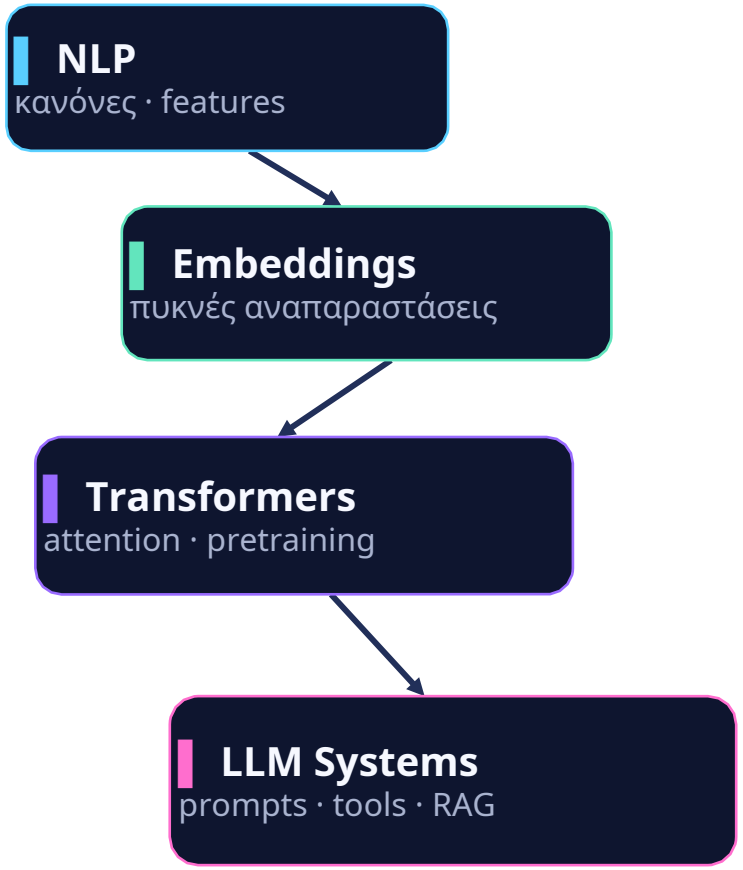
Από το NLP στα LLMs

Θεμελιώδεις έννοιες, τεχνικές,
αρχιτεκτονικές και πρακτικά συστήματα



Συμεών Συμεωνίδης
Βασίλης Περηφάνης

14/03/2026

Στόχοι μάθησης και τρόπος χρήσης του υλικού

1. Βάσεις

- Να ξεχωρίζουμε αναπαράσταση, μοντέλο και pipeline.
- Να βλέπουμε πού ταιριάζουν preprocessing, features, embeddings και sequence models.
- Να συνδέουμε τα παλιά NLP tasks με τα σημερινά generative workflows.

2. Αρχιτεκτονικές

- Να εξηγούμε attention, tokenization, pretraining και transformer blocks χωρίς υπερβολικό μαθηματικό βάθος.
- Να κατανοούμε τι ακριβώς είναι ένα LLM και γιατί η κλιμάκωση άλλαξε το πεδίο.
- Να διαβάζουμε σωστά όρους όπως context window, decoding, instruction tuning.

3. Εφαρμογές

- Να διαλέγουμε ανάμεσα σε prompt engineering, RAG, fine-tuning και tool use.
- Να συζητάμε latency, cost, safety, evaluation και deployment με επιχειρησιακούς όρους.
- Να μπορούμε να περιγράψουμε μια ρεαλιστική αρχιτεκτονική λύσης.

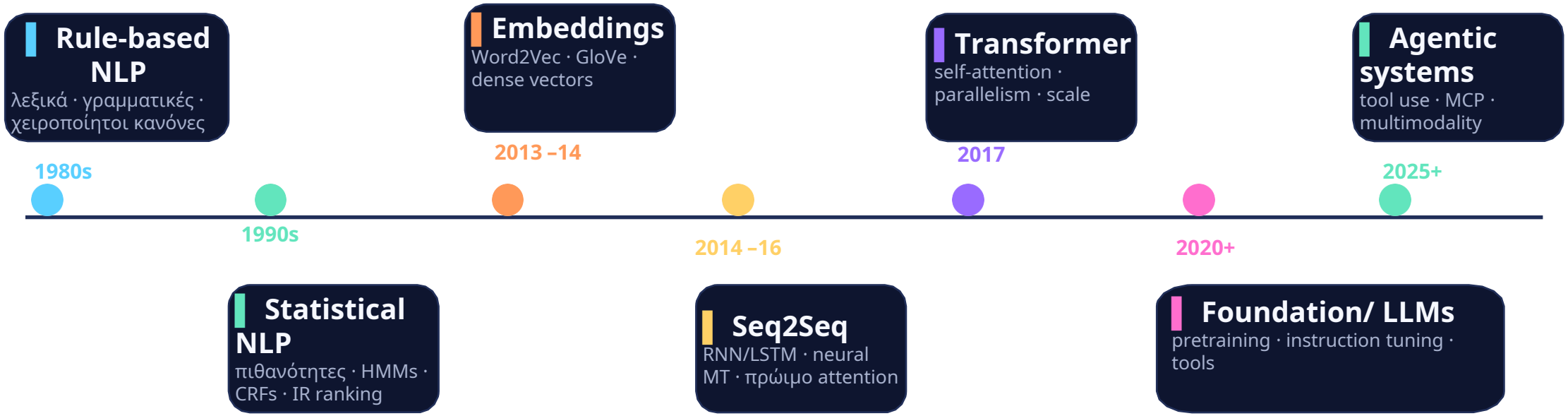
Τι δεν προσπαθεί να καλύψει αυτή η παρουσίαση

Δεν είναι μάθημα βαθιάς μαθηματικής απόδειξης ούτε workshop GPU engineering. Είναι ένα αναλυτικό, καθαρό mental model για να φτάσουμε από το κλασικό NLP στα σύγχρονα LLM systems.

Τι είναι το NLP;

- Πεδίο της Τεχνητής Νοημοσύνης
- Εστιάζει στην αλληλεπίδραση ανθρώπου–υπολογιστή μέσω γλώσσας
- Κατανόηση και παραγωγή φυσικής γλώσσας

Από τους κανόνες στα foundation models: μια σύντομη ιστορική διαδρομή



Εφαρμογές NLP

- Μηχανική μετάφραση
- Chatbots
- Ανάλυση συναισθήματος
- Question Answering
- Εξαγωγή πληροφοριών

Προκλήσεις στη γλώσσα

Γιατί η φυσική γλώσσα είναι δύσκολη για τα υπολογιστικά συστήματα

Παράδειγμα πρότασης

«Μπορείς να ανοίξεις το παράθυρο;»

Ένας άνθρωπος ακούει αίτημα.
Ένα απλό σύστημα μπορεί να την εκλάβει ως ερώτηση ικανότητας.

Το ίδιο string απαιτεί διαφορετικά επίπεδα ερμηνείας.

Μορφολογία

κλίσεις, καταλήξεις, σύνθετες λέξεις, υποκοριστικά, χρόνοι και πρόσωπα

Σύνταξη

ποιος είναι το υποκείμενο, ποιο το αντικείμενο και πώς συνδέονται μεταξύ τους

Σημασιολογία

πολυσημία, συνωνυμία, αναφορές όπως «εκείνη», «αυτό», «η τράπεζα»

Πραγματολογία

πρόθεση, ειρωνεία, κοινωνικές συμβάσεις, συμφραζόμενα και γνώση κόσμου

Κλασικές εργασίες NLP και πώς επανεμφανίζονται στην εποχή των LLMs

Classification
«Το feedback είναι θετικό ή αρνητικό;»

NER
«Η Microsoft συνεργάζεται με τον ΟΤΕ.»

Machine Translation
«Καλημέρα» → «Good morning»

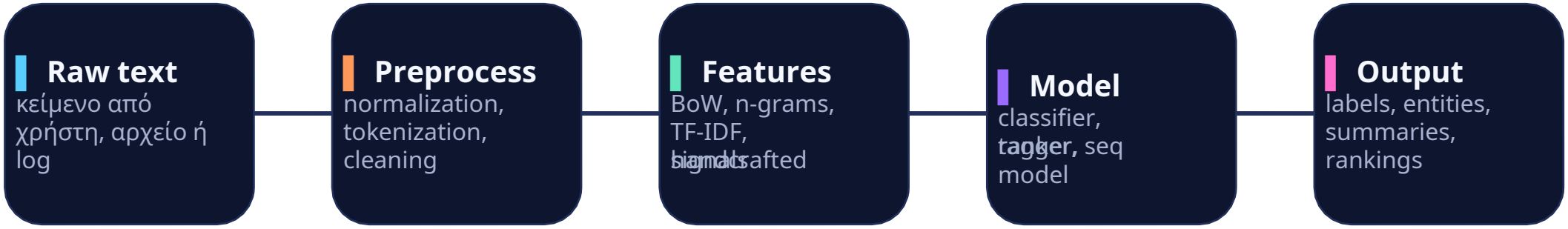
Summarization
Μακρύ report → σύντομη περίληψη

Question Answering
«Πότε λήγει η σύμβαση;»

Semantic Search
Βρίσκω παρόμοια αποσπάσματα σε βάση γνώσης

Σήμερα πολλές από αυτές τις εργασίες “κρύβονται” πίσω από ένα chat interface, αλλά εξακολουθούν να υπάρχουν ως υπο-προβλήματα.

Το κλασικό pipeline NLP



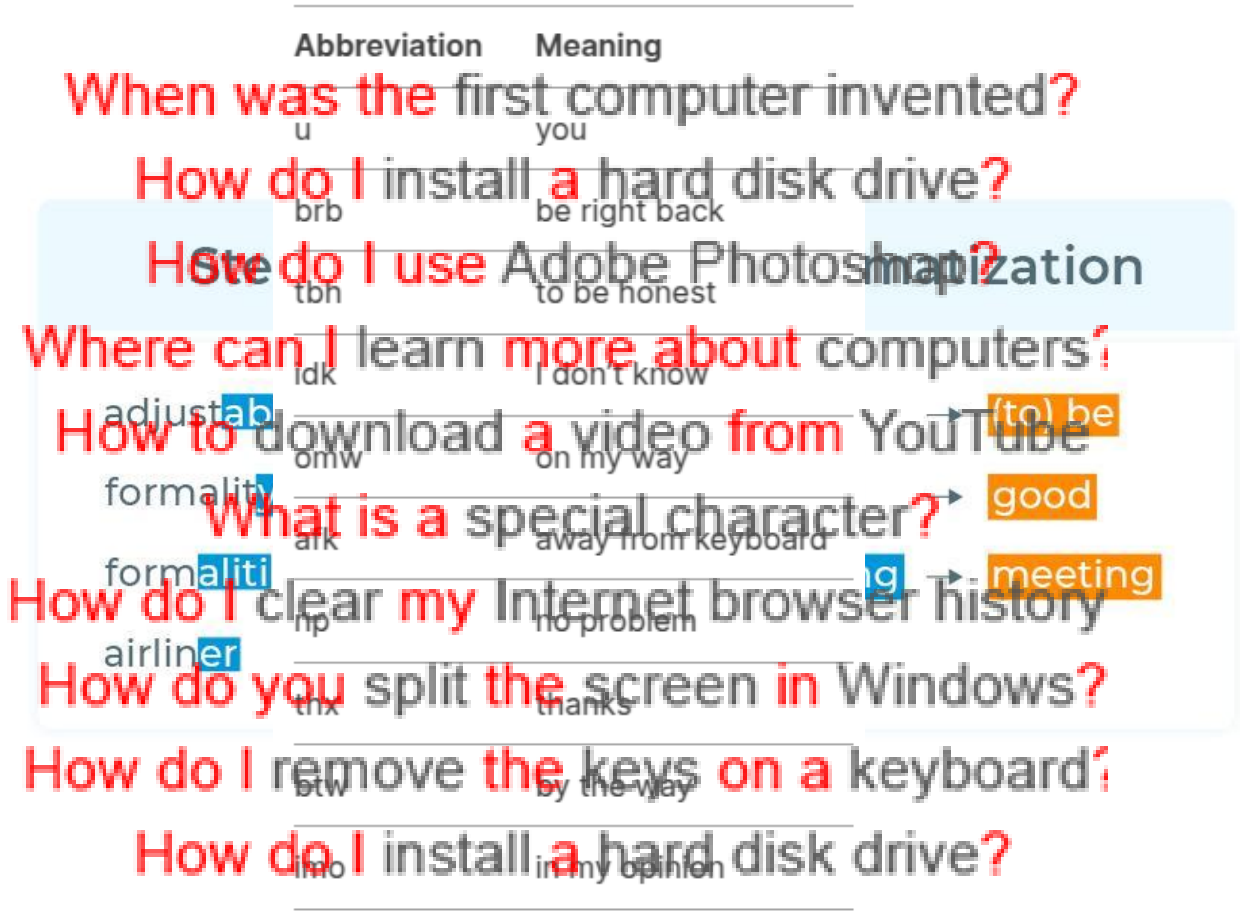
Τι κερδίζαμε

Έλεγχο, ερμηνευσιμότητα, μικρότερα μοντέλα, σαφείς ενδιάμεσες αναπαραστάσεις.

