Παραγωγική Τεχνητή Νοημοσύνη: Generative AI

Κωνσταντίνος Καραμανής

The University of Texas at Austin & Archimedes/Athena RC

constantine@utexas.edu

https://caramanis.github.io/

Ας θυμηθούμε τα προηγούμενα...

Word2Vec: δεν αρκεί για σημασιολογική αναζήτηση

Στατική vs Συμφραζόμενη Ενσωμάτωση

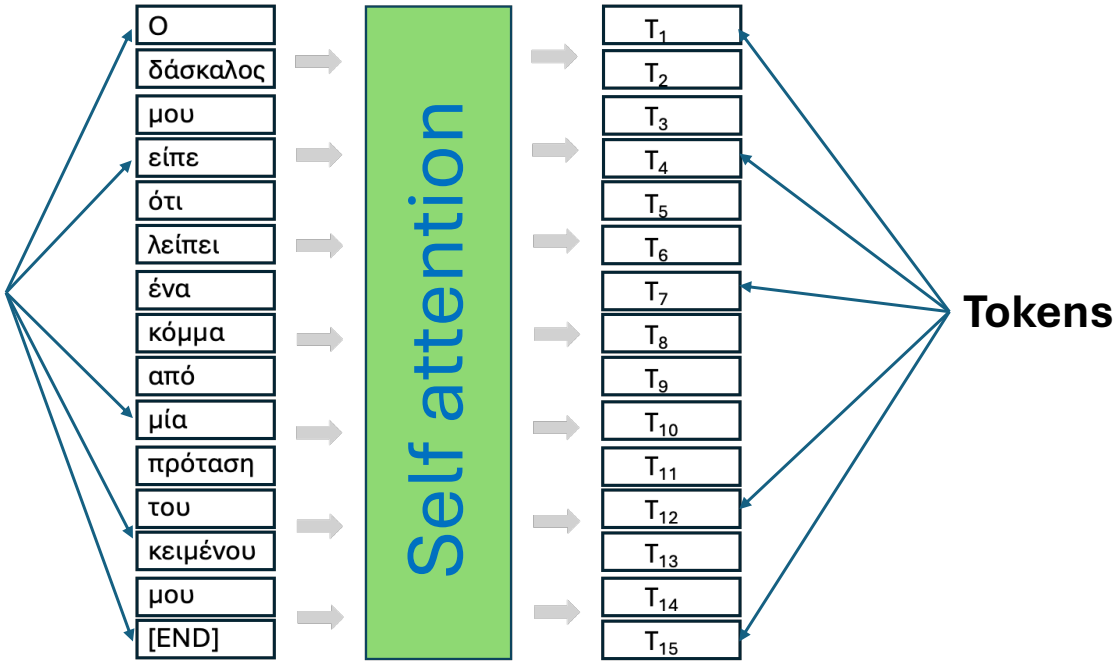
“Ο δάσκαλος είπε ότι λείπει ένα κόμμα από μια πρόταση του κειμένου μου.”

“Ο σχολιαστής είπε ότι λείπει ένα κόμμα από την παρουσίαση των αποτελεσμάτων της δημοσκόπησης.”

Το Word2Vec παράγει στατικές ενσωματώσεις: η ενσωμάτωση της λέξης «κόμμα» στο Word2Vec είναι πάντα το ίδιο διάνυσμα, ανεξάρτητα από τα συμφραζόμενα. Για αποδοτική αναζήτηση με βάση την σημασία, χρειαζόμαστε **συμφραζόμενες** ενσωματώσεις, όπου το διάνυσμα της λέξης εξαρτάται από το πλαίσιο της πρότασης.

Το «Transformer» Layer και Attention

“[Ο] [δάσκαλος] [είπε] [ότι] [λείπει] [ένα] [κόμμα] [από] [μια] [πρόταση] [του] [κειμένου] [μου].”



Επιτηρούμενη Μάθηση

Επιτηρούμενη Μάθηση:
Βλέπουμε *n* παραδείγματα:
(X_1,y_1), (X_2,y_2), ..., (X_n,y_n)

Το αρχικό μας πρόβλημα είναι να βρούμε σημασιολογικές ενσωματώσεις. Για να χρησιμοποιήσουμε όλα τα εργαλεία της επιτηρούμενης μάθησης, πρέπει να διαμορφώσουμε και να δημιουργήσουμε ένα πρόβλημα με τη σωστή μορφή.

Χρειαζόμαστε Ν «παραδείγματα» (X,y)

Self-supervision + σώμα κειμένων (text corpus)

Constantine Caramanis
Professor, Dept. of Electrical and Computer Engineering
Chandra Family Endowed Distinguished Professorship in Electrical and Computer Engineering
Member of the Computer Science Graduate Studies Committee
Office: 2501 Speedway, EER Building Room 6.820
e-mail: constantine at utexas.edu

I am a Professor in the ECE department of The University of Texas at Austin. I received a PhD in EECS from The Massachusetts Institute of Technology, in the Laboratory for Information and Decision Systems (LIDS), and an AB in Mathematics from Harvard University. I received the NSF CAREER award in 2011, and I am an IEEE Fellow.

My current research interests focus on autonomous decision-making in large-scale complex systems, with a focus on learning and computation. Specifically, I am interested in robust and adaptable optimization, high dimensional statistics and machine learning, reinforcement learning and agents, and applications to generative models. I've worked on applications to large-scale networks, including social networks, wireless networks, transportation networks, energy networks and financial applications. I have also worked on applications of machine learning and optimization to computer-aided design.

Κωνσταντίνος Καραμανής

Το διαδίκτυο και άλλες πηγές μας προσφέρουν ανεξάντλητο πλήθος κειμένων. Πρέπει να διαμορφώσουμε πρόβλημα επιτηρούμενης μάθησης.

Αυτό απαιτεί: ζεύγη (X_i, y_i) και συνάρτηση απώλειας που ορίζουν ένα πρόβλημα στο πρότυπο της επιτηρούμενης μάθησης (supervised learning).

Self-supervision + σώμα κειμένων

Masked Language Modeling:

1. 15%: επιλέγουμε τυχαία 15% των «tokens»
2. 80%: αντικαθιστούμε 80% με [MASK]
3. 10%: αντικαθιστούμε 10% με τυχαία επιλεγμένη λέξη
4. 10%: τα υπόλοιπα 10% των tokens παραμένουν ίδια

X = κείμενο, Y = σωστά (αρχικά) tokens

Next Sentence Prediction:

1. Επιλέγουμε δύο προτάσεις (S1,S2) από το σώμα κειμένων. Στις μισές περιπτώσεις, η πρόταση S2 ακολουθεί την S1 στο κείμενο

X = (S1, S2), Y = 1 or 0 (ακόλουθη πρόταση)

Self-supervision + σώμα κειμένων

1. 15% των tokens επιλέγονται

TASK: Masked Language Modeling (MLM)

X_1: My current research interests focus on autonomous decision-making in large-scale complex systems, with a focus on learning and computation

Y_1: {research, autonomous, learning, computation}

X_2: I am interested in robust and adaptable optimization, high dimensional statistics and machine learning, reinforcement learning and agents.

Y_2: {robust, statistics, machine, agents}

Self-supervision + σώμα κειμένων

1. 15% των tokens επιλέγονται

2. 80% των επιλεγμένων token αντικαθίστανται με [MASK]

TASK: Masked Language Modeling (MLM)

X_1: My current [MASK] interests focus on [MASK] decision-making in large-scale complex systems, with a focus on [MASK] and computation

Y_1: {research, autonomous, learning, computation}

X_2: I am interested in [MASK] and adaptable optimization, high dimensional [MASK] and machine learning, reinforcement learning and [MASK]

Y_2: {robust, statistics, machine, agents}

Self-supervision + σώμα κειμένων

1. 15% των tokens επιλέγονται
2. 80% των επιλεγμένων token αντικαθίστανται με [MASK]
3. 10% των επιλεγμένων token αντικαθίστανται τυχαία

TASK: Masked Language Modeling (MLM)

X_1: My current [redacted] interests focus on [redacted] decision-making in large-scale complex systems, with a focus on [redacted] and accordion.

Y_1: {research, autonomous, learning, computation}

X_2: I am interested in [redacted] and adaptable optimization, high dimensional [redacted] and machine learning, reinforcement learning and [redacted]

Y_2: {robust, statistics, machine, agents}

Self-supervision + σώμα κειμένων

1. 15% των tokens επιλέγονται
2. 80% των επιλεγμένων token αντικαθίστανται με [MASK]
3. 10% των επιλεγμένων token αντικαθίστανται τυχαία
4. 10% των επιλεγμένων token παραμένουν απαράλλαχτα

TASK: Masked Language Modeling (MLM)

X_1: My current [redacted] interests focus on [redacted] decision-making in large-scale complex systems, with a focus on [redacted] and accordion.

Y_1: {research, autonomous, learning, computation}

X_2: I am interested in [redacted] and adaptable optimization, high dimensional [redacted] and machine learning, reinforcement learning and [redacted]

Y_2: {robust, statistics, machine, agents}

Self-supervision + σώμα κειμένων

1. 15% των tokens επιλέγονται

2. 80% των επιλεγ
αντικαθίστανται μ

3. 10% των επιλεγ
αντικαθίστανται τι

4. 10% των επιλεγμ
παραμένουν απαράλλαχτα

Συνάρτηση Απώλειας: cross-entropy στα επιλεγμένα Tokens

$$\mathcal{L}_{MLM}(\theta) = -\sum_{i \in M} \log p(y_i | \hat{y}_i; \theta)$$

TASK: Masked Language Modeling (MLM)

X_1: My current [redacted] interests focus on [redacted] decision-making in large-scale [redacted] focus on [redacted]

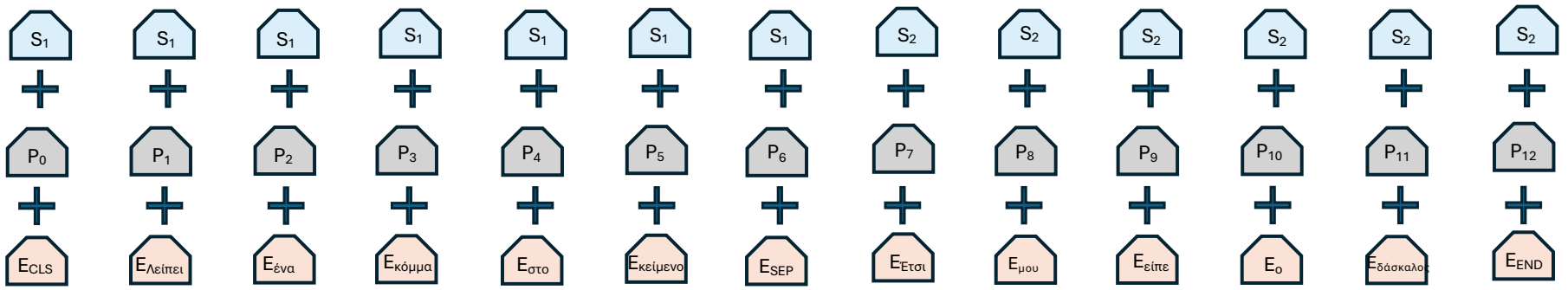
learning, computation}

[redacted] and adaptable [redacted] onal [redacted] forcement [redacted]

