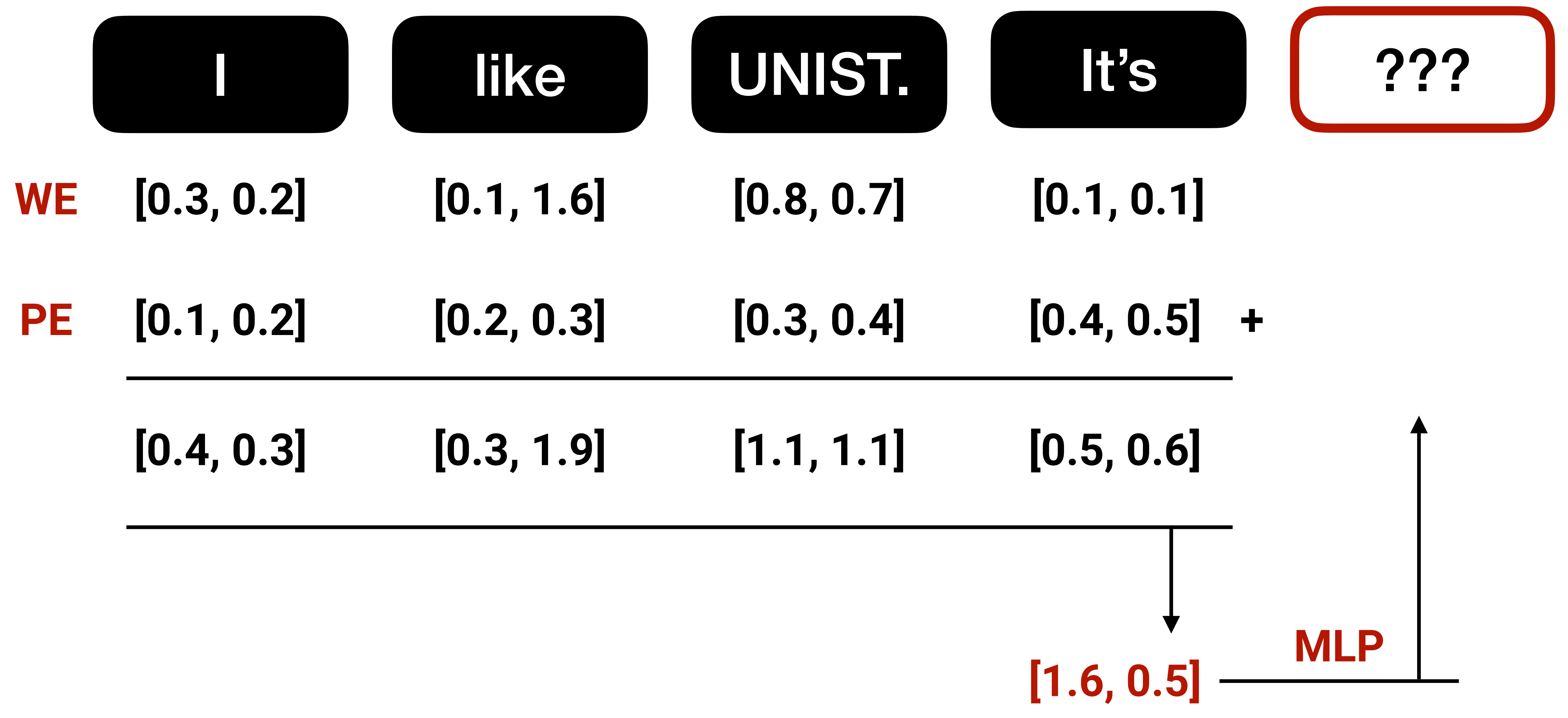
Positional Embeddings

- Context Vectors (CVs)
  - Made by a certain combination of current and the previous WEs
  - Develops from the “averaging” method
    - -> Still, “Positional infos” are lost!!
    - “I love you” == “You love me”
- Positional Embeddings (PEs)
  - For every word positions, we prepare unique embeddings



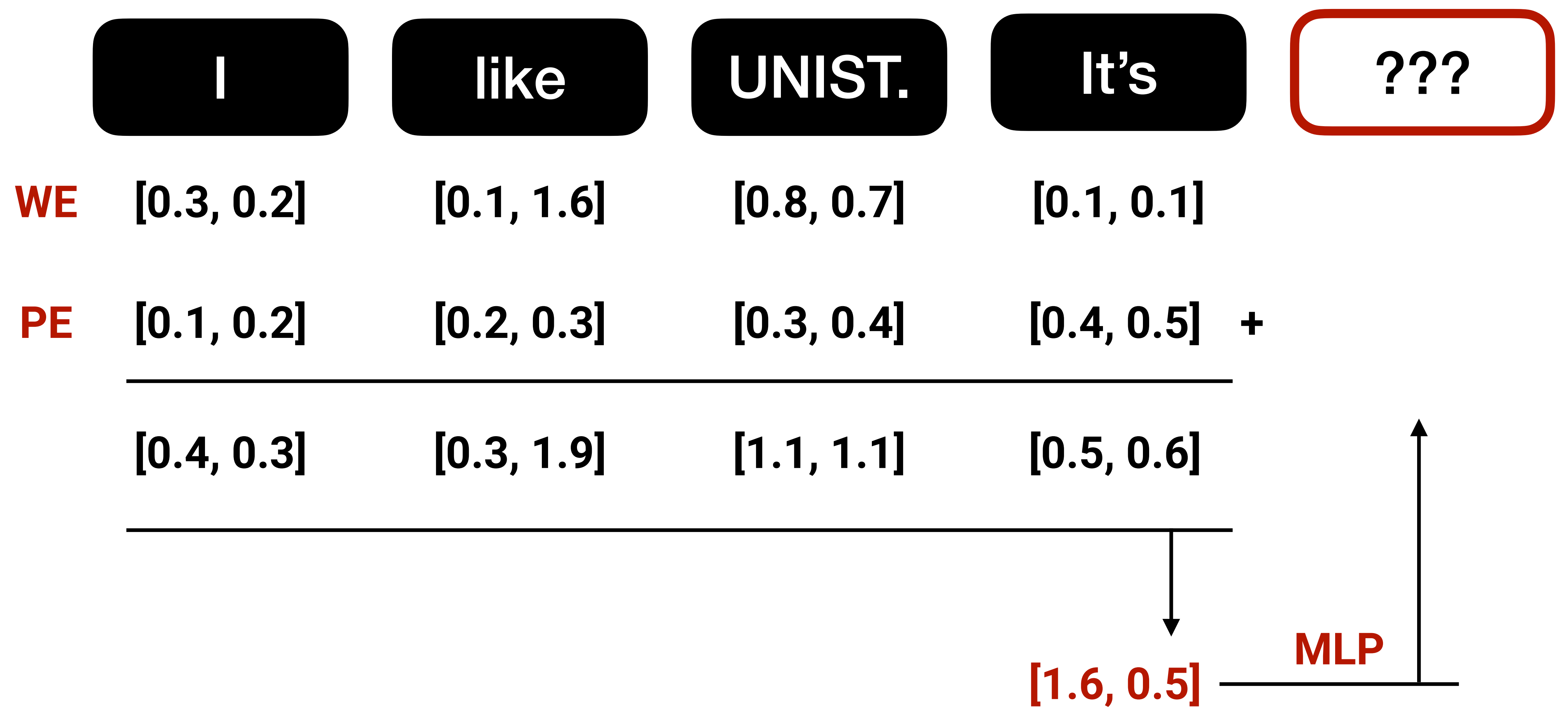
Positional Embeddings

- Context Vectors (CVs)
  - Made by a certain combination of current and the previous WEs
  - Develops from the "averaging" method
    - -> Still, "Positional infos" are lost!!
    - "I love you" == "You love me"
- Positional Embeddings (PEs)
  - For every word positions, we prepare unique embeddings
  - For each word and position, we add WE and PE and use those to ingredients to compute the CV



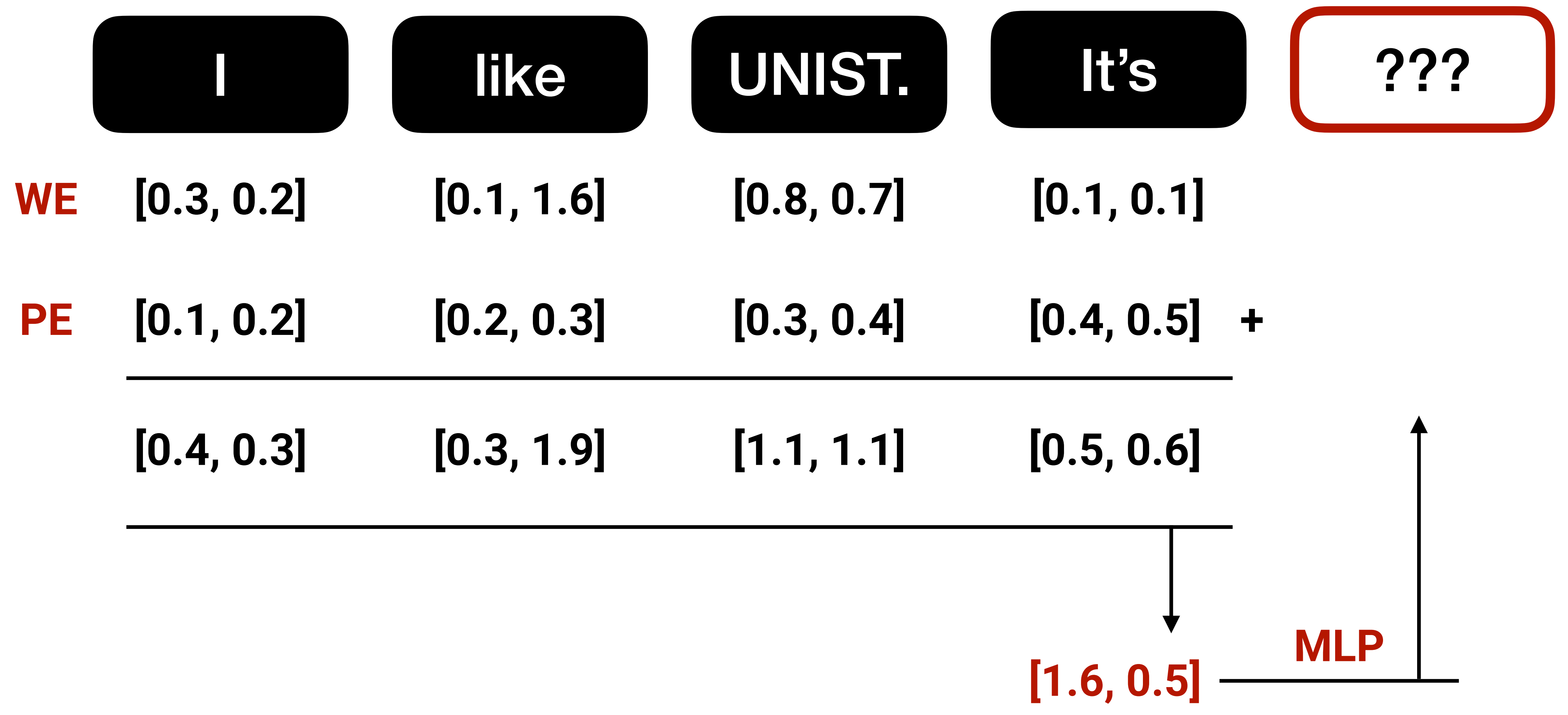
Positional Embeddings

- Context Vectors (CVs)
  - Made by a certain combination of current and the previous WEs
  - Develops from the “averaging” method
    - -> Still, “Positional infos” are lost!!
    - “I love you” == “You love me”
- Positional Embeddings (PEs)
  - For every word positions, we prepare unique embeddings
  - For each word and position, we add WE and PE and use those to ingredients to compute the CV
    - In fact, even for averaging, the loss of positional info can simply be resolved using PEs :)



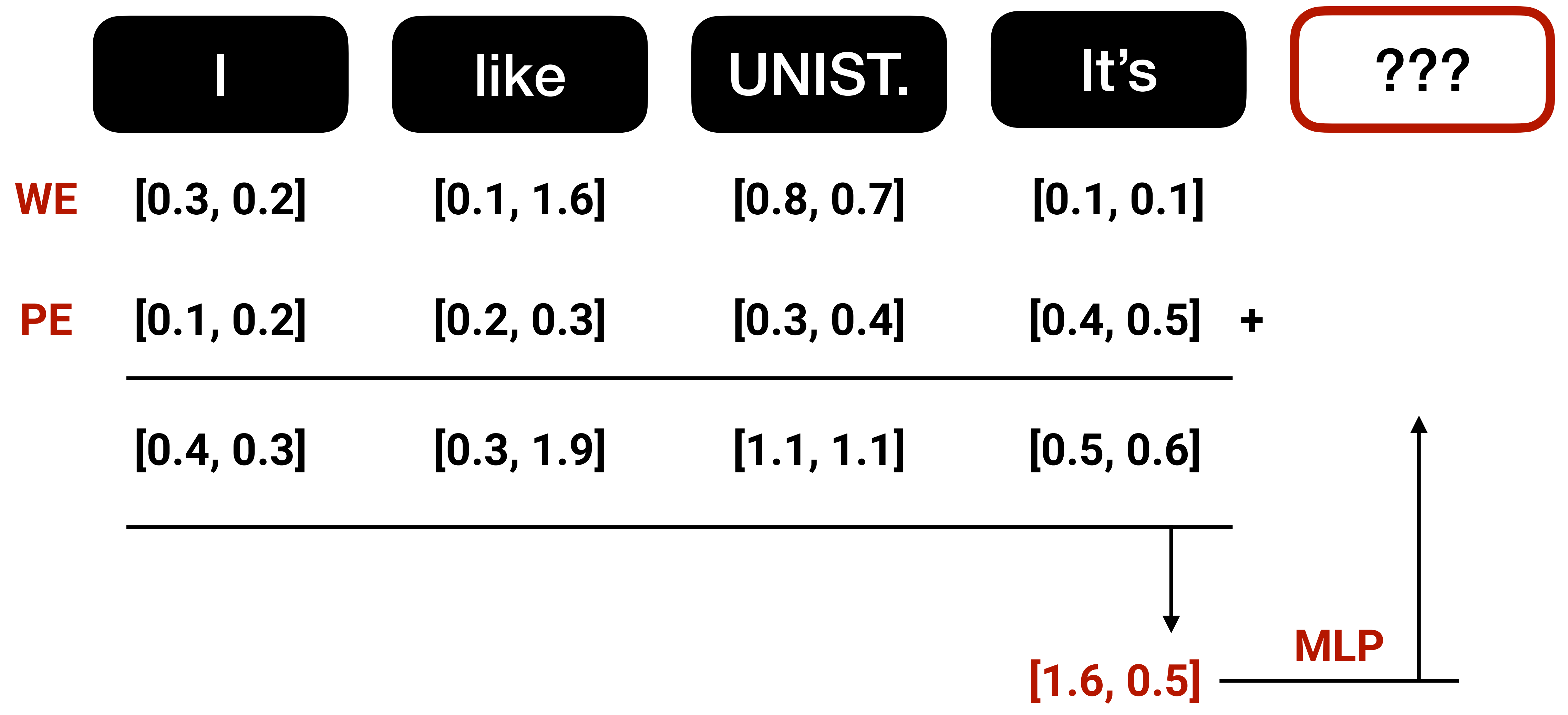
Positional Embeddings

- Positional Embeddings (PEs)
  - For every word positions, we prepare unique embeddings
  - For each word and position, we add WE and PE and use those to ingredients to compute the CV
    - In fact, even for averaging, the loss of positional info can simply be resolved using PEs :)
  - Here, just like WEs, we assume that PEs are given
    - In reality, we can use the likes of sin, cosine functions to explicitly designate PEs, or,
    - Let the model (transformers) compute, just likes they do for WEs



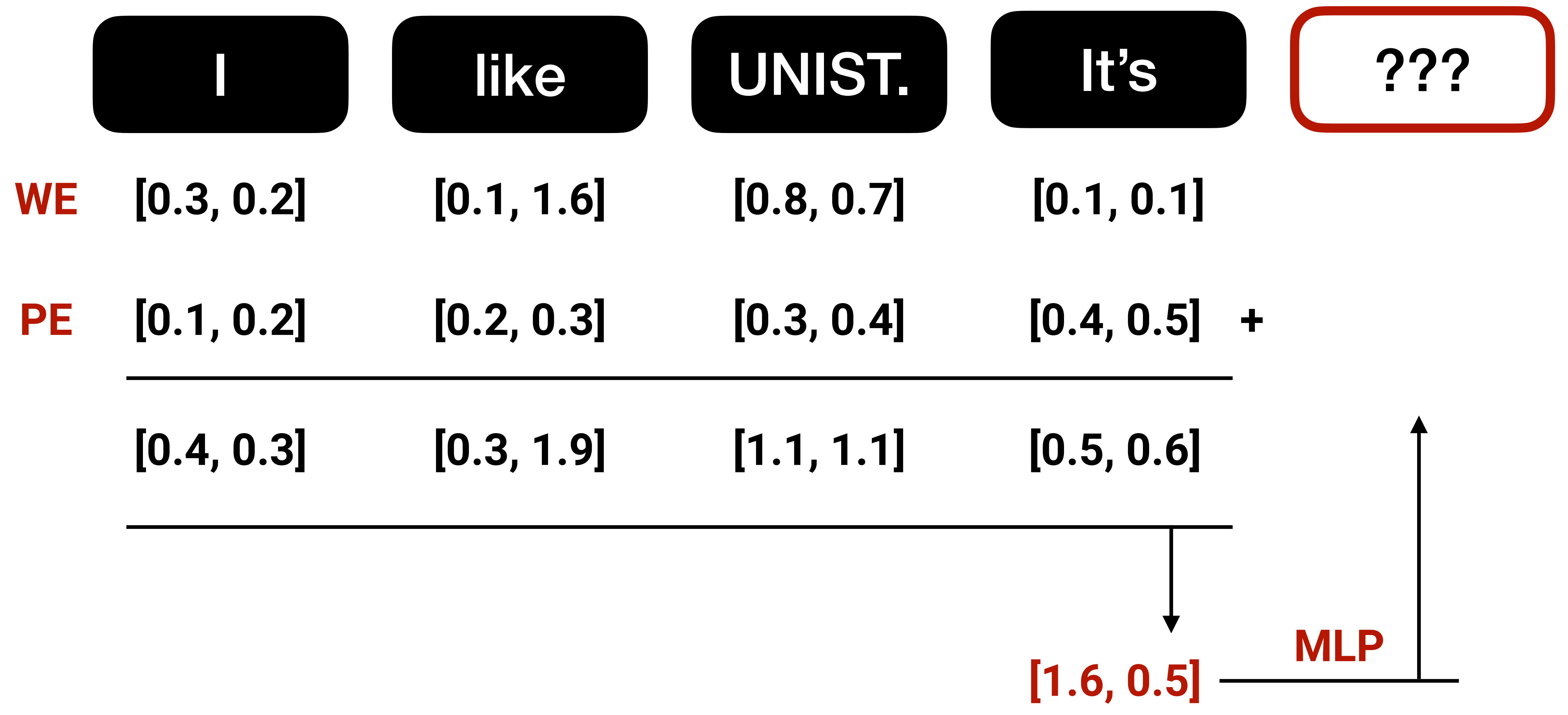
Logics behind CV computation

- Logic for calculating CV
  - The WPE (Word Embedding + Positional Encoding) at the current position (e.g., the 4th word: “It’s”) is combined with the WPEs of the preceding words to calculate the value.
  - A concept similar to averaging
  - Averaging is a very simple combination method. What would be a more "advanced" or sophisticated method?



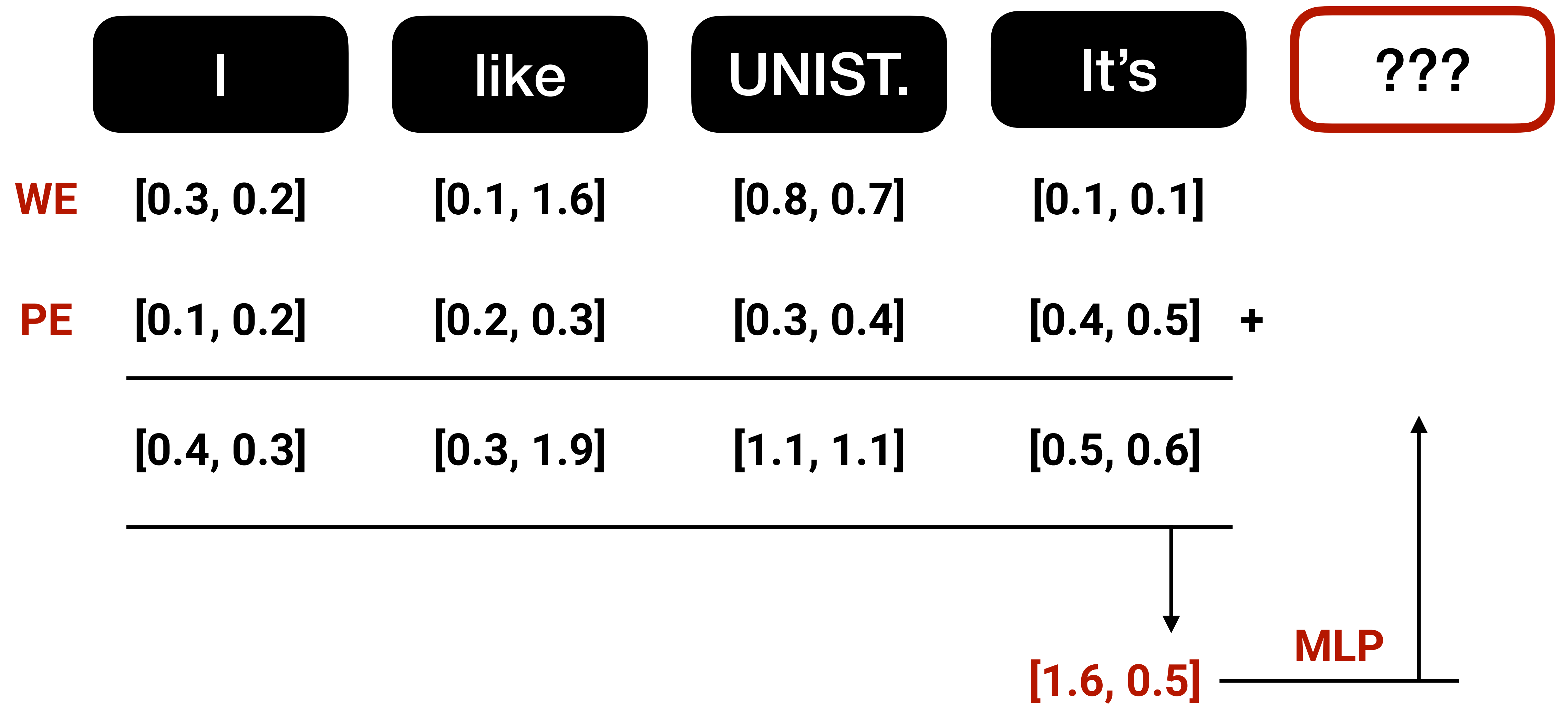
Logics behind CV computation

- Logic for calculating CV
  - The WPE (Word Embedding + Positional Encoding) at the current position (e.g., the 4th word: “It’s”) is combined with the WPEs of the preceding words to calculate the value.
  - A concept similar to averaging
  - Averaging is a very simple combination method. What would be a more "advanced" or sophisticated method?
  - “Attention”!
    - Words and positions that have WPEs “similar” to my WPE are given more weight when combined.
    - Thus, a concept similar to weighted averaging
    - Here, to determine the “weights,” the inner product is used (cosine similarity is also based on the inner product).



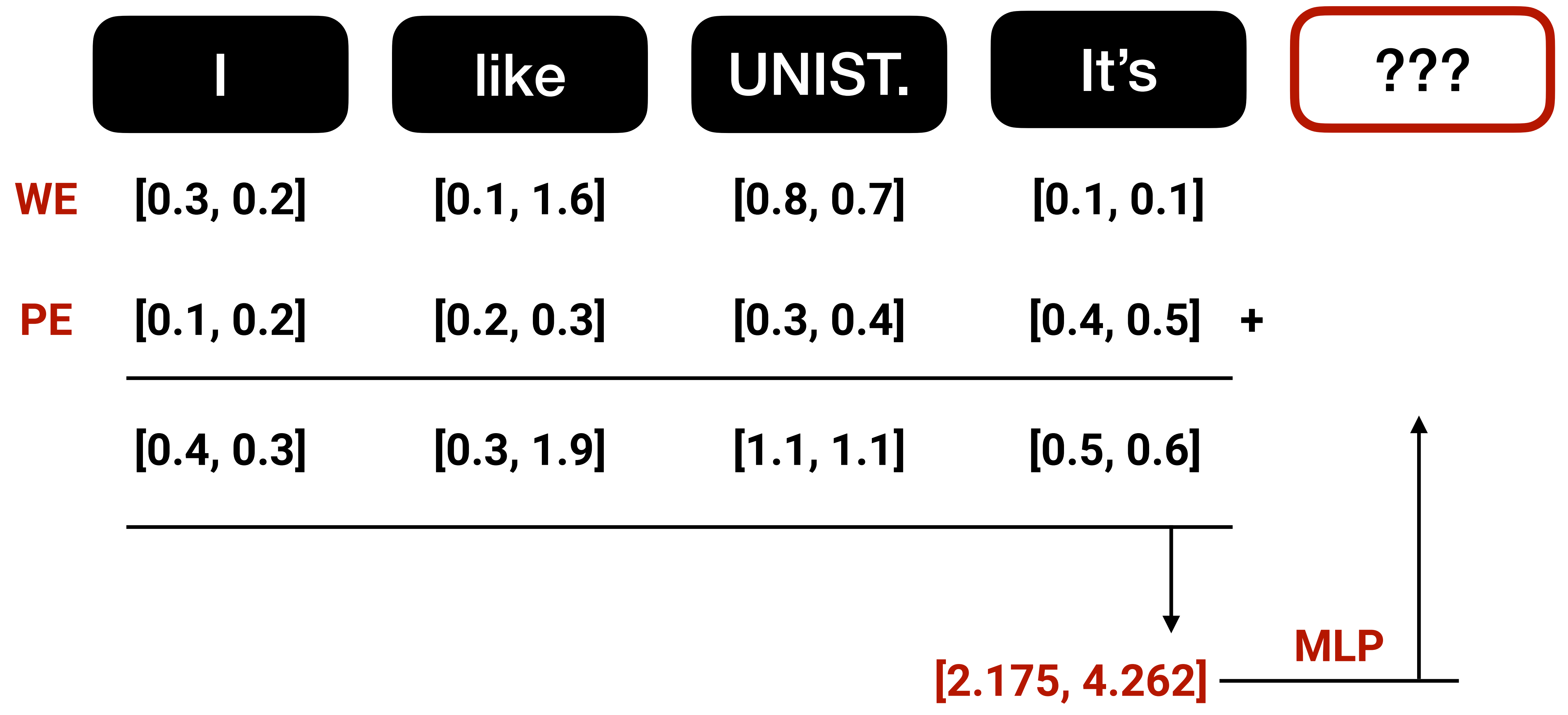
Actual computation of CV

- 실제 CV 계산 과정
  - 지금 타겟 위치는 4번째. 해당 단어는 It’s.
  - 해당 WPE는 [0.5, 0.6]
  - 첫번째 단어와의 “유사도” : [0.4, 0.3]과 [0.5, 0.6]의 내적
    - = 0.4 * 0.5 + 0.3*0.6 = 0.38
  - 두번째 단어와의 “유사도” : [0.3, 1.9]과 [0.5, 0.6]의 내적
    - = 0.3 * 0.5 + 1.9*0.6 = 1.29
  - 세번째 단어와의 “유사도” : [1.1, 1.1]과 [0.5, 0.6]의 내적
    - = 1.1 * 0.5 + 1.1*0.6 = 1.21
  - 네번째 단어 (자기 자신!)과 “유사도” : [0.5, 0.6]과 [0.5, 0.6]의 내적
    - = 0.5 * 0.5 + 0.6*0.6 = 0.61



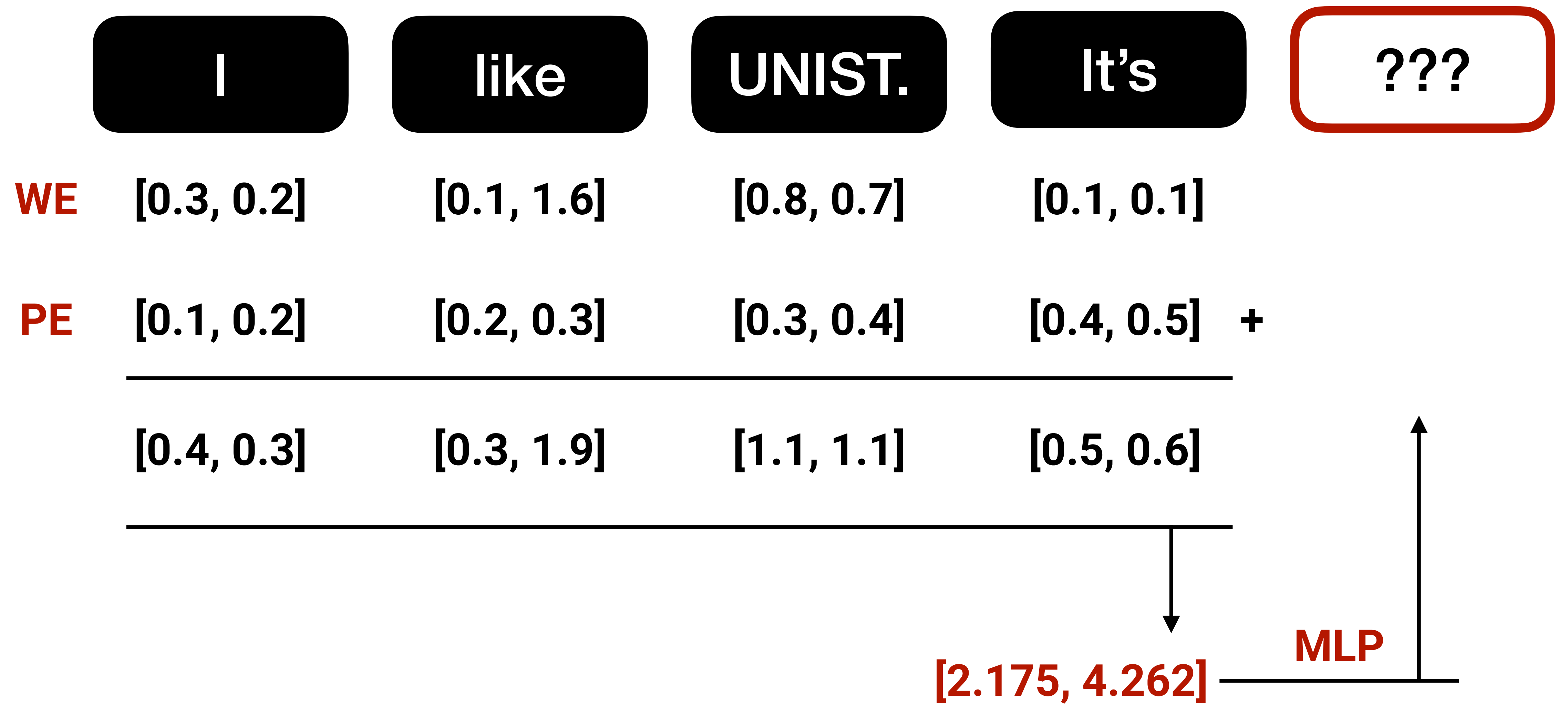
Actual computation of CV

- 실제 CV 계산 과정
  - 구해진 해당 WPE와 다른 위치,단어들의 WPE의 유사도를 기반으로, 가중평균을 구함
    - 0.38 * [0.4, 0.3] + 1.29 * [0.3, 1.9] + 1.21 * [1.1, 1.1] + 0.61 * [0.5, 0.6]
    - = **[2.175, 4.262]**
  -



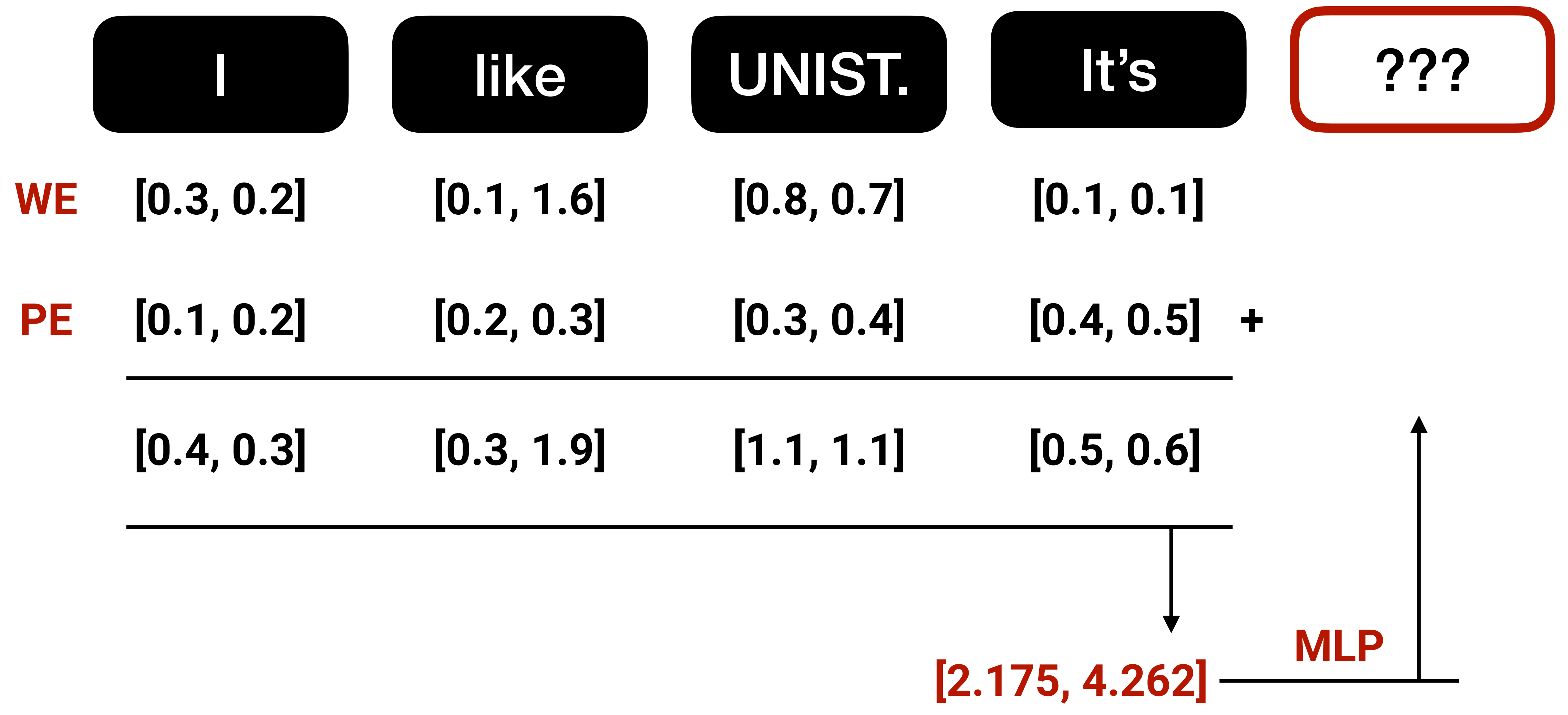
Questions

- Query, Key, Value
  - **Question:** Isn't Transformer a neural network (deep learning)?
  - Where are the model parameters being "trained"?“
  - Assuming WE and PE are given, the only parameters to train are in the MLP
  - But Transformers have hundreds of millions to trillions of parameters—MLP alone seems insufficient.