Τι πληρώναμε

Πολλή χειρωνακτική feature engineering, domain-specific tuning και αδύναμη γενίκευση σε νέες περιπτώσεις.

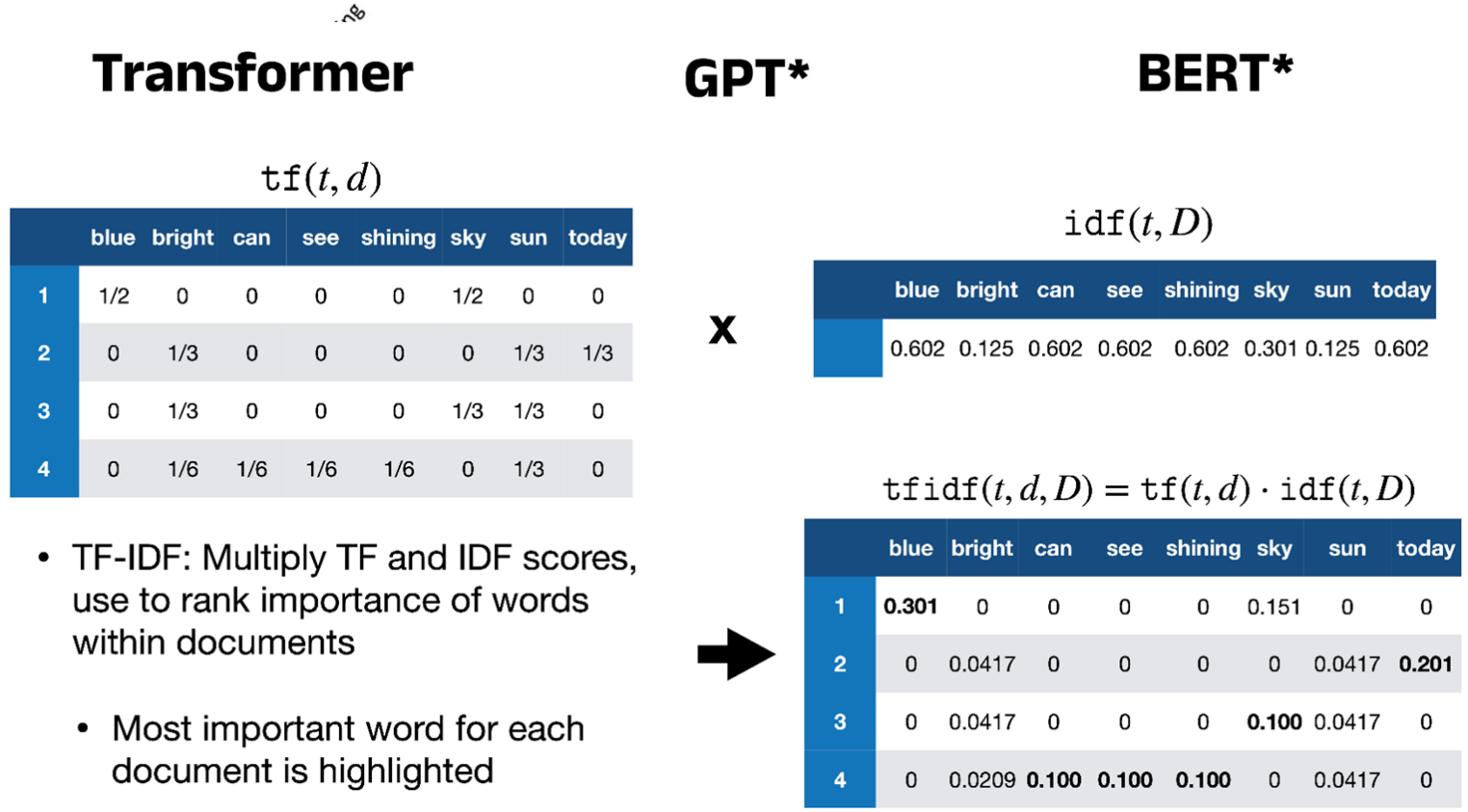
- Stopwords,normalization
- Tokenization
- Part-of-speech (POS)
- Lemmatization / stemming
- Emojis,slang



Επεξεργασία Δεδομένων (Preprocessing)

Αναπαράσταση Κειμένου

- Bag of Words / TF-IDF
- Word embeddings (Word2Vec, GloVe)
- Contextual embeddings (BERT, GPT)



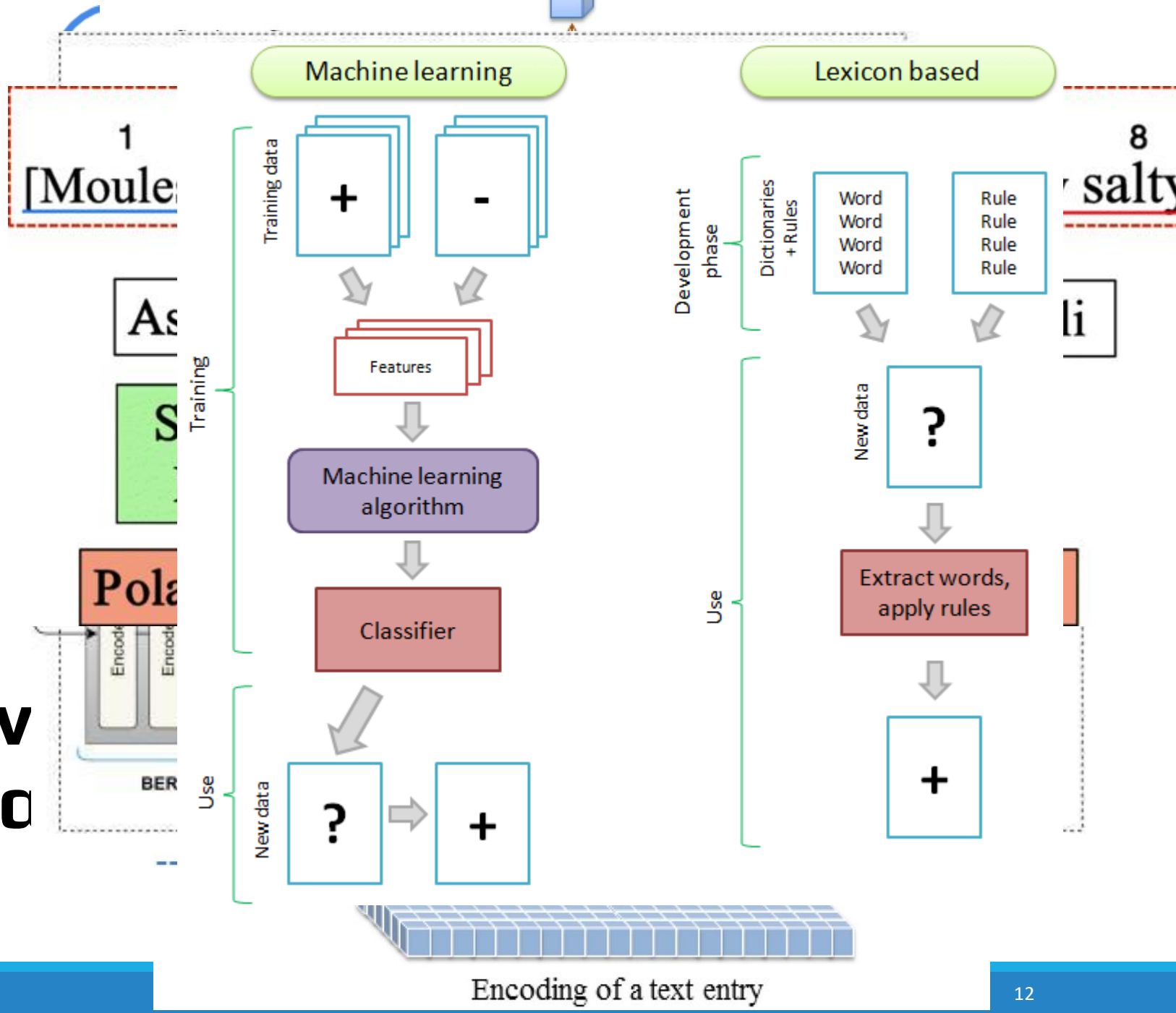
	blue	bright	can	see	shining	sky	sun	today
1	1/2	0	0	0	0	1/2	0	0
2	0	1/3	0	0	0	0	1/3	1/3
3	0	1/3	0	0	0	1/3	1/3	0
4	0	1/6	1/6	1/6	1/6	0	1/3	0

	blue	bright	can	see	shining	sky	sun	today
	0.602	0.125	0.602	0.602	0.602	0.301	0.125	0.602

	blue	bright	can	see	shining	sky	sun	today
1	**0.301**	0	0	0	0	0.151	0	0
2	0	0.0417	0	0	0	0	0.0417	**0.201**
3	0	0.0417	0	0	0	**0.100**	0.0417	0
4	0	0.0209	**0.100**	**0.100**	**0.100**	0	0.0417	0

*Illustrative example, exact model architecture may vary slightly

- Lexicon-based (positive/negative dictionaries)
- Classical ML: Naive Bayes, SVM
- Deep Learning: LSTM, GRU
- Transformers: BERT, RoBERTa
- Aspect-based Sentiment



Preprocessing: normalization, tokenization, lemmatization

Παράδειγμα εισόδου

«Τα LLMs, τελικά, ΔΕΝ “καταλαβαίνουν” όπως εμείς!!!»

Normalize
πεζά/κεφαλαία, σημεία στίξης, unicode

Tokenize
χωρισμός σε λέξεις ή subwords

Lemma / stem
μείωση παραλλαγών όπου χρειάζεται

Filter
stopwords, hashtags, URLs, noise

Ενδεικτικό αποτέλεσμα

tokens = ["τα", "llms", "τελικά", "δεν", "καταλαβαίνουν", "όπως", "εμείς"]

lemma-like μορφή = ["LLM", "τελικά", "δεν", "καταλαβαίνω", "εγώ"]

Σημείωση: σε σύγχρονα LLM pipelines δεν αφαιρούμε πάντα stopwords· εξαρτάται από το μοντέλο και τον στόχο.

Αναπαράσταση κειμένου: Bag-of-Words, n- grams και TF-IDF

BoW / n-gram λογική

Η πρόταση «το μοντέλο παράγει κείμενο» γίνεται vector πάνω σε vocabulary.

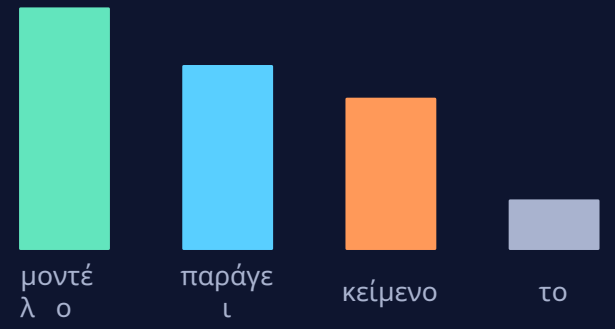
Με n-grams κρατάμε λίγη τοπική σειρά: «το μοντέλο», «παράγει κείμενο».