Y_2: {robust, statistics, machine, agents}

BERT και RoBERTa: token embeddings (E+P+S)

Segment embedding **(learned):** για το BERT-base, ένα 768-διάστατο διάνυσμα κωδικοποιεί S1 ή S2 (RoBERTa δεν έχει S1/S2)

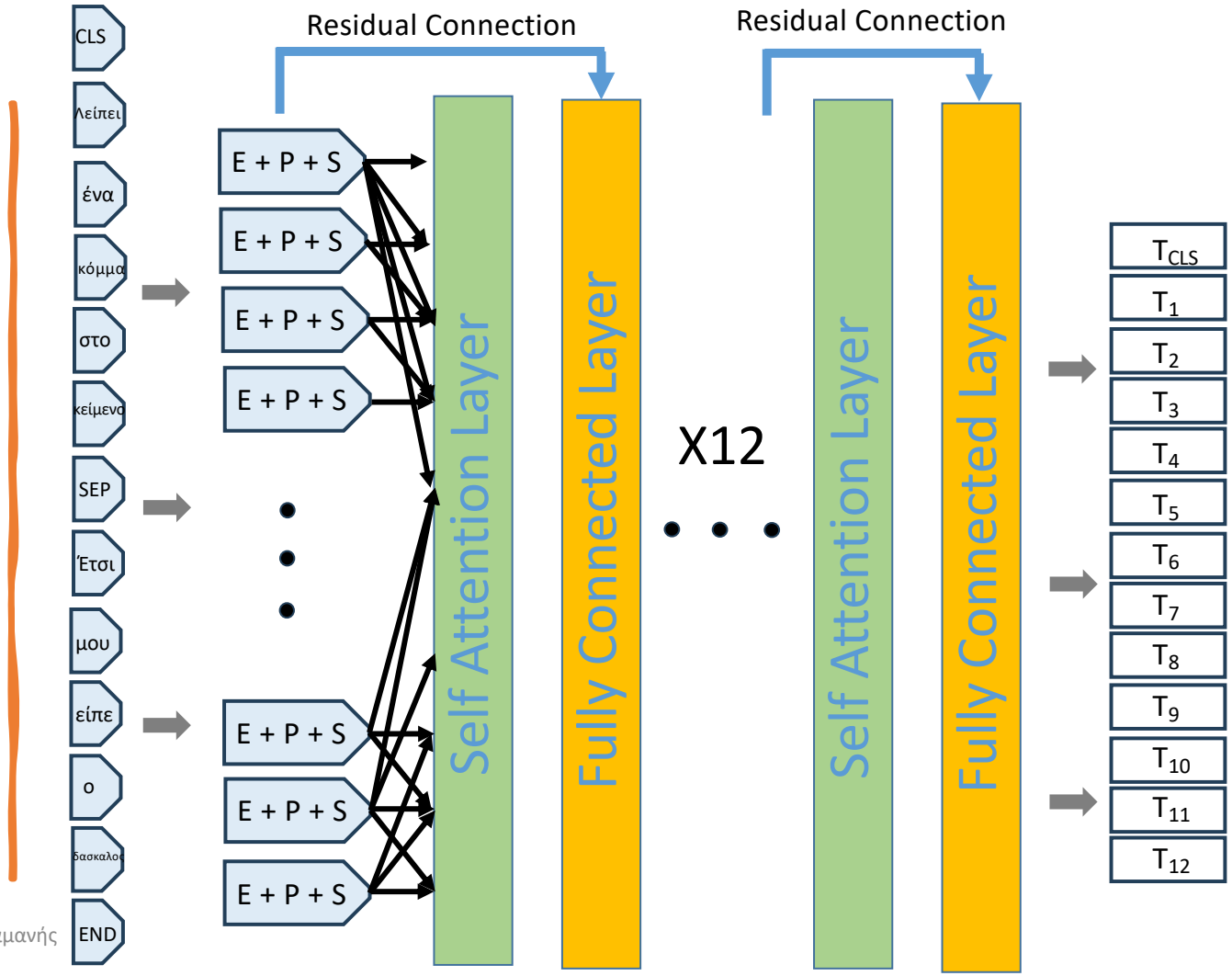


Πρόταση #1

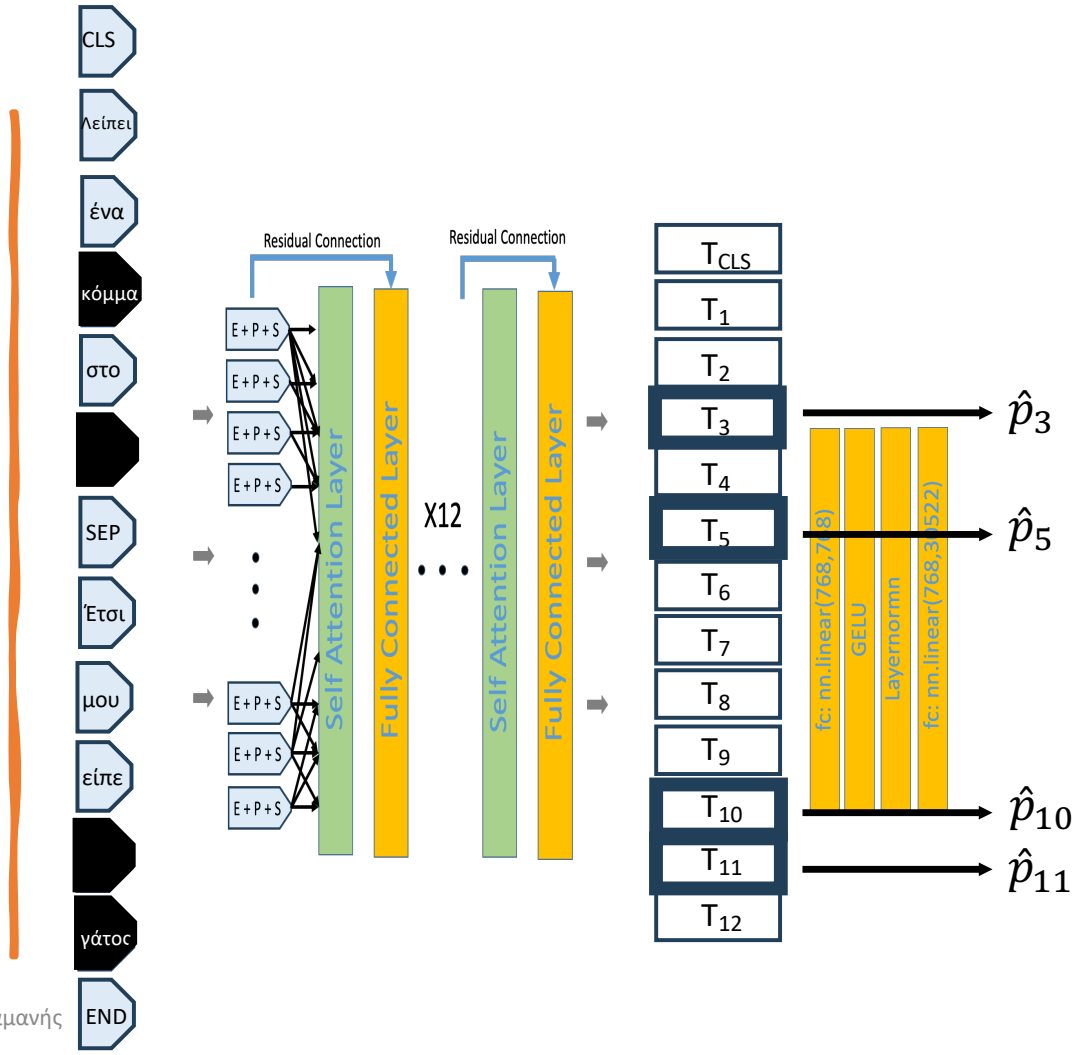
Πρόταση #2

BERT Family: Αρχιτεκτονική του μοντέλου

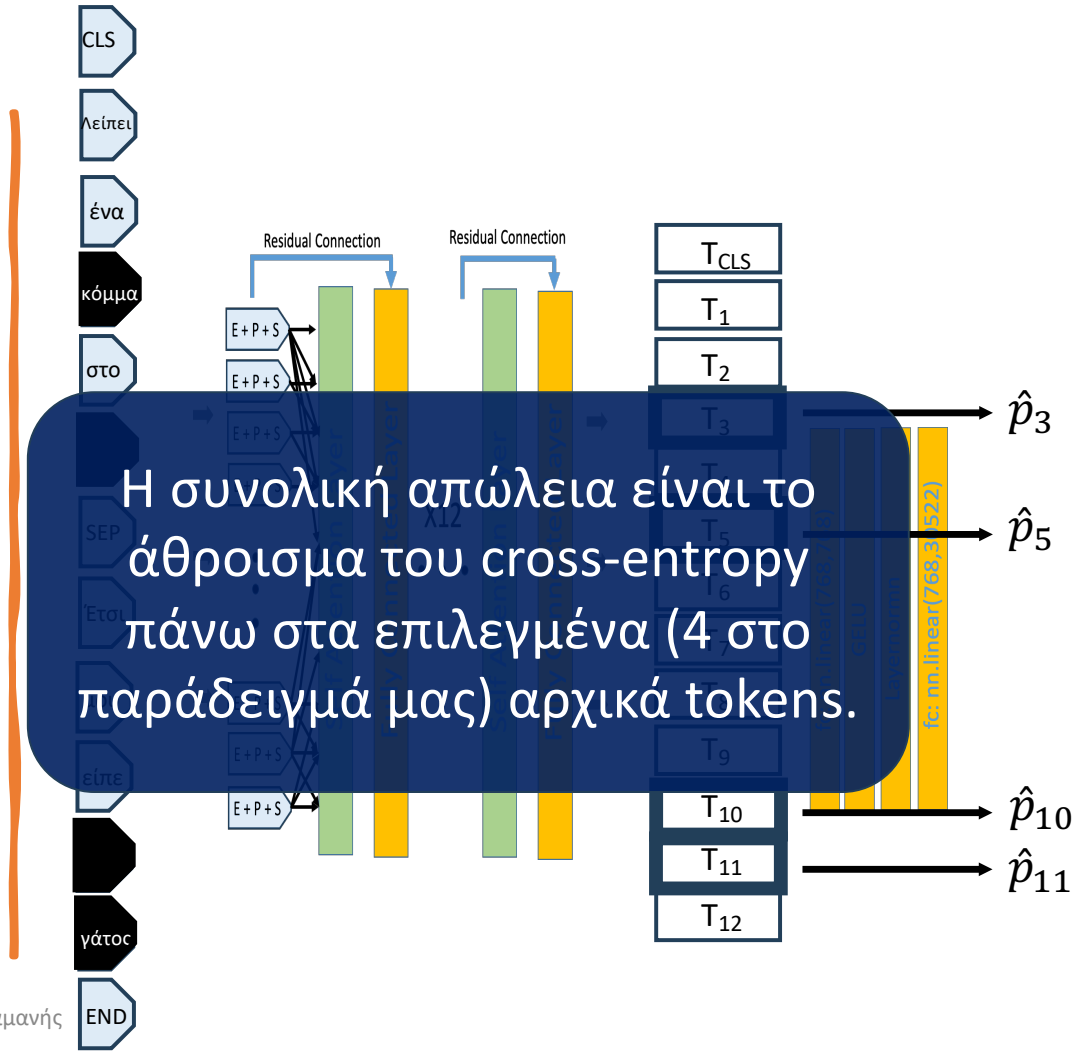
Πώς γίνεται η εκπαίδευση με τα δεδομένα που δημιουργήσαμε με το Masked Language Modeling?