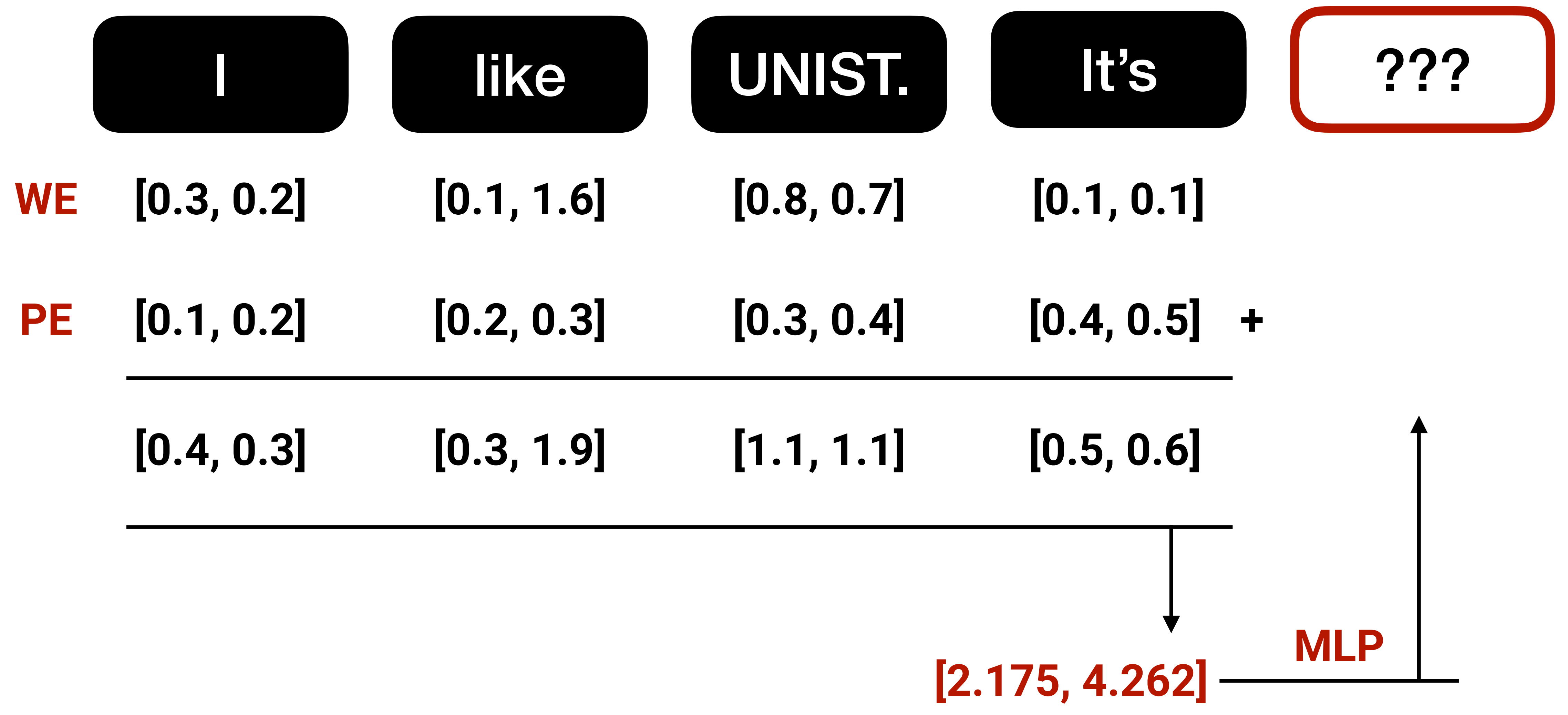
Query, Key, Value (QKV)

- Query, Key, Value
  - In practice, three matrices—query, key, and value—are introduced to calculate attention scores and weighted averages
  - If WPE is a k-dimensional vector (k=2 in our example), the query, key, and value matrices are each k x k in size



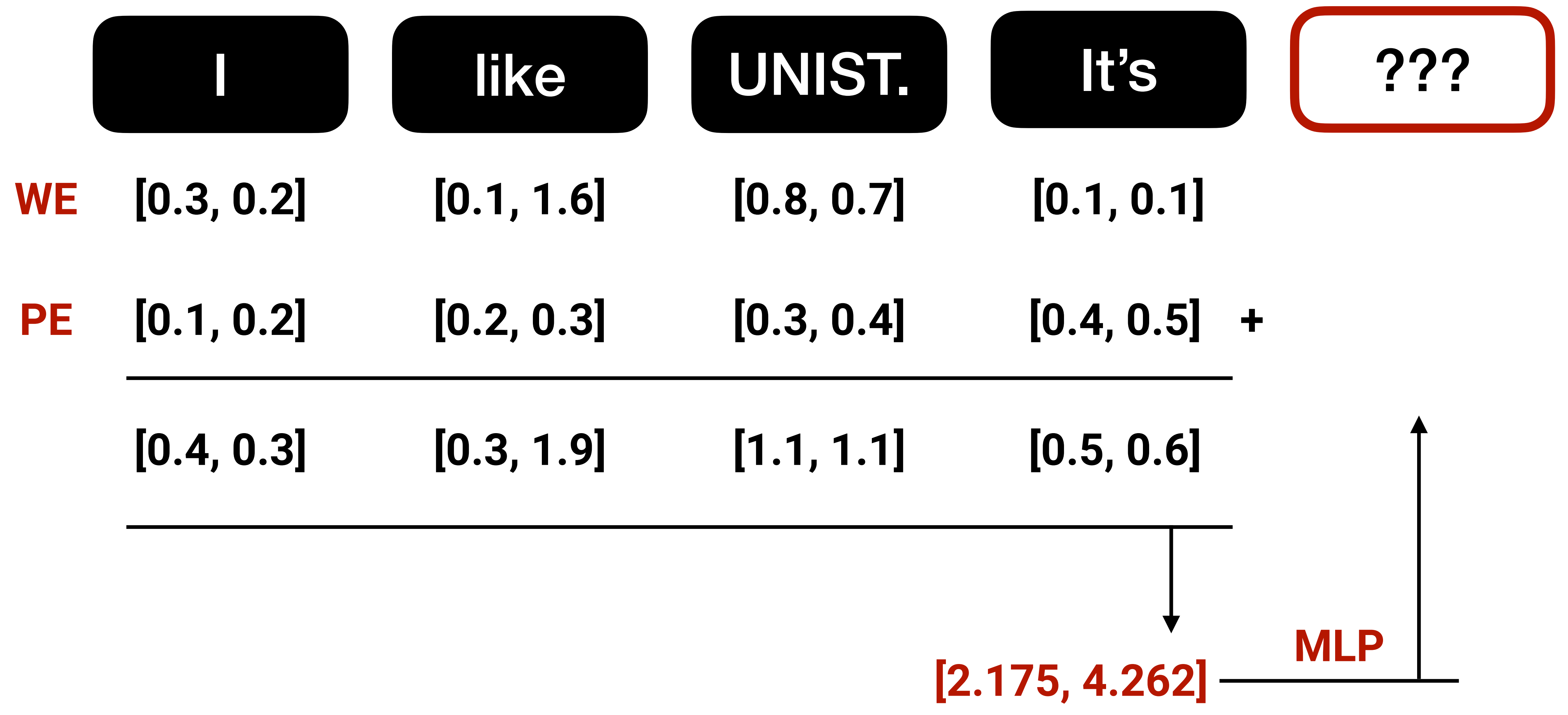
CV computation using QKV

- Query, Key, Value
  - In practice, three matrices—query, key, and value—are introduced to calculate attention scores and weighted averages
  - If WPE is a k-dimensional vector (k=2 in our example), the query, key, and value matrices are each k x k in size
  - The Transformer learning process involves assigning arbitrary values to query, key, and value, and then learning these values over time!
  - Let's actually compute the CV using the followings:
  - Q = [[1, 2], [3, 4]]
  - K = [[5, 6], [7, 8]]
  - V = [[9, 10], [11, 12]]



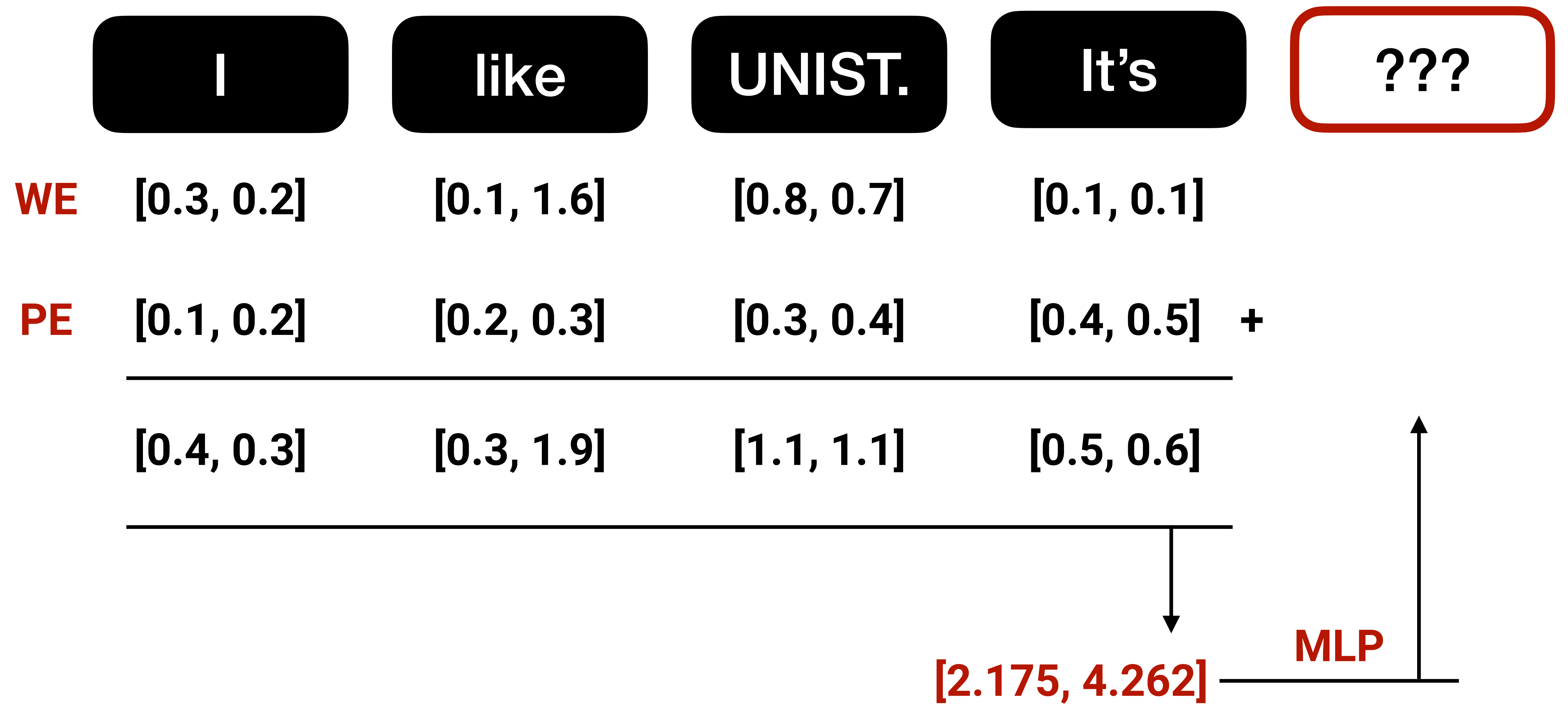
CV computation using QKV

- Query, Key, Value
  - Q = [[1, 2], [3, 4]]
  - K = [[5, 6], [7, 8]]
  - V = [[9, 10], [11, 12]]
  - The attention score of 4-th word (target) “It’s” with respect to the first word “I”
    - Inner product between [0.5, 0.6] and [0.4, 0.3] ->
    - Between Q([0.5, 0.6])^T and K([0.4, 0.3])^T
    - Q([0.5, 0.6])^T = [1.7, 3.9]
    - K([0.4, 0.3])^T = [3.8, 5.2]
    - [1.7, 3.9] dot [3.8, 5.2] = 26.74



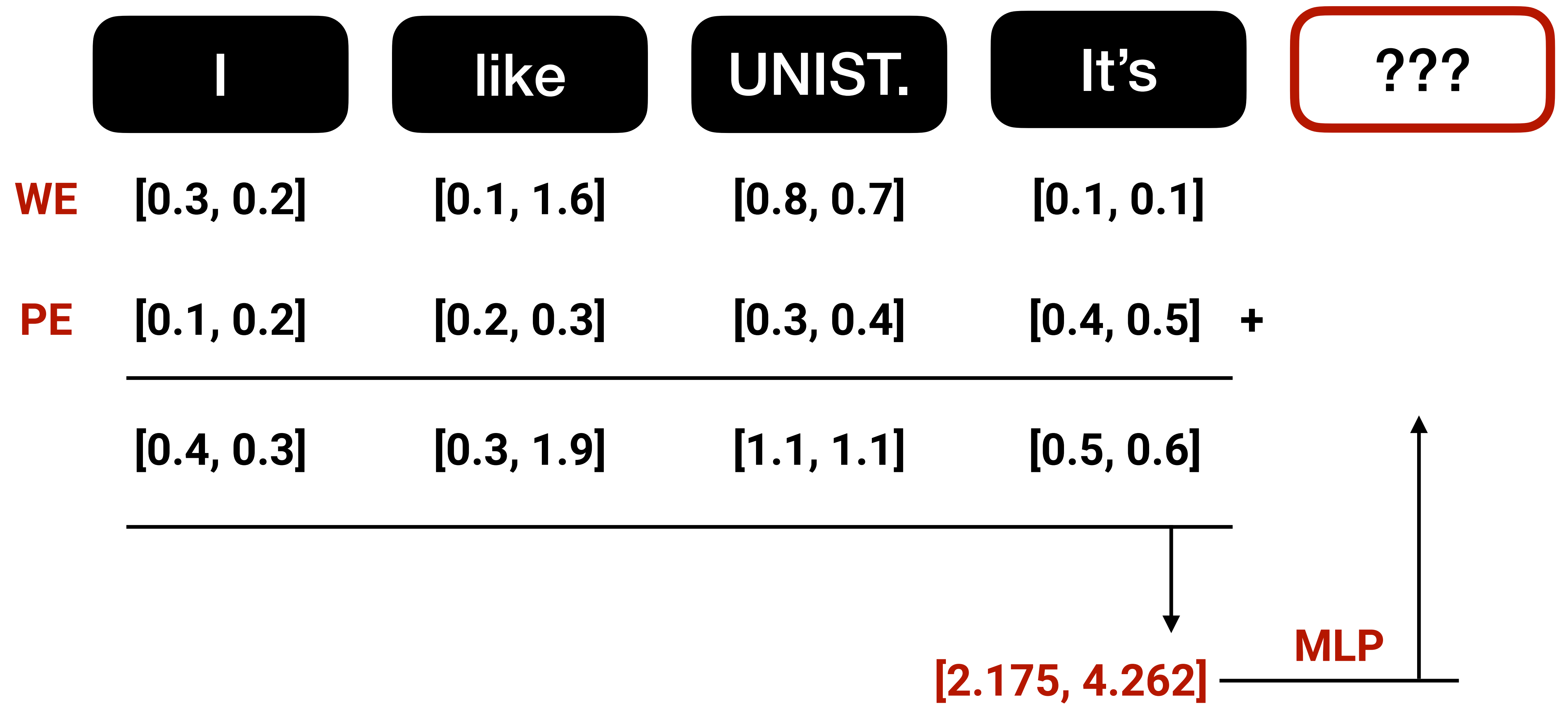
CV computation using QKV

- Query, Key, Value
  - The attention score of the 4th target word "It’s" for the 1st word "I" is 26.74.
  - In the “weighted ” averaging, we should be doing:
  - 26.74 * [0.4, 0.3]
  - However, here comes “V”
  - 26.74 * V ([0.4, 0.3])^T = [176.484, 213.92]



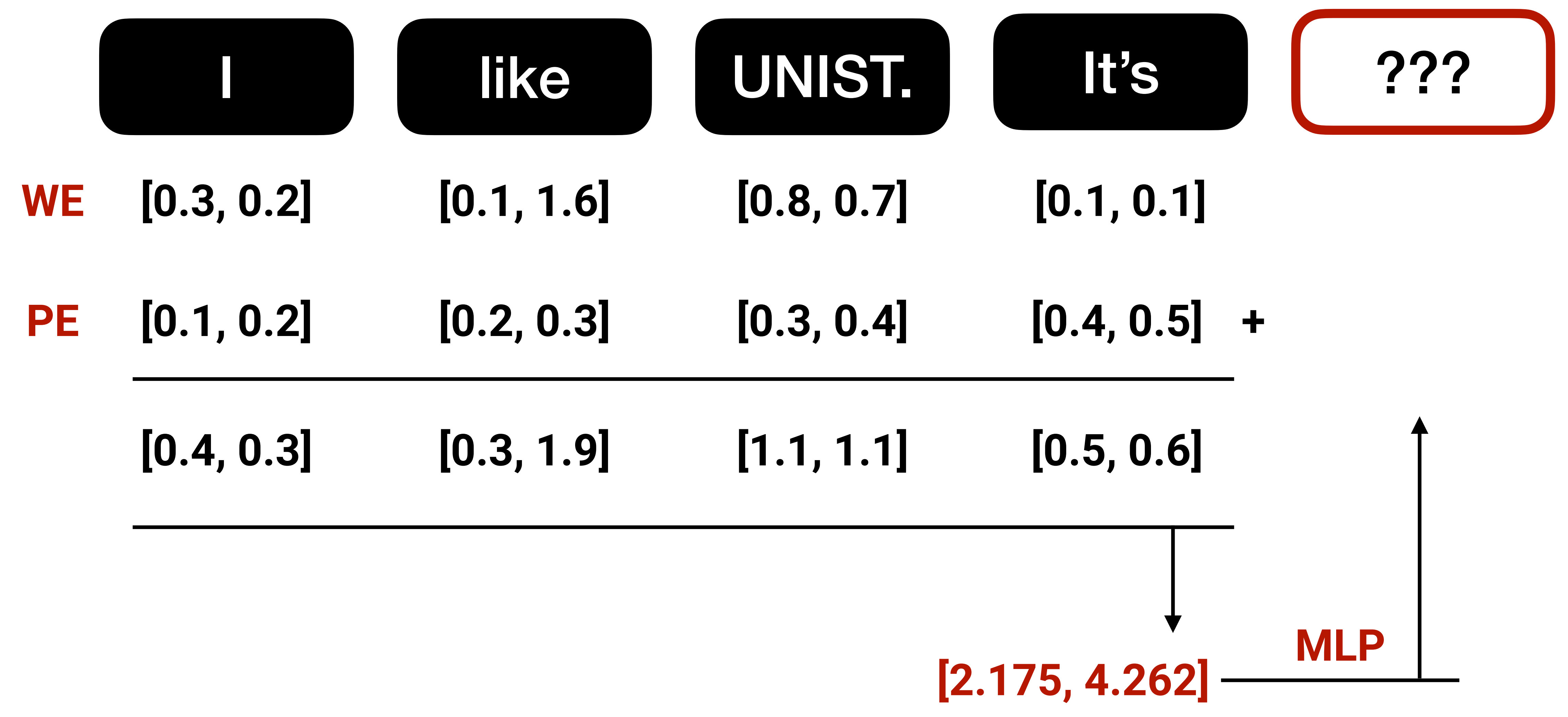
CV computation using QKV

- Query, Key, Value
  - Similarly, the attention score of the 4th target word "It’s" for the 2nd word "like" is 89.4
  - In a regular weighted average, the weight is 89.4
  - The calculated vector value is [1939.98, 2333.34]



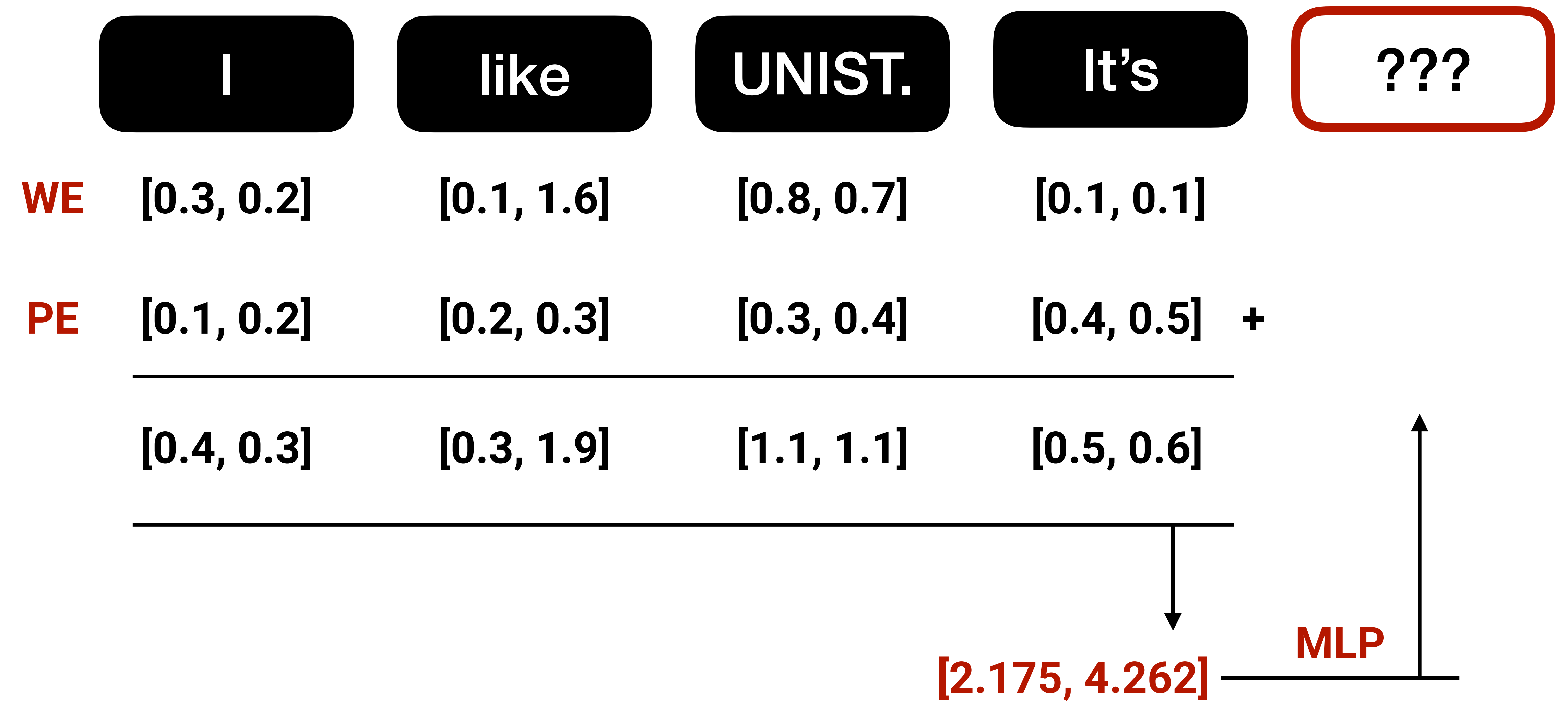
CV computation using QKV

- Query, Key, Value
  - Similarly, the attention score of the 4th target word "It’s" for the 3rd word "UNIST" is 84.92
  - In a regular weighted average, the weight is 84.92
  - The calculated vector value is [1774.828, 2148.476]



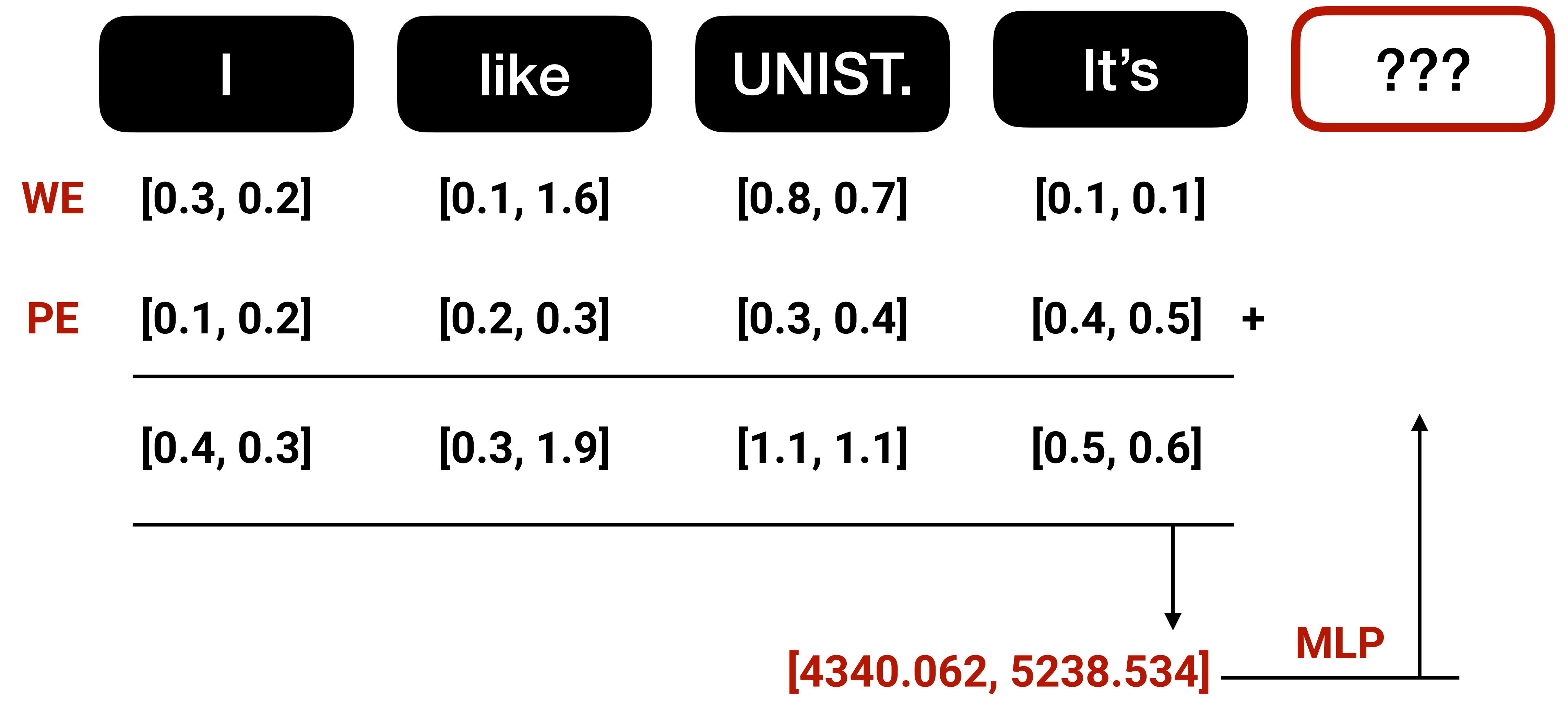
CV computation using QKV

- Query, Key, Value
  - Similarly, the attention score of the 4th target word "It’s" for the 4th word “It's" is 42.74
  - [448.77, 542.798]



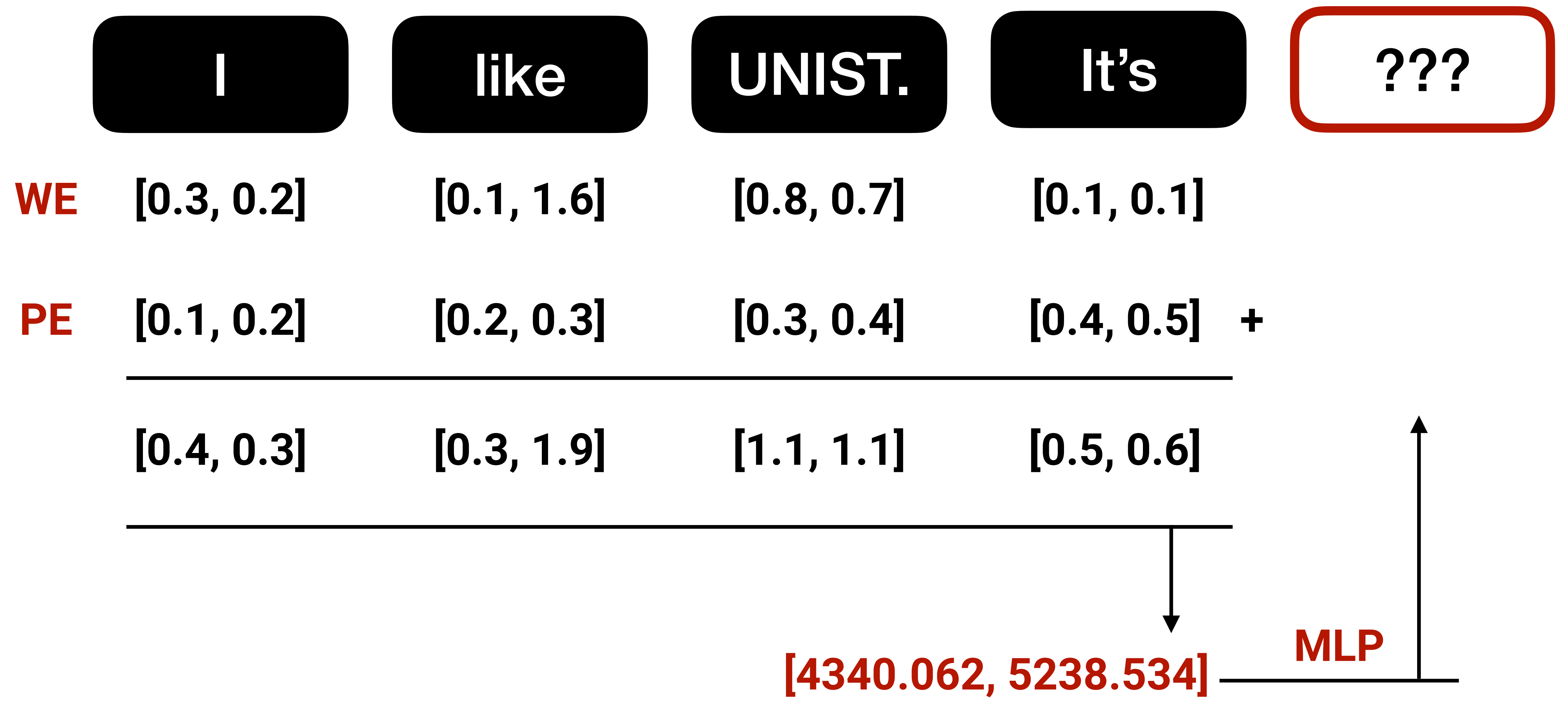
CV computation using QKV

- Query, Key, Value
  - The final CV for “It’s” is:
  - [4340.062, 5238.534] !