Mini vocabulary

term	count
το	1
μοντέλο	1
παράγει	1
κείμενο	1
δεδομένα	0

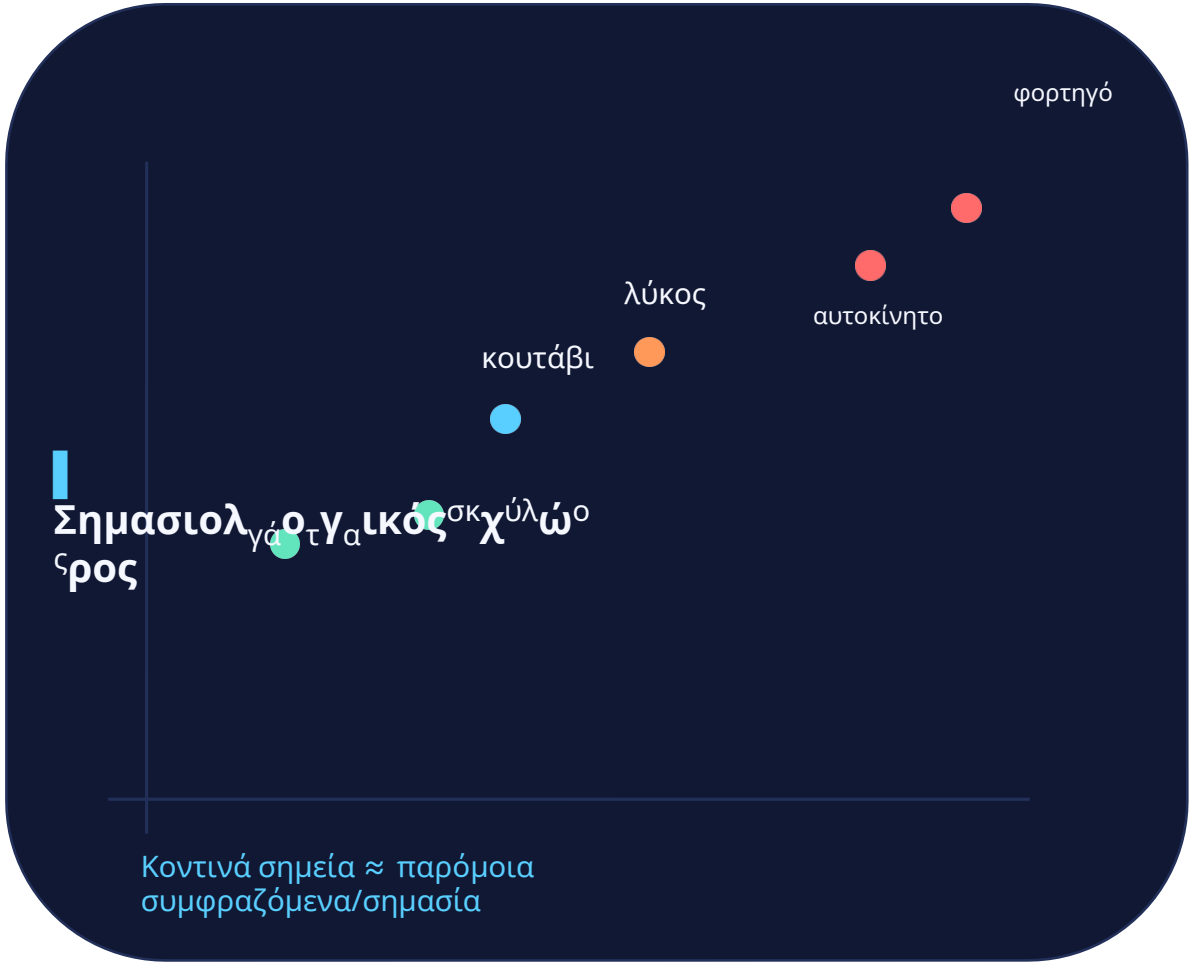
TF-IDF intuition

Μια λέξη παίρνει μεγαλύτερο βάρος όταν είναι συχνή μέσα στο συγκεκριμένο κείμενο αλλά όχι σε όλα τα κείμενα του corpus



Ισχυρή baseline, αλλά sparse vectors, ασθενής σημασιολογία και περιορισμένη κατανόηση συμφραζομένων.

Embeddings: από sparse vectors σε πυκνές σημασιολογικές αναπαραστάσεις



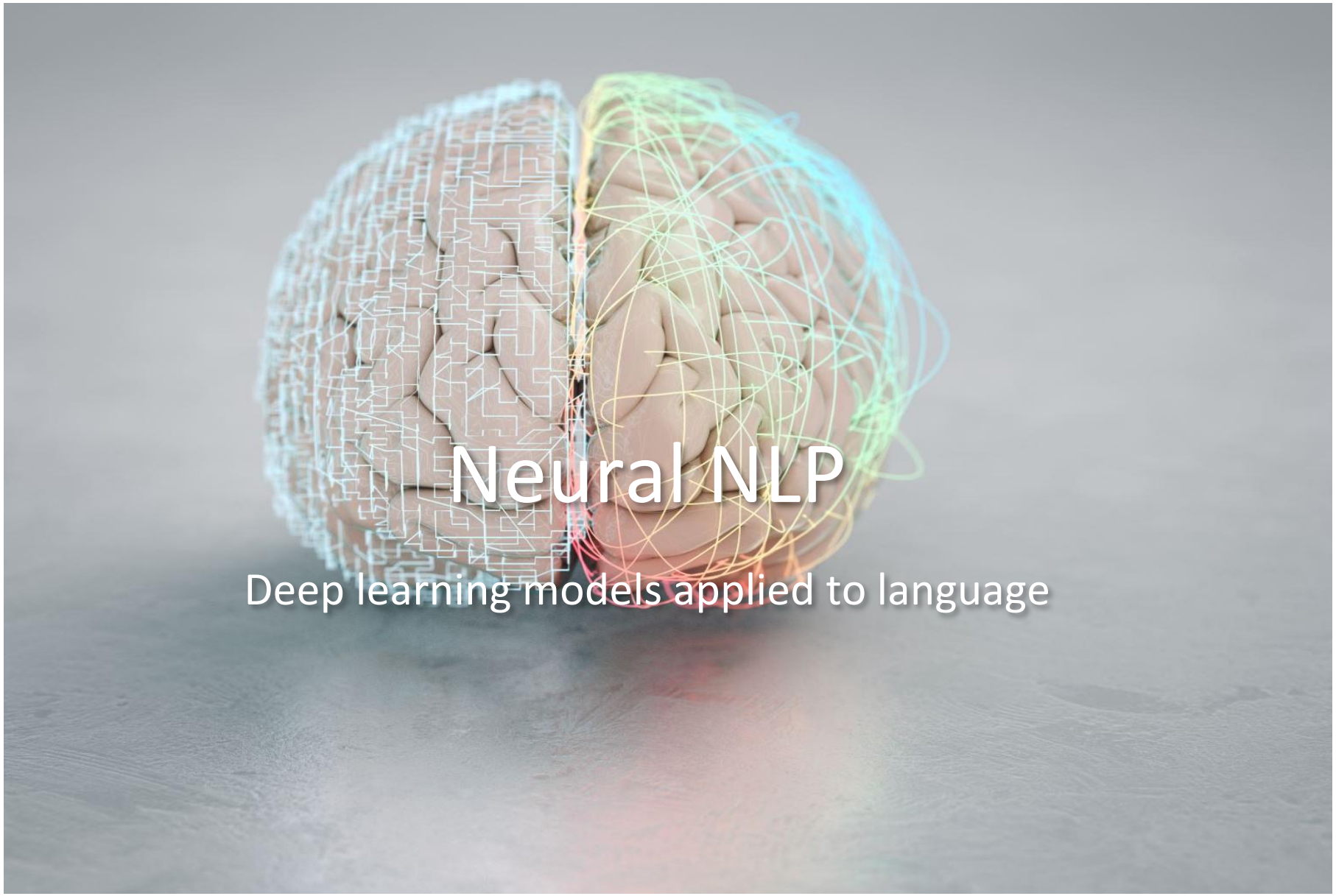
Τι κερδίζουμε

- dense vectors μικρής διάστασης
- σημασιολογική ομοιότητα
- μεταφορά γνώσης σε downstream tasks

Τι να προσέχω

- κλασικά embeddings είναι context-independent
- οι προκαταλήψεις των δεδομένων περνούν στα vectors
- η πολυσημία παραμένει δύσκολη

Αναλογία: βασιλιάς − άνδρας + γυναίκα ≈ βασίλισσα


Neural NLP
Deep learning models applied to language

Recurrent Neural Networks

Τι είναι

- Νευρωνικά δίκτυα σχεδιασμένα για **sequential data**
- Διατηρούν **μνήμη προηγούμενων βημάτων** μέσω επαναληπτικών συνδέσεων

Πώς λειτουργούν

- Η έξοδος ενός χρονικού βήματος τροφοδοτεί το επόμενο
- Το μοντέλο χρησιμοποιεί **προηγούμενο context** για να επεξεργαστεί νέα δεδομένα

Κύρια χρήση

- **Language modelling**: πρόβλεψη της επόμενης λέξης σε μια πρόταση

Παραδείγματα εφαρμογών

- Natural Language Processing
- Speech recognition
- Machine translation
- Time-series prediction

LSTM

Το **LSTM** είναι ένας τύπος **Recurrent Neural Network (RNN)** που σχεδιάστηκε για να μαθαίνει **μακροχρόνιες εξαρτήσεις (long-term dependencies)** σε ακολουθιακά δεδομένα.

- Αντιμετωπίζει το πρόβλημα του **vanishing gradient** που εμφανίζεται στα κλασικά RNNs.
- Διαθέτει μια **memory cell (cell state)** που επιτρέπει την αποθήκευση και μεταφορά πληροφορίας σε μεγάλα χρονικά διαστήματα.

Η λειτουργία του ελέγχεται από τρεις βασικές πύλες (gates):

- **Forget Gate** – αποφασίζει ποια πληροφορία θα διαγραφεί από τη μνήμη
- **Input Gate** – καθορίζει ποια νέα πληροφορία θα αποθηκευτεί
- **Output Gate** – ελέγχει ποια πληροφορία θα μεταφερθεί στο επόμενο βήμα

Χρησιμοποιείται ευρέως σε:

- **Natural Language Processing (NLP)**
- **Machine Translation**
- **Speech Recognition**

GRU

Το **GRU (Gated Recurrent Unit)** είναι ένας τύπος **Recurrent Neural Network (RNN)** που σχεδιάστηκε για να μαθαίνει εξαρτήσεις σε ακολουθιακά δεδομένα.

Αποτελεί μια **απλοποιημένη εκδοχή του LSTM**, με λιγότερες πύλες και μικρότερη υπολογιστική πολυπλοκότητα.

Χρησιμοποιεί κυρίως **δύο πύλες**:

- **Update gate**: ελέγχει πόση παλιά πληροφορία θα διατηρηθεί
- **Reset gate**: ελέγχει πόση προηγούμενη πληροφορία θα αγνοηθεί

Δεν διαθέτει ξεχωριστό **cell state** όπως το LSTM· η μνήμη ενσωματώνεται απευθείας στο hidden state.

Πλεονεκτήματα:

- πιο απλή αρχιτεκτονική
- ταχύτερη εκπαίδευση

Χρησιμοποιείται σε:

- χρονοσειρές
- NLP

Συμπέρασμα:
Το GRU προσφέρει μια πιο ελαφριά και αποδοτική εναλλακτική του LSTM, διατηρώντας την ικανότητα μοντελοποίησης μακροχρόνιων εξαρτήσεων.

Προβλήματα των RNNs

- Τα **Recurrent Neural Networks (RNNs)** σχεδιάστηκαν για την επεξεργασία **sequential data**, όπως κείμενο, χρονοσειρές και ομιλία, λαμβάνοντας υπόψη την προηγούμενη πληροφορία μέσω επαναληπτικών συνδέσεων.
- Ένα βασικό πρόβλημα των RNNs είναι το **vanishing gradient problem**. Κατά την εκπαίδευση με backpropagation through time, τα gradients μπορεί να γίνουν πολύ μικρά, με αποτέλεσμα το μοντέλο να δυσκολεύεται να μάθει **μακροχρόνιες εξαρτήσεις** μέσα σε μεγάλες ακολουθίες.
- Επιπλέον, η εκπαίδευση των RNNs είναι **αργή**, επειδή η επεξεργασία γίνεται **διαδοχικά (sequentially)** και όχι παράλληλα, γεγονός που αυξάνει σημαντικά τον χρόνο εκπαίδευσης σε μεγάλα datasets.
- Τα προβλήματα αυτά περιορίζουν την ικανότητα των RNNs να διαχειρίζονται **μεγάλα context windows** και σύνθετες γλωσσικές δομές.

Περιορισμοί Κλασικού NLP

- Τα παραδοσιακά μοντέλα NLP βασίζονται συχνά σε **sparse vector representations** (π.χ. Bag-of-Words, TF-IDF), όπου κάθε λέξη αντιμετωπίζεται ως ανεξάρτητο χαρακτηριστικό χωρίς βαθύτερη γλωσσική κατανόηση.
- Η αναπαράσταση αυτή οδηγεί σε **υψηλής διάστασης και αραιά διανύσματα**, γεγονός που δυσκολεύει τη γενίκευση και αυξάνει την υπολογιστική πολυπλοκότητα.
- Τα κλασικά μοντέλα έχουν **περιορισμένη κατανόηση σημασιολογίας**, καθώς δεν μπορούν να αποτυπώσουν αποτελεσματικά τις σχέσεις μεταξύ λέξεων, συνωνύμων ή πολυσημίας.
- Επίσης εμφανίζουν **περιορισμένη αξιοποίηση του συμφραζομένου (context)**, καθώς συνήθως εξετάζουν λέξεις μεμονωμένα ή μέσα σε πολύ μικρά παράθυρα κειμένου.
- Ως αποτέλεσμα, τα συστήματα αυτά δυσκολεύονται να κατανοήσουν **τη συνολική σημασία μιας πρότασης ή ενός κειμένου**.