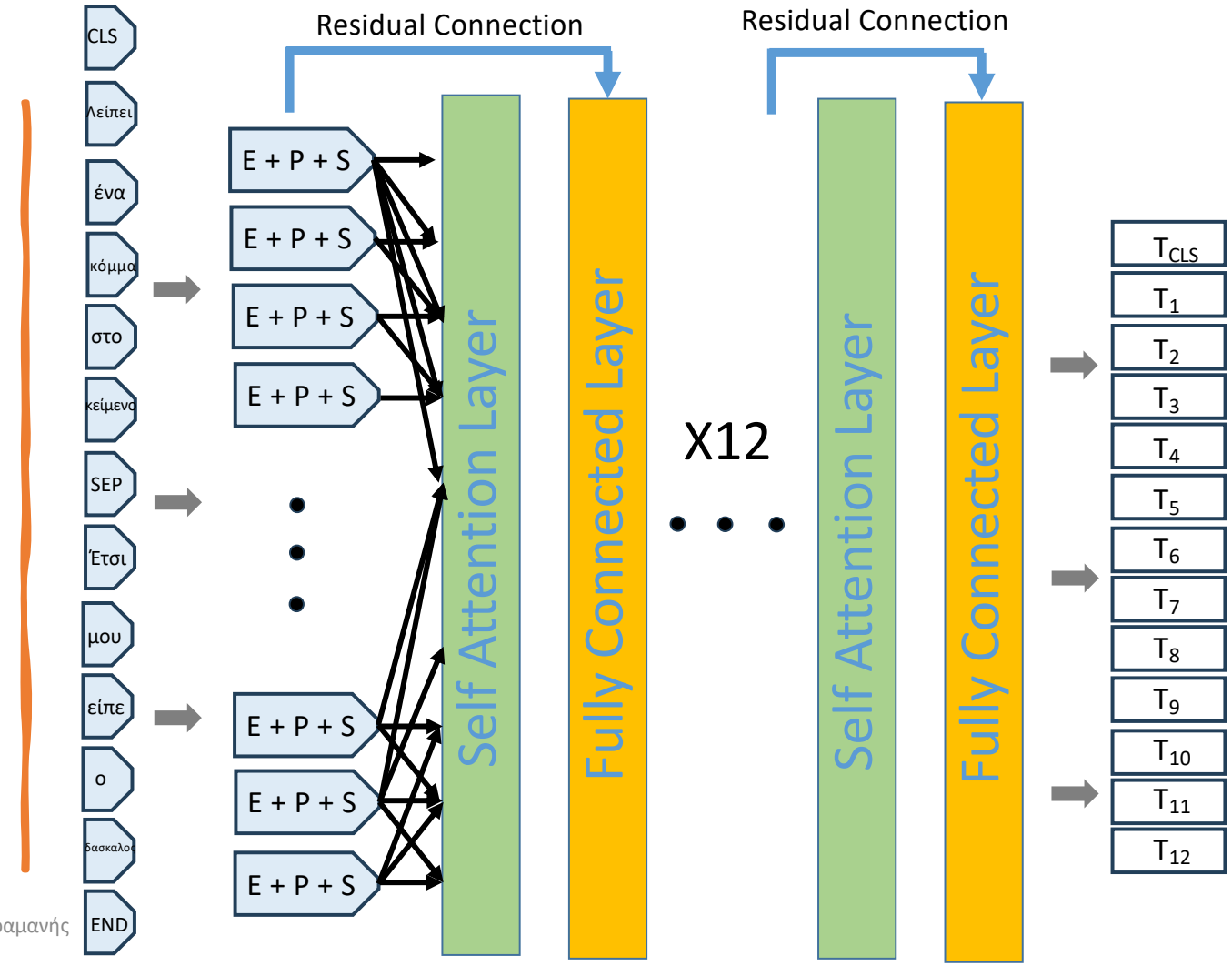
Training and Masked Language Modeling (MLM)



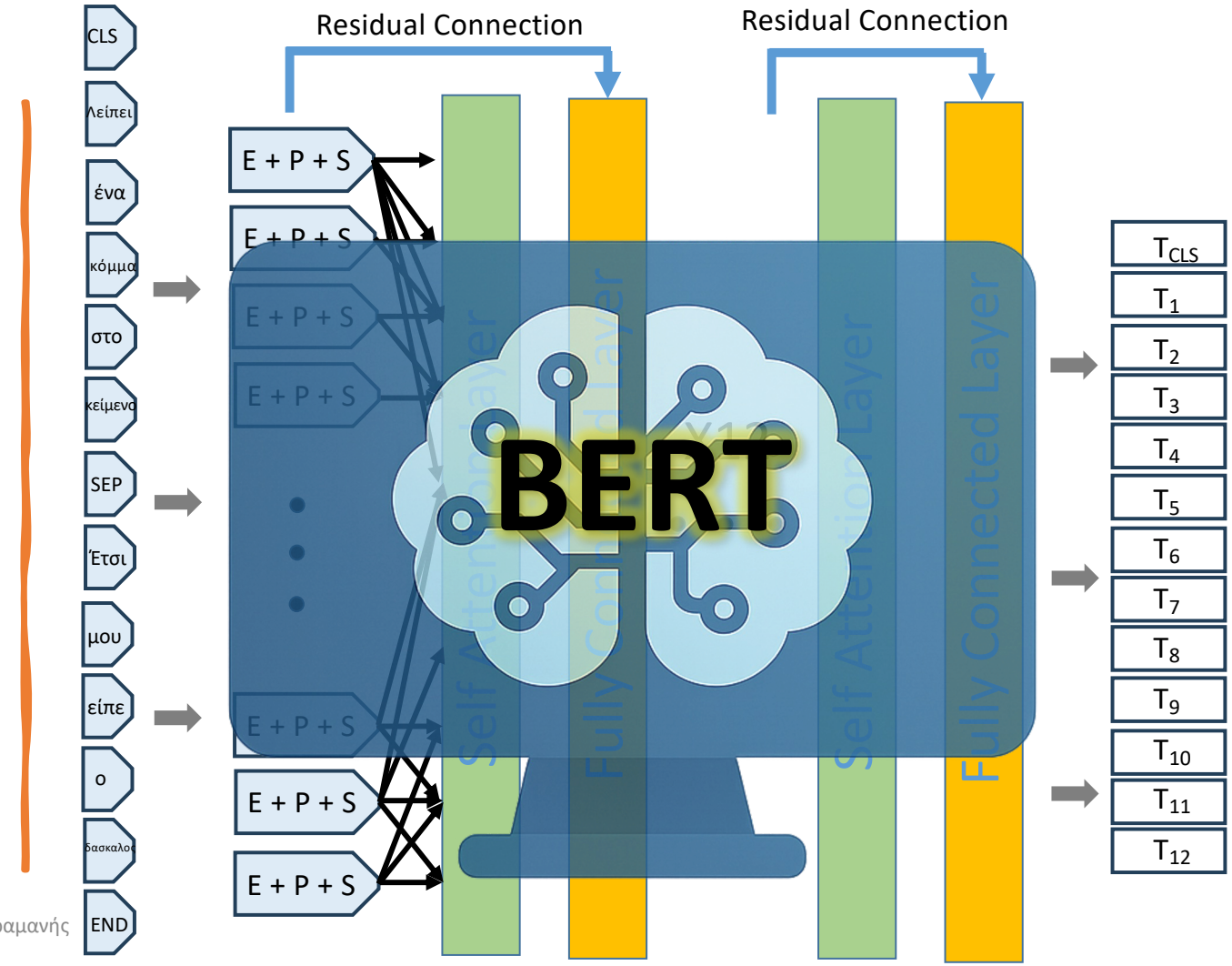
Training and Masked Language Modeling (MLM)