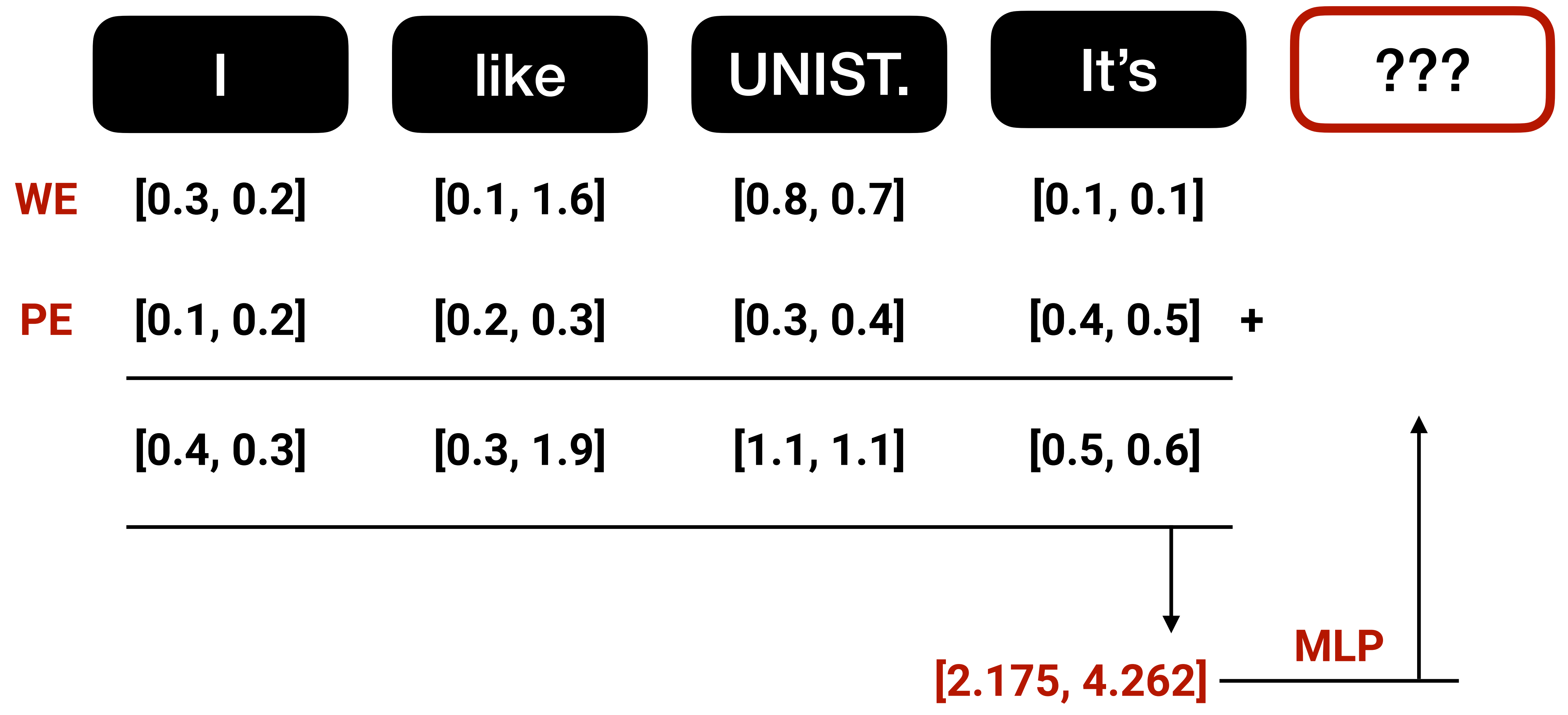
Softmax

- Softmax
  - In an actual Transformer, there are dozens to hundreds of attention layers.
  - However, repeating this process can cause attention scores to become very large or very small.
    - Overflow, underflow -> “Break down” (worst-case)
  - To remedy this, we take the softmax function with respect to the attention score
  - -> Softmax([26.74, 89.4, 84,92, 42.74])
  - = [5.5*1e-20, 0.96, 0.04, 4.5*1e-15]
  - (Sum to 1)



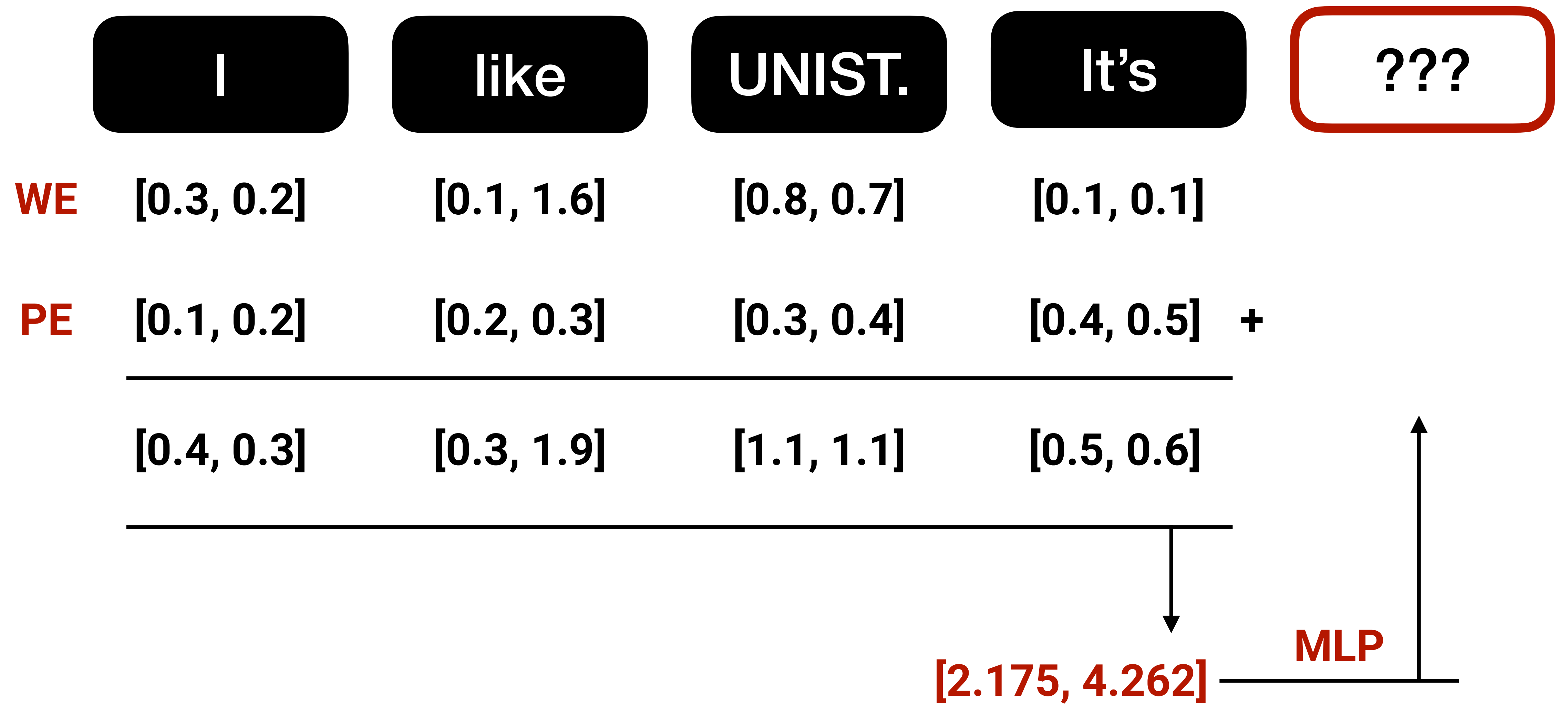
CV computation using QKV

- Query, Key, Value
  - The attention score of the 4th target word "It’s" for the 1st word "I" is 26.74.
  - In the “weighted ” averaging, we should be doing:
  - 26.74 * [0.4, 0.3]
  - However, here comes “V”
  - 26.74 * V ([0.4, 0.3])^T = [176.484, 213.92]
  - -> 6*1e-28 * V ([0.4, 0.3])^T ~= **0.**



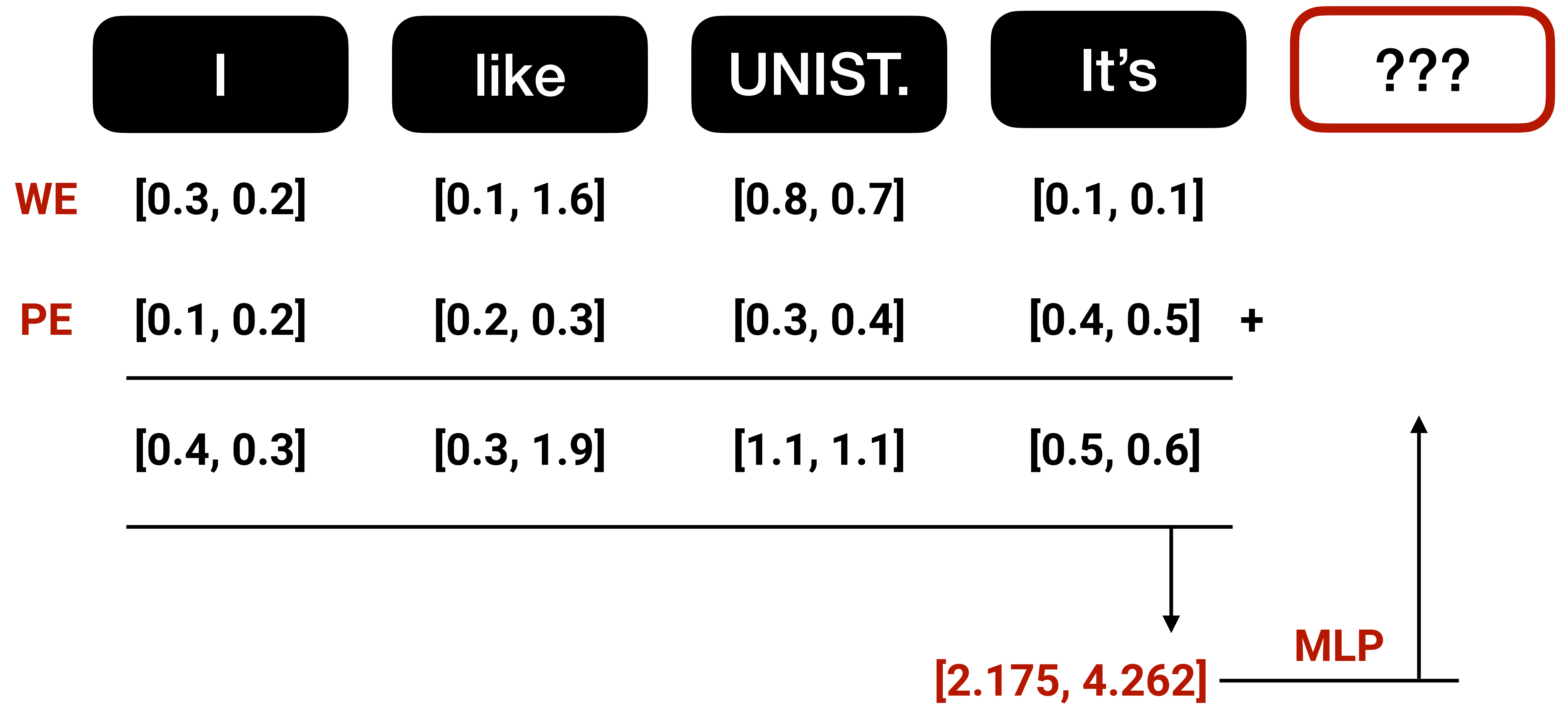
CV computation using QKV

- Query, Key, Value
  - Similarly, the attention score of the 4th target word "It's" for the 2nd word "like" is 89.4
  - In a regular weighted average, the weight is 89.4
  - The calculated vector value is [1939.98, 2333.34] ->
  - [20.7452, 24.9516]



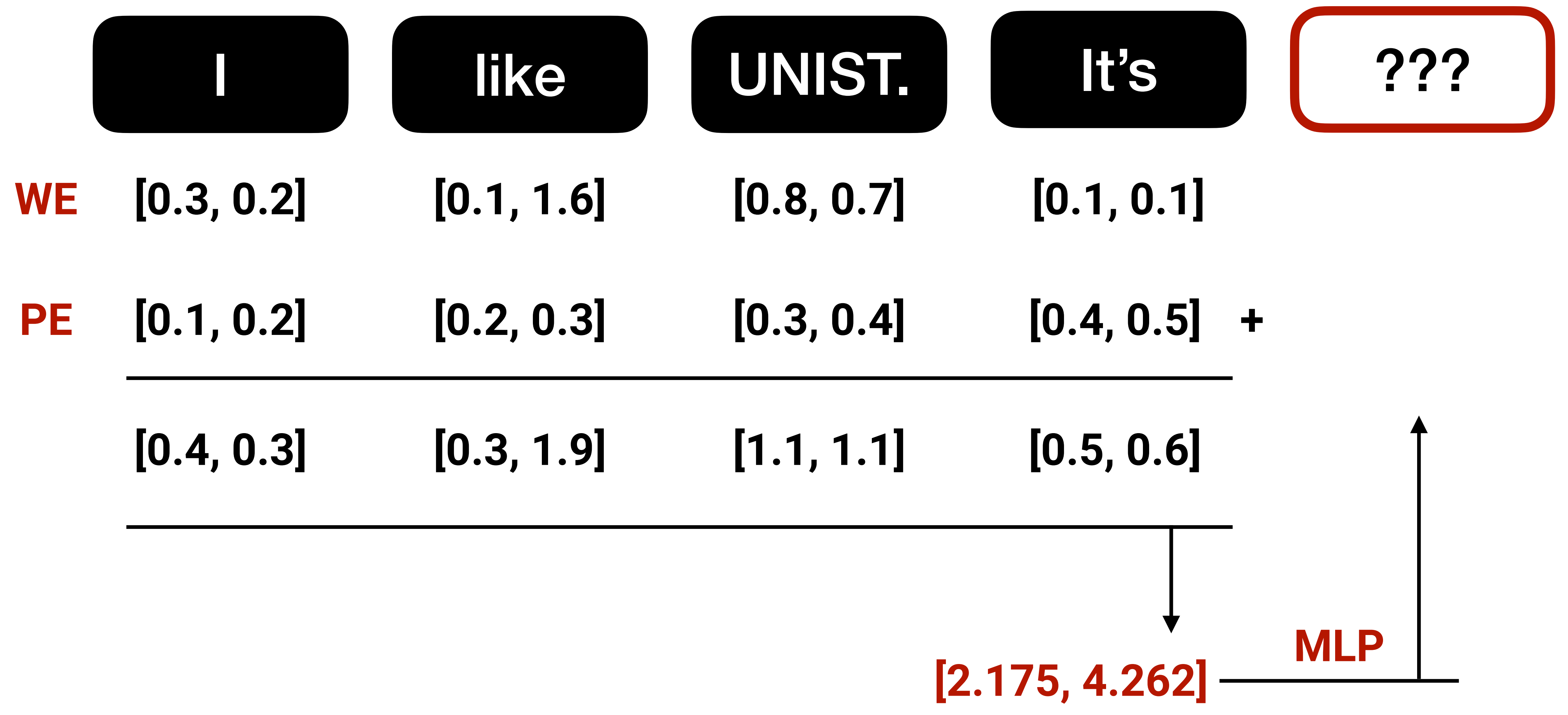
CV computation using QKV

- Query, Key, Value
  - Similarly, the attention score of the 4th target word "It’s" for the 3rd word "UNIST" is 84.92
  - In a regular weighted average, the weight is 84.92
  - The calculated vector value is [1774.828, 2148.476] ->
  - ->[0.836, 1.012]



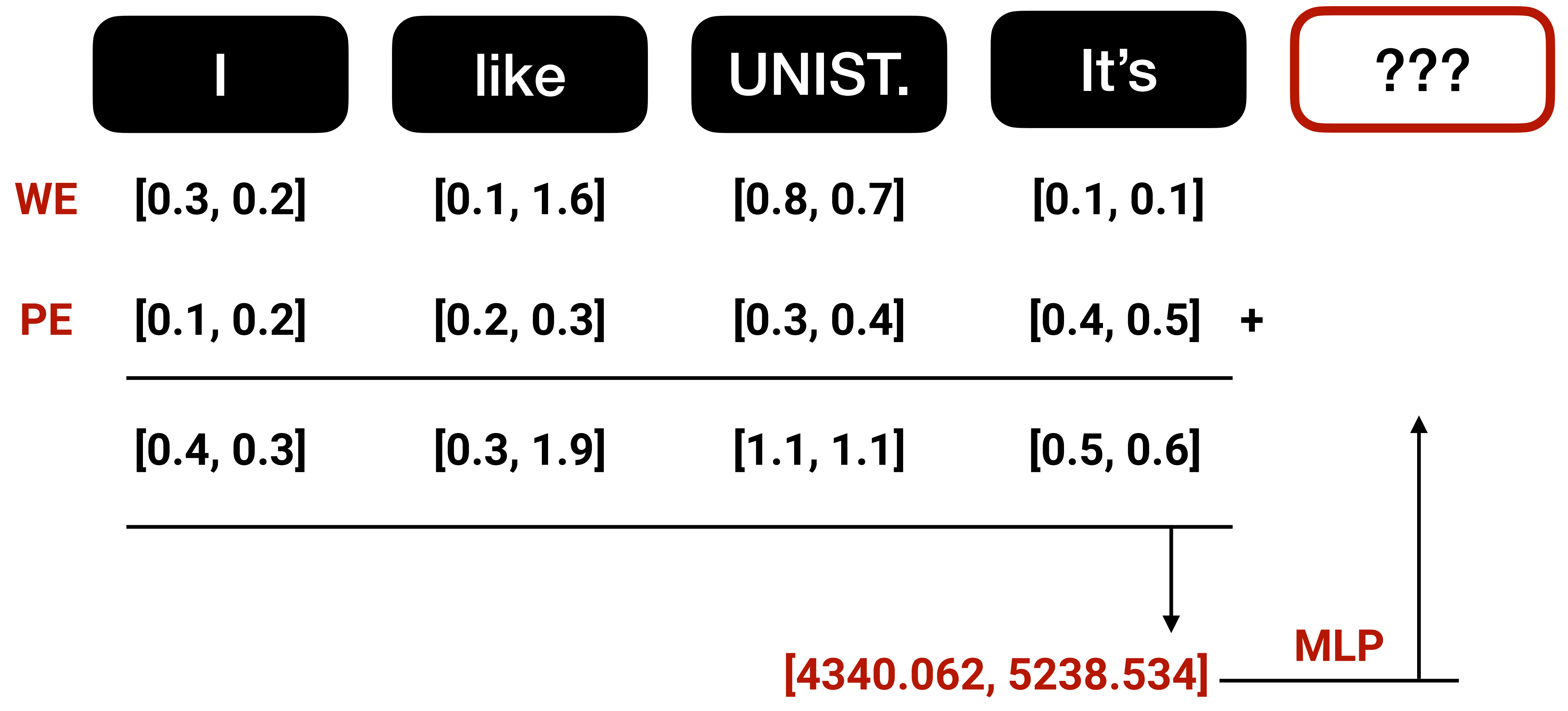
CV computation using QKV

- Query, Key, Value
  - Similarly, the attention score of the 4th target word "It’s" for the 4th word “It's" is 42.74
  - [448.77, 542.798]
  - -> This goes to zero as well



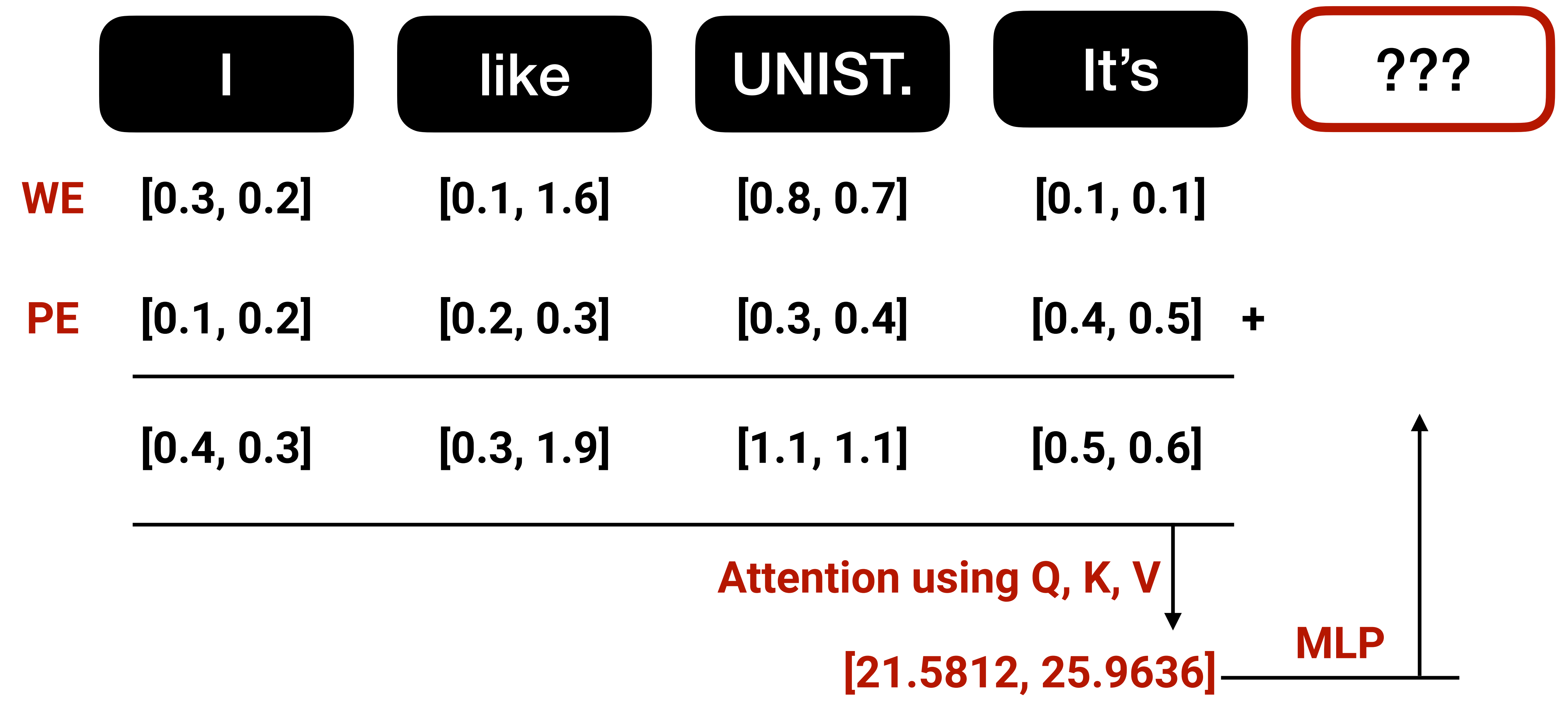
CV computation using QKV

- Query, Key, Value
  - The final CV is:
  - [4340.062, 5238.534]
  - -> [21.5812, 25.9636]!



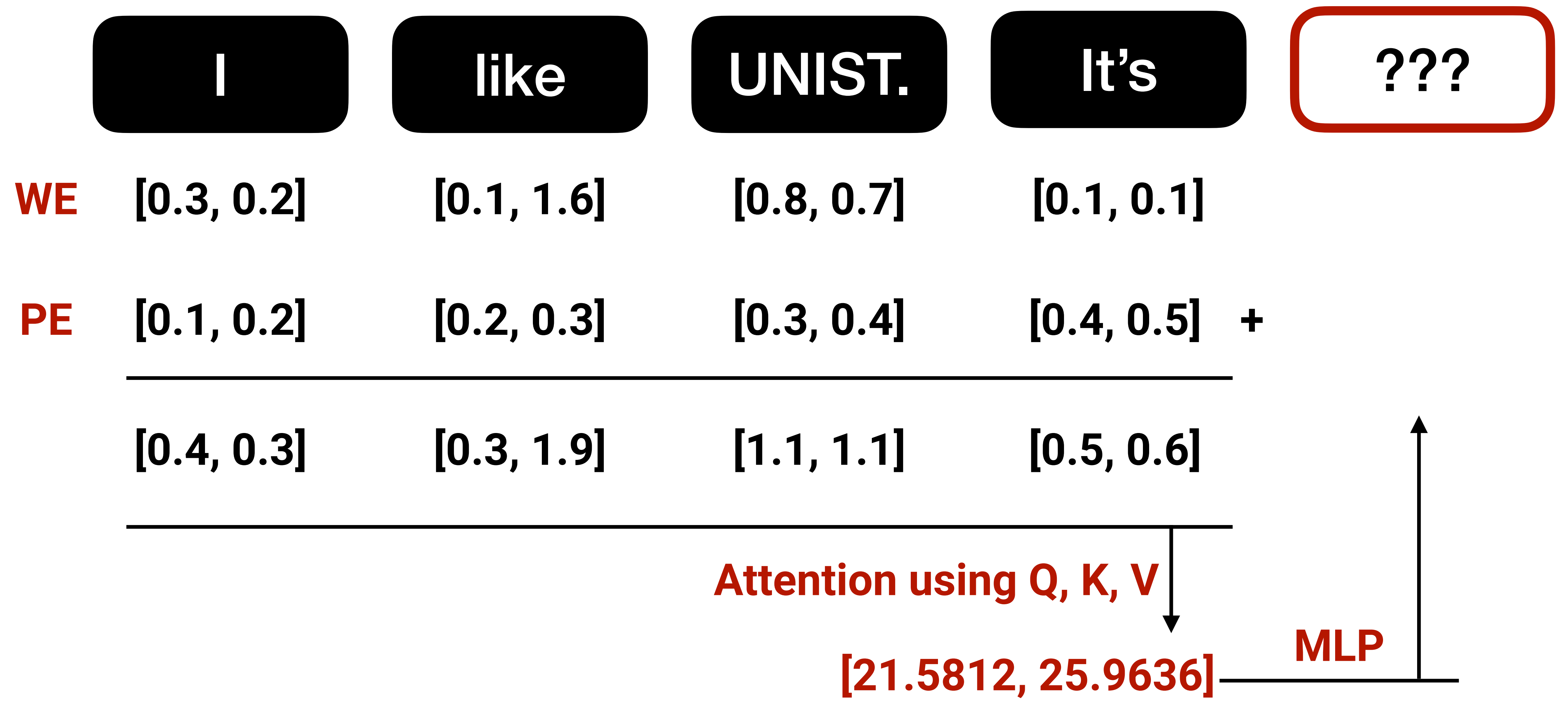
Transformer in a nutshell

- Initialize the word, position embeddings
  - Word (token)
- Compute context vector
  - Using attentions (query, key, value)
  - Parameters to learn: entries in Q, K, V matrices
- MLP for final prediction



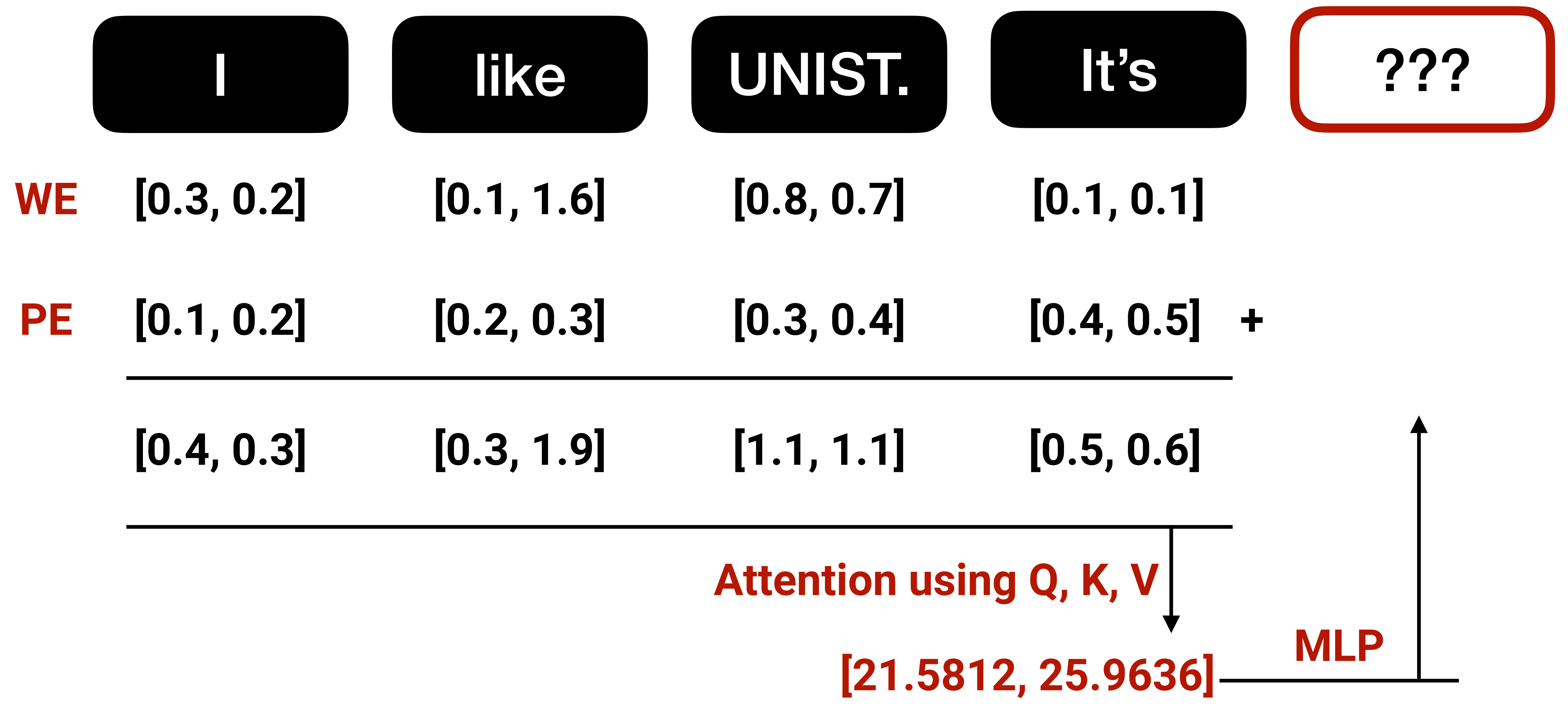
Multi-layer transformer

- Multi-layer
  - 원래는 4번째 단어에 대한 WPE가 [0.5, 0.6]
  - 두번째 레이어에서는 WPE가 [21.58, 25.96]이 됨
- Self-attention
  - Q, K, V의 대상이 모두 동일
- Cross-attention
  - Q의 대상이 K, V의 대상과 다름
- Causal attention
  - attention을 계산하는 방향이 왼쪽에서 오른쪽으로만
  - Bert: 양방향
- GPT는 self, causal attention을 사용