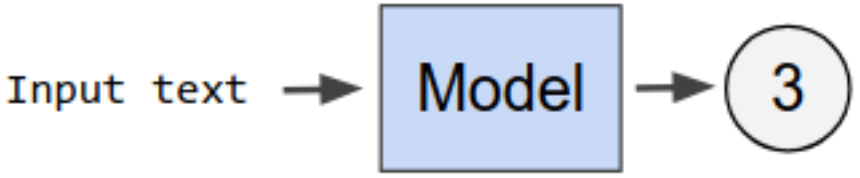
LLMs, RAG & Agentic AI

Agenda

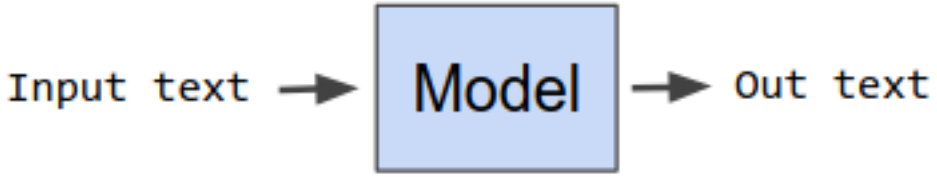
- Transformers
- LLMs
- Retrieval Augmented Generation
- Agentic AI

NLP tasks in a nutshell

- Ταξινόμηση
  - Ανάλυση συναισθήματος, Αναγνώριση πρόθεσης και γλώσσας



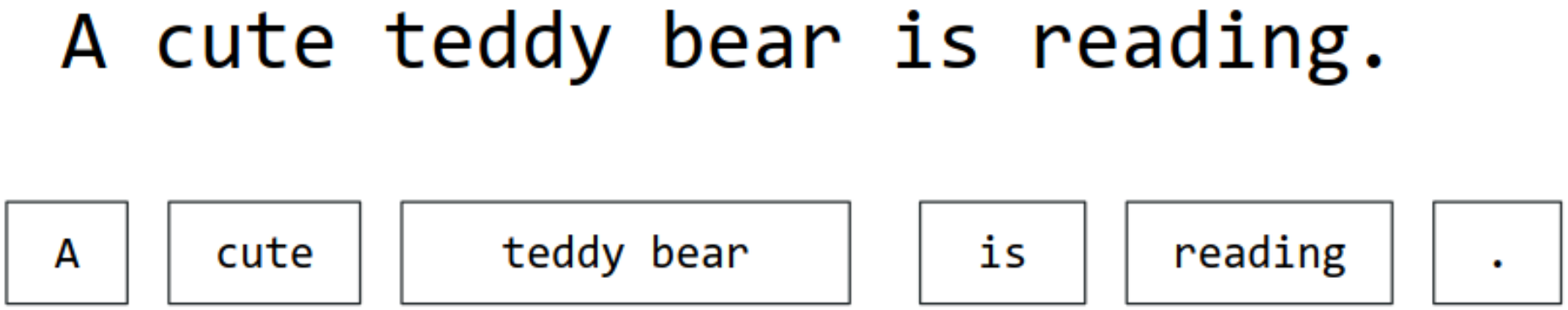
- Δημιουργία Κειμένου
  - Μετάφραση, απάντηση σε ερωτήσεις, δημιουργία περίληψης



Ιστορικό

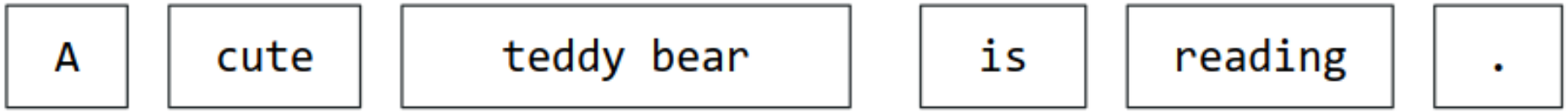
- 1980: Recurrent Neural Networks (1980)
- 1997: Long Short-Term Memory (LSTM)
- 2013: Word2Vec
- 2017: Transformers
- Σήμερα: Large Language Models

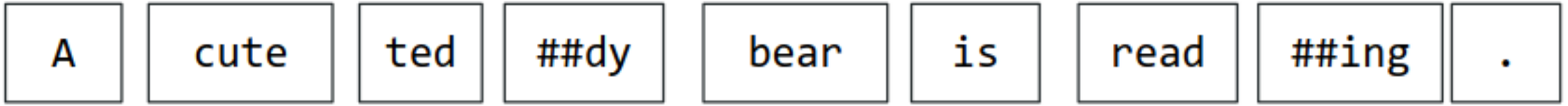
Tokenization



Tokenization

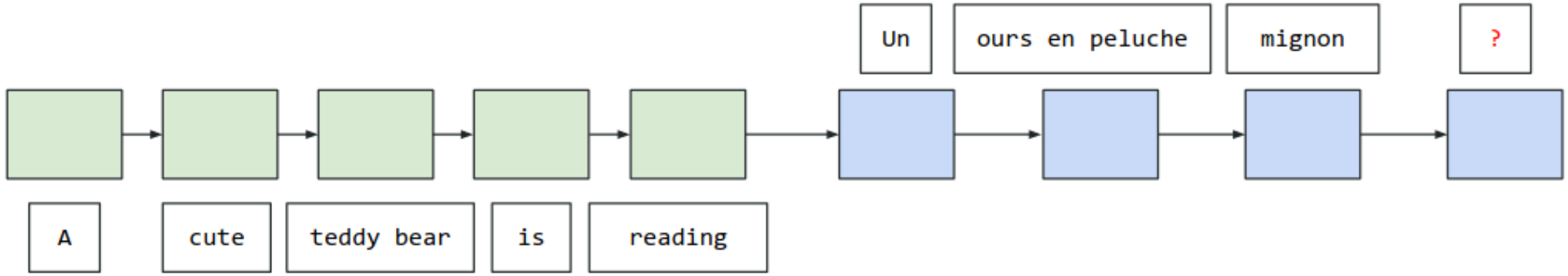
A cute teddy bear is reading.





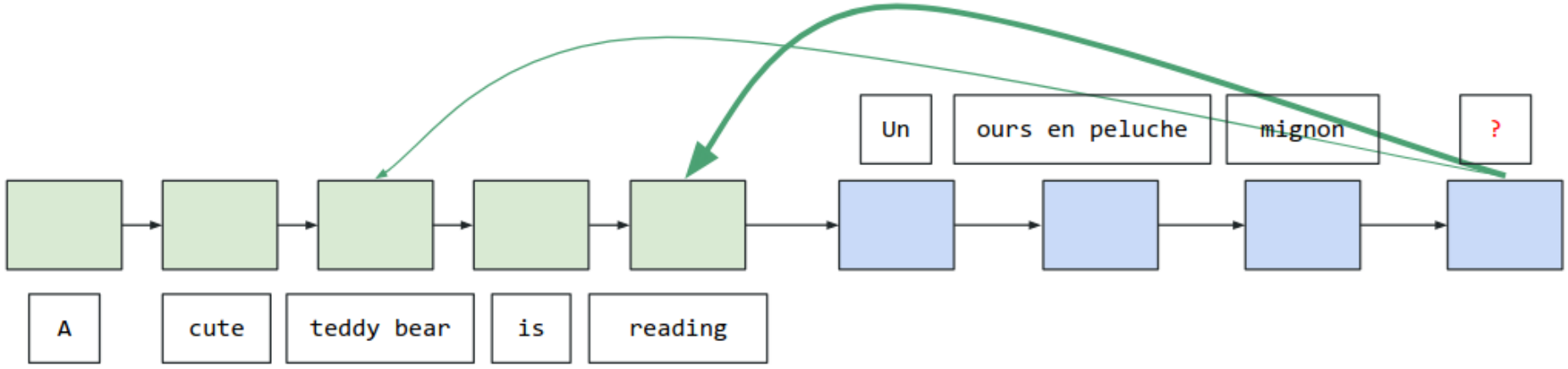
Attention Layer

- Προτάθηκε το 2014
- Τα προβλήματα σχετικά με τη μετάφραση είχαν ζητήματα σε σχέση με τις αλληλουχίες
- Δεν ήταν εφικτό για ένα μοντέλο να “θυμάται” τι έλεγε η πρόταση εισόδου



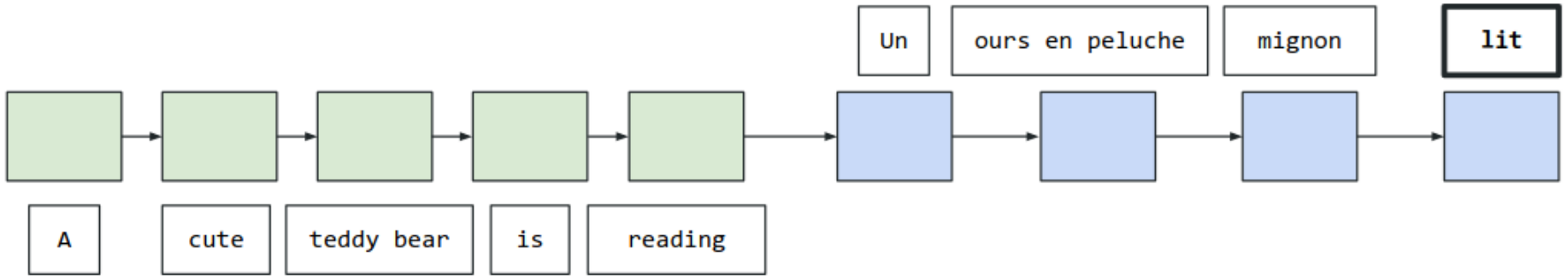
Attention Layer

- Προτάθηκε το 2014
- Τα προβλήματα σχετικά με τη μετάφραση είχαν ζητήματα σε σχέση με τις αλληλουχίες
- Δεν ήταν εφικτό για ένα μοντέλο να “θυμάται” τι έλεγε η πρόταση εισόδου



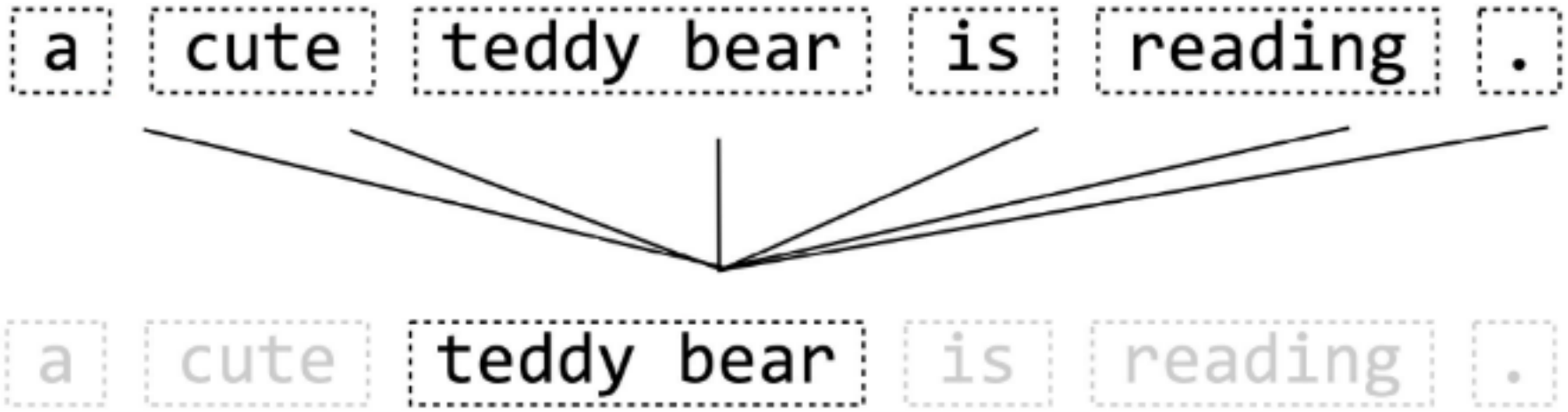
Attention Layer

- Προτάθηκε το 2014
- Τα προβλήματα σχετικά με τη μετάφραση είχαν ζητήματα σε σχέση με τις αλληλουχίες
- Δεν ήταν εφικτό για ένα μοντέλο να “θυμάται” τι έλεγε η πρόταση εισόδου