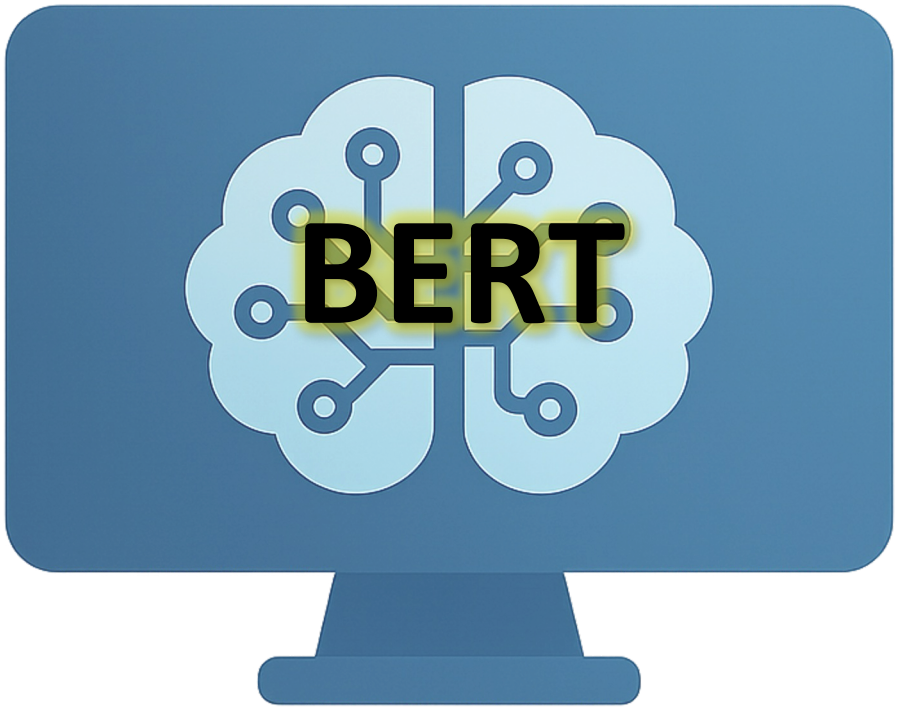
BERT Family: Αρχιτεκτονική του μοντέλου



BERT Family: Αρχιτεκτονική του μοντέλου



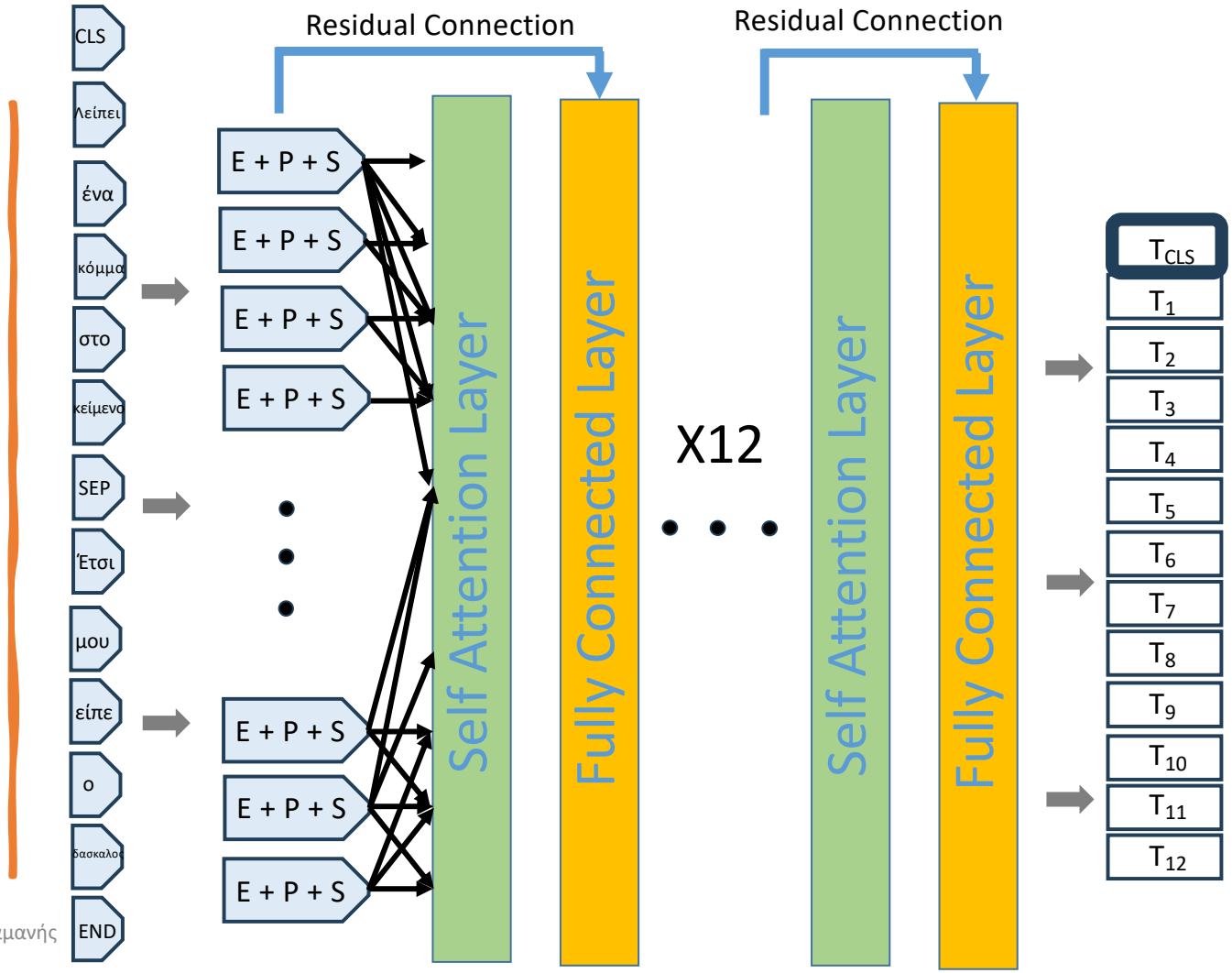
Το Masked Language Modeling εκπαιδεύει ένα ισχυρό μοντέλο, με τη δυνατότητα να προσαρμοστεί ώστε να λύσει μια ποικιλία προβλημάτων ταξινόμησης.



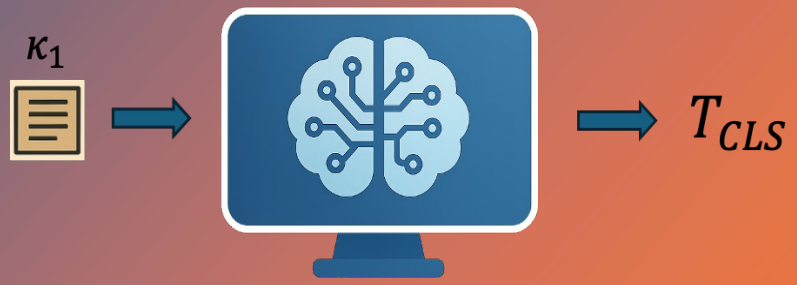
BERT Family: Αρχιτεκτονική του μοντέλου

Παρατηρήστε το $T_{CLS} \in \mathbb{R}^{768}$ τελικό token. Λόγω του "attention" του encoder, αυτό το token ενσωματώνει πληροφορία («κατανόηση») του κειμένου που εισήγαμε.

Χρησιμοποιείται για προβλήματα ταξινόμησης σε φυσική γλώσσα.



Εφαρμογές του BERT and T_{CLS}

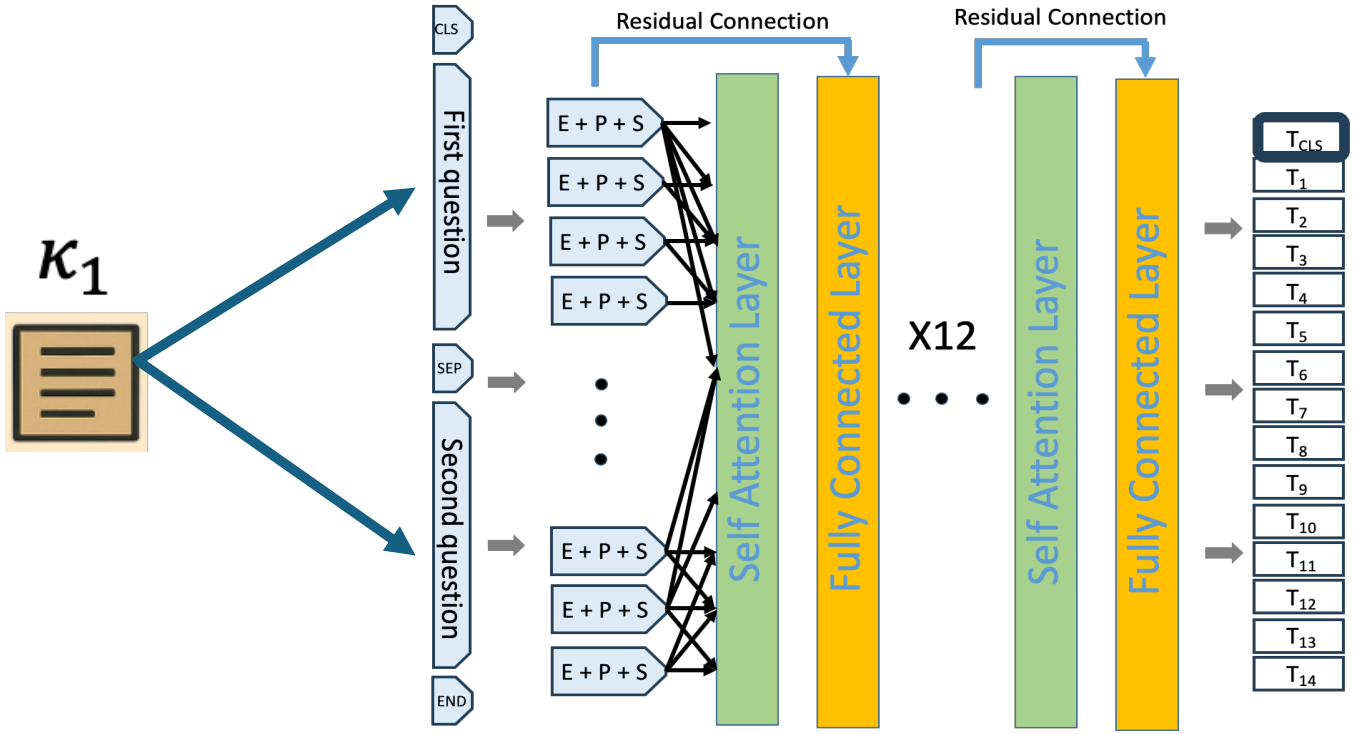


1ο Παράδειγμα: «Quora Question Pairs»: Δεδομένων δύο ερωτήσεων στην πλατφόρμα Quora, πρέπει να προσδιορίσουμε εάν διατυπώνουν ουσιαστικά την ίδια ερώτηση. (binary classification)

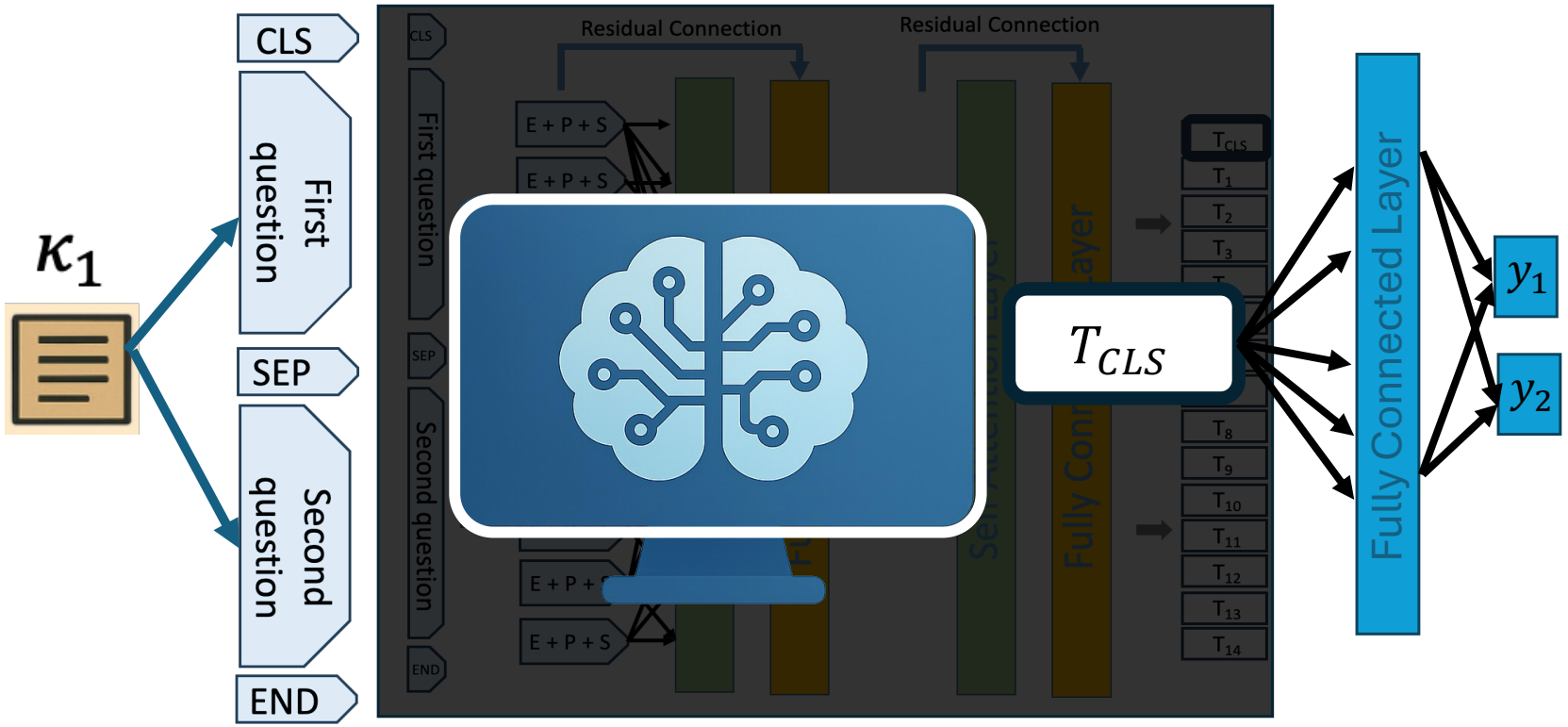
2ο Παράδειγμα: Το «Corpus of Linguistic Acceptability»: Δεδομένης μιας πρότασης (στα αγγλικά), πρέπει να προσδιορίσουμε εάν είναι γλωσσικά (όχι μόνο γραμματικά) αποδεκτή. (binary classification).