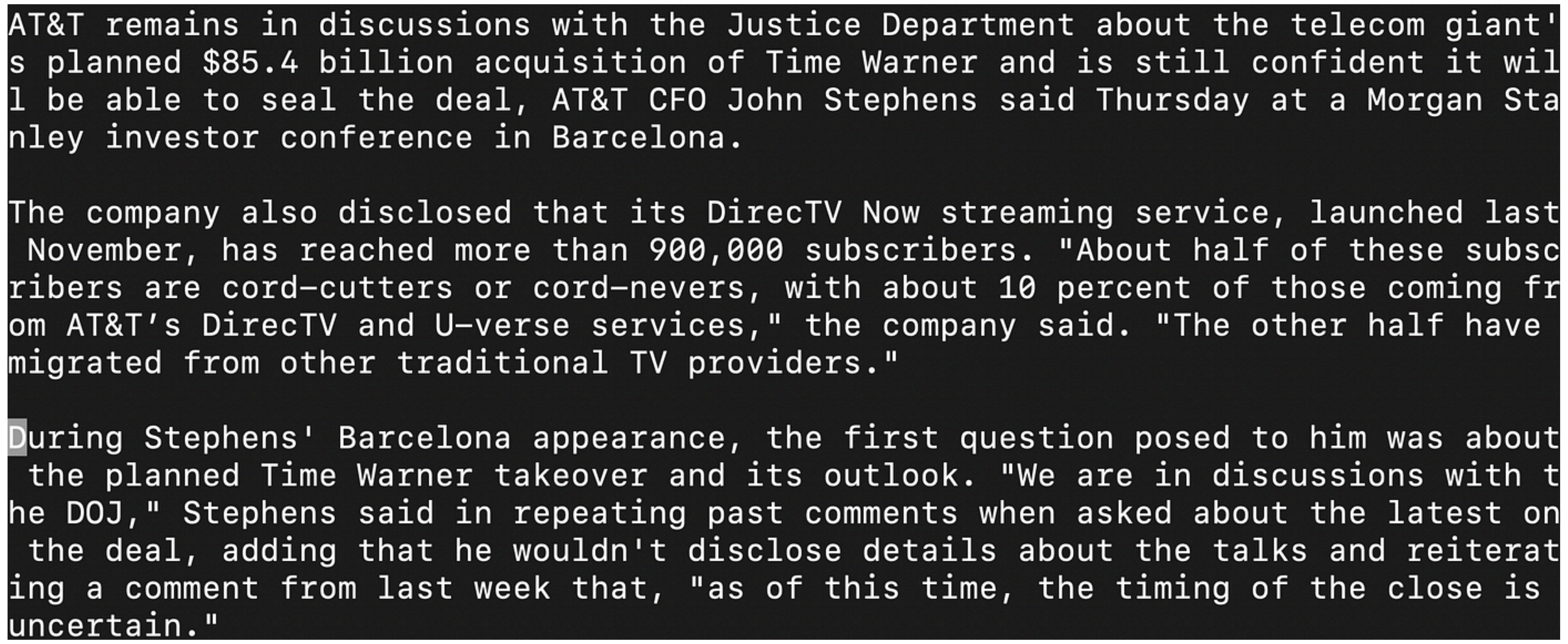
NanoGPT

- https://github.com/karpathy/nanoGPT
  - 실재로 파라미터 수 수억~수십억 짜리 GPT 모델을 ~300줄 이내의 Pytorch코드로 구현
  - train.py
    - 데이터 로딩, 모델 로딩, 멀티GPU 구현, optimizer 구현
  - model.py
    - attention, mlp layers



NanoGPT: Data

- OpenWebText
  - 매우 단순하게, 약 9B (90억)개의 토큰으로 이뤄짐.
  - 각 토큰은 0 ~ 50257이 숫자로 매핑
    - uint16으로 가능
  - 즉, data.bin은 0부터 50257까지의 숫자의 약 90억개의 배열 (array)
  - 1B (10억 토큰) -> 약 2GB (rule of thumb)
    - Uint32 면 4GB
    - 총 약 18GB
  - 이걸 train, validation set으로 나눔
    - Validation set은 매우매우 작음 (10MB 이내)

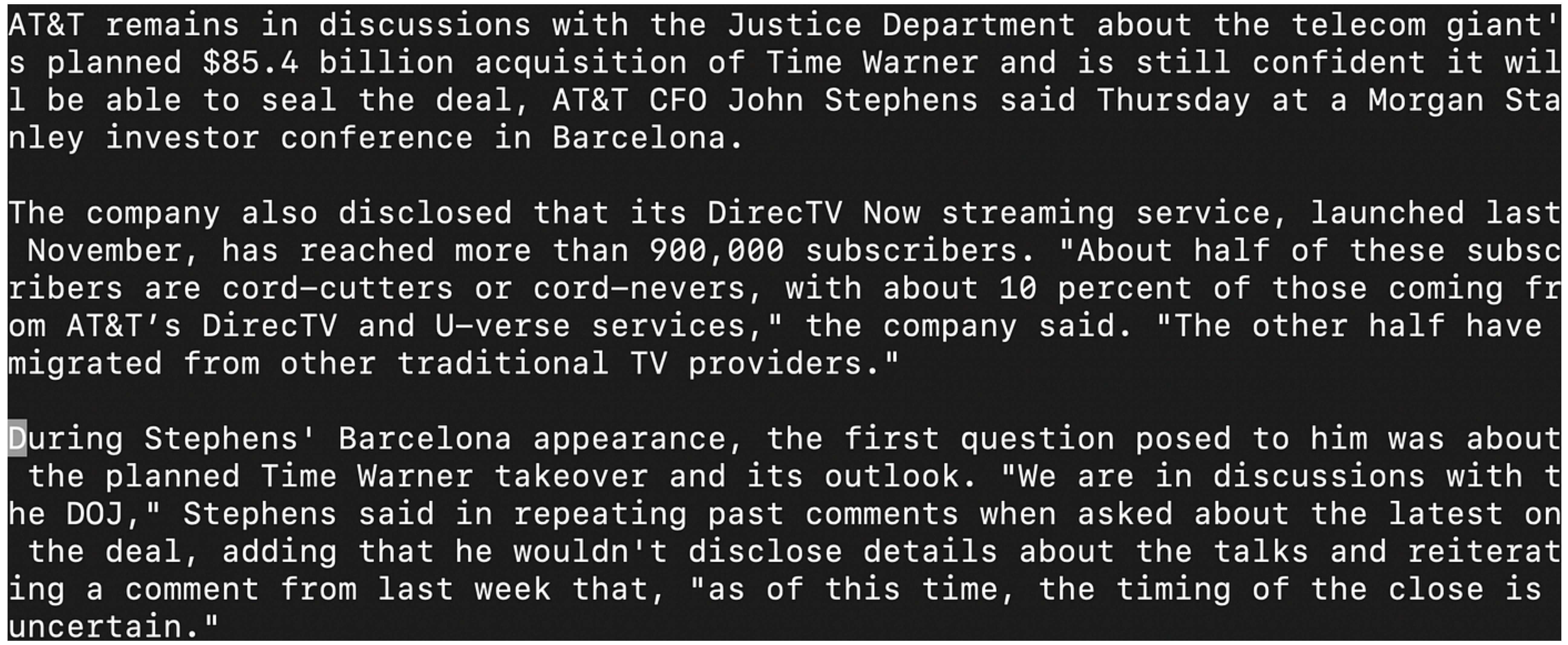


```
AT&T remains in discussions with the Justice Department about the telecom giant'
s planned $85.4 billion acquisition of Time Warner and is still confident it wil
l be able to seal the deal, AT&T CFO John Stephens said Thursday at a Morgan Sta
nley investor conference in Barcelona.

The company also disclosed that its DirecTV Now streaming service, launched last
 November, has reached more than 900,000 subscribers. "About half of these subsc
ribers are cord-cutters or cord-nevers, with about 10 percent of those coming fr
om AT&T's DirecTV and U-verse services," the company said. "The other half have
migrated from other traditional TV providers."

During Stephens' Barcelona appearance, the first question posed to him was about
 the planned Time Warner takeover and its outlook. "We are in discussions with t
he DOJ," Stephens said in repeating past comments when asked about the latest on
 the deal, adding that he wouldn't disclose details about the talks and reiterat
ing a comment from last week that, "as of this time, the timing of the close is
uncertain."
```

https://medium.com/datafabrica/mastering-context-extraction-for-fine-tuning-gpt-models-03c5b2f3dc02

NanoGPT: Data

- 트랜스포머가 하는일:
  - [나는 오늘 학교에 갔다]를 토큰화: [20, 31, 58, 29, 49]
  - 실제로는 openwebtext의 데이터에서 랜덤하게 context vector 사이즈 + 1 만큼의 배열 (array)를 샘플
  - [20, 31, 58, 29, ..., 84, 49]
  - 여기서, x = [20, 31, 58, ..., 84]
  - y = [31, 58, 29, ..., 49]
  - 즉, model(x) 와 y의 차이 (로스)를 최소화해서 model(x)가 y를 잘 예측하도록 하는 작업
  - How? By computing the context vectors!

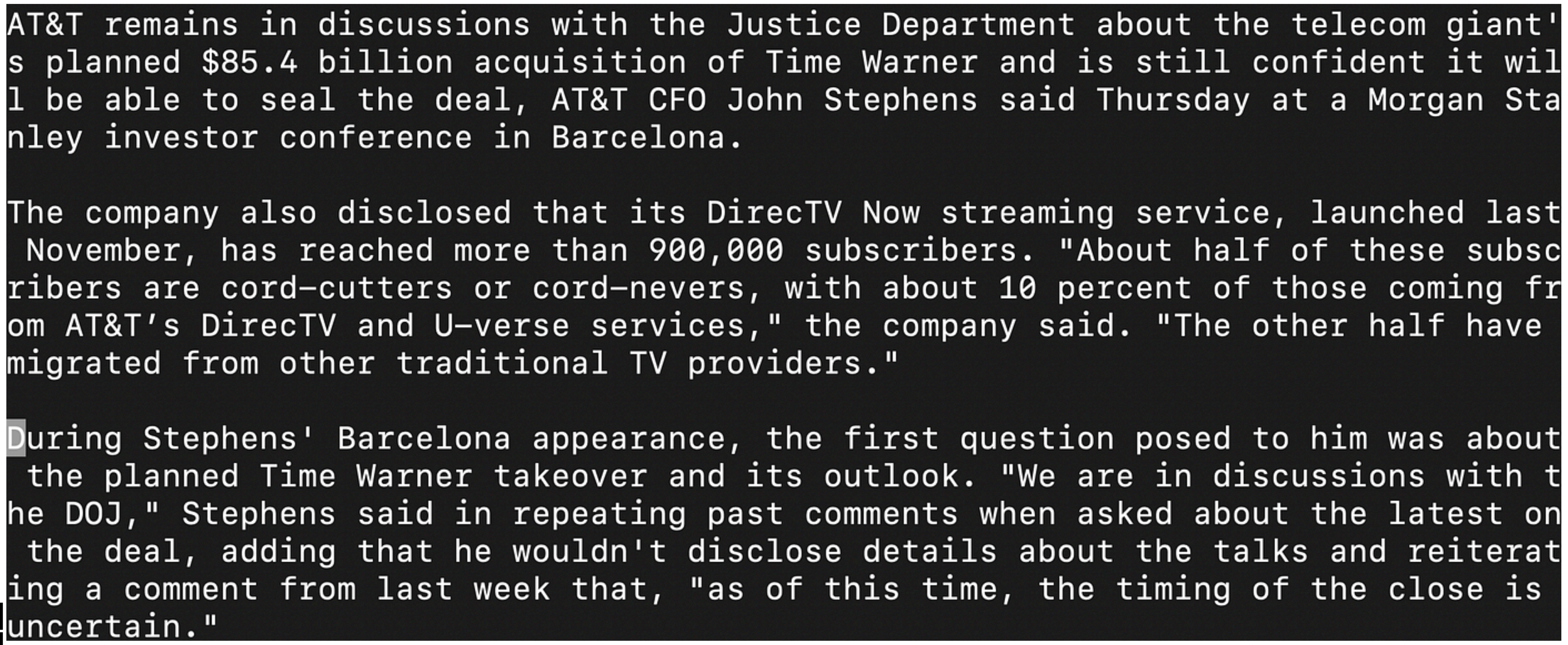


```
AT&T remains in discussions with the Justice Department about the telecom giant'
s planned $85.4 billion acquisition of Time Warner and is still confident it wil
l be able to seal the deal, AT&T CFO John Stephens said Thursday at a Morgan Sta
nley investor conference in Barcelona.

The company also disclosed that its DirecTV Now streaming service, launched last
 November, has reached more than 900,000 subscribers. "About half of these subsc
ribers are cord-cutters or cord-nevers, with about 10 percent of those coming fr
om AT&T's DirecTV and U-verse services," the company said. "The other half have
migrated from other traditional TV providers."

During Stephens' Barcelona appearance, the first question posed to him was about
 the planned Time Warner takeover and its outlook. "We are in discussions with t
he DOJ," Stephens said in repeating past comments when asked about the latest on
 the deal, adding that he wouldn't disclose details about the talks and reiterat
ing a comment from last week that, "as of this time, the timing of the close is
uncertain."
```

https://medium.com/datafabrica/mastering-context-extraction-for-fine-tuning-gpt-models-03c5b2f3dc02

NanoGPT: Data

- 트랜스포머가 하는일:
  - [나는 오늘 학교에 갔다]를 토큰화: [20, 31, 58, 29, 49]
  - 실제로는 openwebtext의 데이터에서 랜덤하게 context vector 사이즈 + 1 만큼의 배열 (array)를 샘플
  - [20, 31, 58, 29, …, 84, 49]
  - 여기서, x = [20, 31, 58, …, 84]
  - y = [31, 58, 29, …, 49]
  - 실제로는, batch size 만큼의 x, y를 한꺼번에 model에 통과시킴
  - x1 = x
  - x = [x1, x2, … xB]
  - 즉, model(x) 와 y의 차이 (로스)를 최소화해서 model(x)가 y를 잘 예측하도록 하는 작업
  - How? By computing the context vectors!

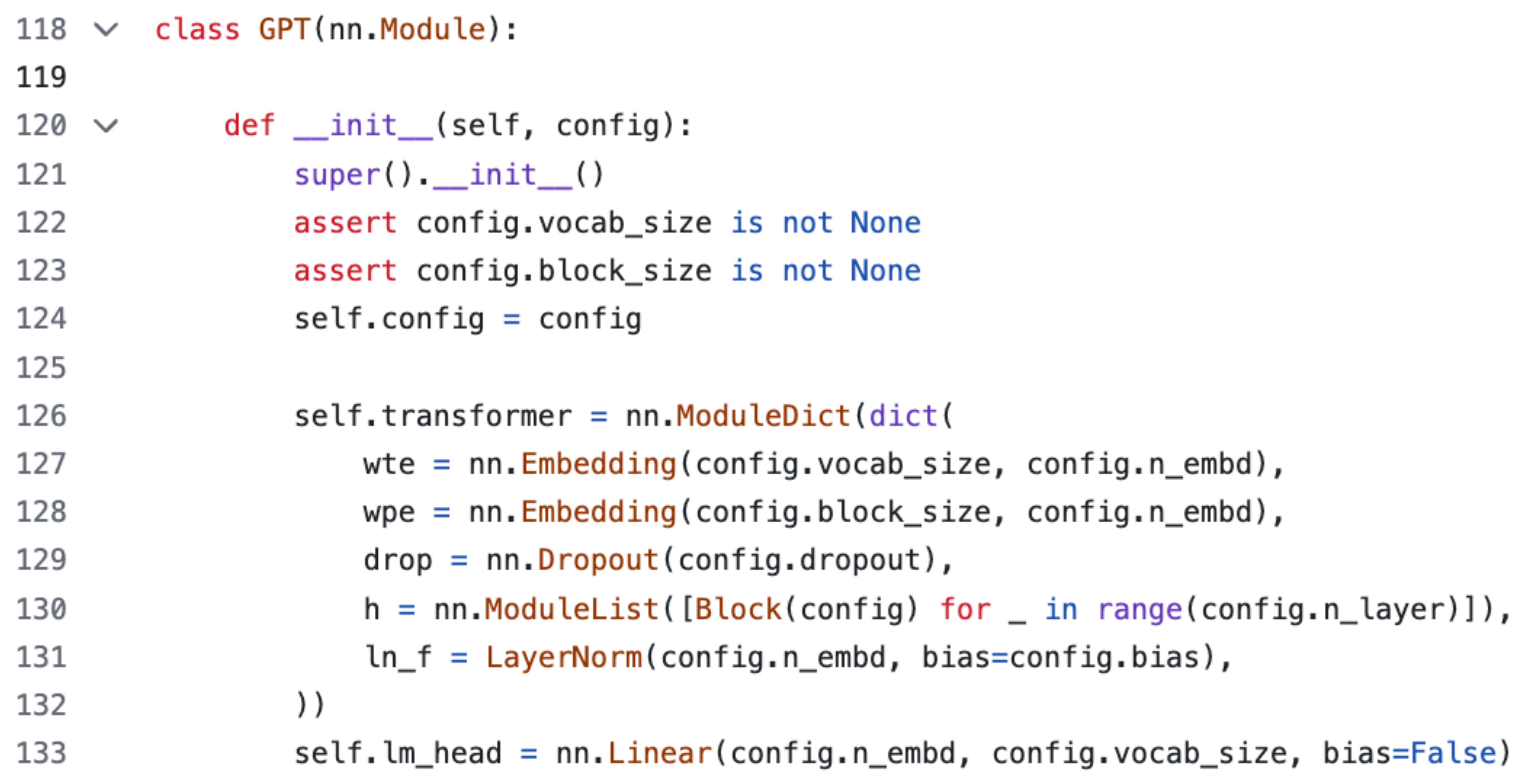


```
AT&T remains in discussions with the Justice Department about the telecom giant'
s planned $85.4 billion acquisition of Time Warner and is still confident it wil
l be able to seal the deal, AT&T CFO John Stephens said Thursday at a Morgan Sta
nley investor conference in Barcelona.

The company also disclosed that its DirecTV Now streaming service, launched last
 November, has reached more than 900,000 subscribers. "About half of these subsc
ribers are cord-cutters or cord-nevers, with about 10 percent of those coming fr
om AT&T's DirecTV and U-verse services," the company said. "The other half have
migrated from other traditional TV providers."

During Stephens' Barcelona appearance, the first question posed to him was about
 the planned Time Warner takeover and its outlook. "We are in discussions with t
he DOJ," Stephens said in repeating past comments when asked about the latest on
 the deal, adding that he wouldn't disclose details about the talks and reiterat
ing a comment from last week that, "as of this time, the timing of the close is
uncertain."
```

https://medium.com/datafabrica/mastering-context-extraction-for-fine-tuning-gpt-models-03c5b2f3dc02

NanoGPT: model.py

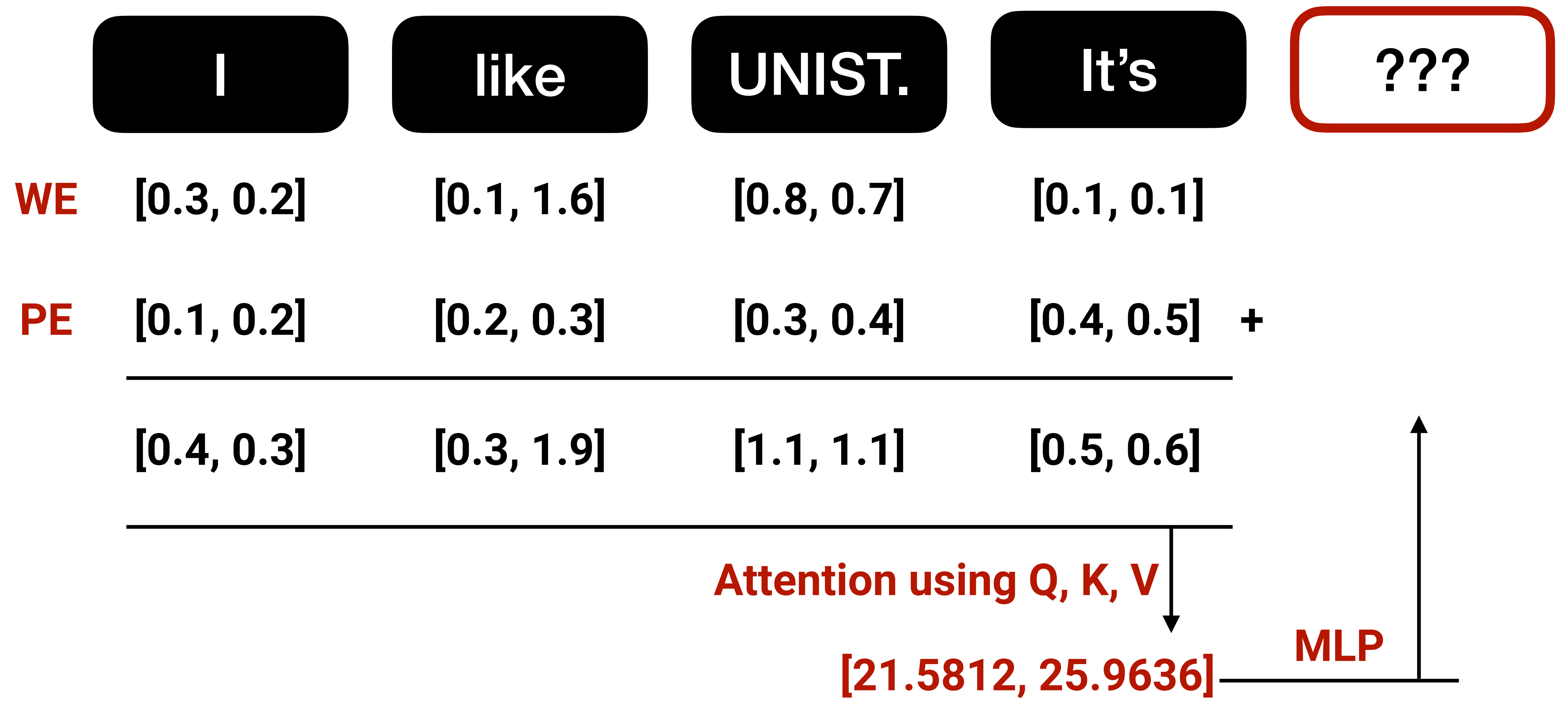
- Token, positional을 초기화
  - n_embd: WE, PE의 벡터 크기 (수백~수천)
  - vocab_size: 총 토큰 개수 (약 5만 ~ 10만)
  - block_size: max 컨텍스트 길이 (수백~수천)



```python
class GPT(nn.Module):

    def __init__(self, config):
        super().__init__()
        assert config.vocab_size is not None
        assert config.block_size is not None
        self.config = config

        self.transformer = nn.ModuleDict(dict(
            wte = nn.Embedding(config.vocab_size, config.n_embd),
            wpe = nn.Embedding(config.block_size, config.n_embd),
            drop = nn.Dropout(config.dropout),
            h = nn.ModuleList([Block(config) for _ in range(config.n_layer)]),
            ln_f = LayerNorm(config.n_embd, bias=config.bias),
        ))
        self.lm_head = nn.Linear(config.n_embd, config.vocab_size, bias=False)
```

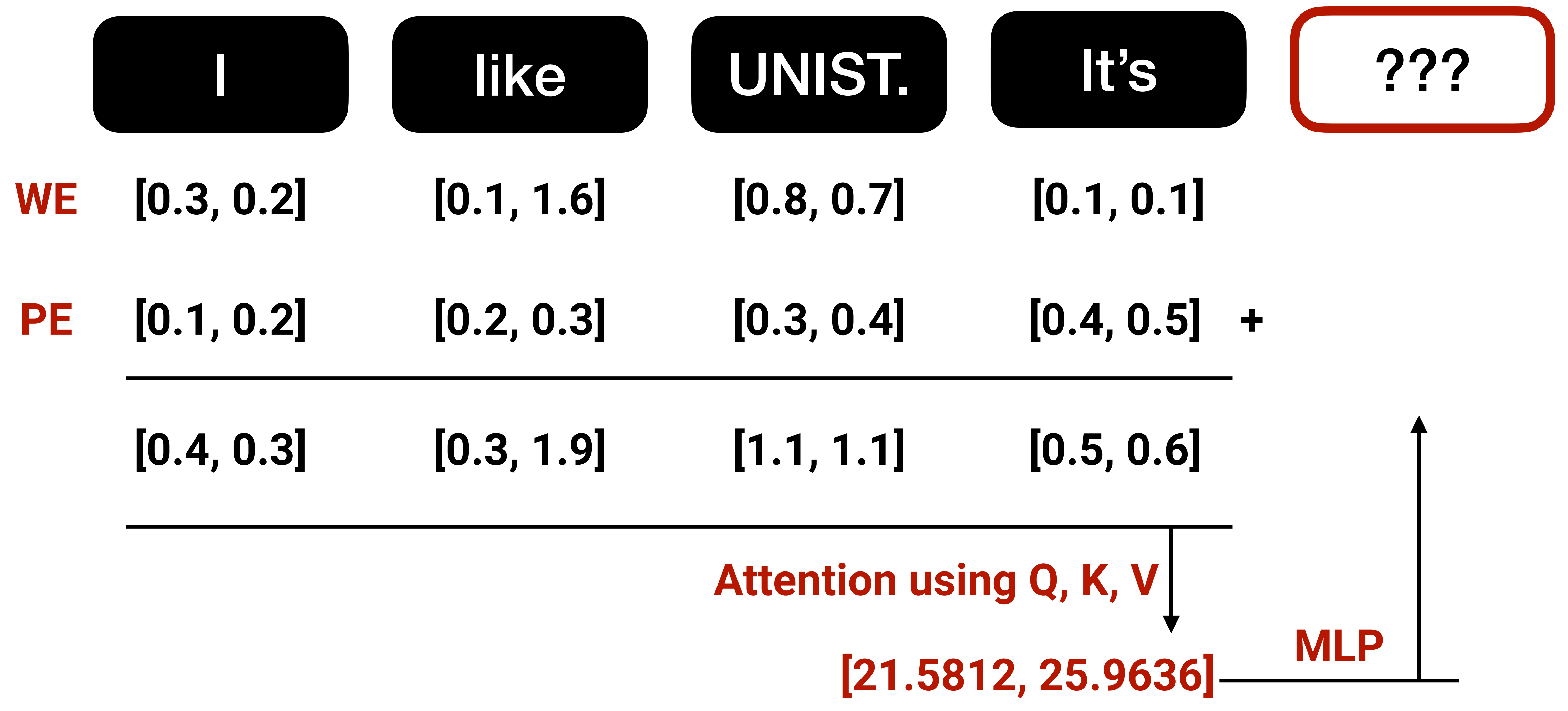
Transformer in a nutshell

- Initialize the word, position embeddings
  - Word (token)
- Compute context vector
  - Using attentions (query, key, value)
  - Parameters to learn: entries in Q, K, V matrices
- MLP for final prediction



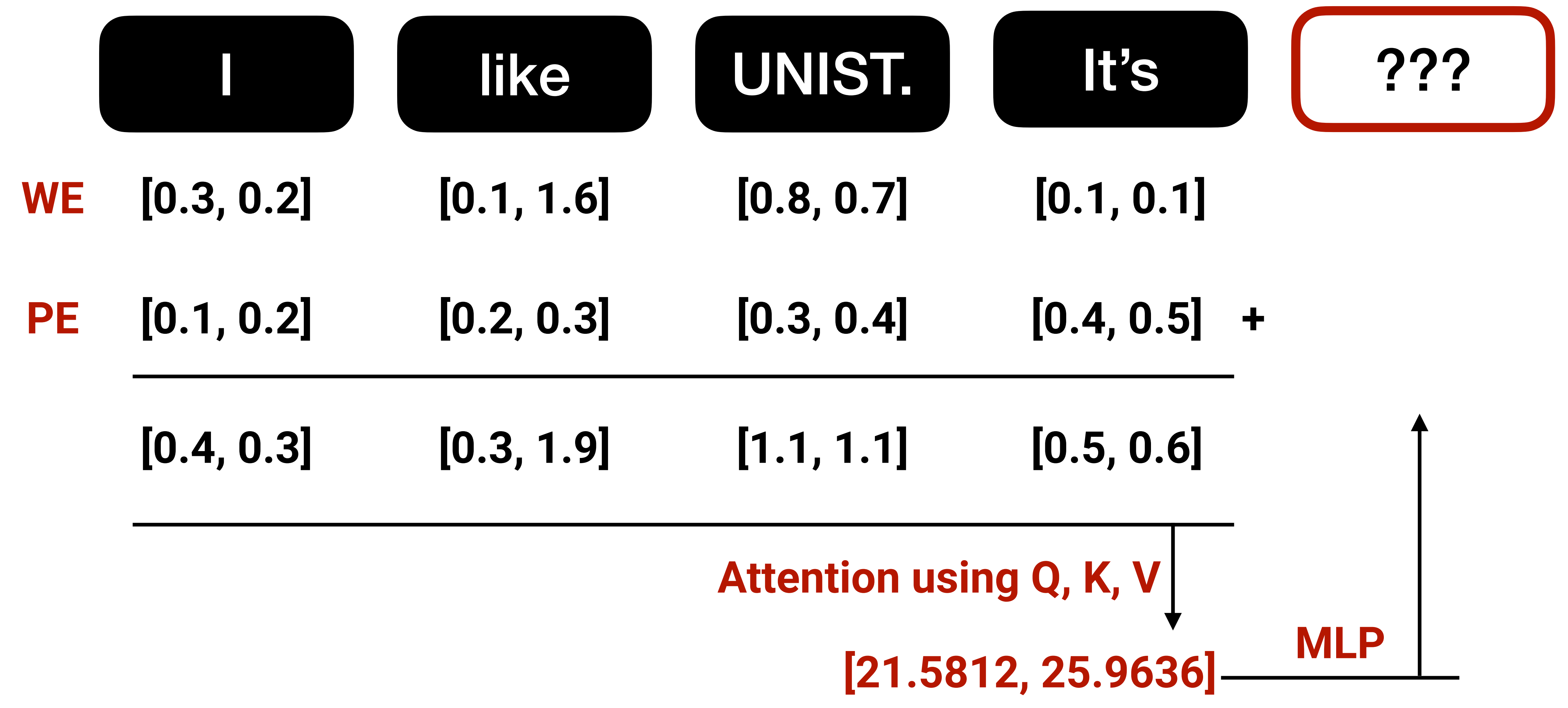
Multi-layer transformer

- Multi-layer
  - 원래는 4번째 단어에 대한 WPE가 [0.5, 0.6]
  - 두번째 레이어에서는 WPE가 [21.58, 25.96]이 됨
- Self-attention
  - Q, K, V의 대상이 모두 동일
- Cross-attention
  - Q의 대상이 K, V의 대상과 다름
- Causal attention
  - attention을 계산하는 방향이 왼쪽에서 오른쪽으로만
  - Bert: 양방향
- GPT는 self, causal attention을 사용