Attention is all you need

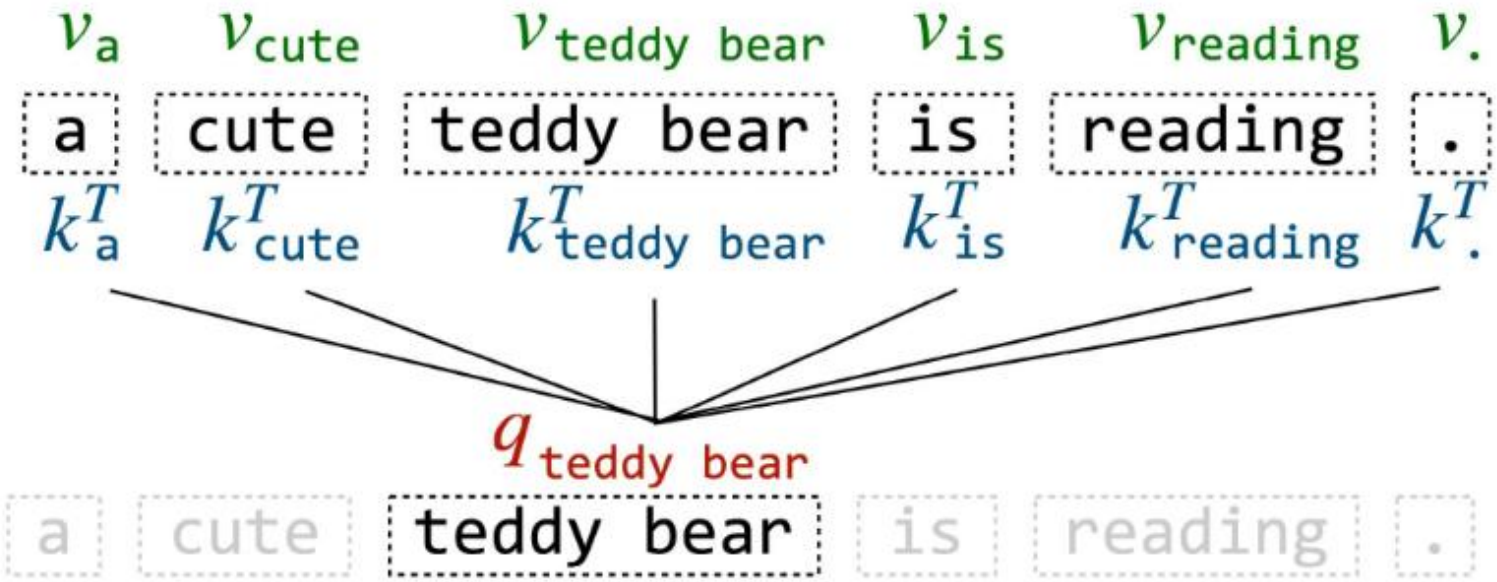
- Προτάθηκε το 2017
- Μηχανισμός self-attention



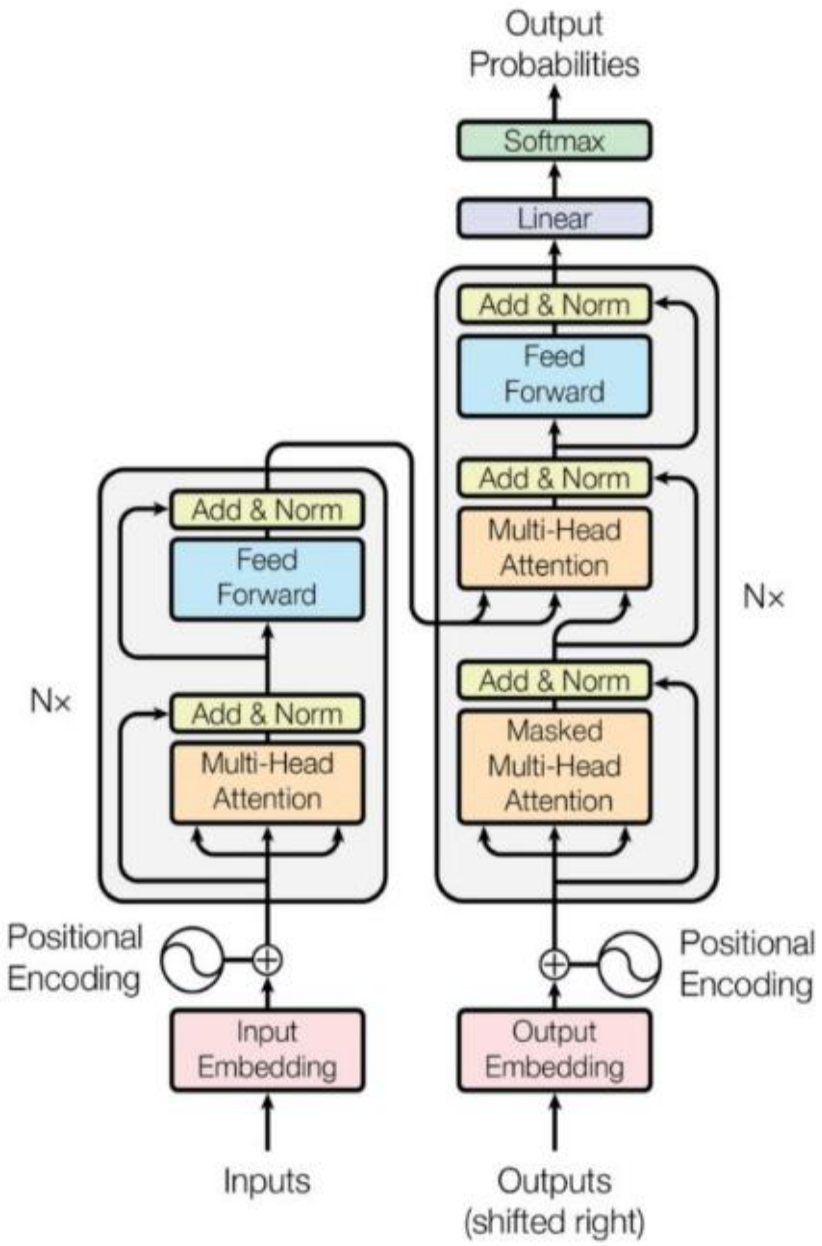
Attention is all you need

$$\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

- Προτάθηκε το 2017
- Μηχανισμός self-attention
- Βασική τριπλέτα: Query, Key, Value (Q, K, V)



Transformer

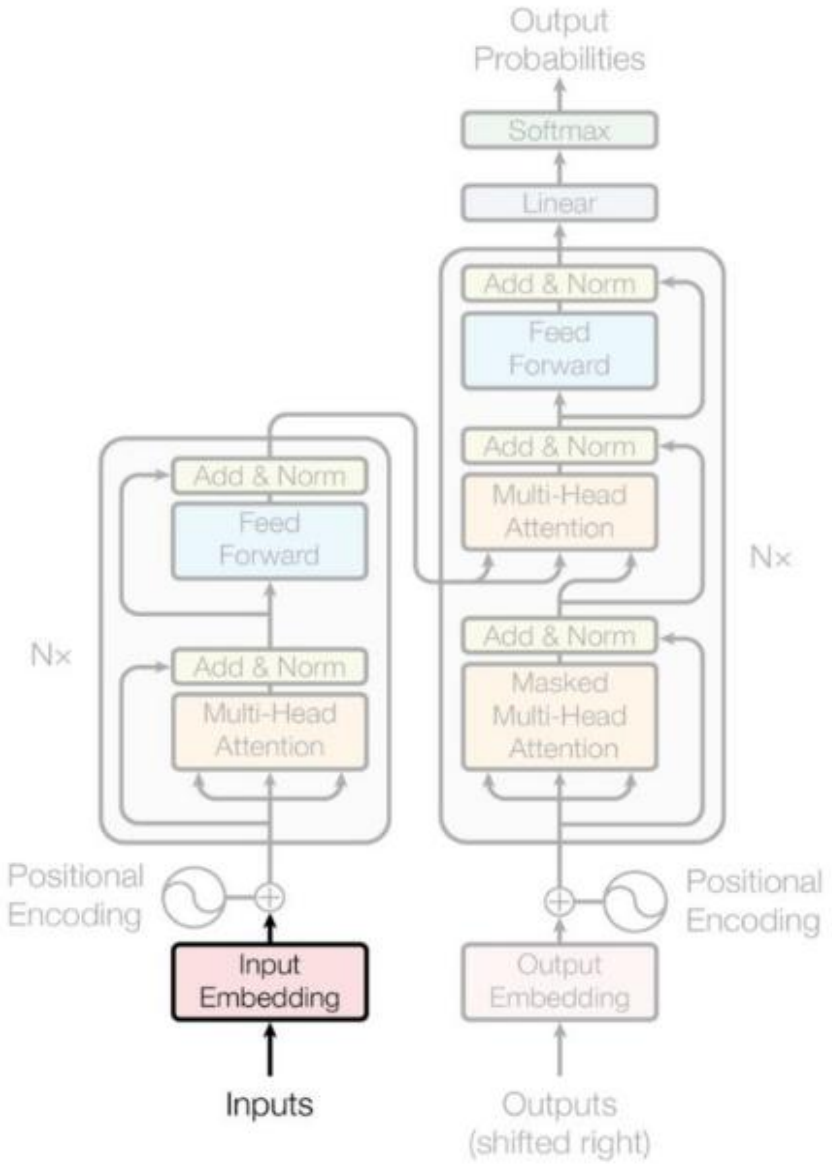


Transformer

Παράμετροι: μέγεθος λεξικού και διάσταση μοντέλου

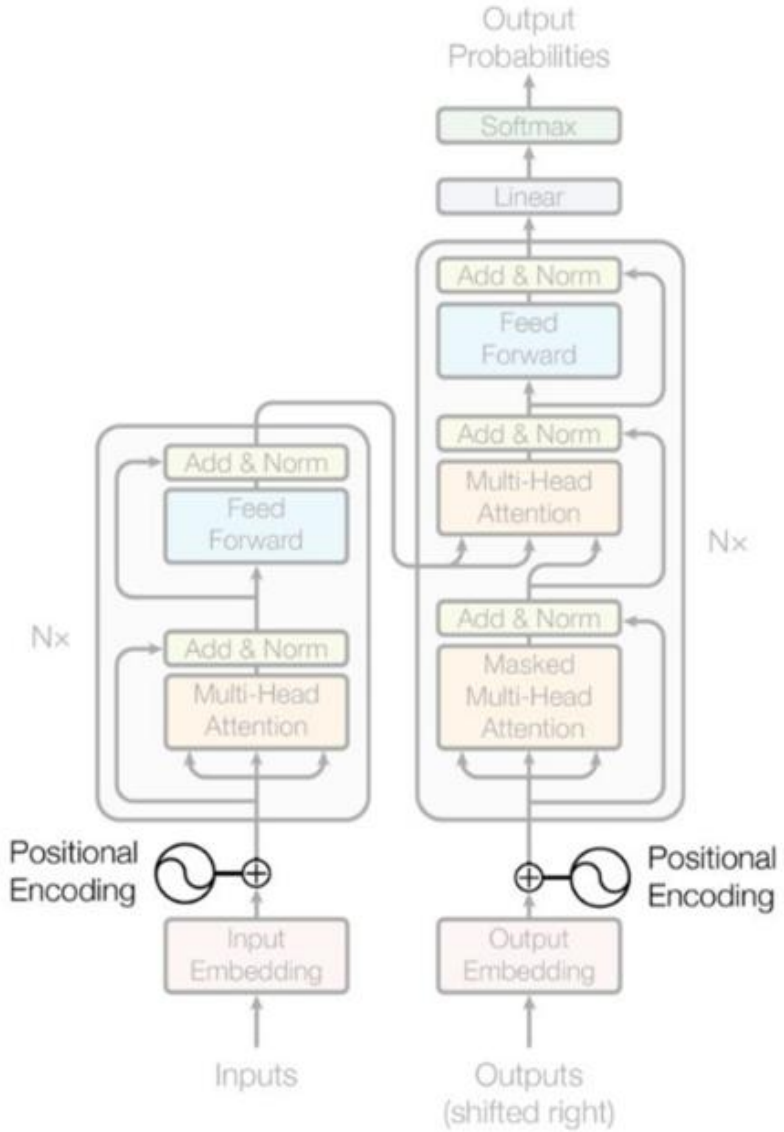
Το κείμενο περνά από μια συνάρτηση tokenization