Για τα παραπάνω παραδείγματα: τι είναι το κ_1 και πως χρησιμοποιούμε το τελικό token T_{CLS}?

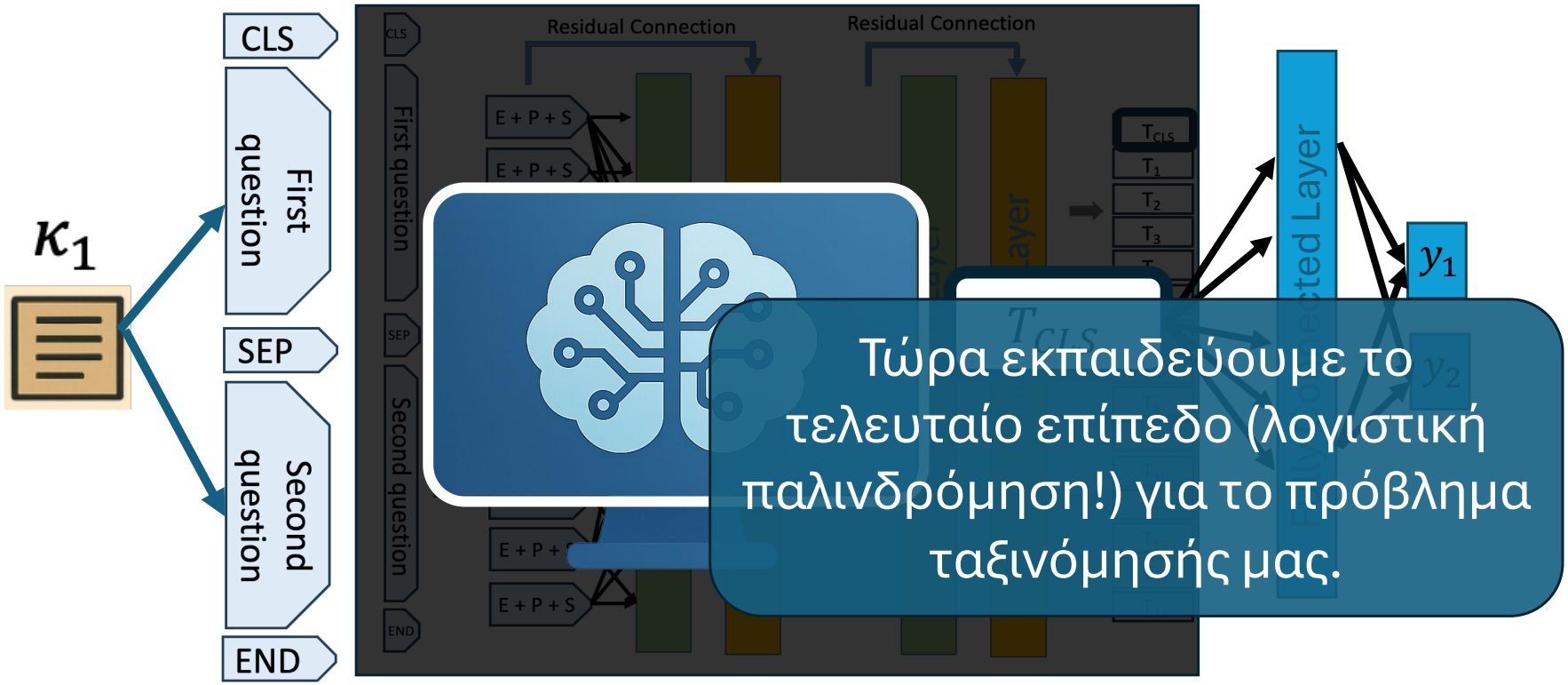
Παράδειγμα 1: Quora Question Pairs: Ισοδυναμούν δύο ερωτήσεις; (binary classification)



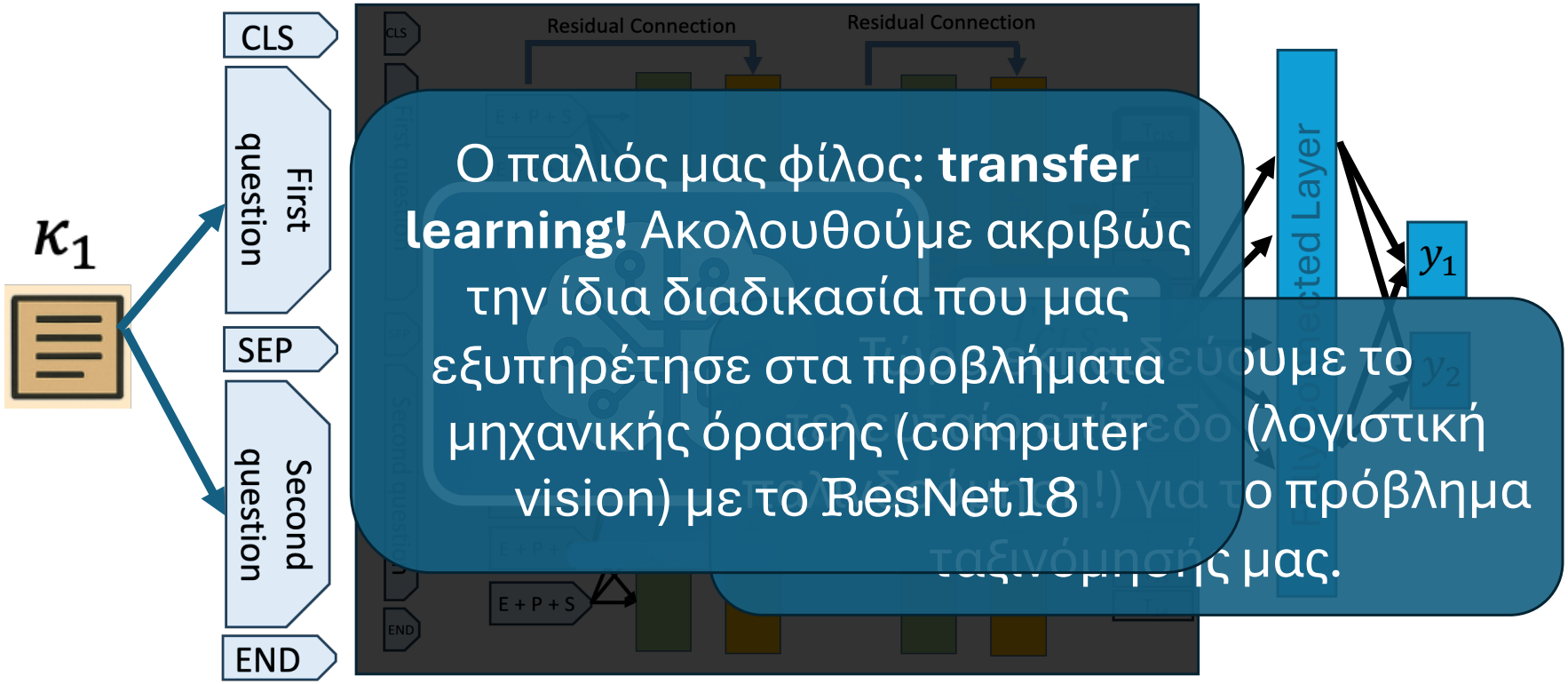
Παράδειγμα 1: Quora Question Pairs: Ισοδυναμούν δύο ερωτήσεις; (binary classification)



Παράδειγμα 1: Quora Question Pairs: Ισοδυναμούν δύο ερωτήσεις; (binary classification)



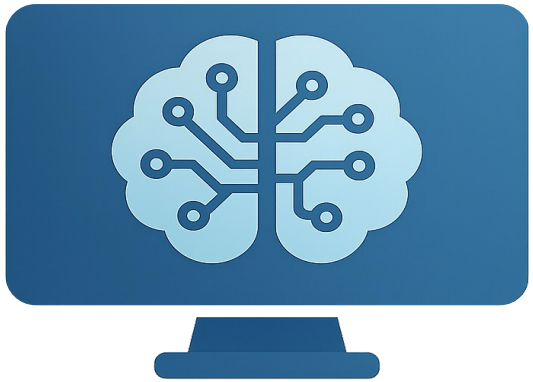
Παράδειγμα 1: Quora Question Pairs: Ισοδυναμούν δύο ερωτήσεις; (binary classification)



Το BERT στην Python

```
model = AutoModelForMaskedLM.from_pretrained("google-bert/bert-base-uncased")
```