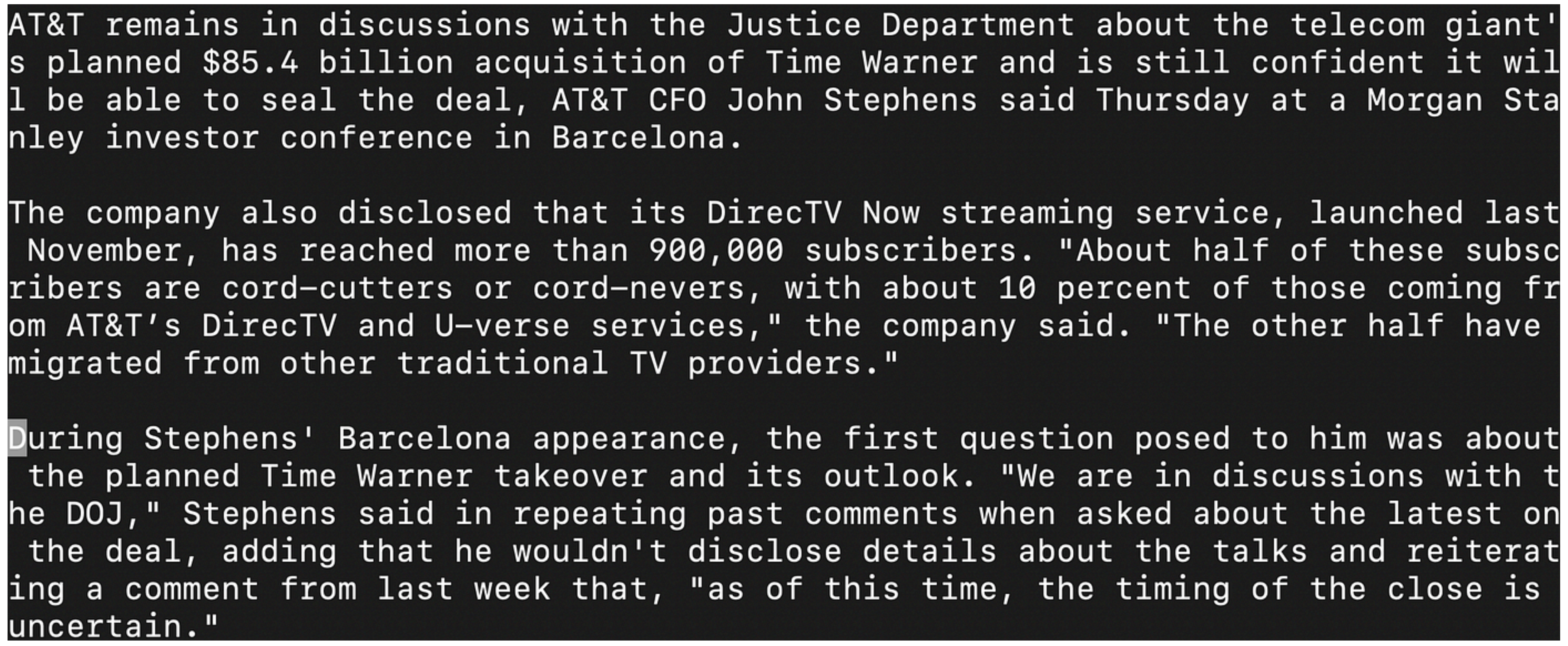
NanoGPT

- https://github.com/karpathy/nanoGPT
  - 실재로 파라미터 수 수억~수십억 짜리 GPT 모델을 ~300줄 이내의 Pytorch코드로 구현
  - train.py
    - 데이터 로딩, 모델 로딩, 멀티GPU 구현, optimizer 구현
  - model.py
    - attention, mlp layers



NanoGPT: Data

- OpenWebText
  - 매우 단순하게, 약 9B (90억)개의 토큰으로 이뤄짐.
  - 각 토큰은 0 ~ 50257이 숫자로 매핑
    - uint16으로 가능
  - 즉, data.bin은 0부터 50257까지의 숫자의 약 90억개의 배열 (array)
  - 1B (10억 토큰) -> 약 2GB (rule of thumb)
    - Uint32 면 4GB
    - 총 약 18GB
  - 이걸 train, validation set으로 나눔
    - Validation set은 매우매우 작음 (10MB 이내)

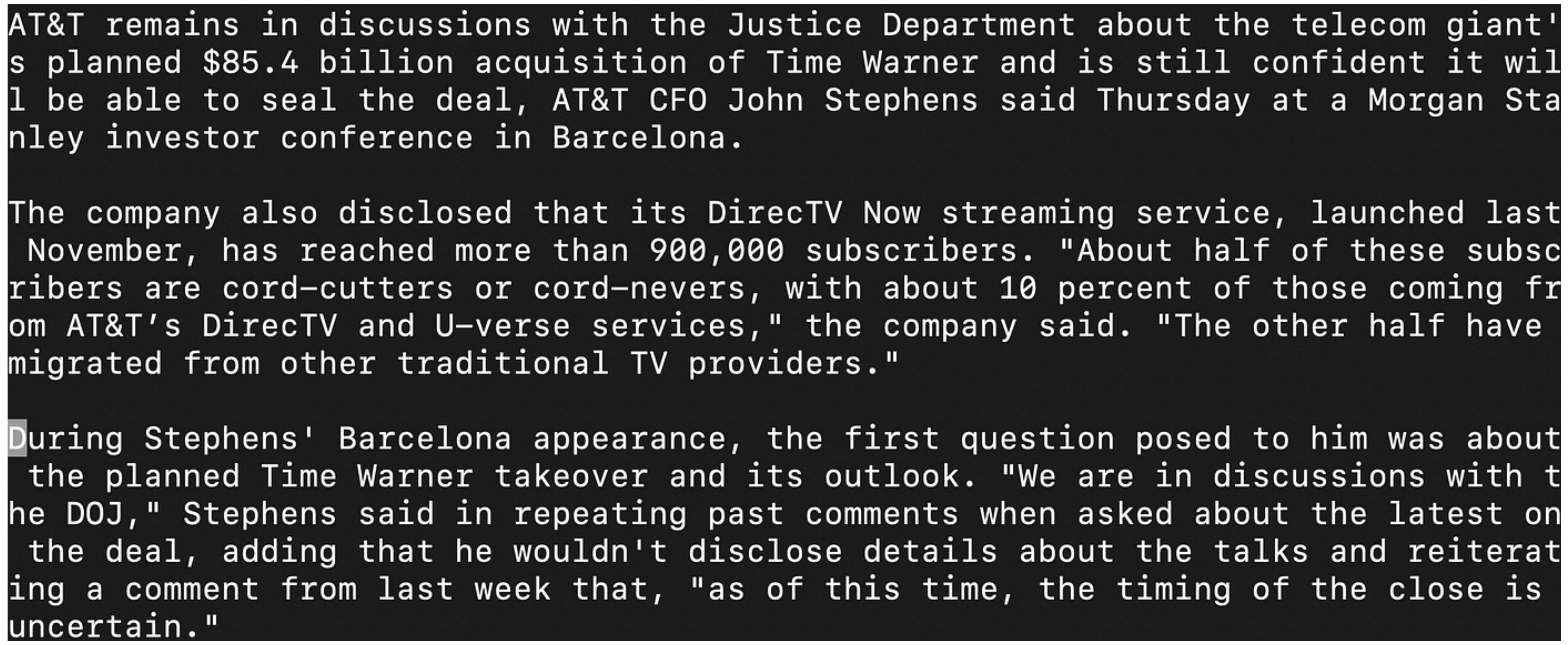


```
AT&T remains in discussions with the Justice Department about the telecom giant'
s planned $85.4 billion acquisition of Time Warner and is still confident it wil
l be able to seal the deal, AT&T CFO John Stephens said Thursday at a Morgan Sta
nley investor conference in Barcelona.

The company also disclosed that its DirecTV Now streaming service, launched last
 November, has reached more than 900,000 subscribers. "About half of these subsc
ribers are cord-cutters or cord-nevers, with about 10 percent of those coming fr
om AT&T's DirecTV and U-verse services," the company said. "The other half have
migrated from other traditional TV providers."

During Stephens' Barcelona appearance, the first question posed to him was about
 the planned Time Warner takeover and its outlook. "We are in discussions with t
he DOJ," Stephens said in repeating past comments when asked about the latest on
 the deal, adding that he wouldn't disclose details about the talks and reiterat
ing a comment from last week that, "as of this time, the timing of the close is
uncertain."
```

https://medium.com/datafabrica/mastering-context-extraction-for-fine-tuning-gpt-models-03c5b2f3dc02

NanoGPT: Data

- 트랜스포머가 하는일:
  - [나는 오늘 학교에 갔다]를 토큰화: [20, 31, 58, 29, 49]
  - 실제로는 openwebtext의 데이터에서 랜덤하게 context vector 사이즈 + 1 만큼의 배열 (array)를 샘플
  - [20, 31, 58, 29, ..., 84, 49]
  - 여기서, x = [20, 31, 58, ..., 84]
  - y = [31, 58, 29, ..., 49]
  - 즉, model(x) 와 y의 차이 (로스)를 최소화해서 model(x)가 y를 잘 예측하도록 하는 작업
  - How? By computing the context vectors!

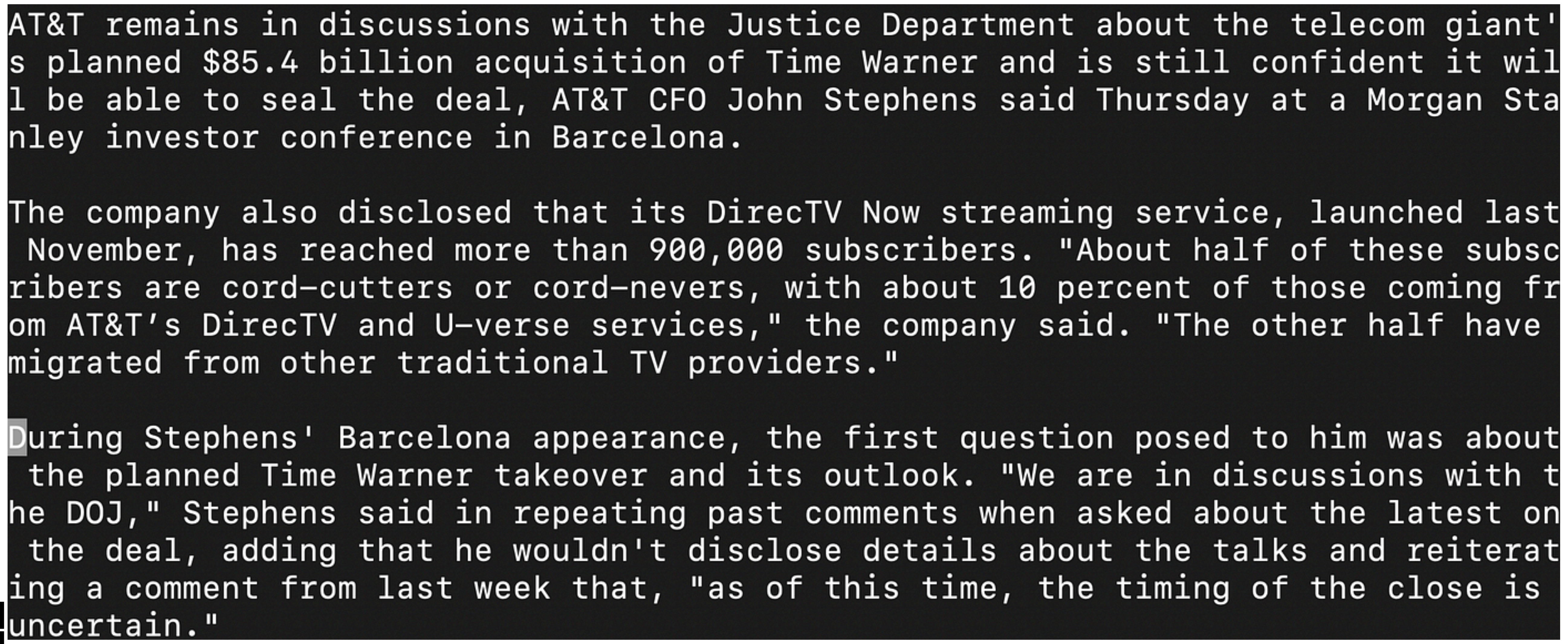


```
AT&T remains in discussions with the Justice Department about the telecom giant'
s planned $85.4 billion acquisition of Time Warner and is still confident it wil
l be able to seal the deal, AT&T CFO John Stephens said Thursday at a Morgan Sta
nley investor conference in Barcelona.

The company also disclosed that its DirecTV Now streaming service, launched last
 November, has reached more than 900,000 subscribers. "About half of these subsc
ribers are cord-cutters or cord-nevers, with about 10 percent of those coming fr
om AT&T's DirecTV and U-verse services," the company said. "The other half have
migrated from other traditional TV providers."

During Stephens' Barcelona appearance, the first question posed to him was about
 the planned Time Warner takeover and its outlook. "We are in discussions with t
he DOJ," Stephens said in repeating past comments when asked about the latest on
 the deal, adding that he wouldn't disclose details about the talks and reiterat
ing a comment from last week that, "as of this time, the timing of the close is
uncertain."
```

https://medium.com/datafabrica/mastering-context-extraction-for-fine-tuning-gpt-models-03c5b2f3dc02

NanoGPT: Data

- 트랜스포머가 하는일:
  - [나는 오늘 학교에 갔다]를 토큰화: [20, 31, 58, 29, 49]
  - 실제로는 openwebtext의 데이터에서 랜덤하게 context vector 사이즈 + 1 만큼의 배열 (array)를 샘플
  - [20, 31, 58, 29, …, 84, 49]
  - 여기서, x = [20, 31, 58, …, 84]
  - y = [31, 58, 29, …, 49]
  - 실제로는, batch size 만큼의 x, y를 한꺼번에 model에 통과시킴
  - x1 = x
  - x = [x1, x2, … xB]
  - 즉, model(x) 와 y의 차이 (로스)를 최소화해서 model(x)가 y를 잘 예측하도록 하는 작업
  - How? By computing the context vectors!

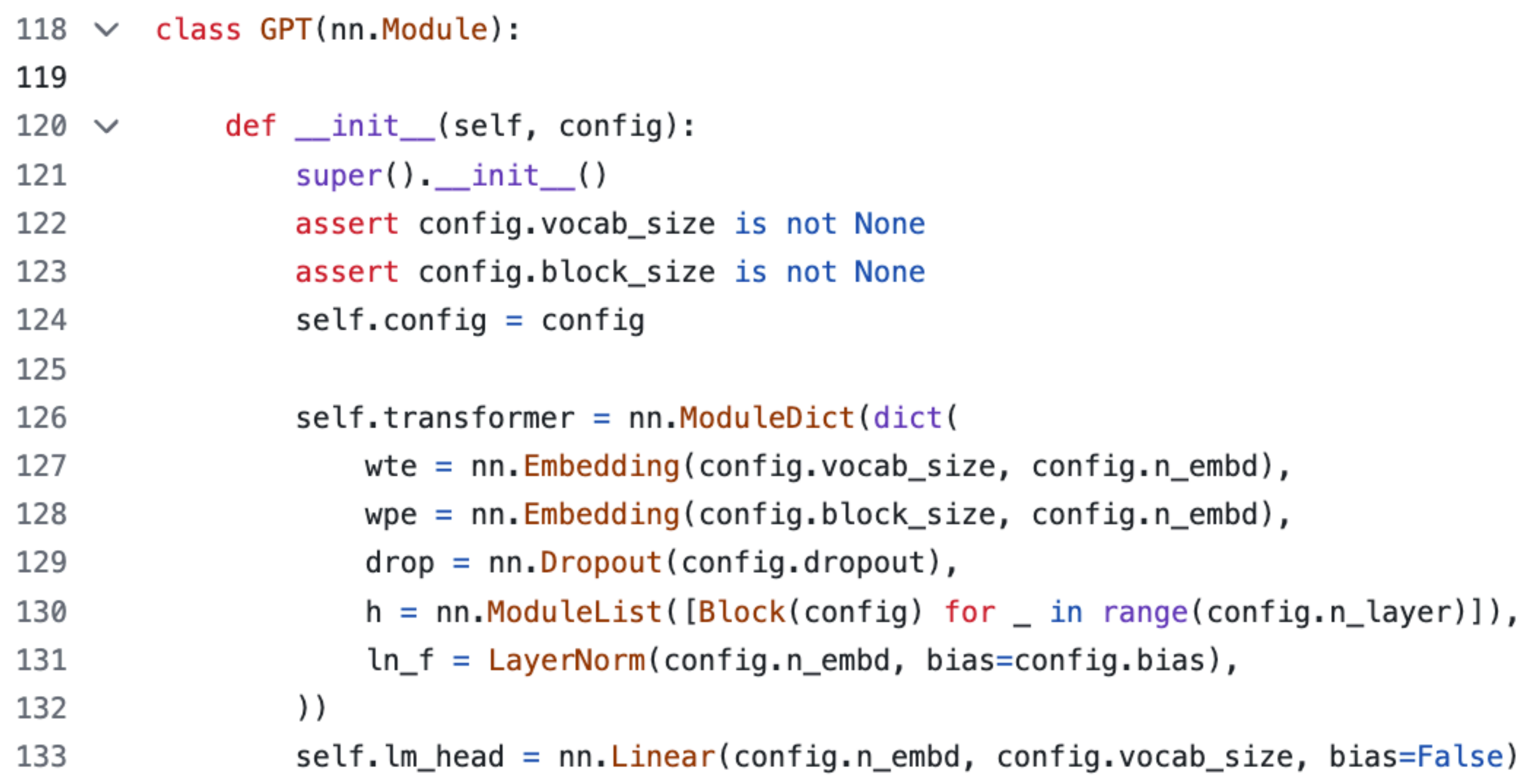


```
AT&T remains in discussions with the Justice Department about the telecom giant'
s planned $85.4 billion acquisition of Time Warner and is still confident it wil
l be able to seal the deal, AT&T CFO John Stephens said Thursday at a Morgan Sta
nley investor conference in Barcelona.

The company also disclosed that its DirecTV Now streaming service, launched last
 November, has reached more than 900,000 subscribers. "About half of these subsc
ribers are cord-cutters or cord-nevers, with about 10 percent of those coming fr
om AT&T's DirecTV and U-verse services," the company said. "The other half have
migrated from other traditional TV providers."

During Stephens' Barcelona appearance, the first question posed to him was about
 the planned Time Warner takeover and its outlook. "We are in discussions with t
he DOJ," Stephens said in repeating past comments when asked about the latest on
 the deal, adding that he wouldn't disclose details about the talks and reiterat
ing a comment from last week that, "as of this time, the timing of the close is
uncertain."
```

https://medium.com/datafabrica/mastering-context-extraction-for-fine-tuning-gpt-models-03c5b2f3dc02

NanoGPT: model.py

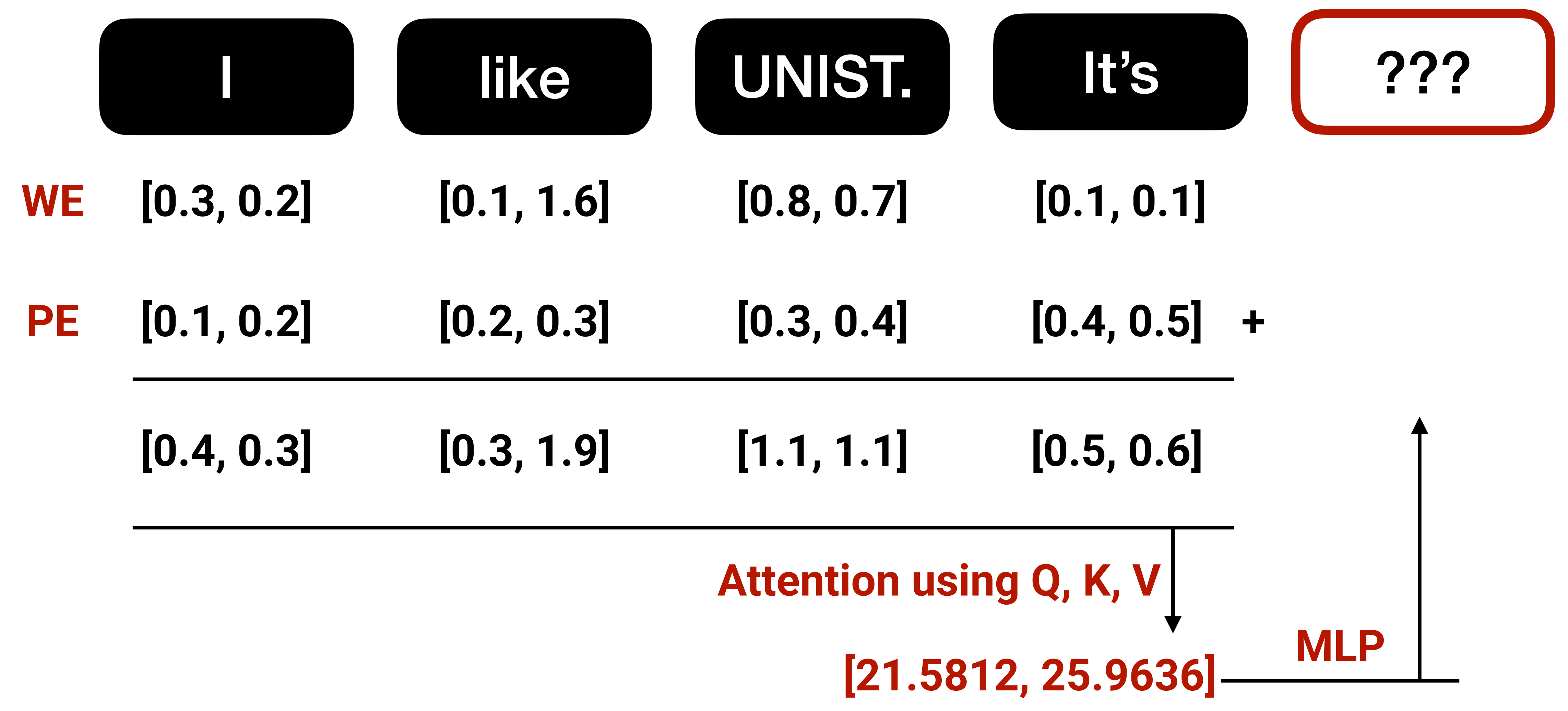
- Token, positional을 초기화
  - n_embd: WE, PE의 벡터 크기 (수백~수천)
  - vocab_size: 총 토큰 개수 (약 5만 ~ 10만)
  - block_size: max 컨텍스트 길이 (수백~수천)



```python
class GPT(nn.Module):

    def __init__(self, config):
        super().__init__()
        assert config.vocab_size is not None
        assert config.block_size is not None
        self.config = config

        self.transformer = nn.ModuleDict(dict(
            wte = nn.Embedding(config.vocab_size, config.n_embd),
            wpe = nn.Embedding(config.block_size, config.n_embd),
            drop = nn.Dropout(config.dropout),
            h = nn.ModuleList([Block(config) for _ in range(config.n_layer)]),
            ln_f = LayerNorm(config.n_embd, bias=config.bias),
        ))
        self.lm_head = nn.Linear(config.n_embd, config.vocab_size, bias=False)
```

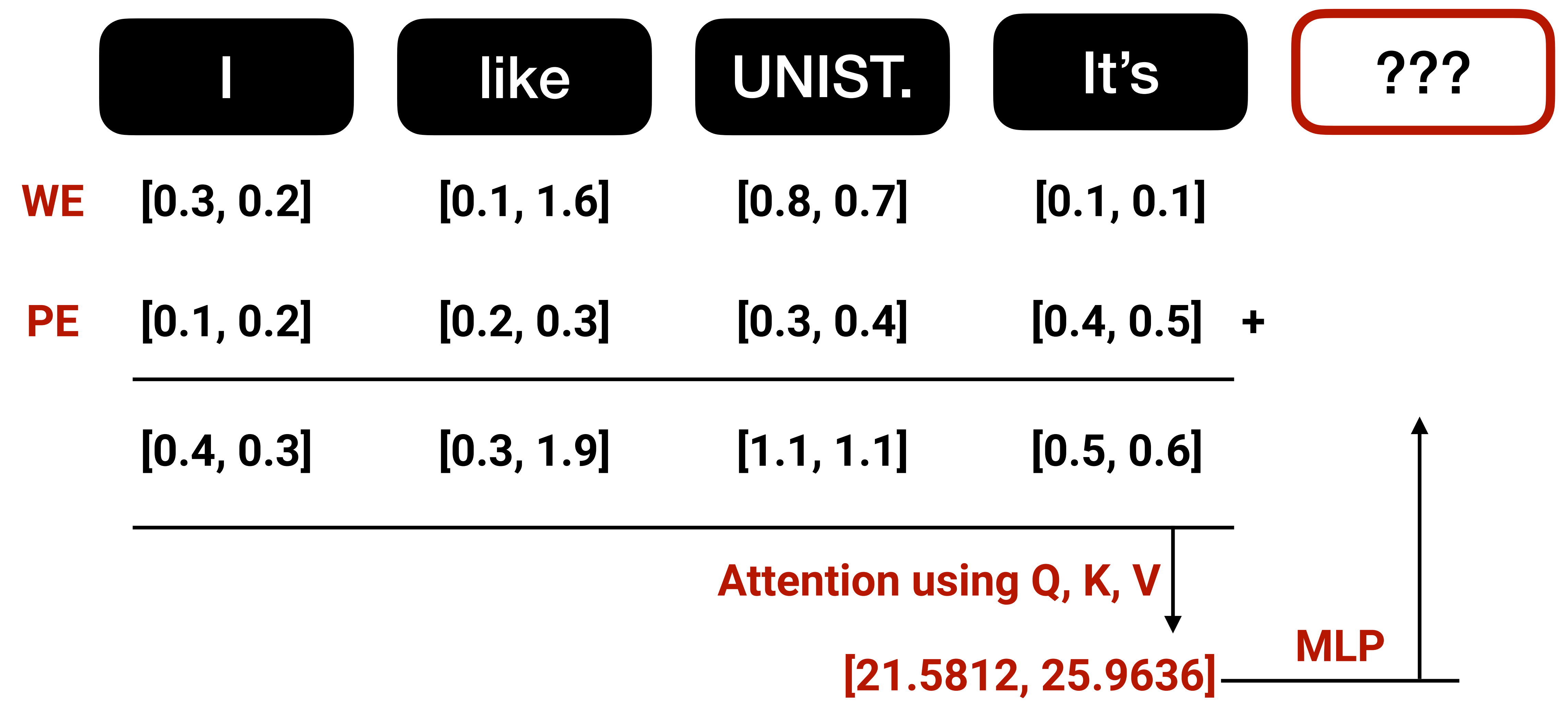
Transformer in a nutshell

- Initialize the word, position embeddings
    - Word (token)
- Compute context vector
    - Using attentions (query, key, value)
    - Parameters to learn: entries in Q, K, V matrices
- MLP for final prediction



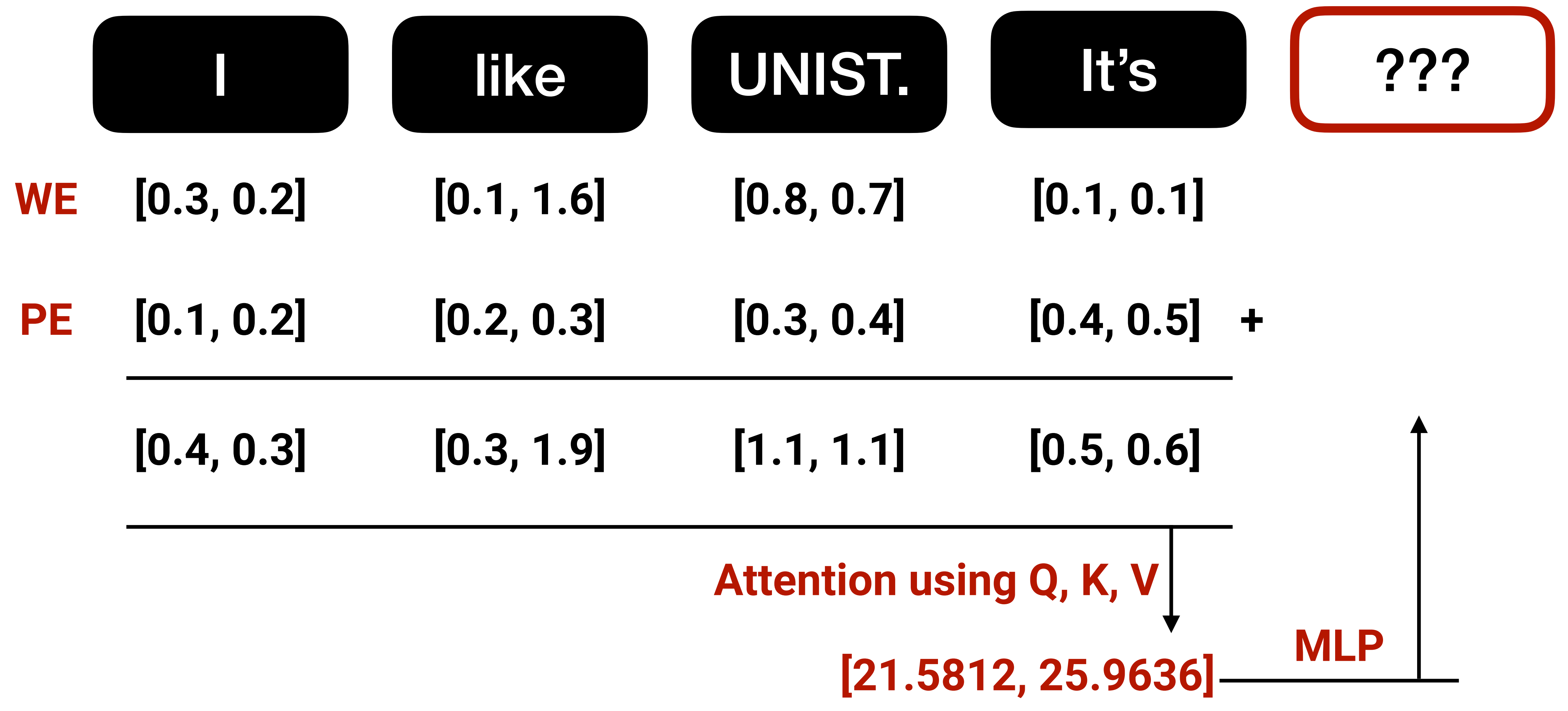
Multi-layer transformer

- Multi-layer
  - 원래는 4번째 단어에 대한 WPE가 [0.5, 0.6]
  - 두번째 레이어에서는 WPE가 [21.58, 25.96]이 됨
- Self-attention
  - Q, K, V의 대상이 모두 동일
- Cross-attention
  - Q의 대상이 K, V의 대상과 다름
- Causal attention
  - attention을 계산하는 방향이 왼쪽에서 오른쪽으로만
  - Bert: 양방향
- GPT는 self, causal attention을 사용