Μετατρέπεται σε embeddings



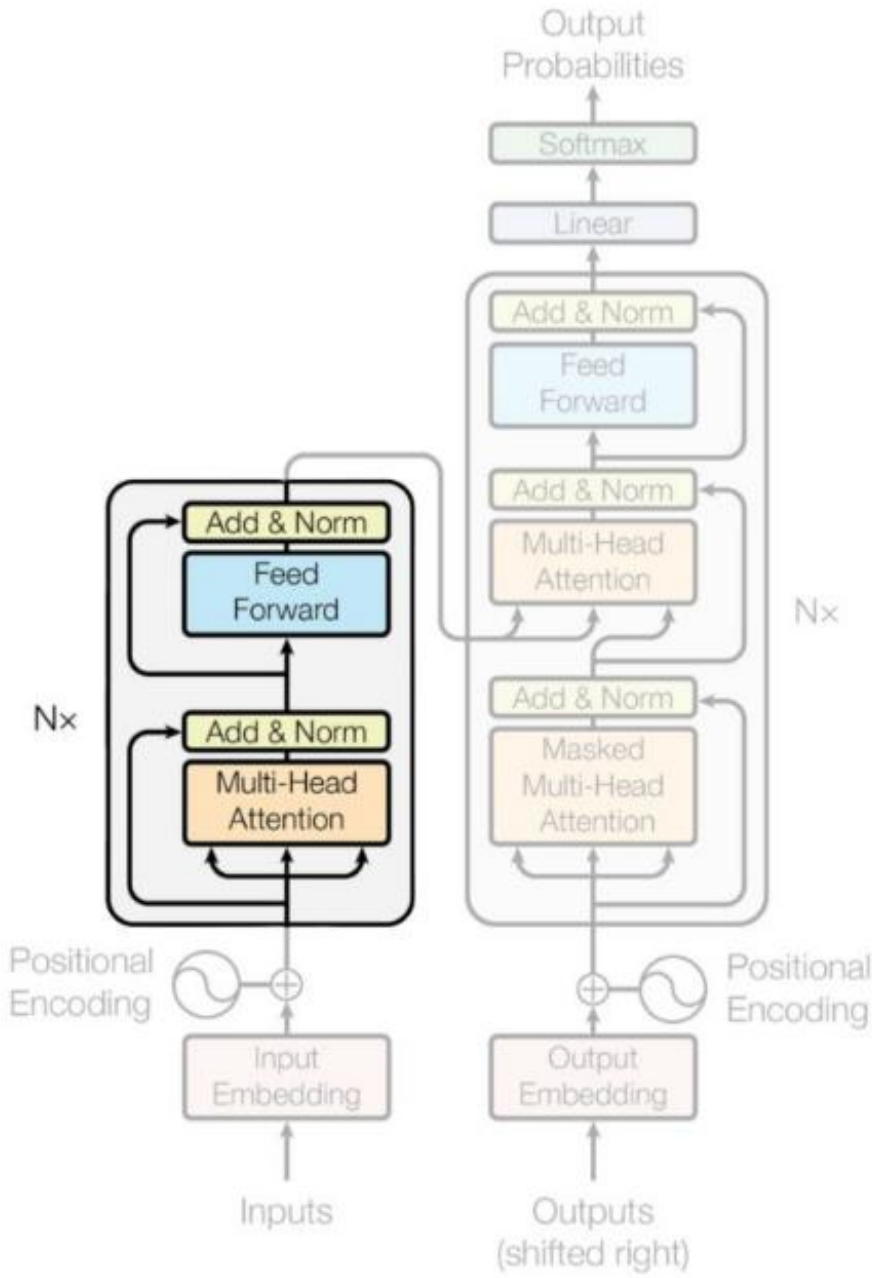
Transformer

- Προσθήκη πληροφορίας θέση στην αρχική είσοδο
- Σκοπός είναι να επιτρέψουμε στο μοντέλο να κατανοήσει την σχετική θέση του token

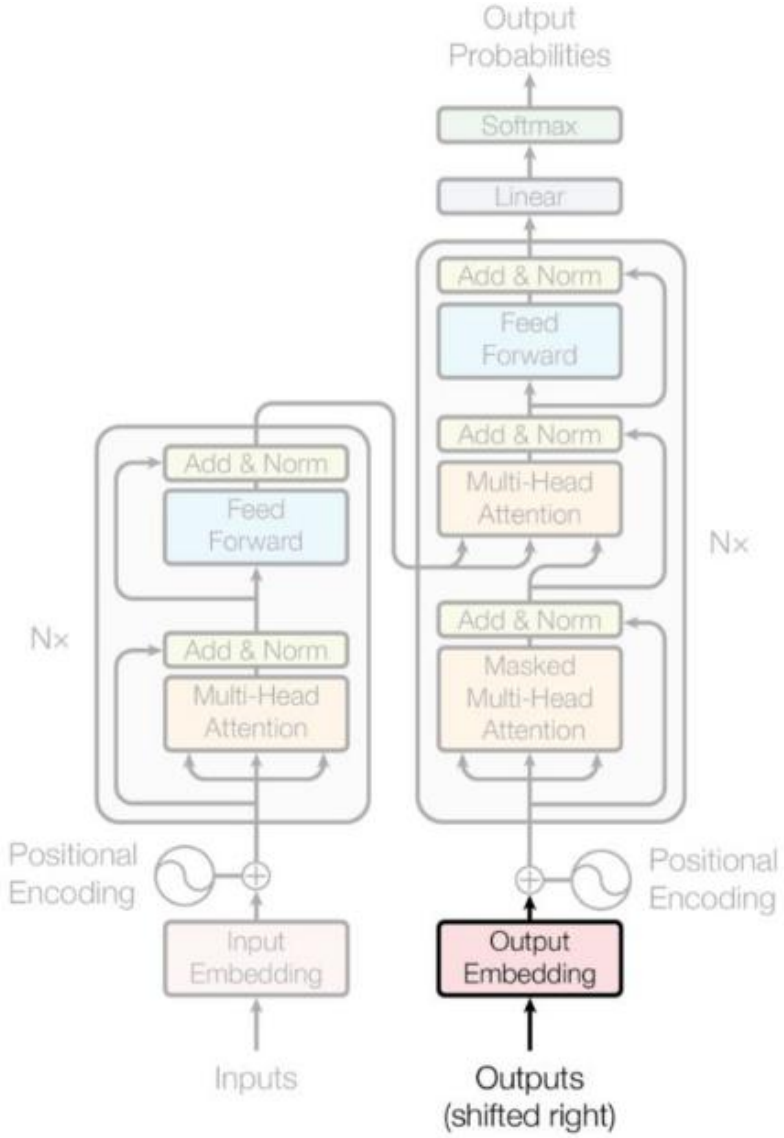


Transformer

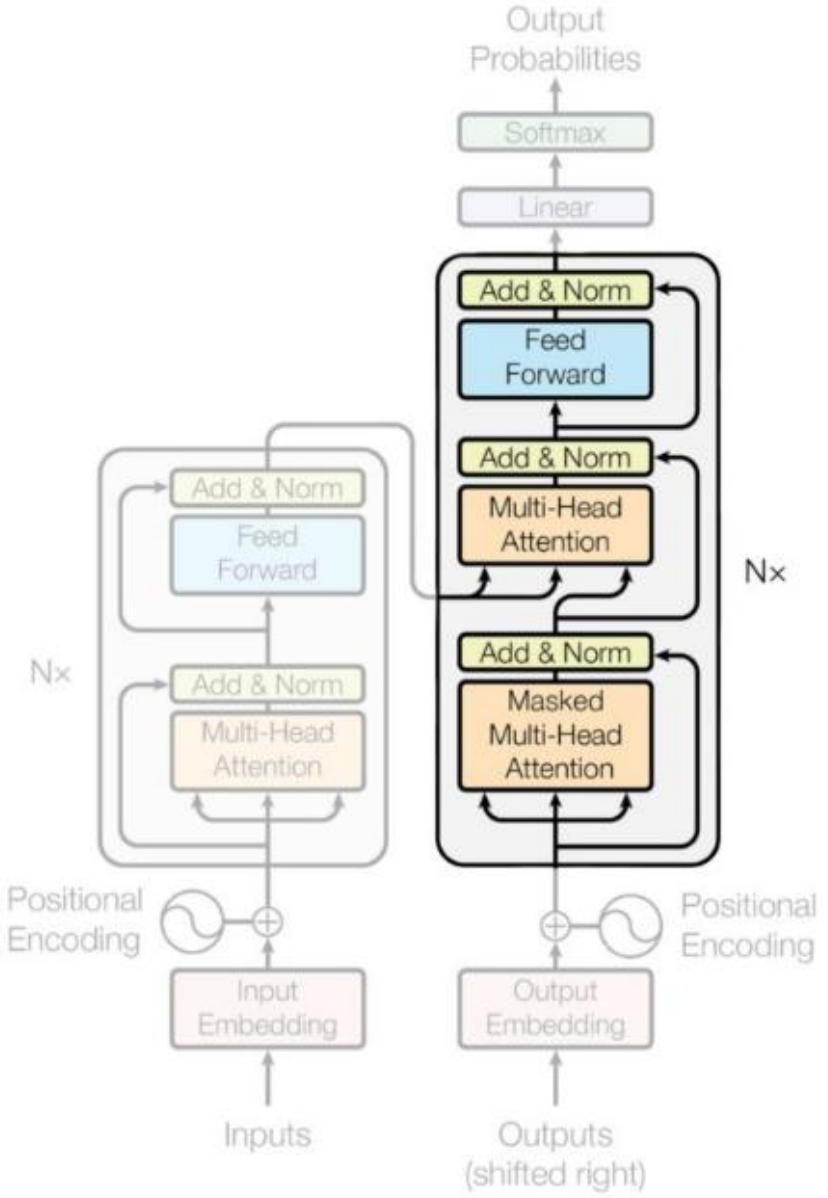
- Self-attention → FFNN →Normalization
- Παράμετροι
  - N: αριθμός από layers
  - h: αριθμός κεφαλών (heads)
  - Διαστάσεις του FFNN δικτύου



Transformer

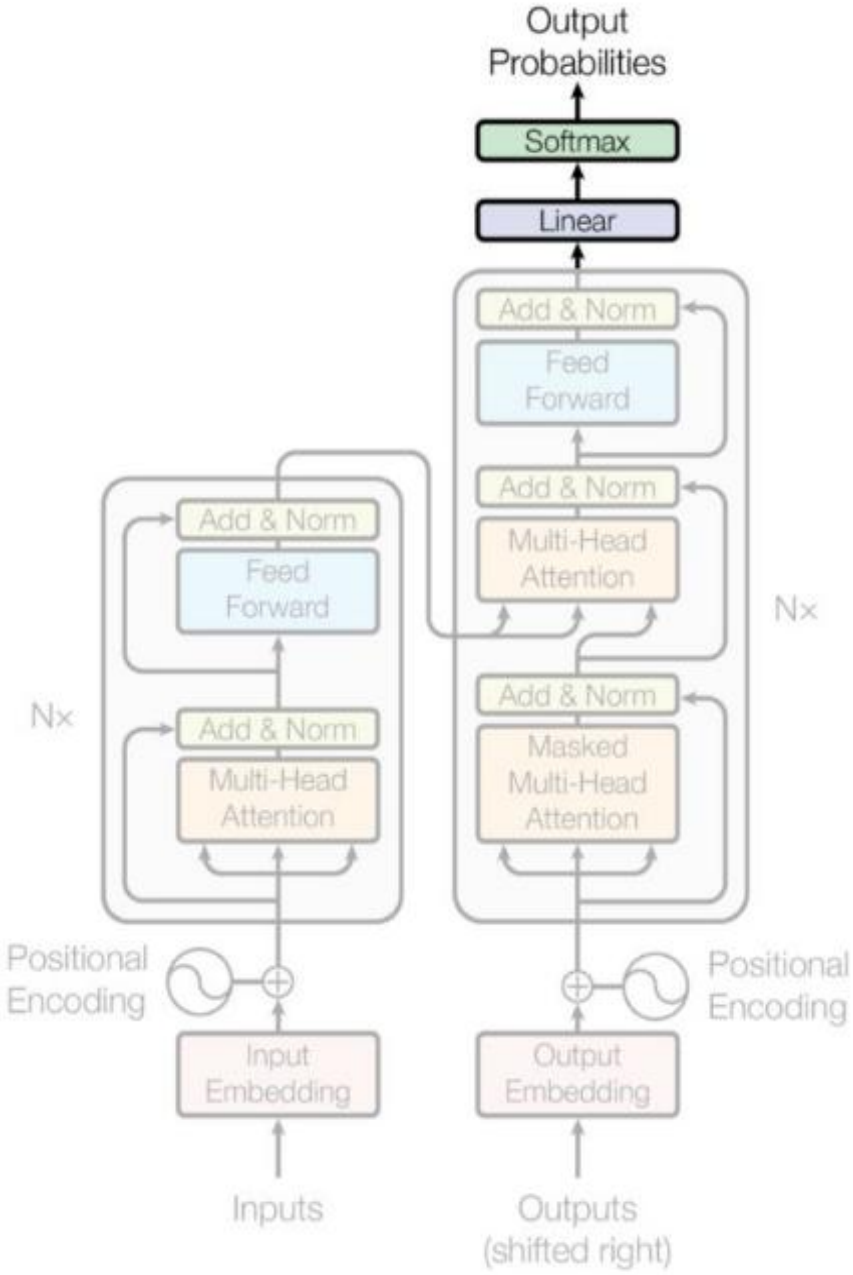


Transformer



Transformer

- Γραμμικός μετασχηματισμός
  - Ανάγεται σε πρόβλημα ταξινόμησης, δηλαδή η έξοδος είναι η πιθανότητα μιας κλάσης (όπου κλάση=λέξη)