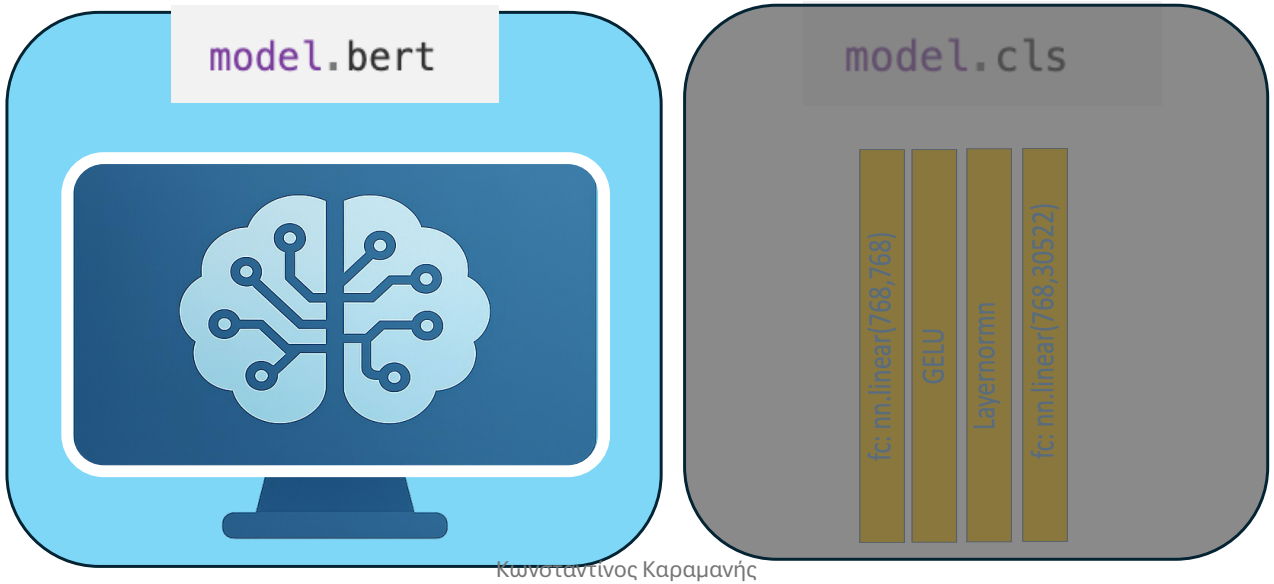
`model.bert`



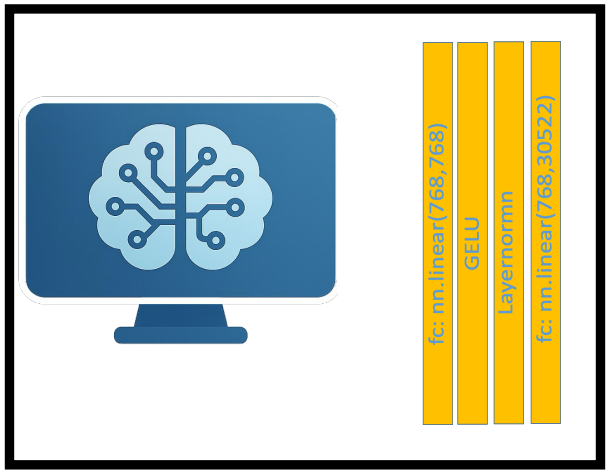
`model.cls`

Το BERT στην Python

```
model = AutoModelForMaskedLM.from_pretrained("google-bert/bert-base-uncased")
```

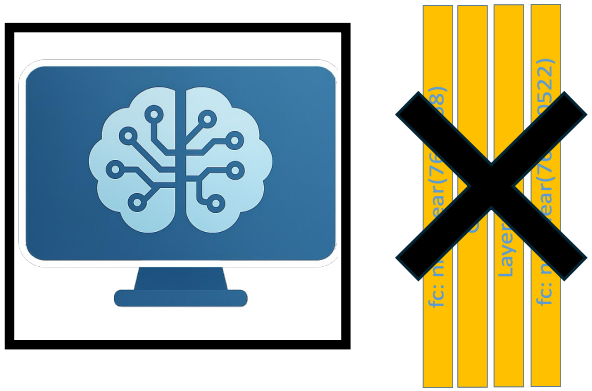


Το BERT στην Python



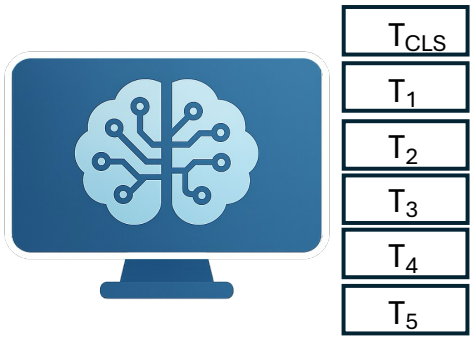

```python
class BertClassifier(torch.nn.Module):
    def __init__(self, model_name):
        super().__init__()
        self.bert = AutoModelForMaskedLM.from_pretrained(model_name).bert
        self.classifier = torch.nn.Linear(self.bert.config.hidden_size, 2)

    def forward(self, **inputs):
        outputs = self.bert(**inputs)
        cls_output = outputs.last_hidden_state[:, 0, :]
        logits = self.classifier(cls_output)
        outputs["logits"] = logits
        return outputs
```

Το BERT στην Python



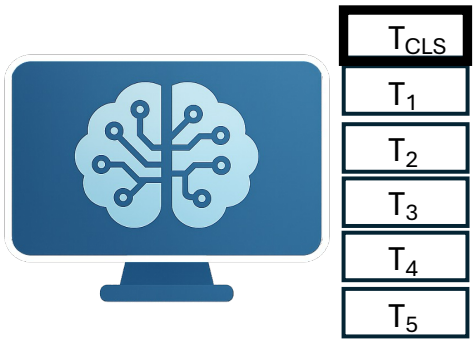
```python
class BertClassifier(torch.nn.Module):
    def __init__(self, model_name):
        super().__init__()
        self.bert = AutoModelForMaskedLM.from_pretrained(model_name).bert
        self.classifier = torch.nn.Linear(self.bert.config.hidden_size, 2)

    def forward(self, **inputs):
        outputs = self.bert(**inputs)
        cls_output = outputs.last_hidden_state[:, 0, :]
        logits = self.classifier(cls_output)
        outputs["logits"] = logits
        return outputs
```

Το BERT στην Python



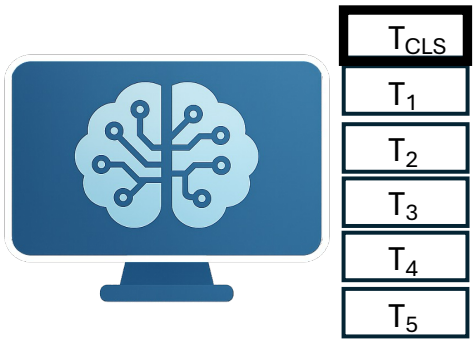

```python
class BertClassifier(torch.nn.Module):
    def __init__(self, model_name):
        super().__init__()
        self.bert = AutoModelForMaskedLM.from_pretrained(model_name).bert
        self.classifier = torch.nn.Linear(self.bert.config.hidden_size, 2)

    def forward(self, **inputs):
        outputs = self.bert(**inputs)
        cls_output = outputs.last_hidden_state[:, 0, :]
        logits = self.classifier(cls_output)
        outputs["logits"] = logits
        return outputs
```

Το BERT στην Python




```python
class BertClassifier(torch.nn.Module):
    def __init__(self, model_name):
        super().__init__()
        self.bert = AutoModelForMaskedLM.from_pretrained(model_name).bert
        self.classifier = torch.nn.Linear(self.bert.config.hidden_size, 2)

    def forward(self, **inputs):
        outputs = self.bert(**inputs)
        cls_output = outputs.last_hidden_state[:, 0, :]
        logits = self.classifier(cls_output)
        outputs["logits"] = logits
        return outputs
```

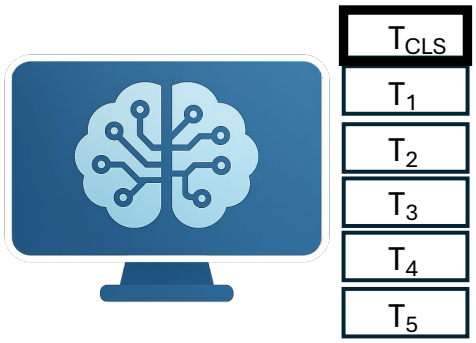
Το BERT στην Python




```python
class BertClassifier(torch.nn.Module):
    def __init__(self, model_name):
        super().__init__()
        self.bert = AutoModelForMaskedLM.from_pretrained(model_name).bert
        self.classifier = torch.nn.Linear(self.bert.config.hidden_size, 2)

    def forward(self, **inputs):
        outputs = self.bert(**inputs)
        cls_output = outputs.last_hidden_state[:, 0, :]
        logits = self.classifier(cls_output)
        outputs["logits"] = logits
        return outputs
```

Το «batch»: κρατάμε όλες τις προτάσεις

Το BERT στην Python

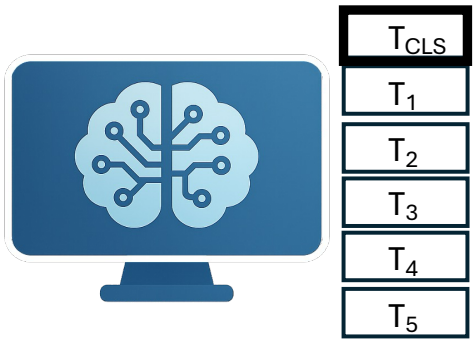


Τα «Tokens»: κρατάμε μόνο το πρώτο: T_{CLS}

```python
class BertClassifier(torch.nn.Module):
    def __init__(self, model_name):
        super().__init__()
        self.bert = AutoModelForMaskedLM.from_pretrained(model_name).bert
        self.classifier = torch.nn.Linear(self.bert.config.hidden_size, 2)

    def forward(self, **inputs):
        outputs = self.bert(**inputs)
        cls_output = outputs.last_hidden_state[:, 0, :]
        logits = self.classifier(cls_output)
        outputs["logits"] = logits
        return outputs
```

batch

Το BERT στην Python



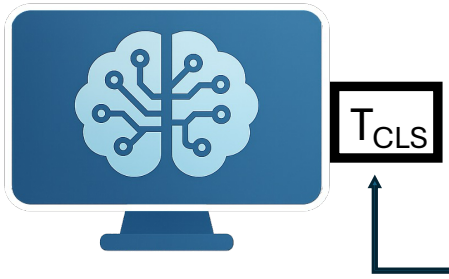
Tokens

```python
class BertClassifier(torch.nn.Module):
    def __init__(self, model_name):
        super().__init__()
        self.bert = AutoModelForMaskedLM.from_pretrained(model_name).bert
        self.classifier = torch.nn.Linear(self.bert.config.hidden_size, 2)

    def forward(self, **inputs):
        outputs = self.bert(**inputs)
        cls_output = outputs.last_hidden_state[:, 0, :]
        logits = self.classifier(cls_output)
        outputs["logits"] = logits
        return outputs
```