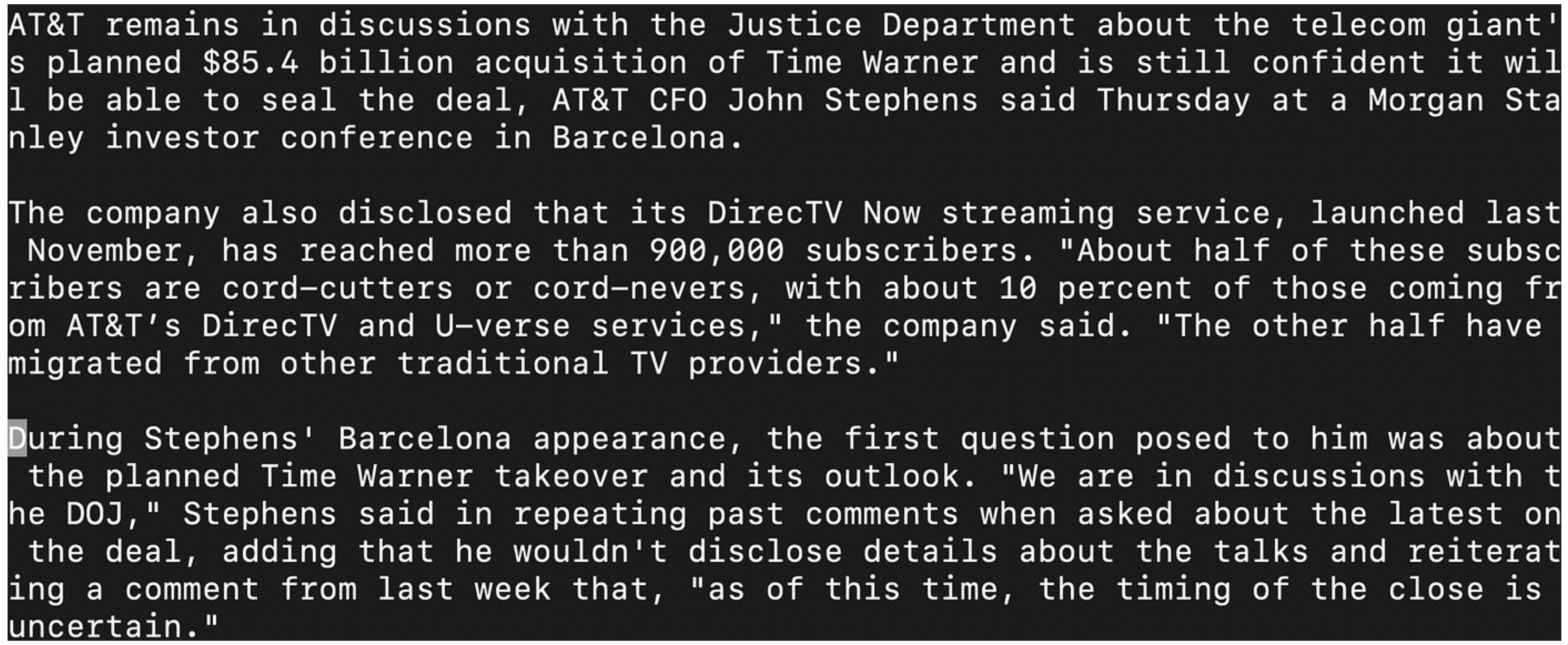
NanoGPT

- https://github.com/karpathy/nanoGPT
  - 실재로 파라미터 수 수억~수십억 짜리 GPT 모델을 ~300줄 이내의 Pytorch코드로 구현
  - train.py
    - 데이터 로딩, 모델 로딩, 멀티GPU 구현, optimizer 구현
  - model.py
    - attention, mlp layers



NanoGPT: Data

- OpenWebText
  - 매우 단순하게, 약 9B (90억)개의 토큰으로 이뤄짐.
  - 각 토큰은 0 ~ 50257이 숫자로 매핑
    - uint16으로 가능
  - 즉, data.bin은 0부터 50257까지의 숫자의 약 90억개의 배열 (array)
  - 1B (10억 토큰) -> 약 2GB (rule of thumb)
    - Uint32 면 4GB
    - 총 약 18GB
  - 이걸 train, validation set으로 나눔
    - Validation set은 매우매우 작음 (10MB 이내)

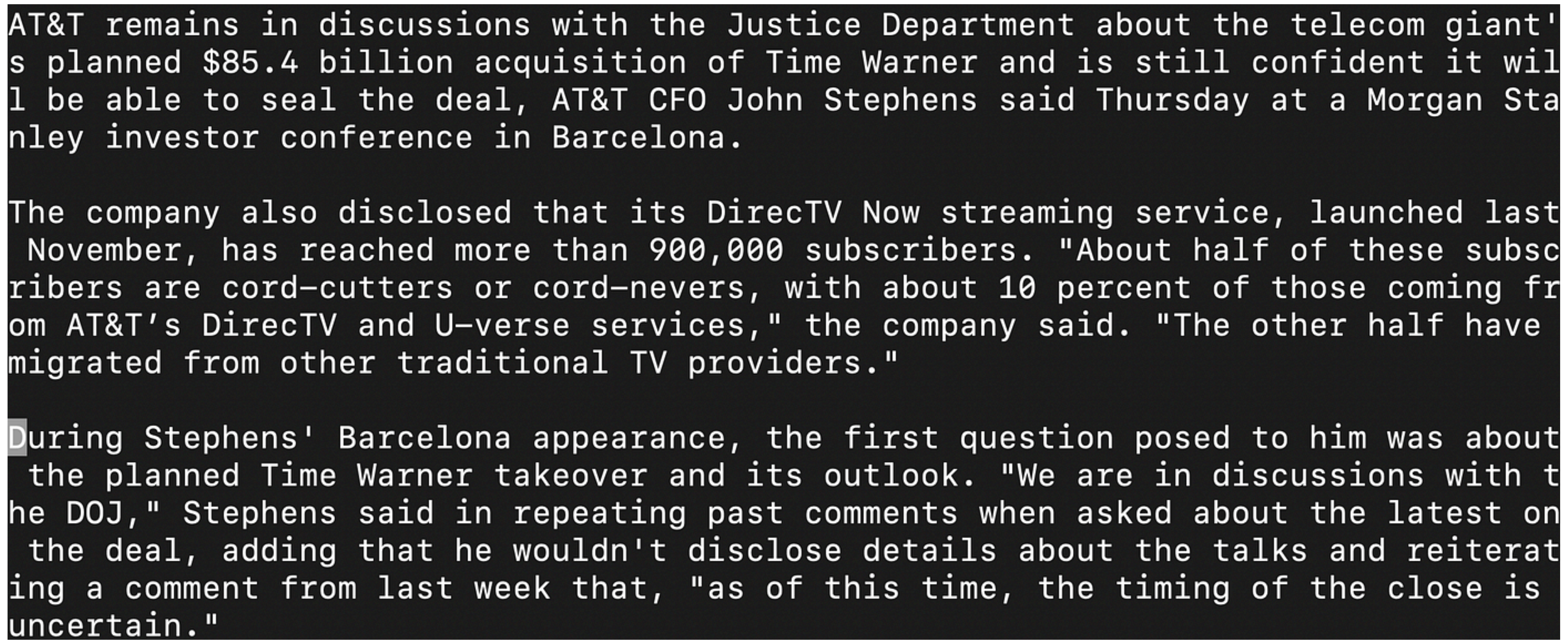


```
AT&T remains in discussions with the Justice Department about the telecom giant'
s planned $85.4 billion acquisition of Time Warner and is still confident it wil
l be able to seal the deal, AT&T CFO John Stephens said Thursday at a Morgan Sta
nley investor conference in Barcelona.

The company also disclosed that its DirecTV Now streaming service, launched last
 November, has reached more than 900,000 subscribers. "About half of these subsc
ribers are cord-cutters or cord-nevers, with about 10 percent of those coming fr
om AT&T's DirecTV and U-verse services," the company said. "The other half have
migrated from other traditional TV providers."

During Stephens' Barcelona appearance, the first question posed to him was about
 the planned Time Warner takeover and its outlook. "We are in discussions with t
he DOJ," Stephens said in repeating past comments when asked about the latest on
 the deal, adding that he wouldn't disclose details about the talks and reiterat
ing a comment from last week that, "as of this time, the timing of the close is
uncertain."
```

https://medium.com/datafabrica/mastering-context-extraction-for-fine-tuning-gpt-models-03c5b2f3dc02

NanoGPT: Data

- 트랜스포머가 하는일:
  - [나는 오늘 학교에 갔다]를 토큰화: [20, 31, 58, 29, 49]
  - 실제로는 openwebtext의 데이터에서 랜덤하게 context vector 사이즈 + 1 만큼의 배열 (array)를 샘플
  - [20, 31, 58, 29, ..., 84, 49]
  - 여기서, x = [20, 31, 58, ..., 84]
  - y = [31, 58, 29, ..., 49]
  - 즉, model(x) 와 y의 차이 (로스)를 최소화해서 model(x)가 y를 잘 예측하도록 하는 작업
  - How? By computing the context vectors!

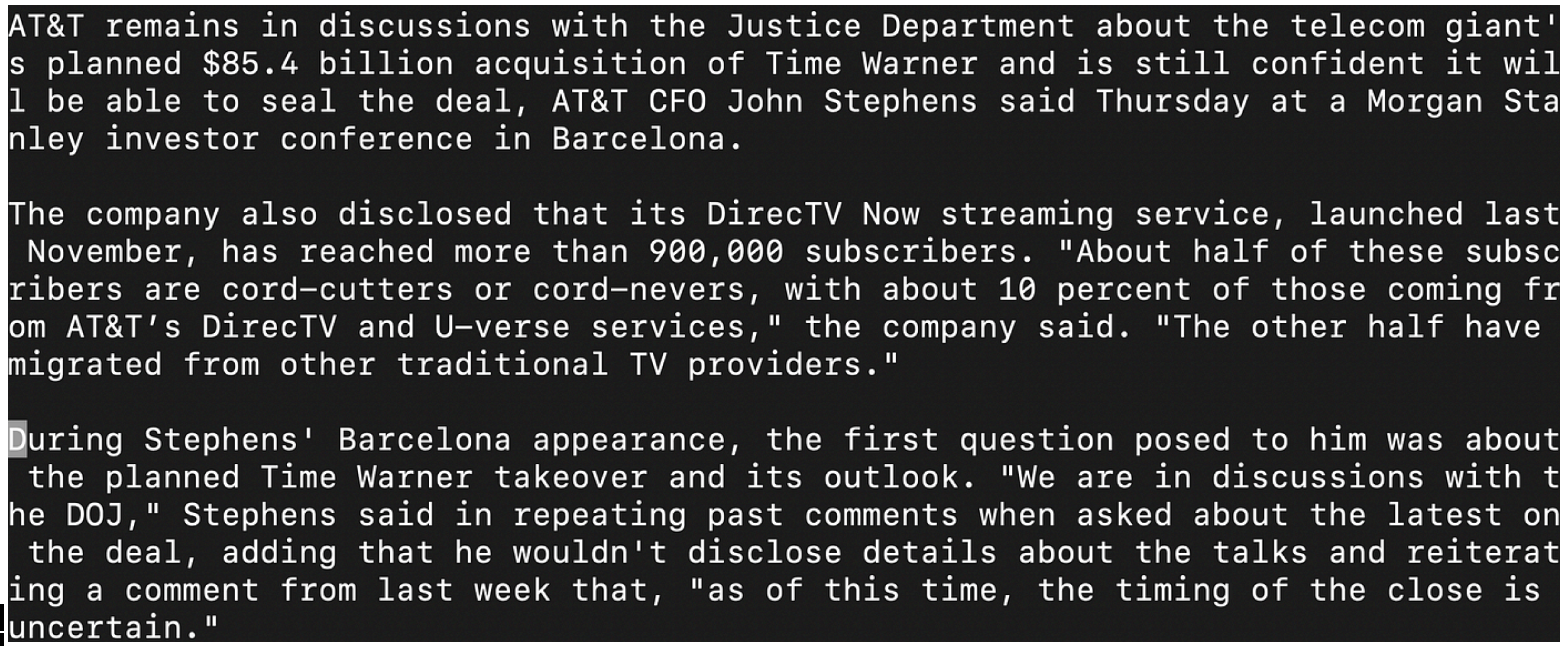


```
AT&T remains in discussions with the Justice Department about the telecom giant'
s planned $85.4 billion acquisition of Time Warner and is still confident it wil
l be able to seal the deal, AT&T CFO John Stephens said Thursday at a Morgan Sta
nley investor conference in Barcelona.

The company also disclosed that its DirecTV Now streaming service, launched last
 November, has reached more than 900,000 subscribers. "About half of these subsc
ribers are cord-cutters or cord-nevers, with about 10 percent of those coming fr
om AT&T's DirecTV and U-verse services," the company said. "The other half have
migrated from other traditional TV providers."

During Stephens' Barcelona appearance, the first question posed to him was about
 the planned Time Warner takeover and its outlook. "We are in discussions with t
he DOJ," Stephens said in repeating past comments when asked about the latest on
 the deal, adding that he wouldn't disclose details about the talks and reiterat
ing a comment from last week that, "as of this time, the timing of the close is
uncertain."
```

https://medium.com/datafabrica/mastering-context-extraction-for-fine-tuning-gpt-models-03c5b2f3dc02

NanoGPT: Data

- 트랜스포머가 하는일:
  - [나는 오늘 학교에 갔다]를 토큰화: [20, 31, 58, 29, 49]
  - 실제로는 openwebtext의 데이터에서 랜덤하게 context vector 사이즈 + 1 만큼의 배열 (array)를 샘플
  - [20, 31, 58, 29, …, 84, 49]
  - 여기서, x = [20, 31, 58, …, 84]
  - y = [31, 58, 29, …, 49]
  - 실제로는, batch size 만큼의 x, y를 한꺼번에 model에 통과시킴
  - x1 = x
  - x = [x1, x2, … xB]
  - 즉, model(x) 와 y의 차이 (로스)를 최소화해서 model(x)가 y를 잘 예측하도록 하는 작업
  - How? By computing the context vectors!

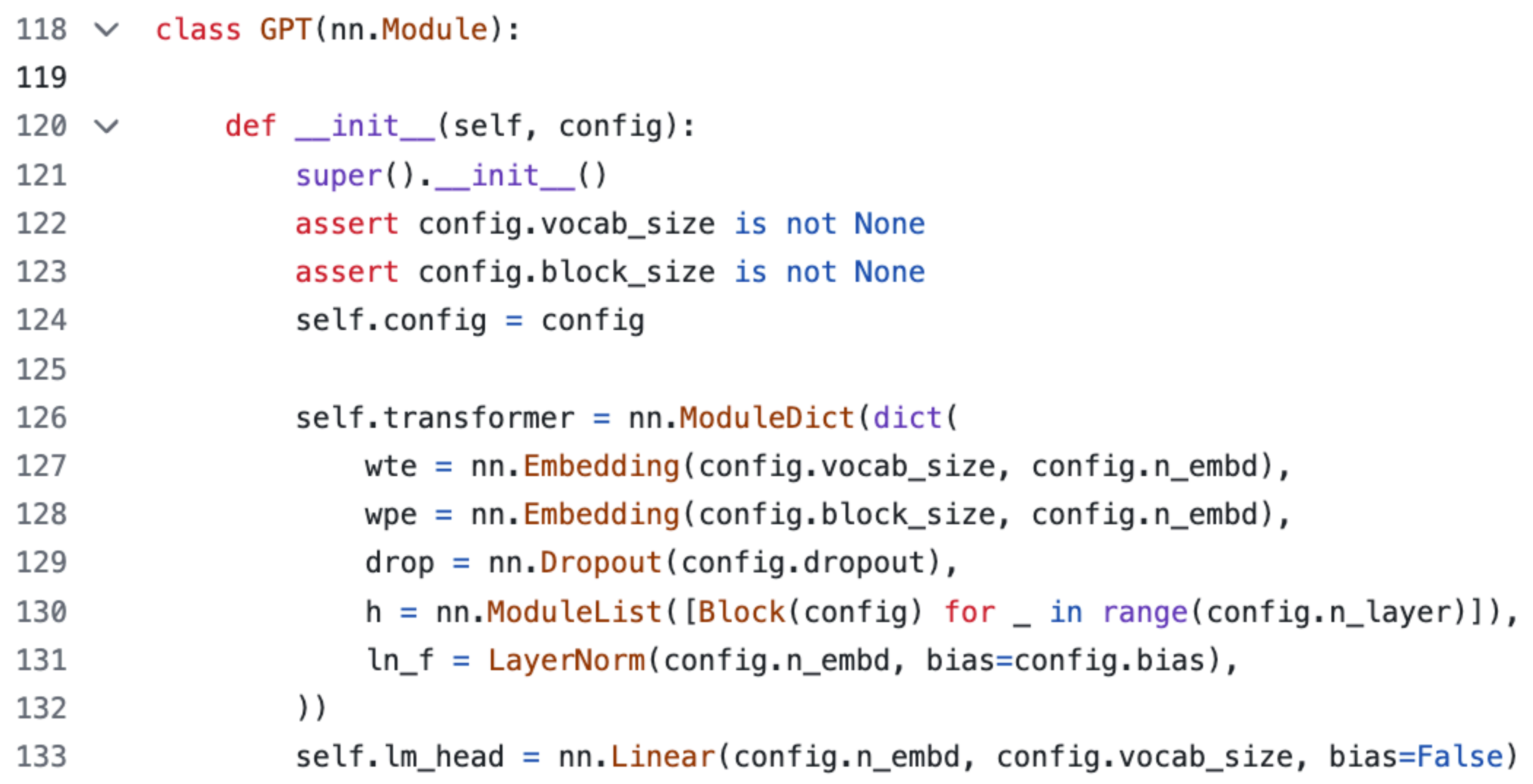


```
AT&T remains in discussions with the Justice Department about the telecom giant'
s planned $85.4 billion acquisition of Time Warner and is still confident it wil
l be able to seal the deal, AT&T CFO John Stephens said Thursday at a Morgan Sta
nley investor conference in Barcelona.

The company also disclosed that its DirecTV Now streaming service, launched last
 November, has reached more than 900,000 subscribers. "About half of these subsc
ribers are cord-cutters or cord-nevers, with about 10 percent of those coming fr
om AT&T's DirecTV and U-verse services," the company said. "The other half have
migrated from other traditional TV providers."

During Stephens' Barcelona appearance, the first question posed to him was about
 the planned Time Warner takeover and its outlook. "We are in discussions with t
he DOJ," Stephens said in repeating past comments when asked about the latest on
 the deal, adding that he wouldn't disclose details about the talks and reiterat
ing a comment from last week that, "as of this time, the timing of the close is
uncertain."
```

https://medium.com/datafabrica/mastering-context-extraction-for-fine-tuning-gpt-models-03c5b2f3dc02

NanoGPT: model.py

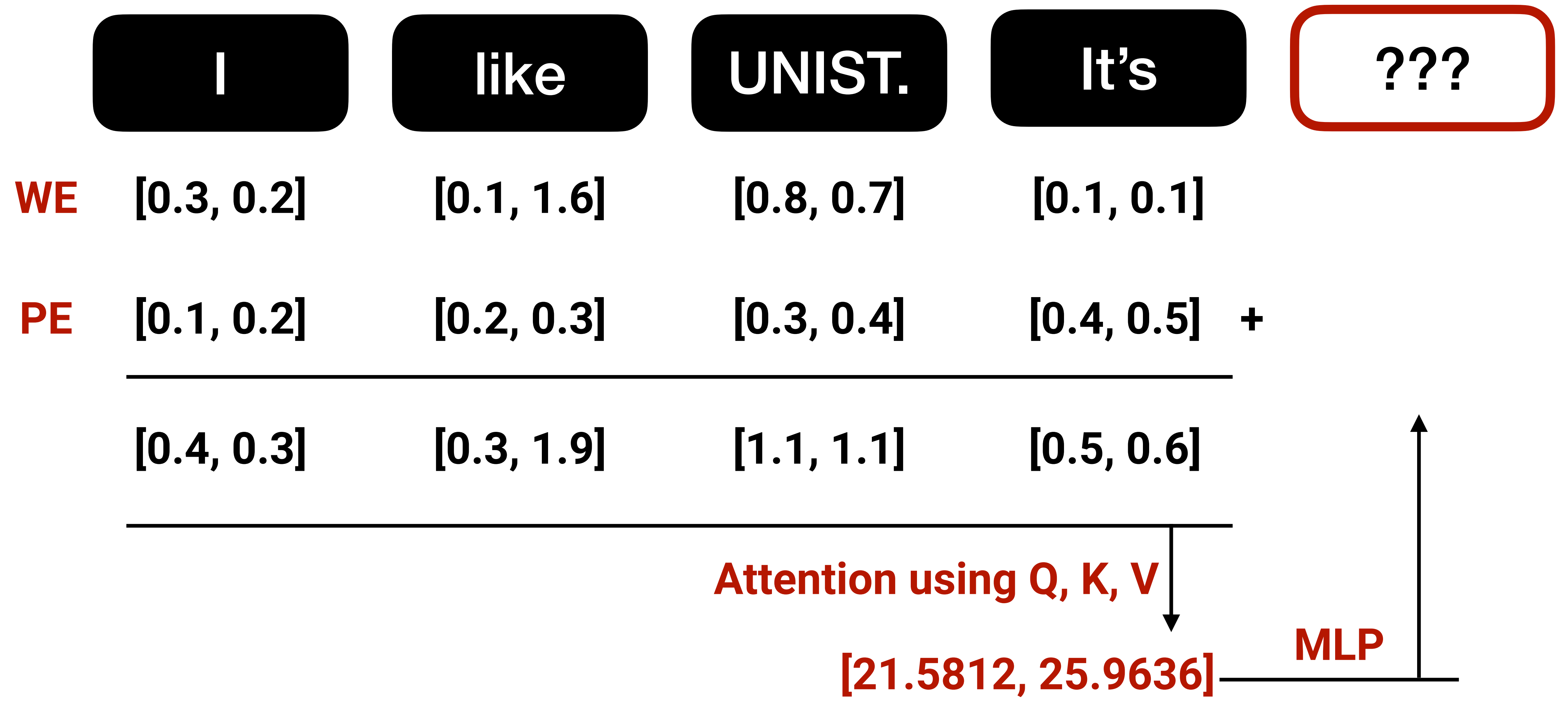
- Token, positional을 초기화
  - n_embd: WE, PE의 벡터 크기 (수백~수천)
  - vocab_size: 총 토큰 개수 (약 5만 ~ 10만)
  - block_size: max 컨텍스트 길이 (수백~수천)



```
class GPT(nn.Module):

    def __init__(self, config):
        super().__init__()
        assert config.vocab_size is not None
        assert config.block_size is not None
        self.config = config

        self.transformer = nn.ModuleDict(dict(
            wte = nn.Embedding(config.vocab_size, config.n_embd),
            wpe = nn.Embedding(config.block_size, config.n_embd),
            drop = nn.Dropout(config.dropout),
            h = nn.ModuleList([Block(config) for _ in range(config.n_layer)]),
            ln_f = LayerNorm(config.n_embd, bias=config.bias),
        ))
        self.lm_head = nn.Linear(config.n_embd, config.vocab_size, bias=False)
```

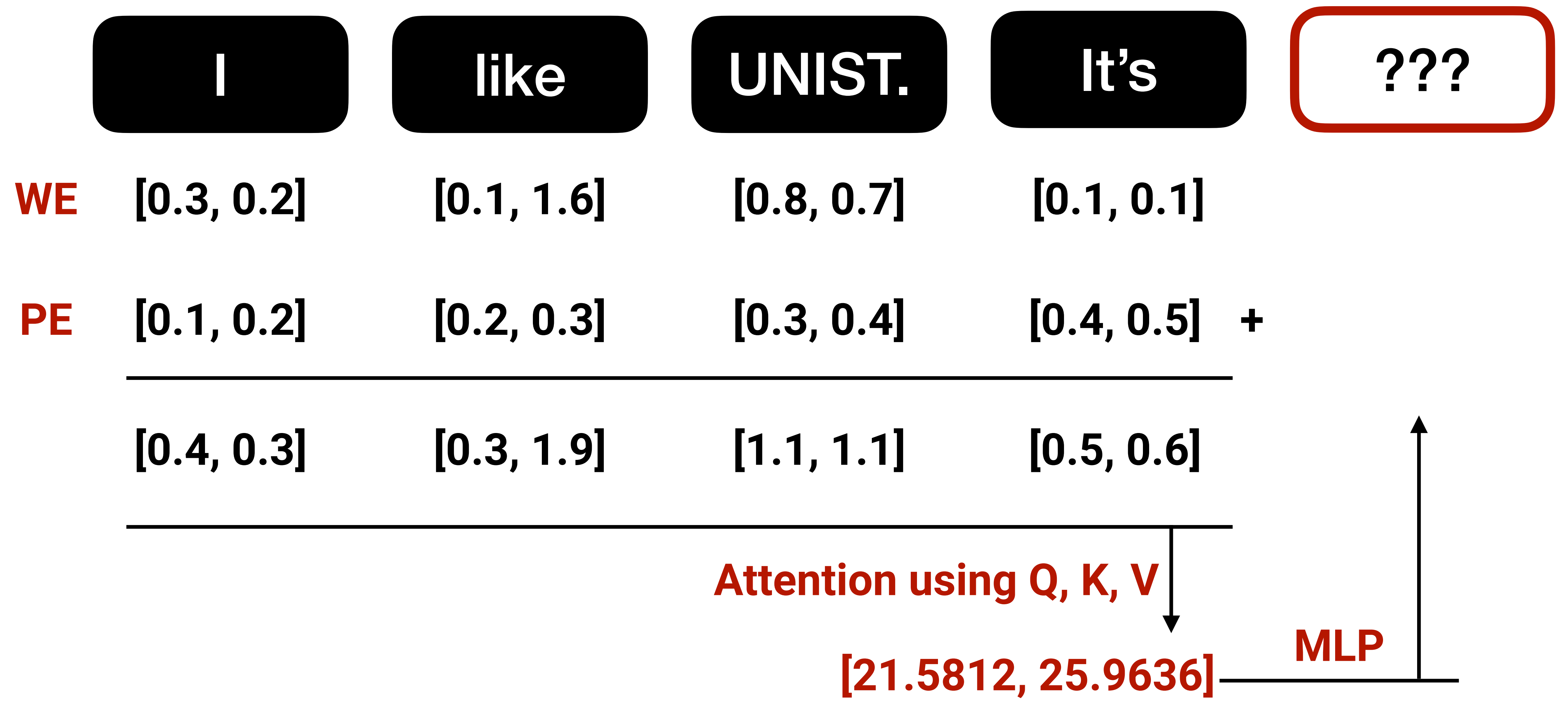
Transformer in a nutshell

- Initialize the word, position embeddings
  - Word (token)
- Compute context vector
  - Using attentions (query, key, value)
  - Parameters to learn: entries in Q, K, V matrices
- MLP for final prediction



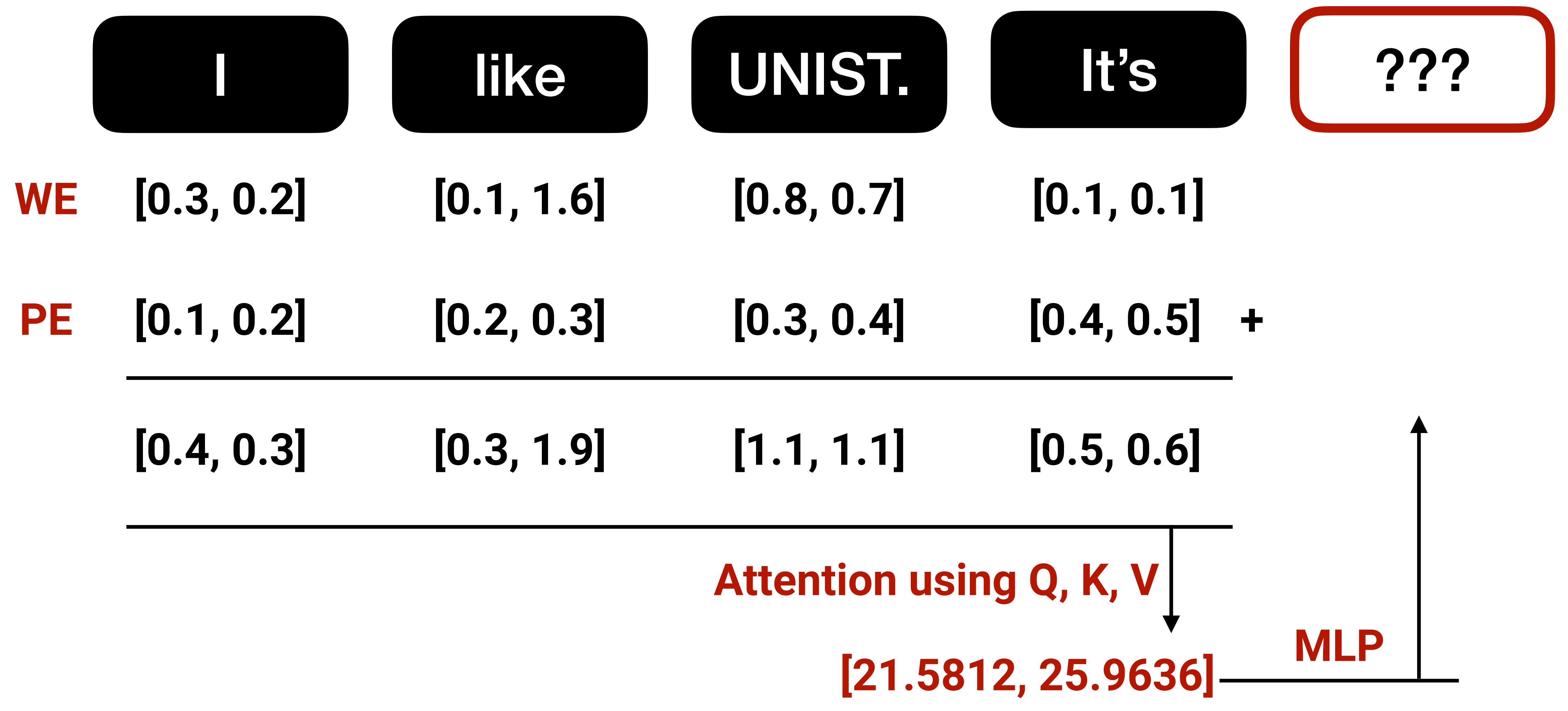
Multi-layer transformer

- Multi-layer
  - 원래는 4번째 단어에 대한 WPE가 [0.5, 0.6]
  - 두번째 레이어에서는 WPE가 [21.58, 25.96]이 됨
- Self-attention
  - Q, K, V의 대상이 모두 동일
- Cross-attention
  - Q의 대상이 K, V의 대상과 다름
- Causal attention
  - attention을 계산하는 방향이 왼쪽에서 오른쪽으로만
  - Bert: 양방향
- GPT는 self, causal attention을 사용



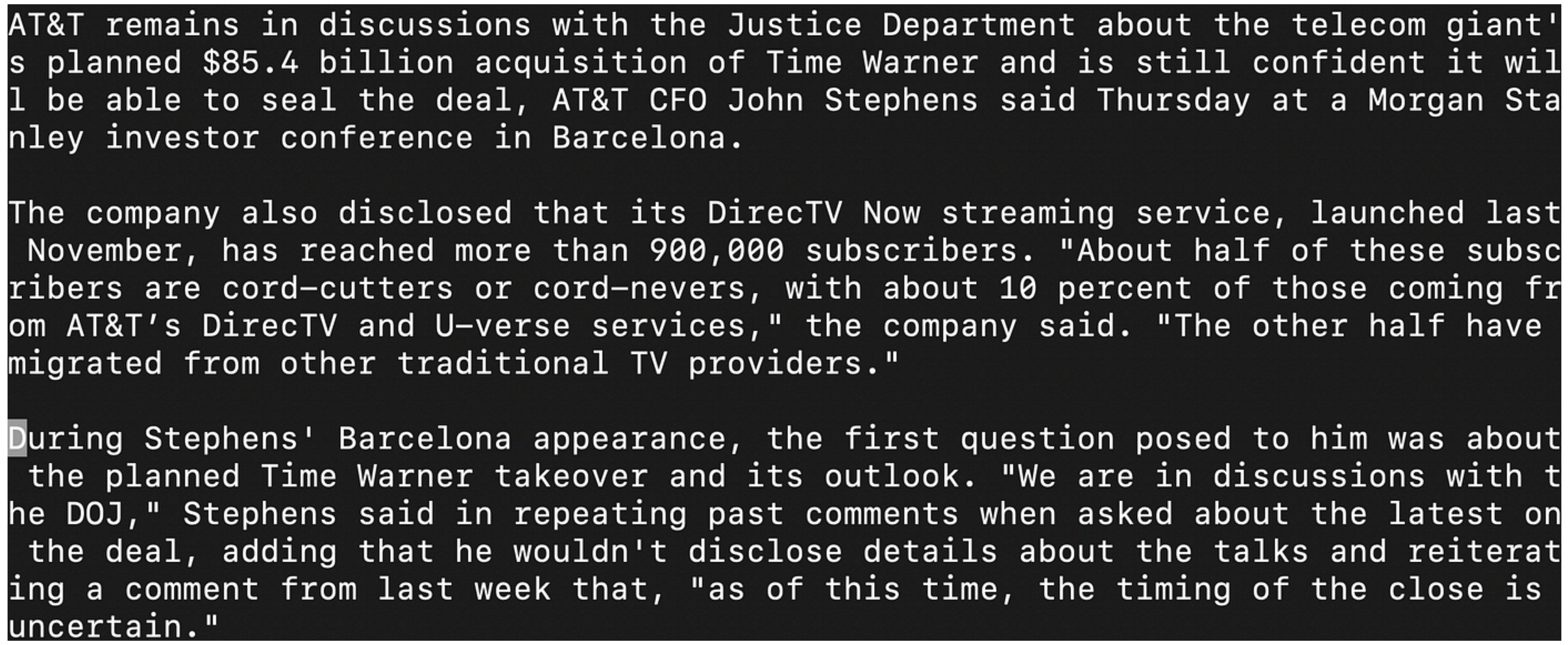
NanoGPT

- https://github.com/karpathy/nanoGPT
  - 실재로 파라미터 수 수억~수십억 짜리 GPT 모델을 ~300줄 이내의 Pytorch코드로 구현
  - train.py
    - 데이터 로딩, 모델 로딩, 멀티GPU 구현, optimizer 구현
  - model.py
    - attention, mlp layers