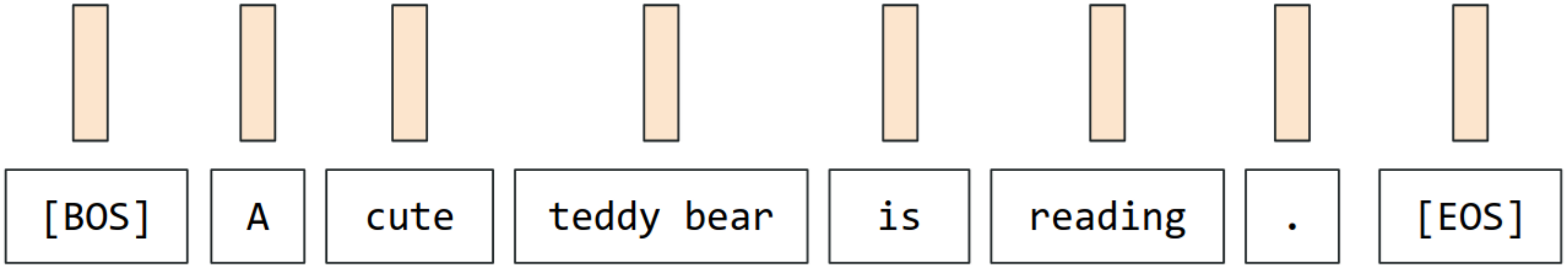


Transformer

- A cute teddy bear is reading.
- |A| |cute| |teddy bear| |is| |reading| |.|
- [BOS] |A| |cute| |teddy bear| |is| |reading| |.| |EOS|

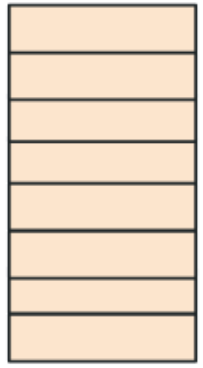
Transformer

- A cute teddy bear is reading.
- |A| |cute| |teddy bear| |is| |reading| |.|
- [BOS] |A| |cute| |teddy bear| |is| |reading| |.| |EOS|



Transformer

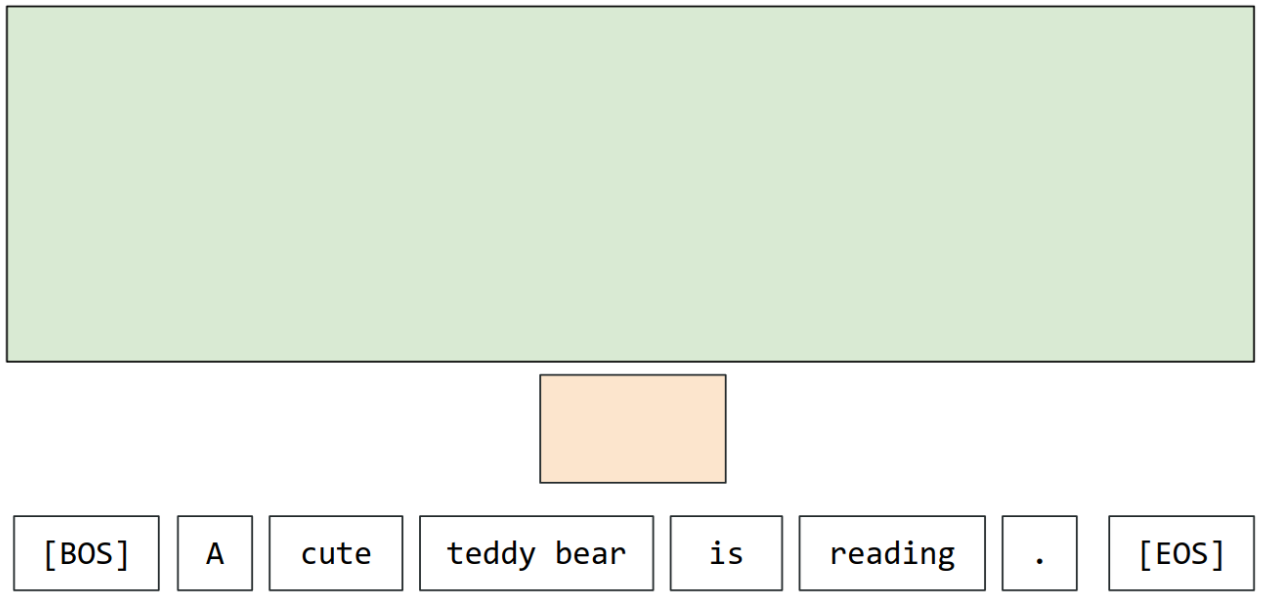
- A cute teddy bear is reading.
- |A| |cute| |teddy bear| |is| |reading| |.|
- [BOS] |A| |cute| |teddy bear| |is| |reading| |.| |EOS|



[BOS]	A	cute	teddy bear	is	reading	.	[EOS]

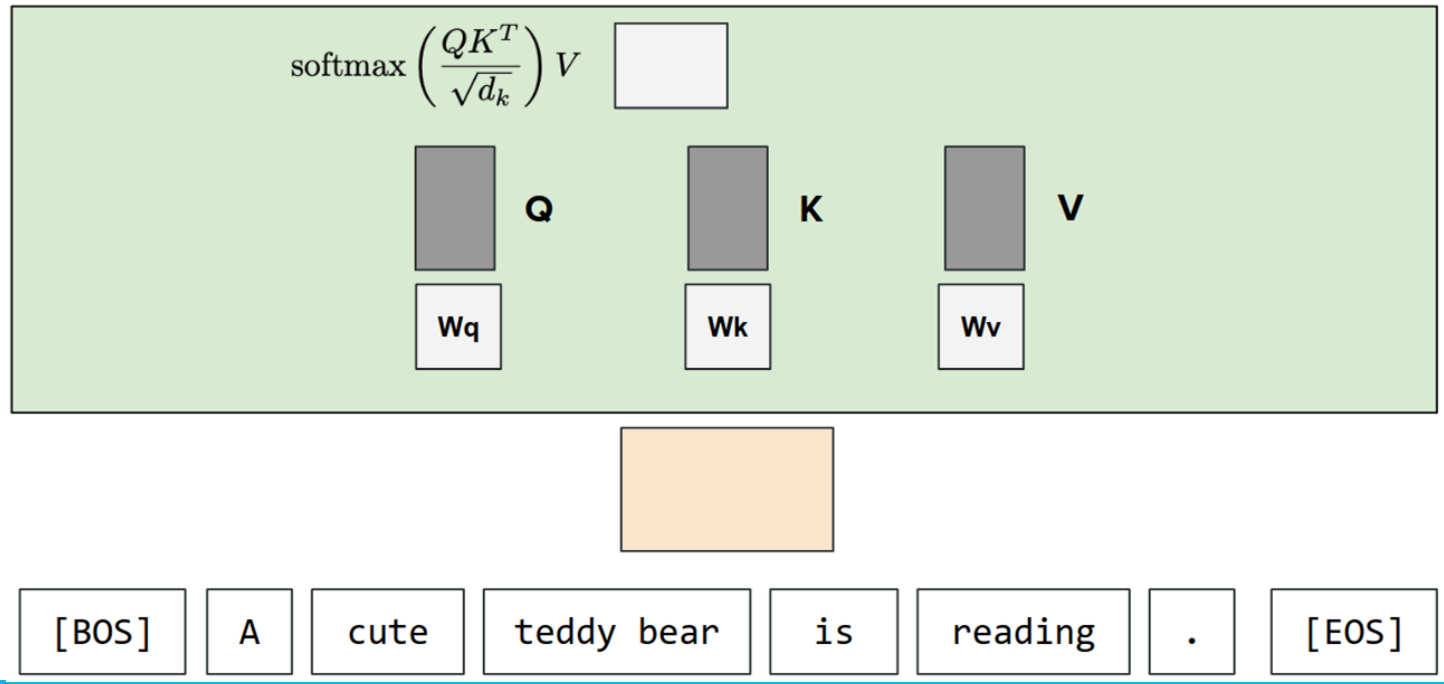
Transformer

- A cute teddy bear is reading.
- |A| |cute| |teddy bear| |is| |reading| |.|
- [BOS] |A| |cute| |teddy bear| |is| |reading| |.| |EOS|



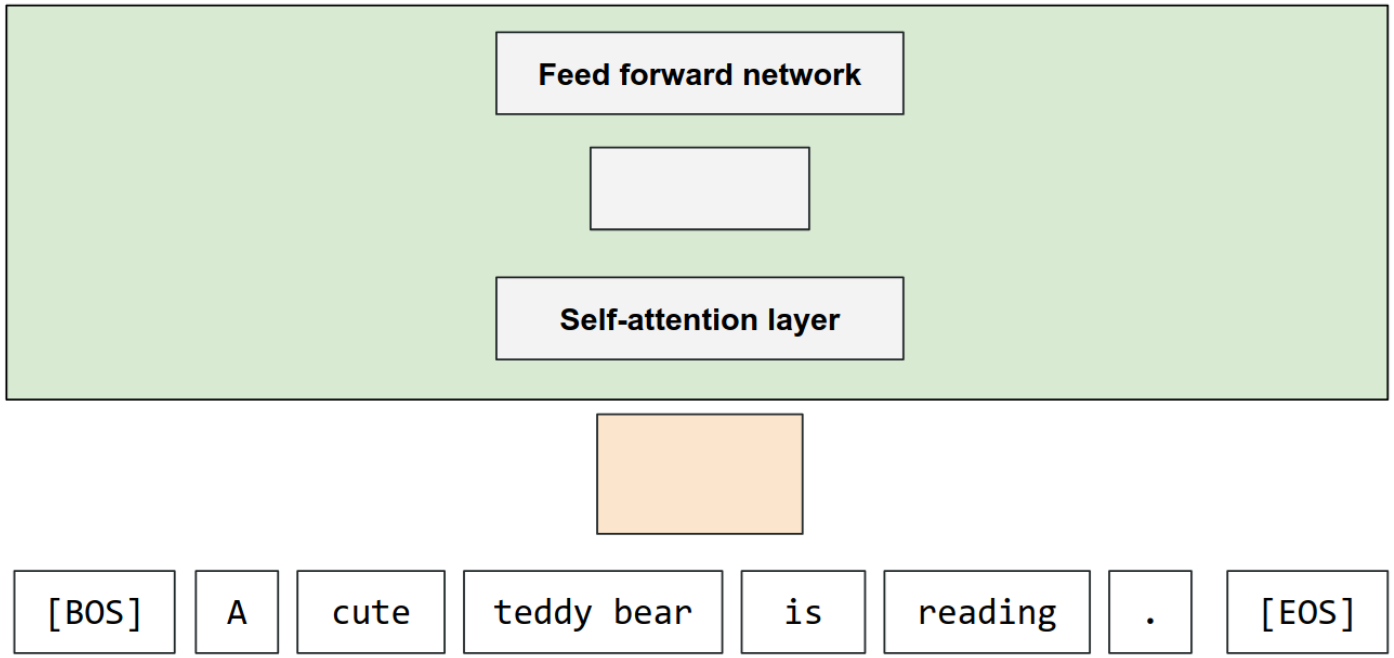
Transformer

- A cute teddy bear is reading.
- |A| |cute| |teddy bear| |is| |reading| |.|
- [BOS] |A| |cute| |teddy bear| |is| |reading| |.| |EOS|



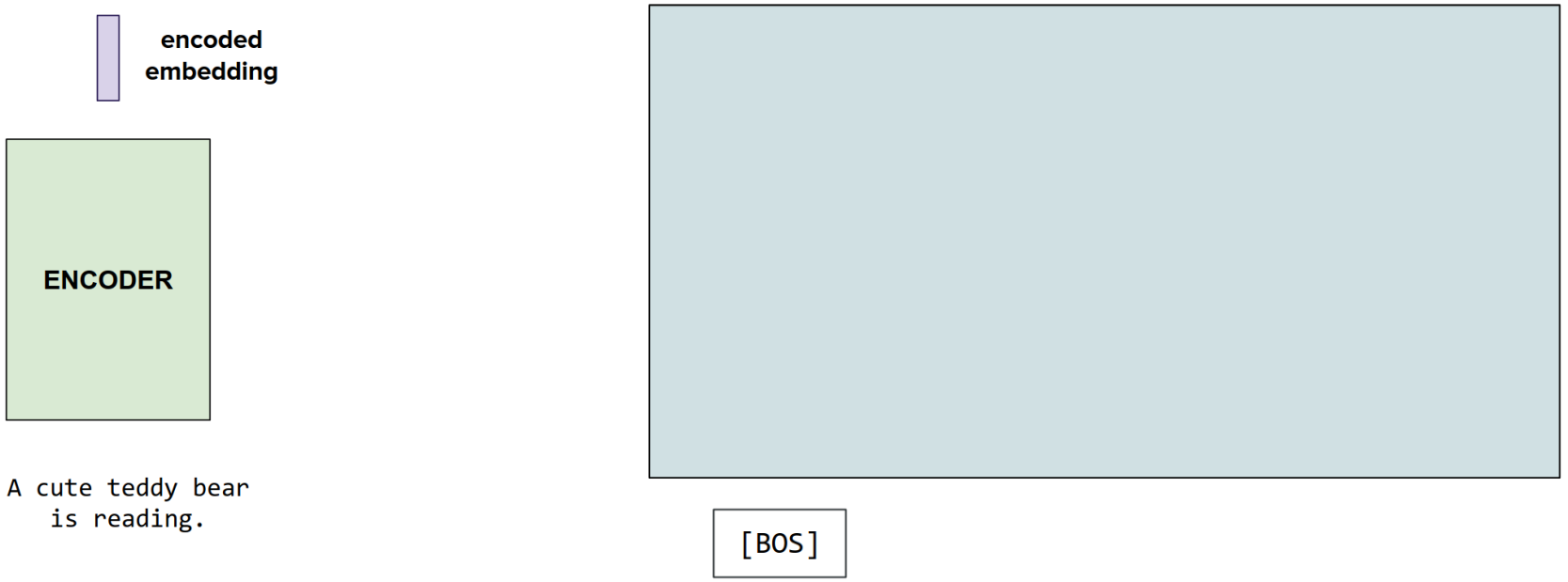
Transformer

- A cute teddy bear is reading.
- |A| |cute| |teddy bear| |is| |reading| |.|
- [BOS] |A| |cute| |teddy bear| |is| |reading| |.| |EOS|