batch

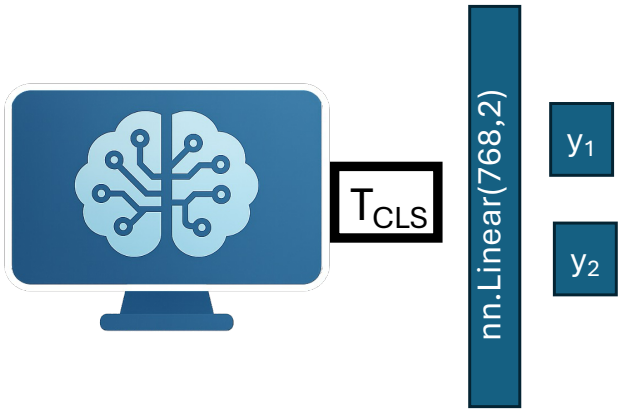
Οι συνιστώσες του διανύσματος

Το BERT στην Python



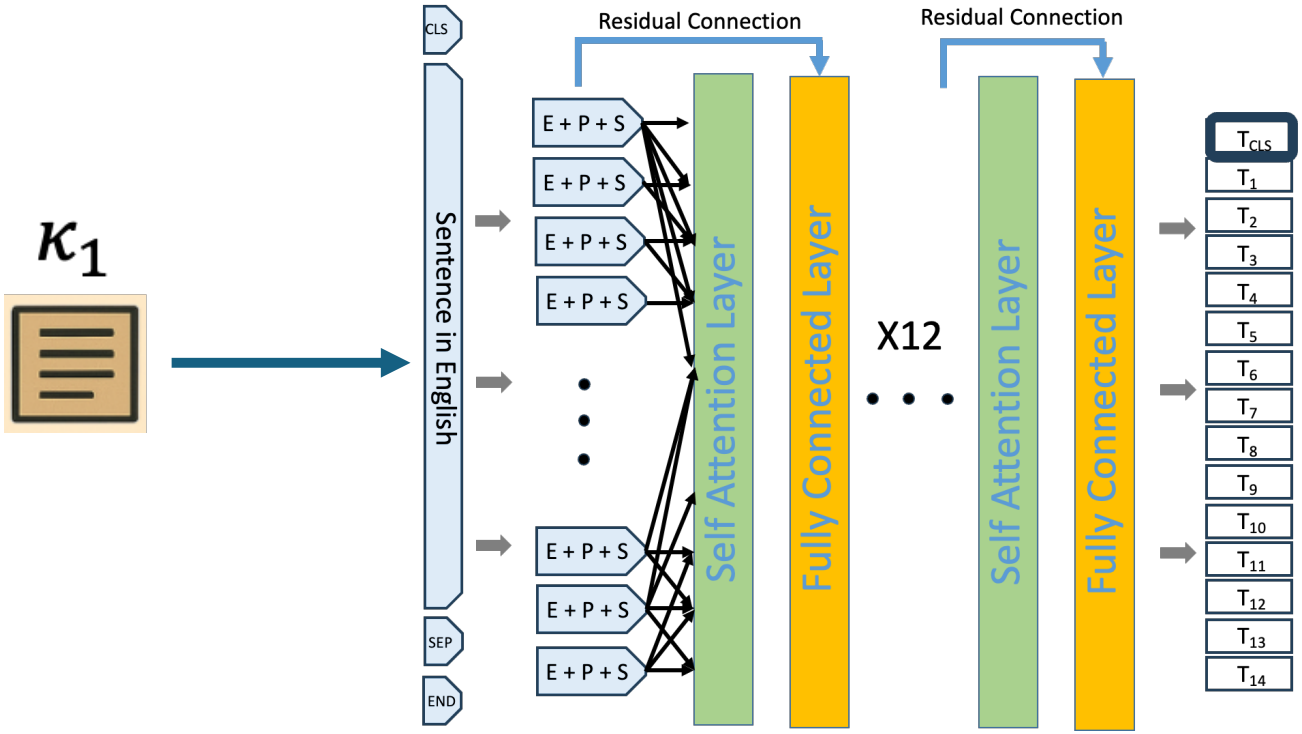

```python
class BertClassifier(torch.nn.Module):
    def __init__(self, model_name):
        super().__init__()
        self.bert = AutoModelForMaskedLM.from_pretrained(model_name).bert
        self.classifier = torch.nn.Linear(self.bert.config.hidden_size, 2)

    def forward(self, **inputs):
        outputs = self.bert(**inputs)
        cls_output = outputs.last_hidden_state[:, 0, :]
        logits = self.classifier(cls_output)
        outputs["logits"] = logits
        return outputs
```

Το BERT στην Python

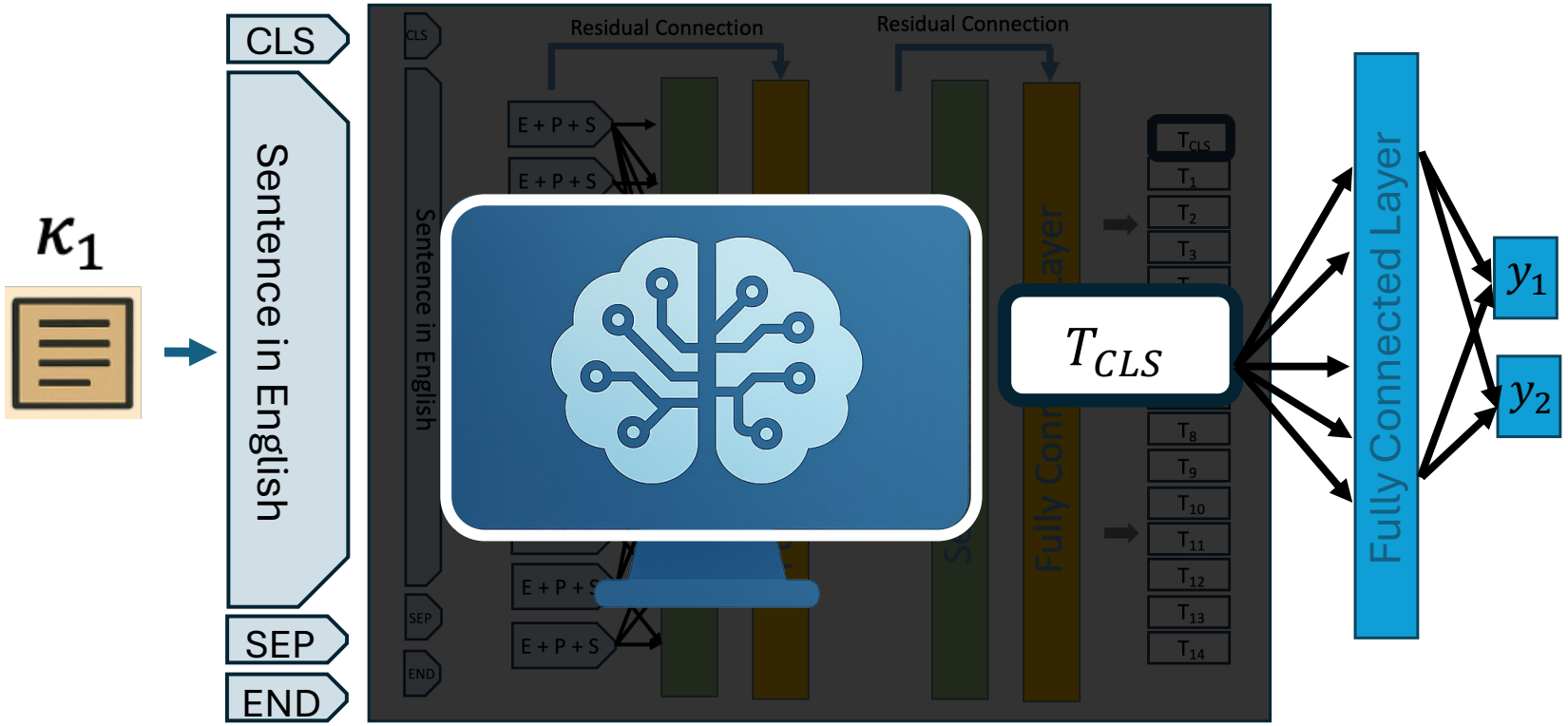


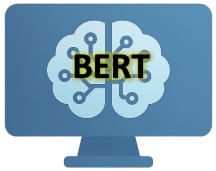

```python
class BertClassifier(torch.nn.Module):
    def __init__(self, model_name):
        super().__init__()
        self.bert = AutoModelForMaskedLM.from_pretrained(model_name).bert
        self.classifier = torch.nn.Linear(self.bert.config.hidden_size, 2)

    def forward(self, **inputs):
        outputs = self.bert(**inputs)
        cls_output = outputs.last_hidden_state[:, 0, :]
        logits = self.classifier(cls_output)
        outputs["logits"] = logits
        return outputs
```

Παράδειγμα 2: The Corpus of Linguistic Acceptability: Προσδιορίζουμε εάν μια πρόταση είναι γραμματικά και γλωσσικά αποδεκτή (binary classification).



Παράδειγμα 2: The Corpus of Linguistic Acceptability: Προσδιορίζουμε εάν μια πρόταση είναι γραμματικά και γλωσσικά αποδεκτή (binary classification).