NanoGPT: Data

- OpenWebText
  - 매우 단순하게, 약 9B (90억)개의 토큰으로 이뤄짐.
  - 각 토큰은 0 ~ 50257이 숫자로 매핑
    - uint16으로 가능
  - 즉, data.bin은 0부터 50257까지의 숫자의 약 90억개의 배열 (array)
  - 1B (10억 토큰) -> 약 2GB (rule of thumb)
    - Uint32 면 4GB
    - 총 약 18GB
  - 이걸 train, validation set으로 나눔
    - Validation set은 매우매우 작음 (10MB 이내)

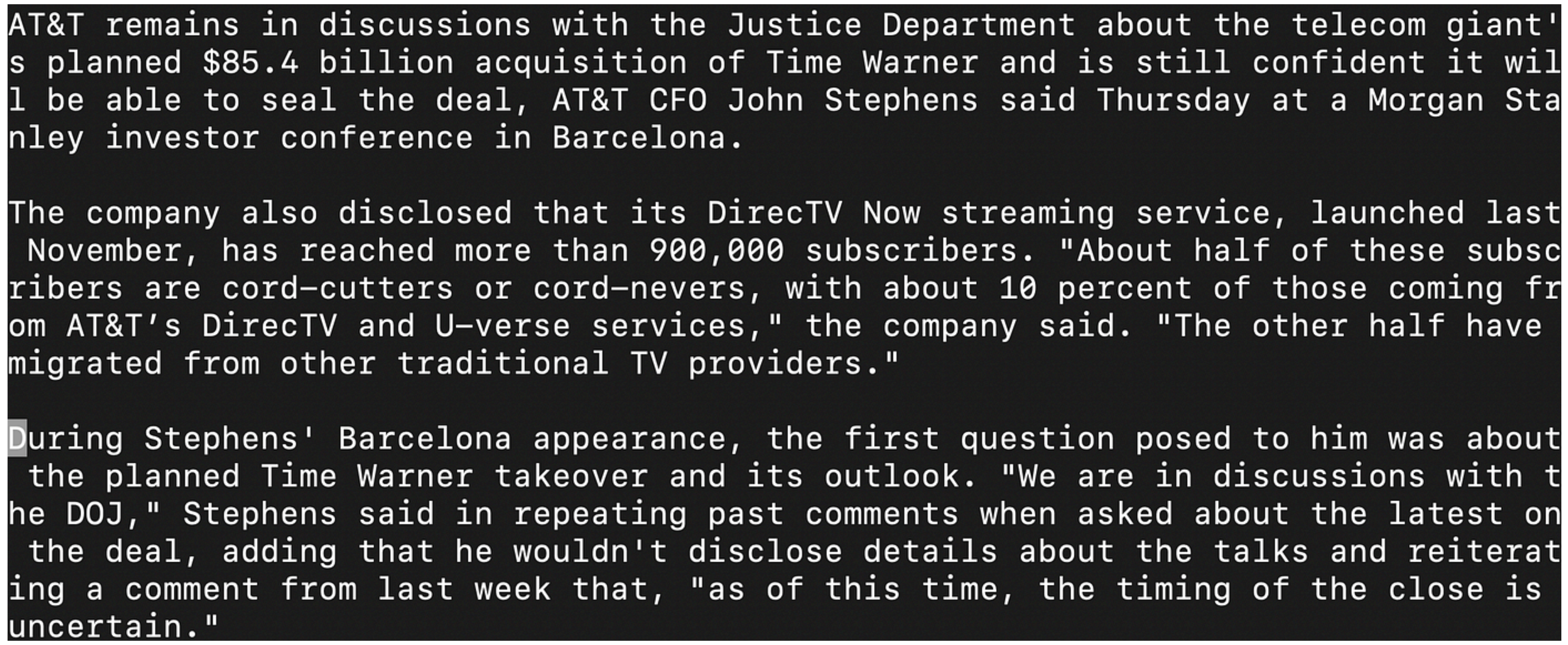


```
AT&T remains in discussions with the Justice Department about the telecom giant'
s planned $85.4 billion acquisition of Time Warner and is still confident it wil
l be able to seal the deal, AT&T CFO John Stephens said Thursday at a Morgan Sta
nley investor conference in Barcelona.

The company also disclosed that its DirecTV Now streaming service, launched last
 November, has reached more than 900,000 subscribers. "About half of these subsc
ribers are cord-cutters or cord-nevers, with about 10 percent of those coming fr
om AT&T's DirecTV and U-verse services," the company said. "The other half have
migrated from other traditional TV providers."

During Stephens' Barcelona appearance, the first question posed to him was about
 the planned Time Warner takeover and its outlook. "We are in discussions with t
he DOJ," Stephens said in repeating past comments when asked about the latest on
 the deal, adding that he wouldn't disclose details about the talks and reiterat
ing a comment from last week that, "as of this time, the timing of the close is
uncertain."
```

https://medium.com/datafabrica/mastering-context-extraction-for-fine-tuning-gpt-models-03c5b2f3dc02

NanoGPT: Data

- 트랜스포머가 하는일:
  - [나는 오늘 학교에 갔다]를 토큰화: [20, 31, 58, 29, 49]
  - 실제로는 openwebtext의 데이터에서 랜덤하게 context vector 사이즈 + 1 만큼의 배열 (array)를 샘플
  - [20, 31, 58, 29, ..., 84, 49]
  - 여기서, x = [20, 31, 58, ..., 84]
  - y = [31, 58, 29, ..., 49]
  - 즉, model(x) 와 y의 차이 (로스)를 최소화해서 model(x)가 y를 잘 예측하도록 하는 작업
  - How? By computing the context vectors!

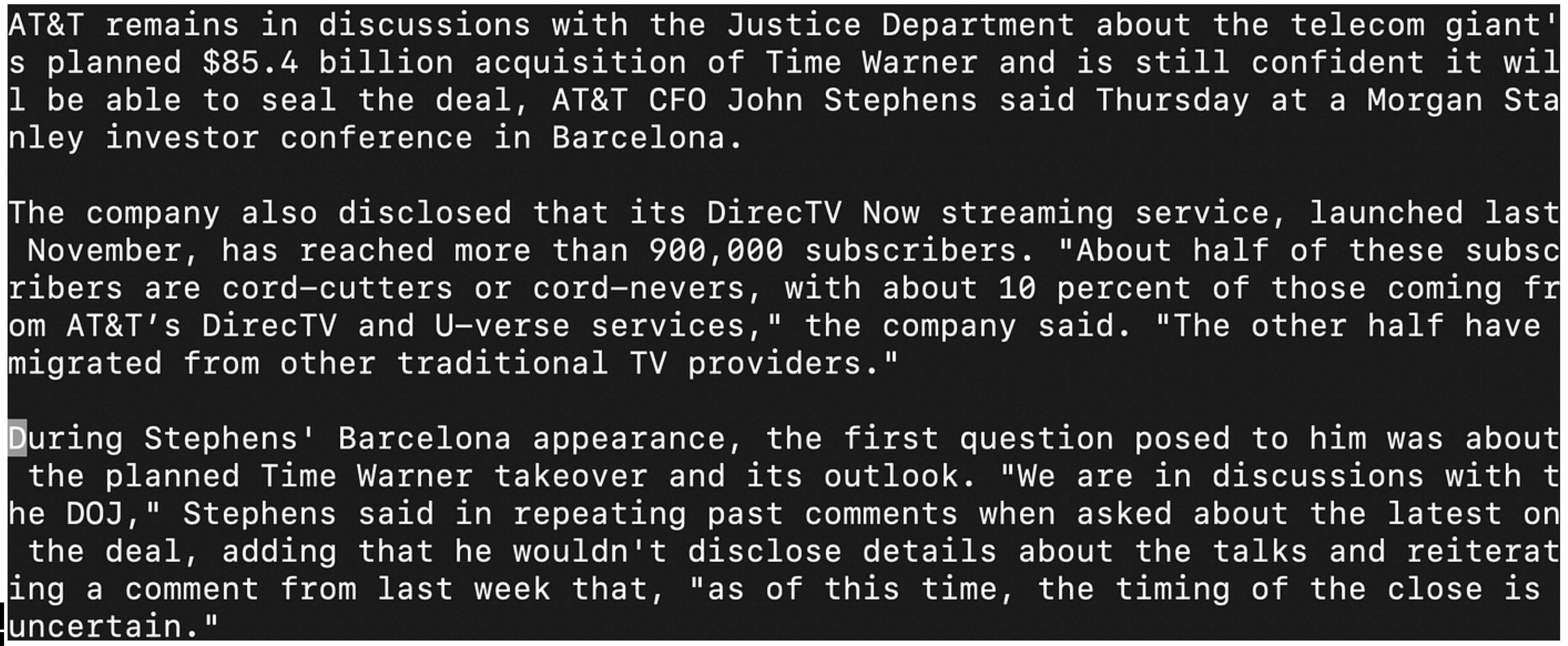


```
AT&T remains in discussions with the Justice Department about the telecom giant'
s planned $85.4 billion acquisition of Time Warner and is still confident it wil
l be able to seal the deal, AT&T CFO John Stephens said Thursday at a Morgan Sta
nley investor conference in Barcelona.

The company also disclosed that its DirecTV Now streaming service, launched last
 November, has reached more than 900,000 subscribers. "About half of these subsc
ribers are cord-cutters or cord-nevers, with about 10 percent of those coming fr
om AT&T's DirecTV and U-verse services," the company said. "The other half have
migrated from other traditional TV providers."

During Stephens' Barcelona appearance, the first question posed to him was about
 the planned Time Warner takeover and its outlook. "We are in discussions with t
he DOJ," Stephens said in repeating past comments when asked about the latest on
 the deal, adding that he wouldn't disclose details about the talks and reiterat
ing a comment from last week that, "as of this time, the timing of the close is
uncertain."
```

https://medium.com/datafabrica/mastering-context-extraction-for-fine-tuning-gpt-models-03c5b2f3dc02

NanoGPT: Data

- 트랜스포머가 하는일:
    - [나는 오늘 학교에 갔다]를 토큰화: [20, 31, 58, 29, 49]
    - 실제로는 openwebtext의 데이터에서 랜덤하게 context vector 사이즈 + 1 만큼의 배열 (array)를 샘플
    - [20, 31, 58, 29, …, 84, 49]
    - 여기서, x = [20, 31, 58, …, 84]
    - y = [31, 58, 29, …, 49]
    - 실제로는, batch size 만큼의 x, y를 한꺼번에 model에 통과시킴
    - x1 = x
    - x = [x1, x2, … xB]
    - 즉, model(x) 와 y의 차이 (로스)를 최소화해서 model(x)가 y를 잘 예측하도록 하는 작업
    - How? By computing the context vectors!



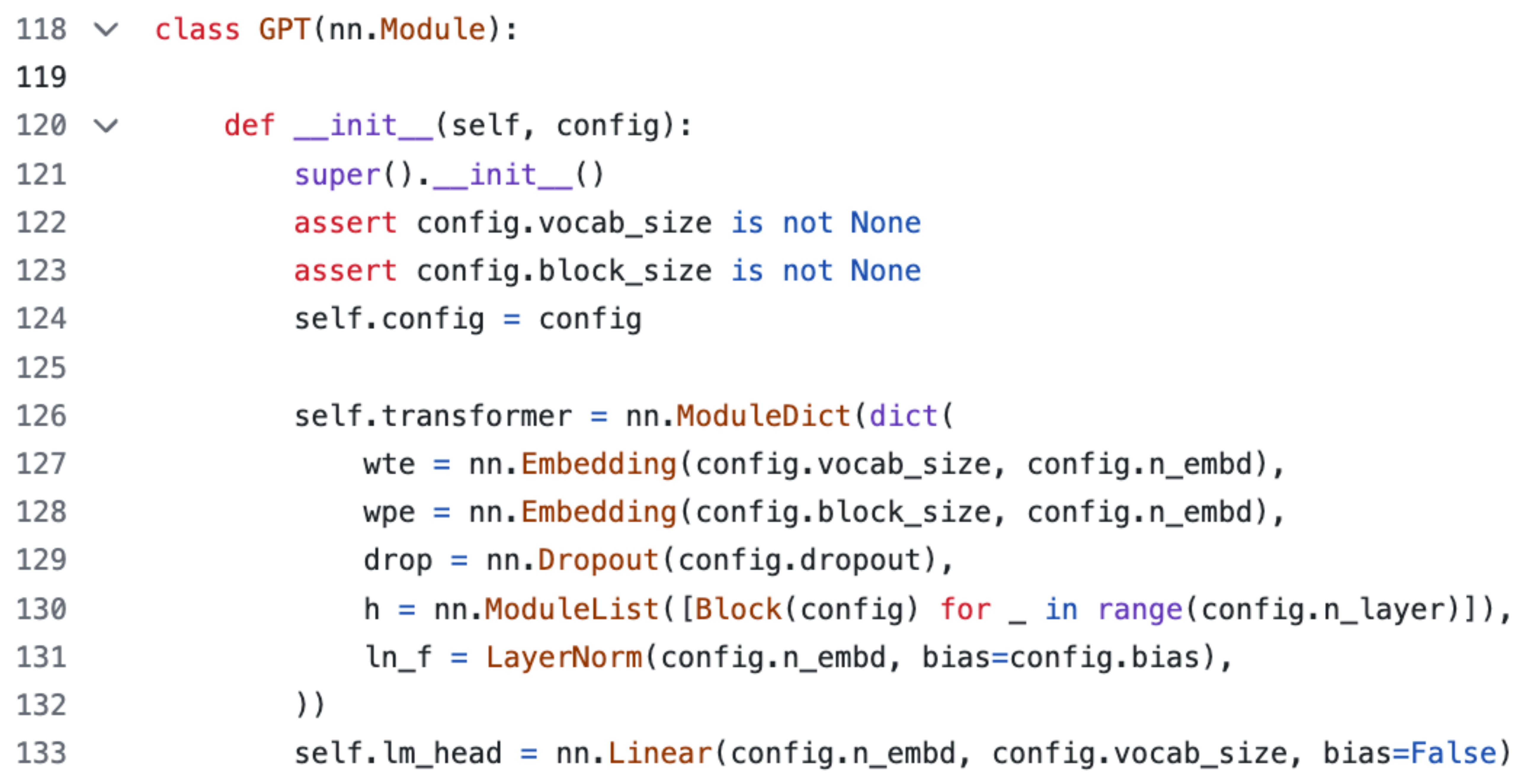
```
AT&T remains in discussions with the Justice Department about the telecom giant'
s planned $85.4 billion acquisition of Time Warner and is still confident it wil
l be able to seal the deal, AT&T CFO John Stephens said Thursday at a Morgan Sta
nley investor conference in Barcelona.

The company also disclosed that its DirecTV Now streaming service, launched last
 November, has reached more than 900,000 subscribers. "About half of these subsc
ribers are cord-cutters or cord-nevers, with about 10 percent of those coming fr
om AT&T's DirecTV and U-verse services," the company said. "The other half have
migrated from other traditional TV providers."

During Stephens' Barcelona appearance, the first question posed to him was about
 the planned Time Warner takeover and its outlook. "We are in discussions with t
he DOJ," Stephens said in repeating past comments when asked about the latest on
 the deal, adding that he wouldn't disclose details about the talks and reiterat
ing a comment from last week that, "as of this time, the timing of the close is
uncertain."
```

https://medium.com/datafabrica/mastering-context-extraction-for-fine-tuning-gpt-models-03c5b2f3dc02

NanoGPT: model.py

- Token, positional embd.를 초기화
  - n_embd: WE, PE의 벡터 크기 (수백~수천)
  - vocab_size: 총 토큰 개수 (약 5만 ~ 10만)
  - block_size: max 컨텍스트 길이 (수백~수천)

  - wte: 토큰 임베딩
  - wpe: 포지셔널 임베딩
  - (중요!) h: 여기서 실제로 context vector를 계산함
    - n_layer 만큼
  - lm_head: 맨 마지막에 최종적으로 계산된 context vec.를 통해 classification 하는 mlp

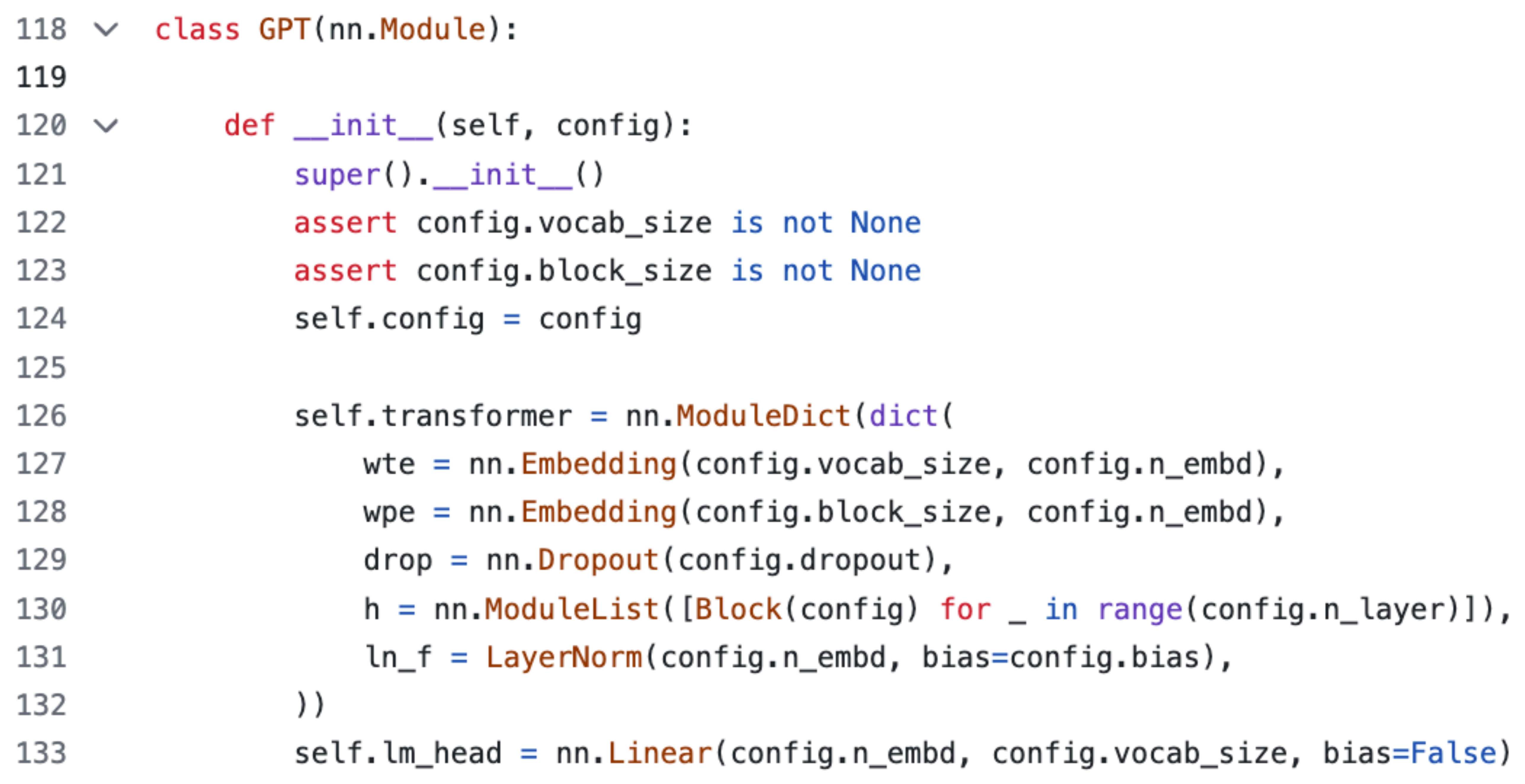


```python
class GPT(nn.Module):

    def __init__(self, config):
        super().__init__()
        assert config.vocab_size is not None
        assert config.block_size is not None
        self.config = config

        self.transformer = nn.ModuleDict(dict(
            wte = nn.Embedding(config.vocab_size, config.n_embd),
            wpe = nn.Embedding(config.block_size, config.n_embd),
            drop = nn.Dropout(config.dropout),
            h = nn.ModuleList([Block(config) for _ in range(config.n_layer)]),
            ln_f = LayerNorm(config.n_embd, bias=config.bias),
        ))
        self.lm_head = nn.Linear(config.n_embd, config.vocab_size, bias=False)
```

NanoGPT: model.py

- Token, positional embd.를 초기화
  - n_embd: WE, PE의 벡터 크기 (수백~수천)
  - vocab_size: 총 토큰 개수 (약 5만 ~ 10만)
  - block_size: max 컨텍스트 길이 (수백~수천)

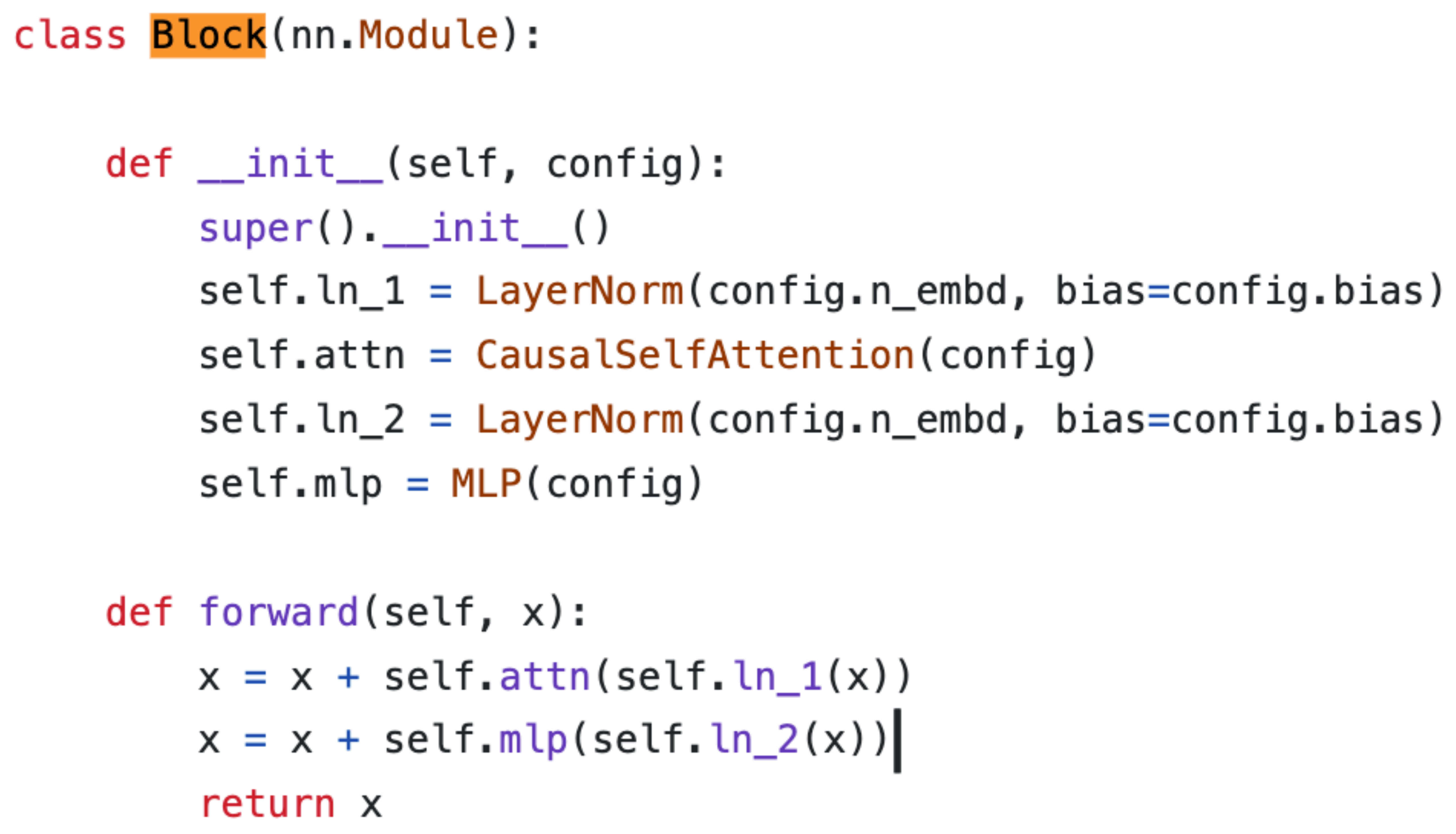
  - wte: 토큰 임베딩
  - wpe: 포지셔널 임베딩
  - (중요!) h: 여기서 실제로 context vector를 계산함
    - n_layer 만큼
  - lm_head: 맨 마지막에 최종적으로 계산된 context vec.를 통해 classification 하는 mlp



```python
class GPT(nn.Module):

    def __init__(self, config):
        super().__init__()
        assert config.vocab_size is not None
        assert config.block_size is not None
        self.config = config

        self.transformer = nn.ModuleDict(dict(
            wte = nn.Embedding(config.vocab_size, config.n_embd),
            wpe = nn.Embedding(config.block_size, config.n_embd),
            drop = nn.Dropout(config.dropout),
            h = nn.ModuleList([Block(config) for _ in range(config.n_layer)]),
            ln_f = LayerNorm(config.n_embd, bias=config.bias),
        ))
        self.lm_head = nn.Linear(config.n_embd, config.vocab_size, bias=False)
```

NanoGPT: model.py

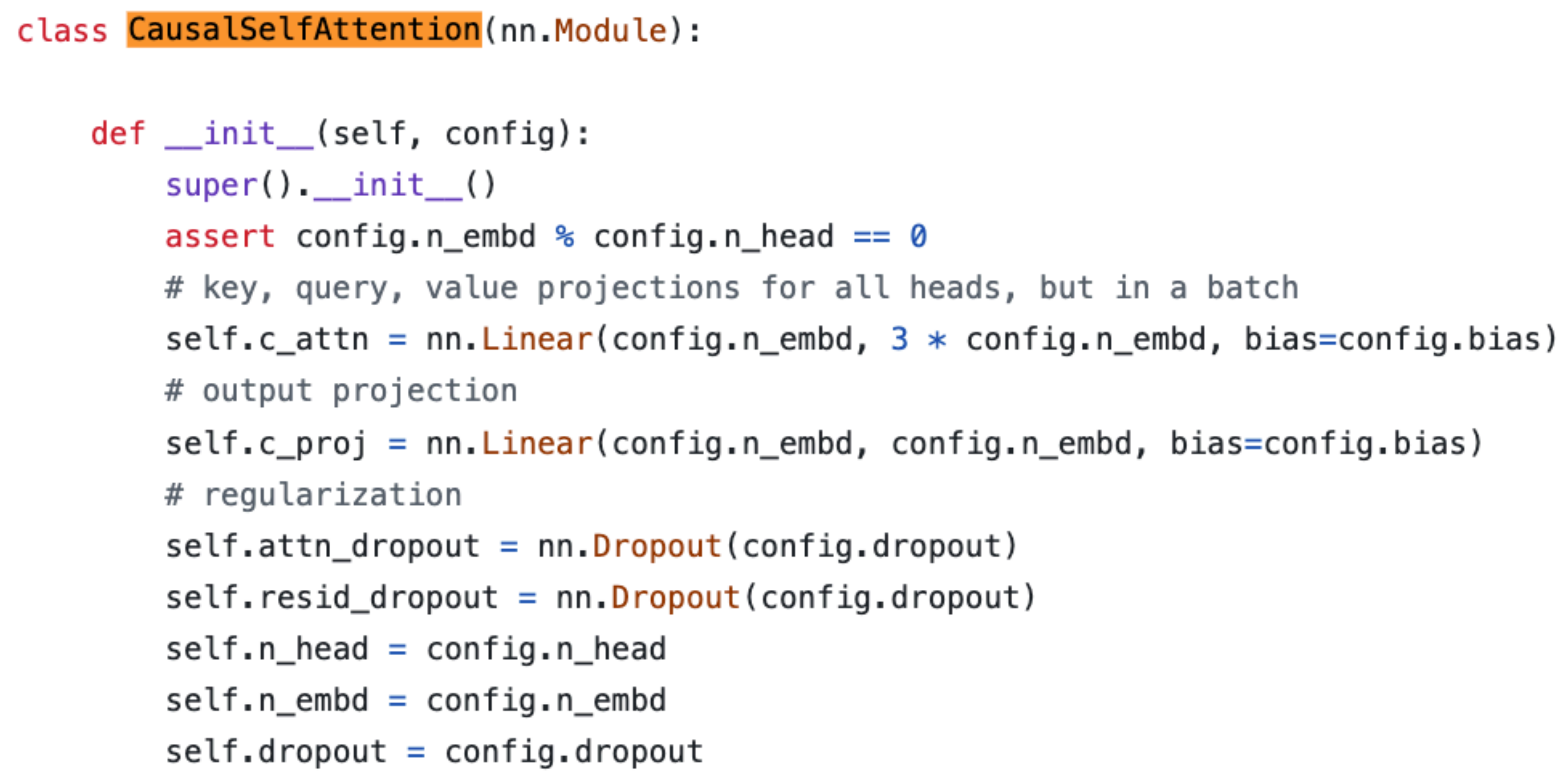
- Class Block()
  - 다시 CausalSelfAttention를 불러옴
  -



```python
class Block(nn.Module):

    def __init__(self, config):
        super().__init__()
        self.ln_1 = LayerNorm(config.n_embd, bias=config.bias)
        self.attn = CausalSelfAttention(config)
        self.ln_2 = LayerNorm(config.n_embd, bias=config.bias)
        self.mlp = MLP(config)

    def forward(self, x):
        x = x + self.attn(self.ln_1(x))
        x = x + self.mlp(self.ln_2(x))
        return x
```

NanoGPT: model.py

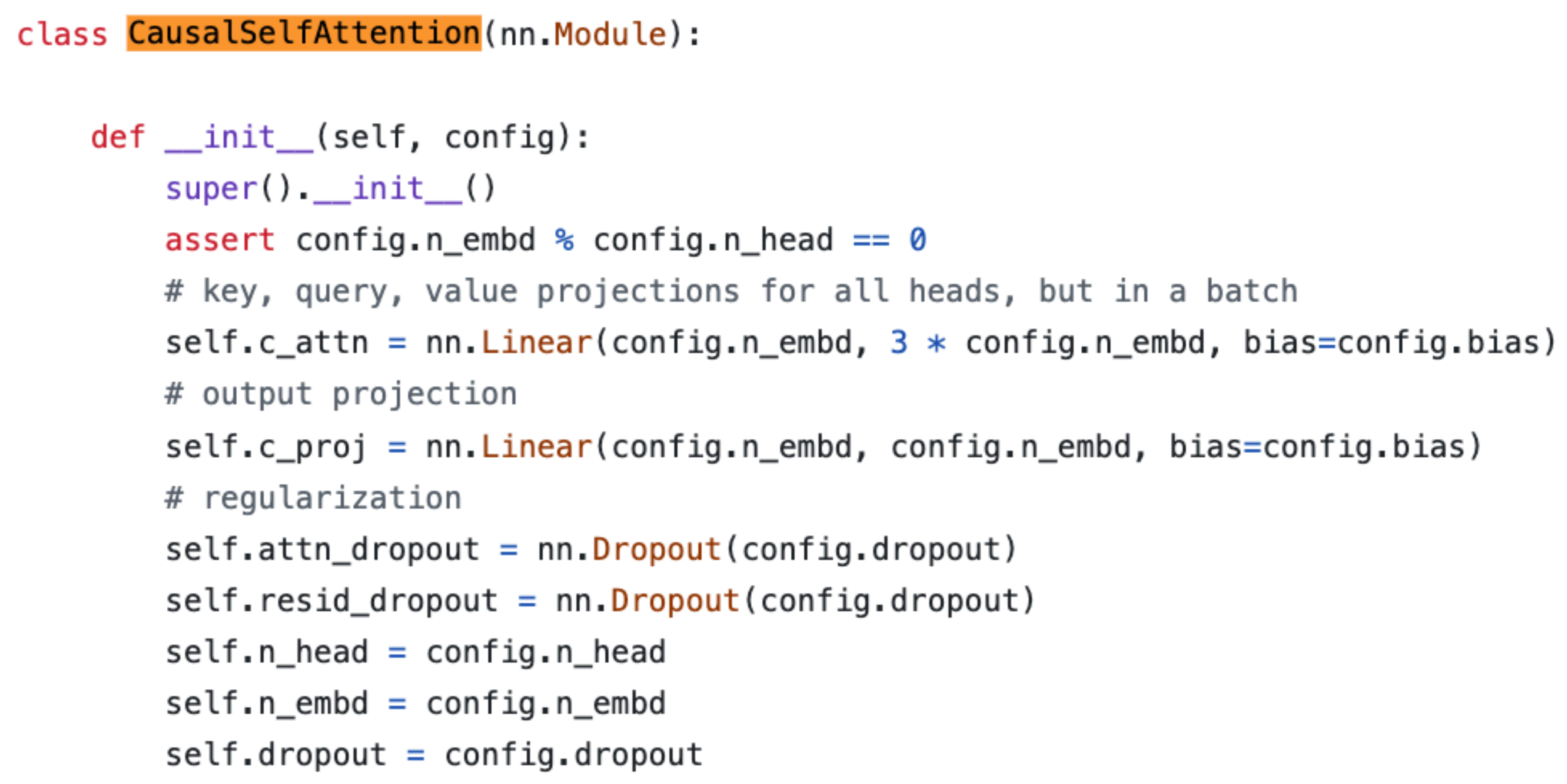
- Class CausalSelfAttention()
  - self.c_attn:
    - 이게 각 block (n_layer만큼 있는)에서의 query, key, value 행렬!
    - 결국, self.c_attn의 행렬값 (weight 값)를 학습하는게 GPT의 메인 과제
  - self.c_proj
    - 각 block에서 계산된 context vector를 매번 MLP에 통과시킴
    - 여기서, 인풋, 아웃풋 크기는 변하지 않음 (n_embd)
    -



```python
class CausalSelfAttention(nn.Module):

    def __init__(self, config):
        super().__init__()
        assert config.n_embd % config.n_head == 0
        # key, query, value projections for all heads, but in a batch
        self.c_attn = nn.Linear(config.n_embd, 3 * config.n_embd, bias=config.bias)
        # output projection
        self.c_proj = nn.Linear(config.n_embd, config.n_embd, bias=config.bias)
        # regularization
        self.attn_dropout = nn.Dropout(config.dropout)
        self.resid_dropout = nn.Dropout(config.dropout)
        self.n_head = config.n_head
        self.n_embd = config.n_embd
        self.dropout = config.dropout
```

NanoGPT: model.py

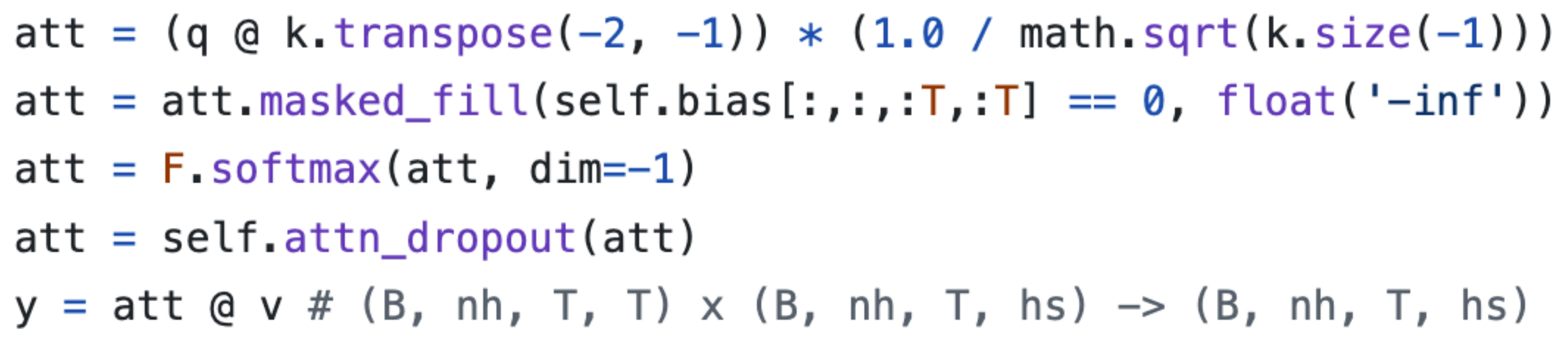
- Class CausalSelfAttention()
  - self.c_attn:
    - 이게 각 block (n_layer만큼 있는)에서의 query, key, value 행렬!
    - 결국, self.c_attn의 행렬값 (weight 값)를 학습하는게 GPT의 메인 과제
  - self.c_proj
    - 각 block에서 계산된 context vector를 매번 MLP에 통과시킴
    - 여기서, 인풋, 아웃풋 크기는 변하지 않음 (n_embd)
    -



```
class CausalSelfAttention(nn.Module):

    def __init__(self, config):
        super().__init__()
        assert config.n_embd % config.n_head == 0
        # key, query, value projections for all heads, but in a batch
        self.c_attn = nn.Linear(config.n_embd, 3 * config.n_embd, bias=config.bias)
        # output projection
        self.c_proj = nn.Linear(config.n_embd, config.n_embd, bias=config.bias)
        # regularization
        self.attn_dropout = nn.Dropout(config.dropout)
        self.resid_dropout = nn.Dropout(config.dropout)
        self.n_head = config.n_head
        self.n_embd = config.n_embd
        self.dropout = config.dropout
```

NanoGPT: model.py

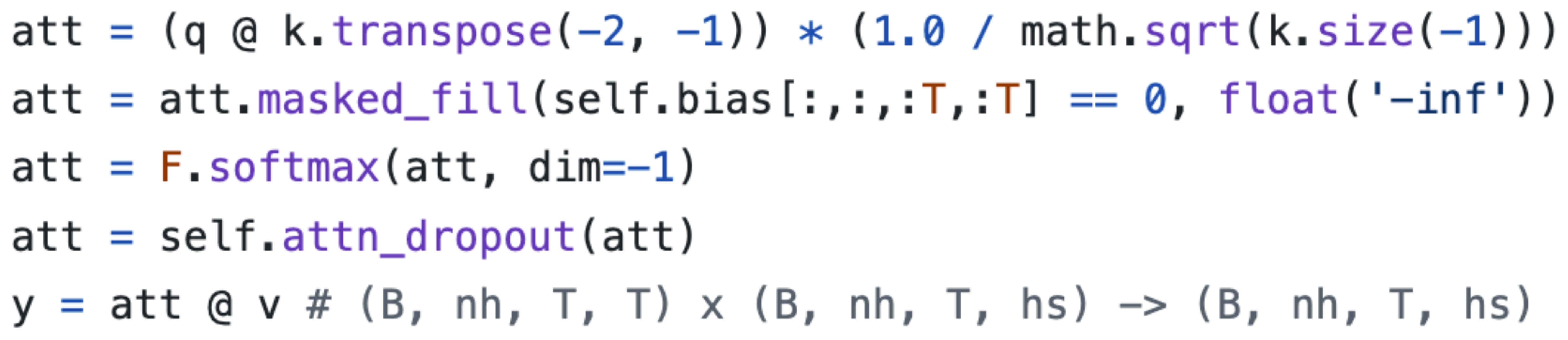
- Class CausalSelfAttention() -> forward()
  - 여기가 가장 중요!
  - 1) query와 key를 내적함
    - 파이토치의 @ 연산자 (matmul이랑 같음)
  - 2) causal attention 일때만 적용
    - 상삼각행렬 upper triangular matrix 를 통해
      - 첫 query는 첫 key
      - 두번째 q -> 1,2번째 k
      - 세번째 q -> 1,2,3번째 k
      - … block size 번째 q -> 모든 k 값들이랑 내적



```
att = (q @ k.transpose(-2, -1)) * (1.0 / math.sqrt(k.size(-1)))
att = att.masked_fill(self.bias[:,:,:T,:T] == 0, float('-inf'))
att = F.softmax(att, dim=-1)
att = self.attn_dropout(att)
y = att @ v # (B, nh, T, T) x (B, nh, T, hs) -> (B, nh, T, hs)
```

NanoGPT: model.py

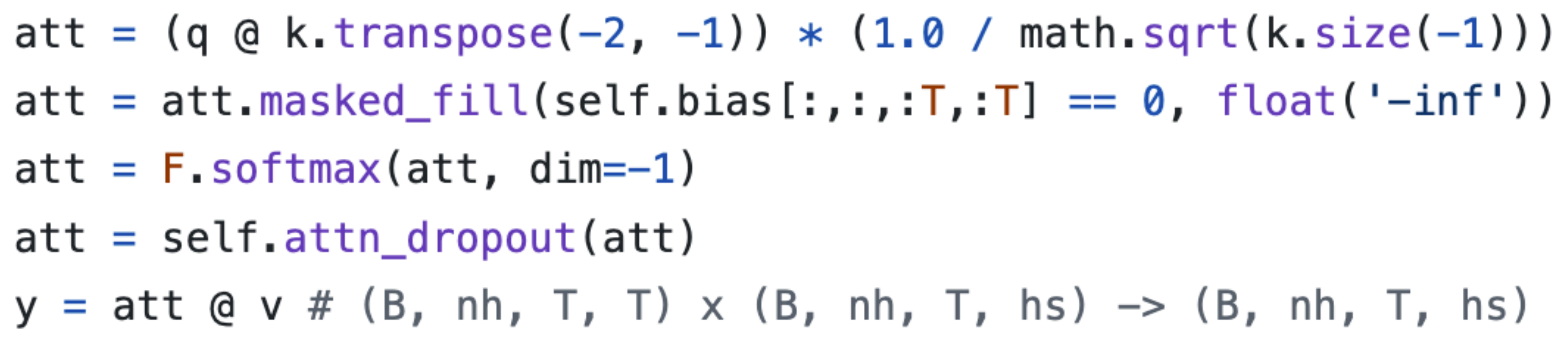
- Class CausalSelfAttention() -> forward()
  - 그 다음 softmax
  - 마지막으로, 계산된 attn값에 value를 곱해 n번째 layer에서의 context vector를 구함



```
att = (q @ k.transpose(-2, -1)) * (1.0 / math.sqrt(k.size(-1)))
att = att.masked_fill(self.bias[:,:,:T,:T] == 0, float('-inf'))
att = F.softmax(att, dim=-1)
att = self.attn_dropout(att)
y = att @ v # (B, nh, T, T) x (B, nh, T, hs) -> (B, nh, T, hs)
```

NanoGPT: model.py

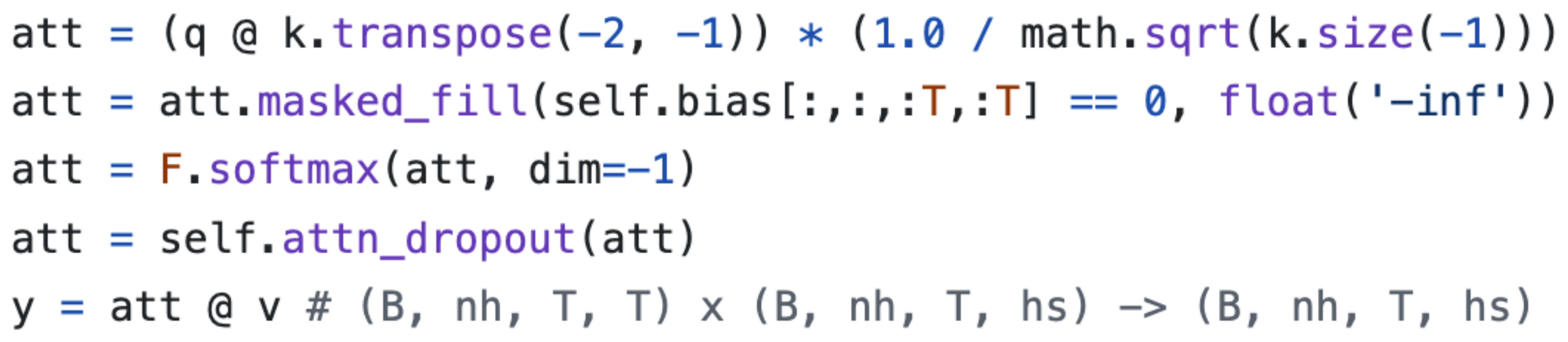
- Tensor 크기
  - 인풋, 아웃풋 (컨텍스트 백터) 둘다
    - 원래는
      - batch size (B)
      - Context length or sequence length (T)
      - n_embd (C)
      - 로 B x T x C



```python
att = (q @ k.transpose(-2, -1)) * (1.0 / math.sqrt(k.size(-1)))
att = att.masked_fill(self.bias[:,:,:T,:T] == 0, float('-inf'))
att = F.softmax(att, dim=-1)
att = self.attn_dropout(att)
y = att @ v # (B, nh, T, T) x (B, nh, T, hs) -> (B, nh, T, hs)
```

NanoGPT: model.py

- Class CausalSelfAttention() -> forward()
  - 그 다음 softmax
  - 마지막으로, 계산된 attn값에 value를 곱해 n번째 layer에서의 context vector를 구함



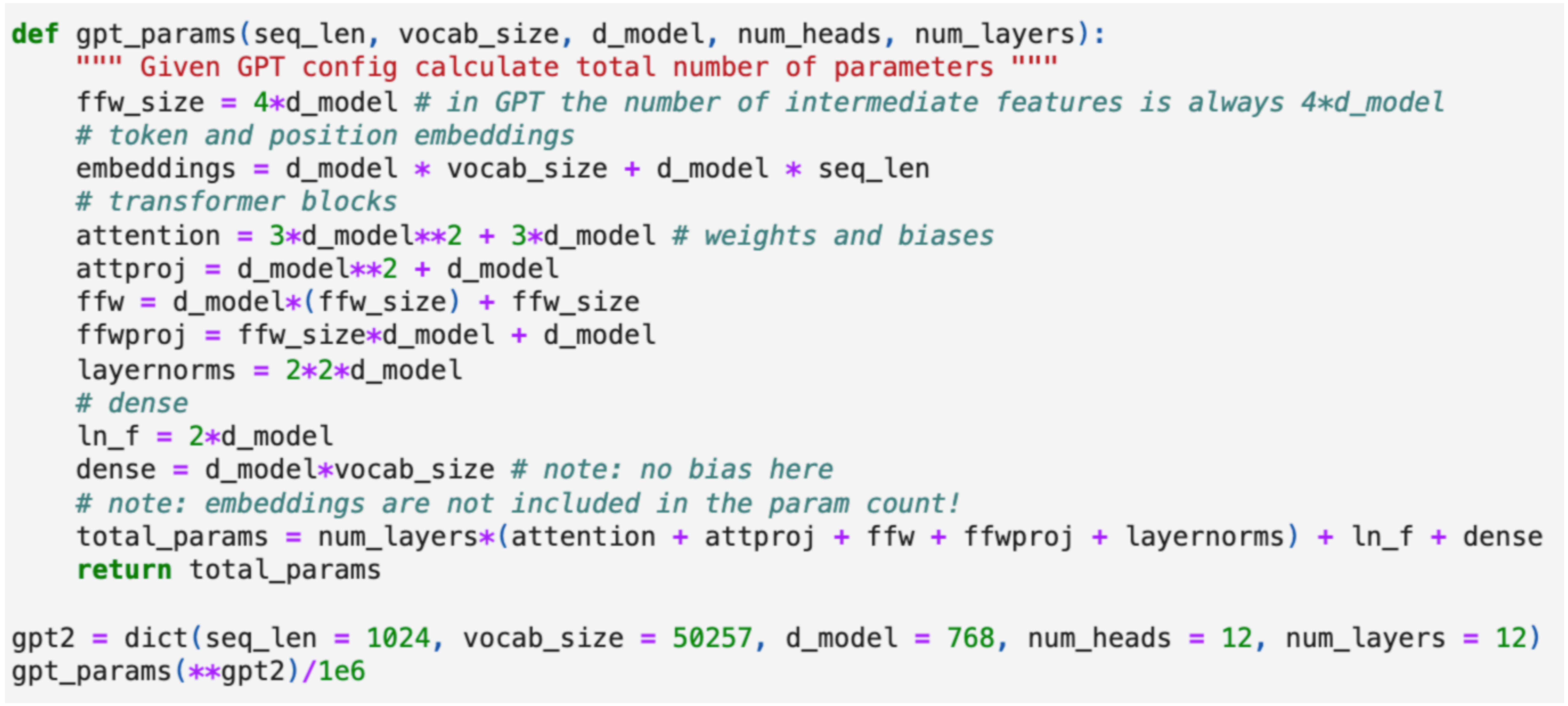
```python
att = (q @ k.transpose(-2, -1)) * (1.0 / math.sqrt(k.size(-1)))
att = att.masked_fill(self.bias[:,:,:T,:T] == 0, float('-inf'))
att = F.softmax(att, dim=-1)
att = self.attn_dropout(att)
y = att @ v # (B, nh, T, T) x (B, nh, T, hs) -> (B, nh, T, hs)
```

NanoGPT: model.py

- Summary
  - model.py는 class GPT()
  - GPT 클래스 안에는 n_layer 만큼의 Block Class가 있음
  - Block 클래스 안에는 1개의 CausalSelfAttention class (이게 메인)과 1개의 MLP class (메인은 아닌데 엄청 큼) 가 있음
  - CausalSelfAttention 클래스 안에서 c_attn (q, k, v 행렬을 합친 거)가 들어있어서 여기서 실제로 attention, context vector 를 구함 (즉 이게 진짜 메인)
    - 부가적으로, 여기 안에서도 c_proj라는 MLP가 존재 (block 안에서의 MLP와 완전 같은 역할)
    - 매 layer마다 존재한다는것도 같음
    - 다른점은, 훨씬 작다는 것 (no hidden layer, 4대신 1)
  - 마지막으로, 모든 레이어를 거친 후 다시 GPT class 안의 forward에서 lm_head를 통해
  - 구한 컨텍스트 벡터에서 다음 단어를 예측하는 classification을 진행함

NanoGPT: model.py

- Total number of params in GPT
  -

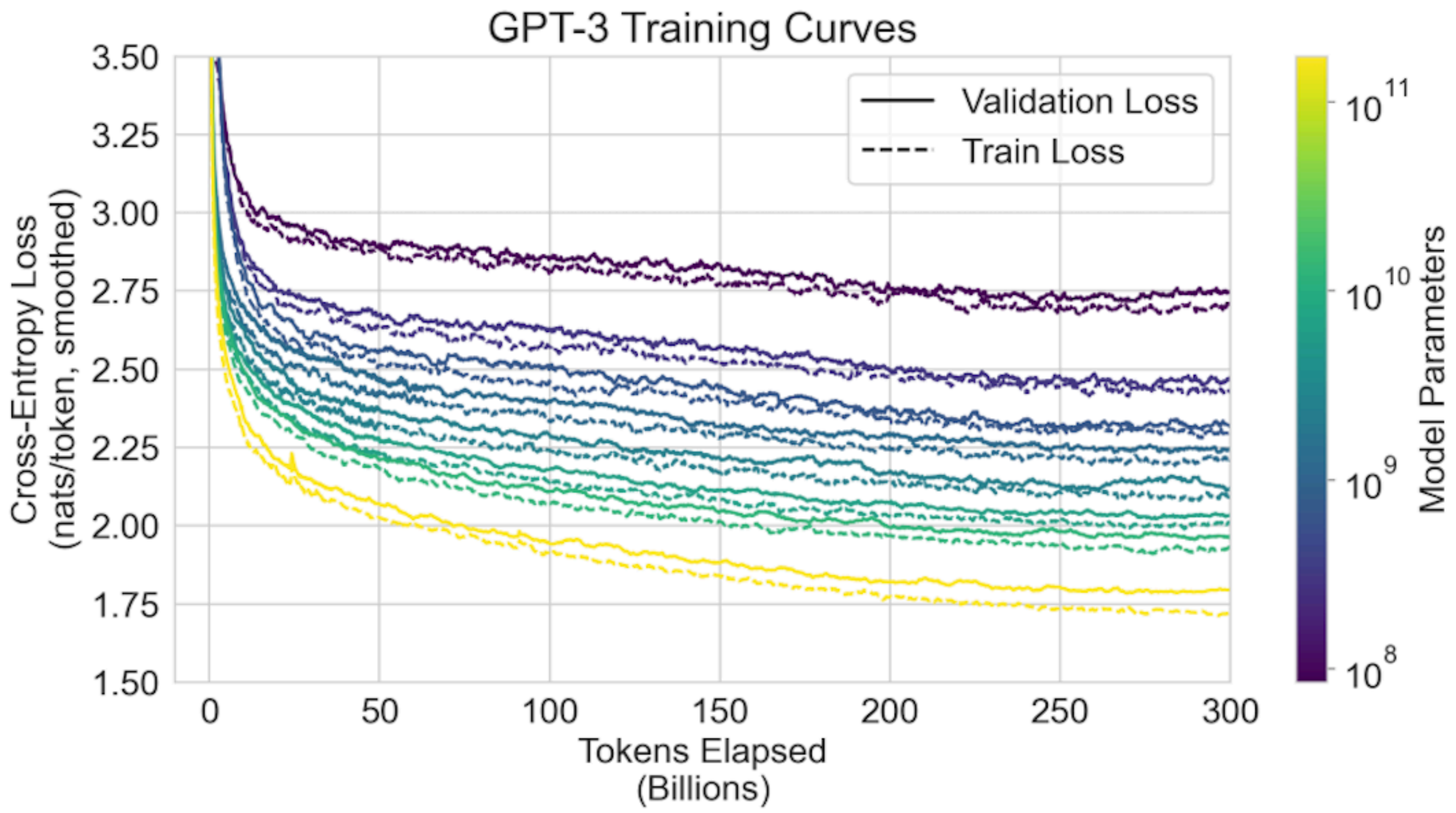


```python
def gpt_params(seq_len, vocab_size, d_model, num_heads, num_layers):
    """ Given GPT config calculate total number of parameters """
    ffw_size = 4*d_model # in GPT the number of intermediate features is always 4*d_model
    # token and position embeddings
    embeddings = d_model * vocab_size + d_model * seq_len
    # transformer blocks
    attention = 3*d_model**2 + 3*d_model # weights and biases
    attproj = d_model**2 + d_model
    ffw = d_model*(ffw_size) + ffw_size
    ffwproj = ffw_size*d_model + d_model
    layernorms = 2*2*d_model
    # dense
    ln_f = 2*d_model
    dense = d_model*vocab_size # note: no bias here
    # note: embeddings are not included in the param count!
    total_params = num_layers*(attention + attproj + ffw + ffwproj + layernorms) + ln_f + dense
    return total_params

gpt2 = dict(seq_len = 1024, vocab_size = 50257, d_model = 768, num_heads = 12, num_layers = 12)
gpt_params(**gpt2)/1e6
```

NanoGPT: train.py

- 총 60만 (600K) 번의 iteration을 거침
    - Using good GPU, 1 iteration per second
    - Ends in 600k seconds
    - A day is 80.64K
    - Therefore, takes about a week…
- For every iter iteration
    - Process Batch size (12) x context length (1024) x GPU 개수 (8) x gradAccumStep (5) tokens
        - 0.5M / sec
        - 600K iter -> About 300B (즉, 처리하는 총 토큰양은 3천억개 정도…)
- OpenWebText dataset has about 9B tokens
- By the end of training, GPT processes each token in the training set about 30 times on average (300/9).

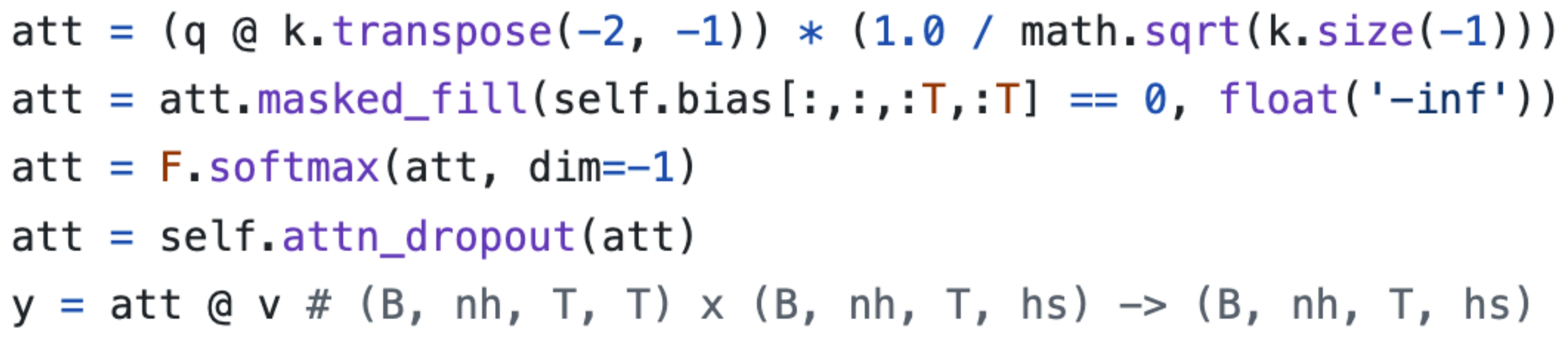
NanoGPT: train.py



* Language Models are Few-Shot Learners

NanoGPT: model.py

- Tensor 크기
  - Input and outputs are
    - Originally
      - batch size (B)
      - Context length or sequence length (T)
      - n_embd (C)
      - therefore B x T x C



```
att = (q @ k.transpose(-2, -1)) * (1.0 / math.sqrt(k.size(-1)))
att = att.masked_fill(self.bias[:,:,:T,:T] == 0, float('-inf'))
att = F.softmax(att, dim=-1)
att = self.attn_dropout(att)
y = att @ v # (B, nh, T, T) x (B, nh, T, hs) -> (B, nh, T, hs)
```

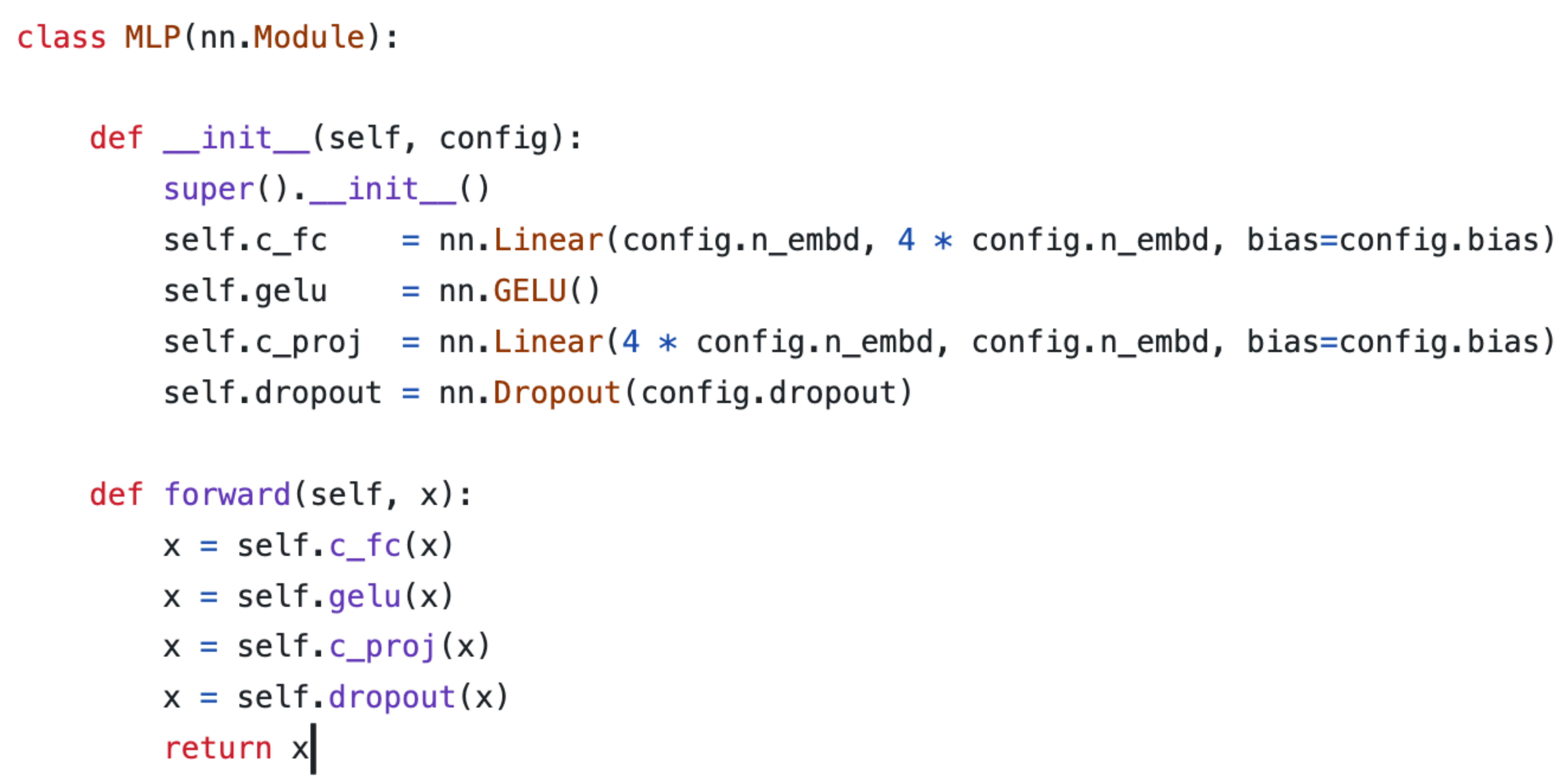
NanoGPT: model.py

- Back to Block()
  - CausalSelfAttention is the main
    - Computes the context vector
  - Here, we have an additional MLP

```python
class Block(nn.Module):

    def __init__(self, config):
        super().__init__()
        self.ln_1 = LayerNorm(config.n_embd, bias=config.bias)
        self.attn = CausalSelfAttention(config)
        self.ln_2 = LayerNorm(config.n_embd, bias=config.bias)
        self.mlp = MLP(config)

    def forward(self, x):
        x = x + self.attn(self.ln_1(x))
        x = x + self.mlp(self.ln_2(x))
        return x
```

NanoGPT: model.py

- class MLP()
  - Nothing special, just pass the computed context vector through an MLP again.
  - Further refines the context vector.
  - The key point is that this happens in every block, for each of the **n_layers**.
  - Both input and output are **n_embd**,
  - but the hidden layer is **4 * n_embd**
  - In other words, it’s a very large MLP.
  - This significantly impacts the overall model size



```
class MLP(nn.Module):

    def __init__(self, config):
        super().__init__()
        self.c_fc    = nn.Linear(config.n_embd, 4 * config.n_embd, bias=config.bias)
        self.gelu    = nn.GELU()
        self.c_proj  = nn.Linear(4 * config.n_embd, config.n_embd, bias=config.bias)
        self.dropout = nn.Dropout(config.dropout)

    def forward(self, x):
        x = self.c_fc(x)
        x = self.gelu(x)
        x = self.c_proj(x)
        x = self.dropout(x)
        return x
```

NanoGPT: model.py

- Summary
  - model.py defines the class **GPT()**.
  - The **GPT** class contains multiple **Block** classes (equal to **n_layers**).
  - Each **Block** class has one **CausalSelfAttention** class (the main component) and one **MLP** class (not the main, but very large).
  - Inside **CausalSelfAttention**, the **c_attn** (combined q, k, v matrices) calculates the attention and context vector (this is the core).
    - Additionally, there's a **c_proj** MLP inside it (similar to the MLP in the block).
    - It's present in every layer but much smaller (no hidden layer, 1 instead of 4).
  - Finally, after passing through all layers, the **lm_head** in the **GPT** class predicts the next word using the context vector