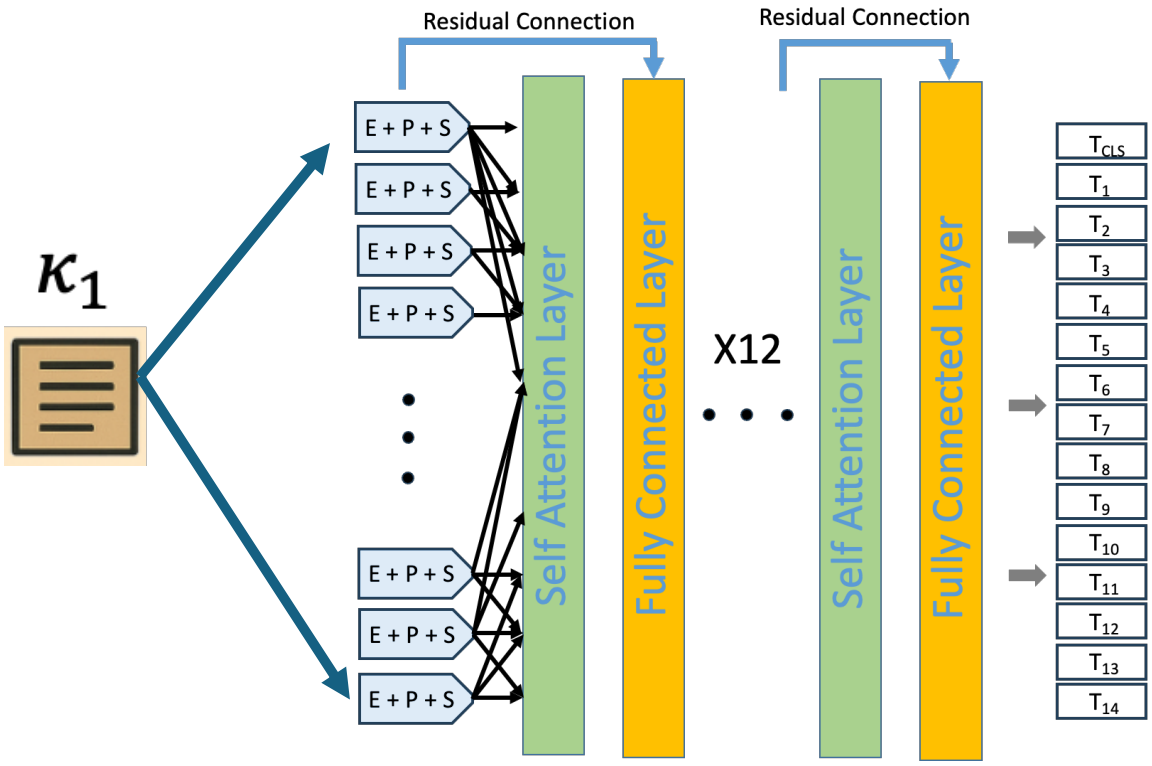




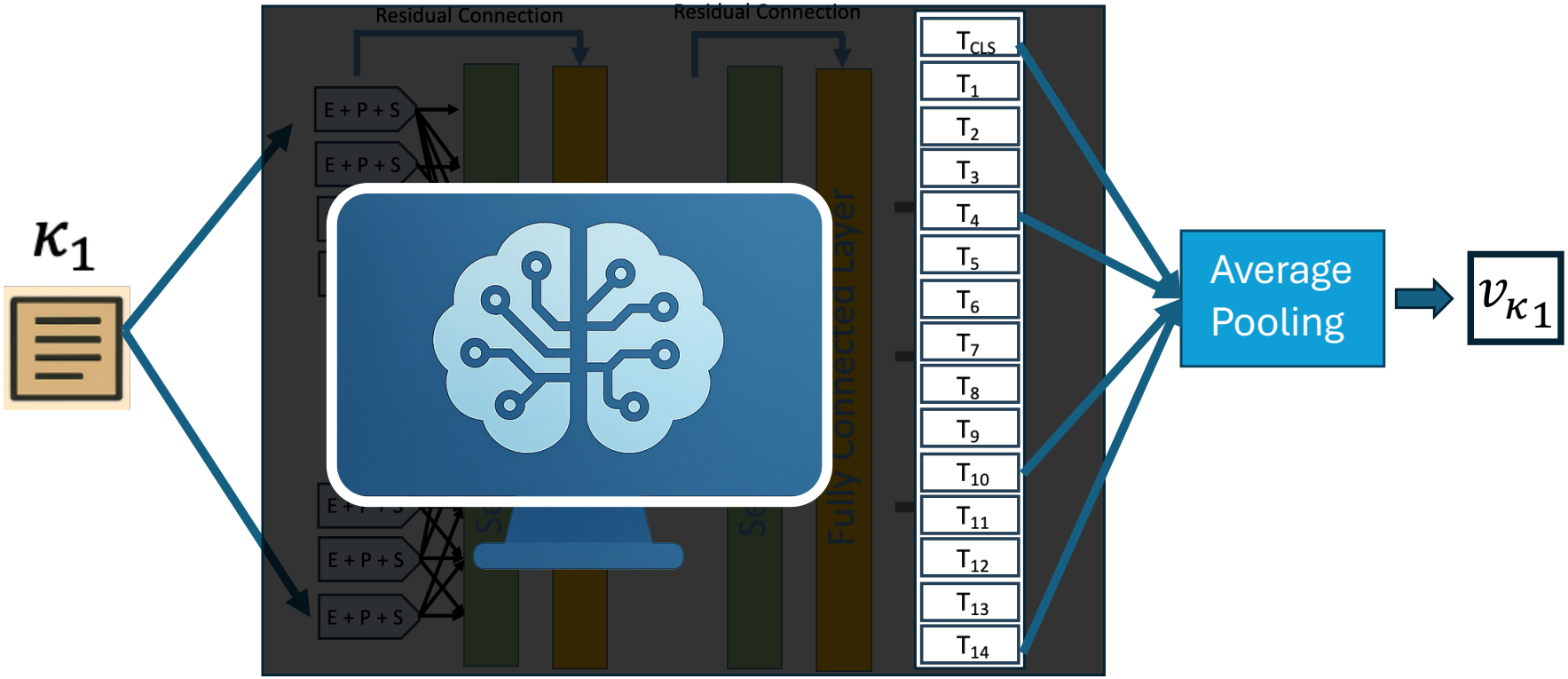
Το BERT έχει εκπαιδευθεί με το Masked Language Modeling (MLM) που απαιτεί μια «κατανόηση» της γλώσσας. Αποτελεί ένα σύστημα ευέλικτο που μπορεί να χρησιμοποιηθεί για να να λύσει μεγάλη ποικιλία προβλημάτων (με λίγη επιπρόσθετη εκπαίδευση – το transfer learning).

Επιστρέφουμε στο αρχικό μας πρόβλημα: Ενσωματώσεις

Επιστρέφουμε στο αρχικό μας πρόβλημα: Ενσωματώσεις



Επιστρέφουμε στο αρχικό μας πρόβλημα: Ενσωματώσεις



Το BERT στην Python

```
model = SentenceTransformer('bert-base-uncased', device=device)
```

Το BERT στην Python

```
model = SentenceTransformer('bert-base-uncased', device=device)
```

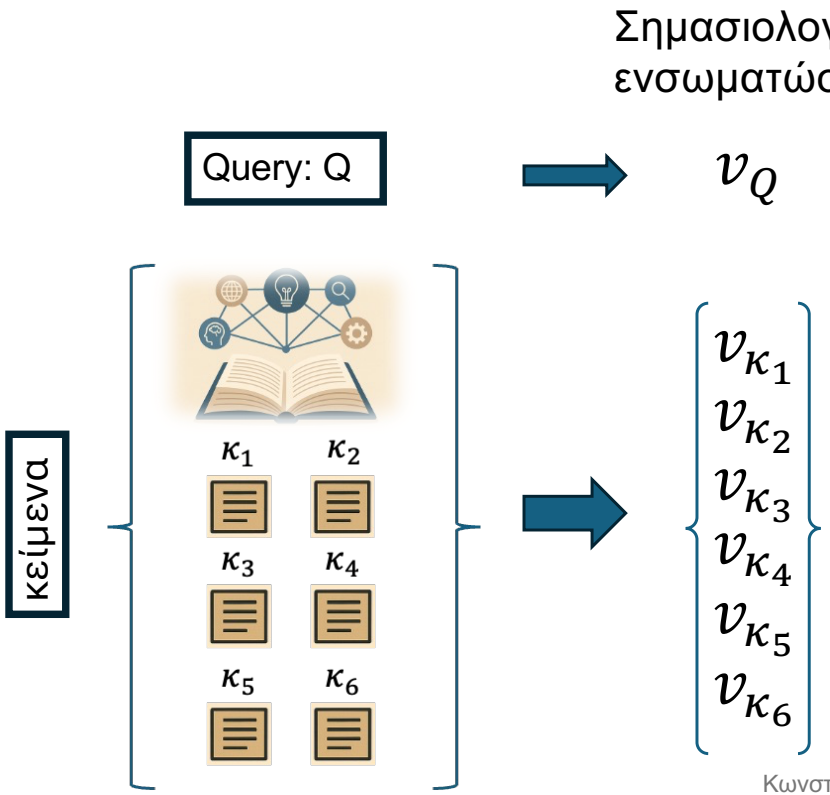
Το SentenceTransformer μας έκανε πολλά βήματα:

1. Κρατάει μόνο το BERT (όχι το MLM Head)
2. Προσθέτει όλα τα τελικά Tokens και παίρνει το μέσο όρο

Το BERT στην Python

```python
class BertEmbedder(nn.Module):
    def __init__(self, model_name):
        super().__init__()
        # AutoModel gives you the base transformer without any task-specific head
        self.bert = AutoModelForMaskedLM.from_pretrained(model_name).bert
        # self.bert = AutoModel.from_pretrained(model_name)  # equivalent to the above

    def mean_pool(self, token_embeddings, attention_mask):
        # token_embeddings: (batch, seq_len, hidden_dim)
        # attention_mask:   (batch, seq_len)
        mask = attention_mask.unsqueeze(-1).float()          # (batch, seq_len, 1)
        sum_embeddings = (token_embeddings * mask).sum(dim=1) # (batch, hidden_dim)
        sum_mask = mask.sum(dim=1).clamp(min=1e-9)            # (batch, 1)
        return sum_embeddings / sum_mask                      # (batch, hidden_dim)

    def forward(self, input_ids, attention_mask, **kwargs):
        outputs = self.bert(input_ids=input_ids, attention_mask=attention_mask, **kwargs)
        return self.mean_pool(outputs.last_hidden_state, attention_mask)
```

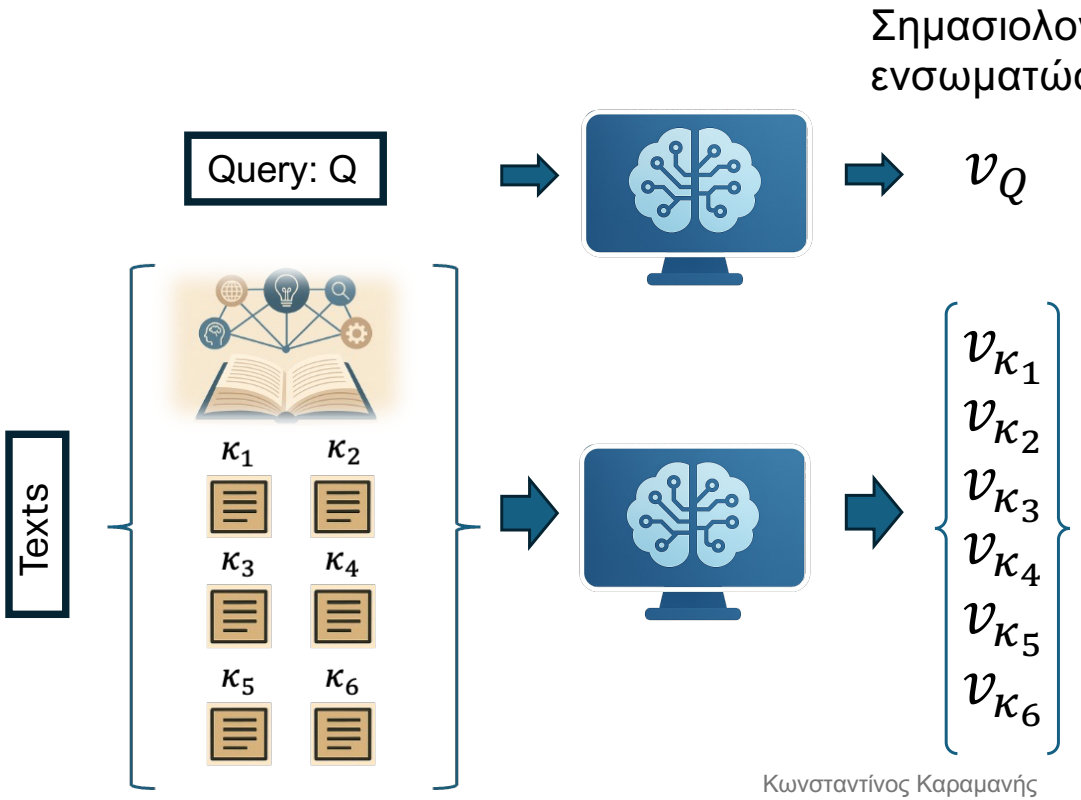
Αναζήτηση σε Βάση Γνώσεων



Κατατάσσουμε τα κείμενα σε φθίνουσα σειρά εσωτερικού γινομένου

$$\langle v_\kappa, v_Q \rangle$$

Αναζήτηση σε Βάση Γνώσεων



Κατατάσσουμε τα κείμενα σε φθίνουσα σειρά εσωτερικού γινομένου

$$\langle v_\kappa, v_Q \rangle$$

Αναζήτηση σε Βάση Γνώσεων