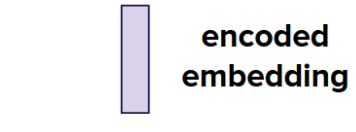
Transformer

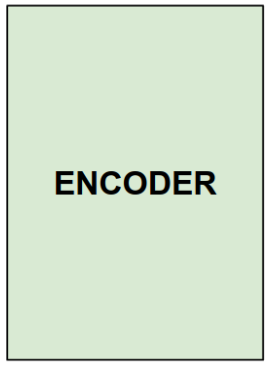
- A cute teddy bear is reading.
- |A| |cute| |teddy bear| |is| |reading| |.|
- [BOS] |A| |cute| |teddy bear| |is| |reading| |.| |EOS|



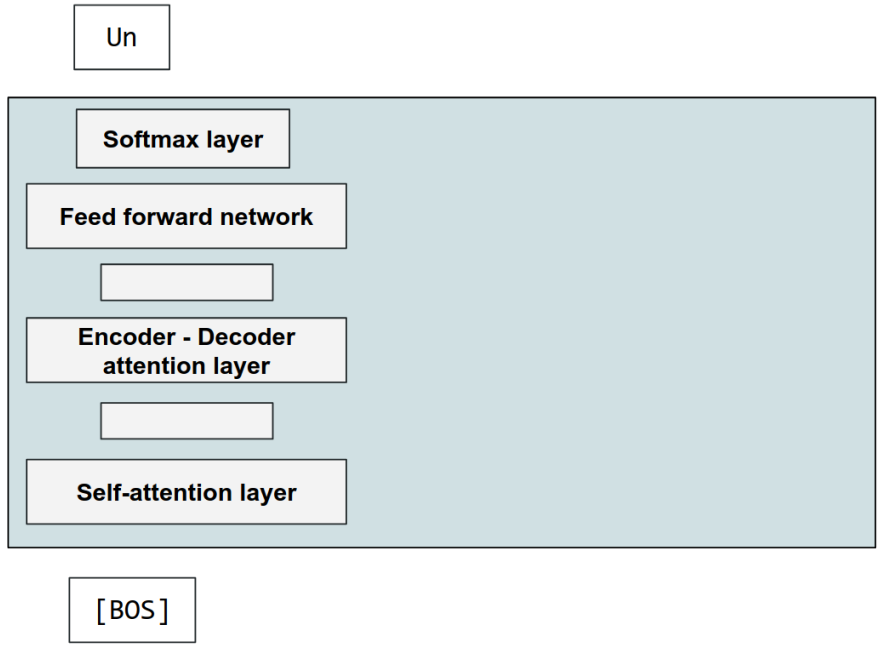
Transformer

- A cute teddy bear is reading.
- |A| |cute| |teddy bear| |is| |reading| |.|
- [BOS] |A| |cute| |teddy bear| |is| |reading| |.| |EOS|



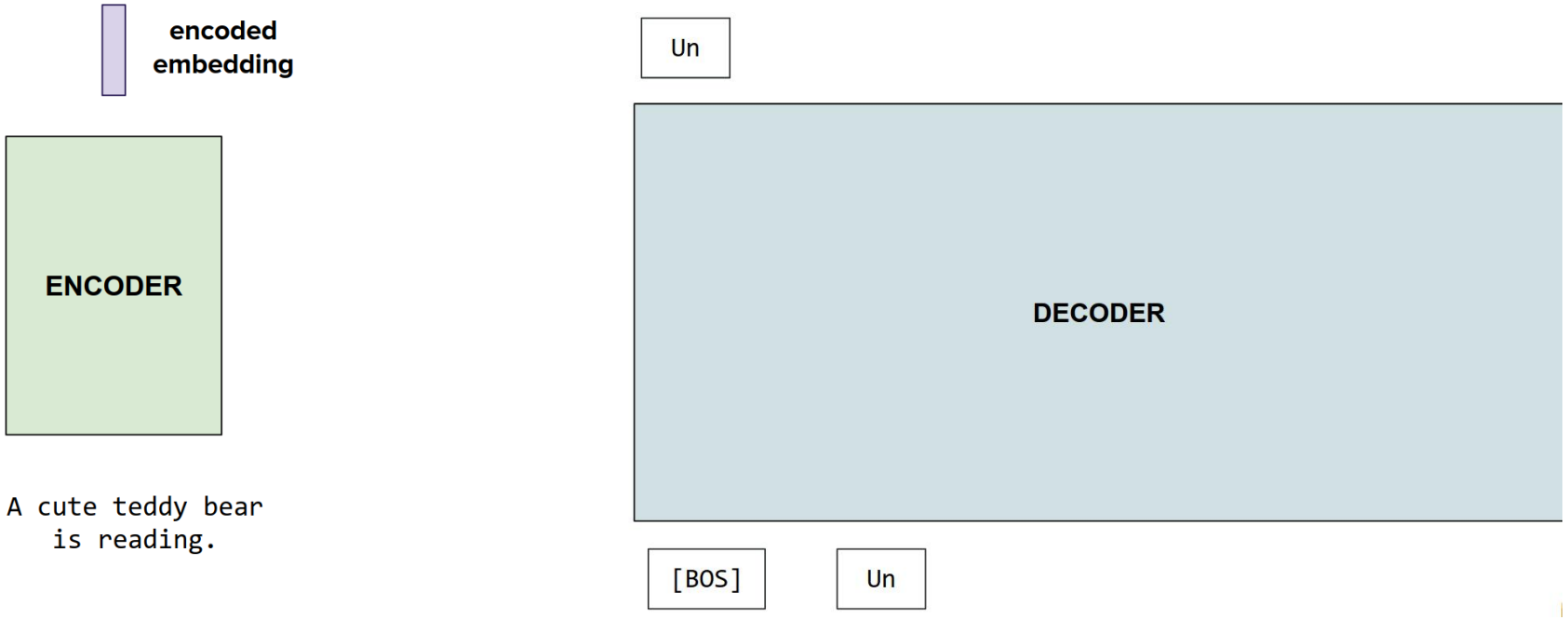


A cute teddy bear
is reading.



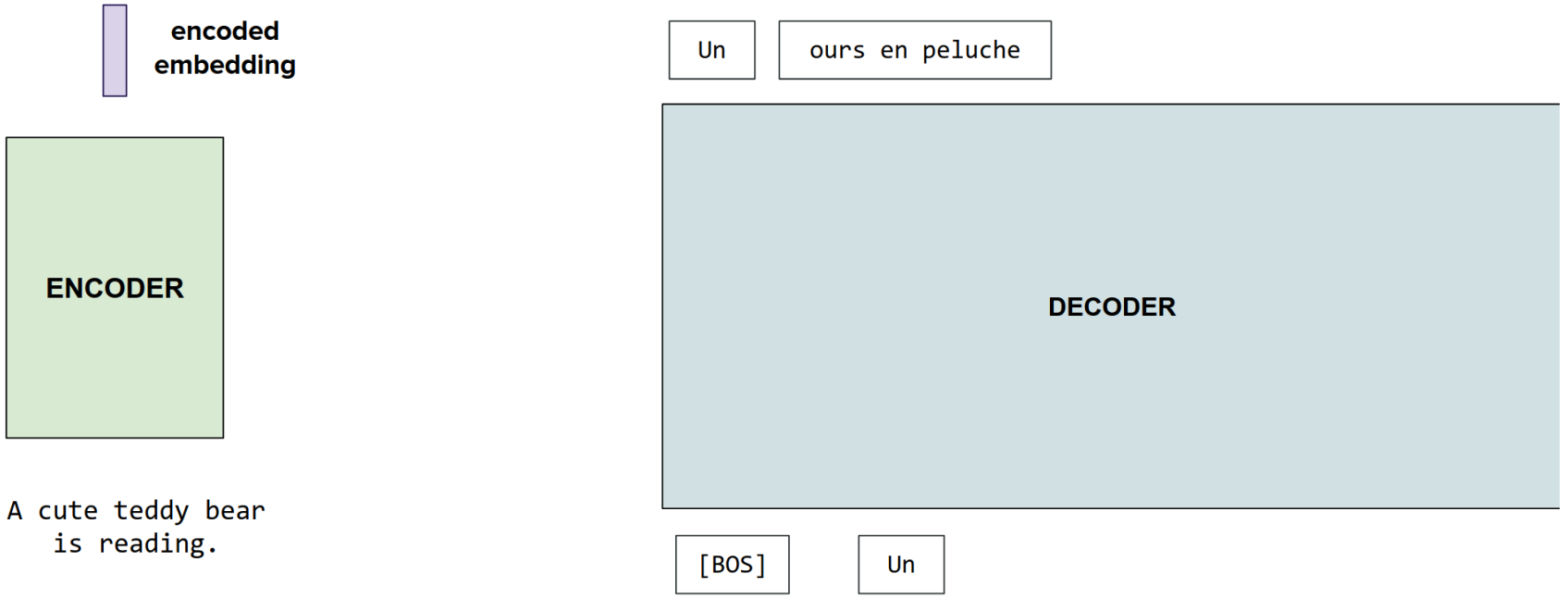
Transformer

- A cute teddy bear is reading.
- |A| |cute| |teddy bear| |is| |reading| |.|
- [BOS] |A| |cute| |teddy bear| |is| |reading| |.| |EOS|



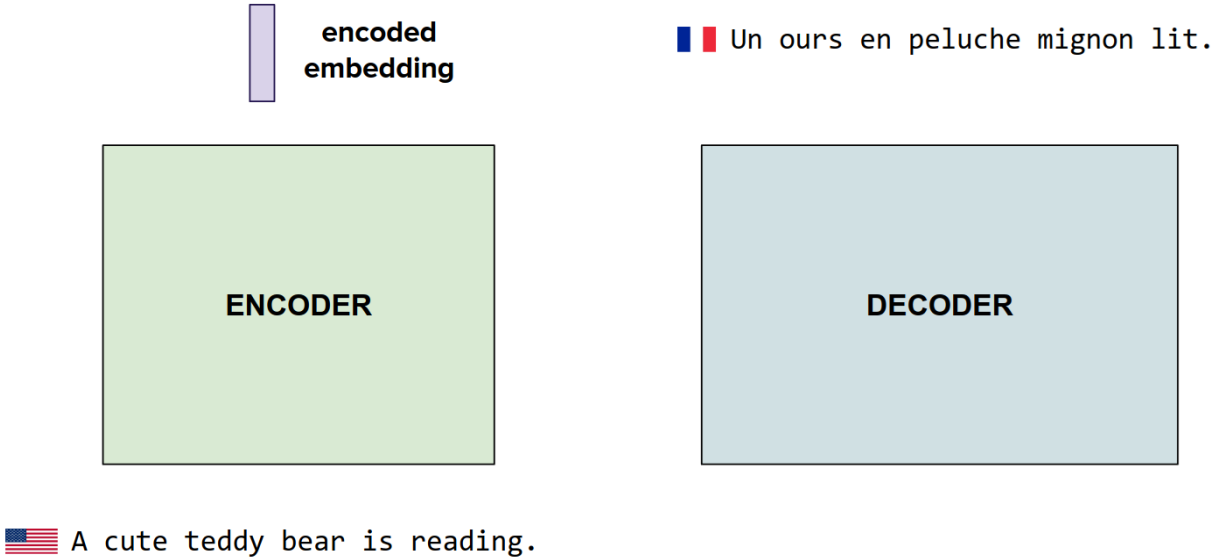
Transformer

- A cute teddy bear is reading.
- |A| |cute| |teddy bear| |is| |reading| |.|
- [BOS] |A| |cute| |teddy bear| |is| |reading| |.| |EOS|



Transformer

- A cute teddy bear is reading.
- |A| |cute| |teddy bear| |is| |reading| |.|
- [BOS] |A| |cute| |teddy bear| |is| |reading| |.| |EOS|



LLMs

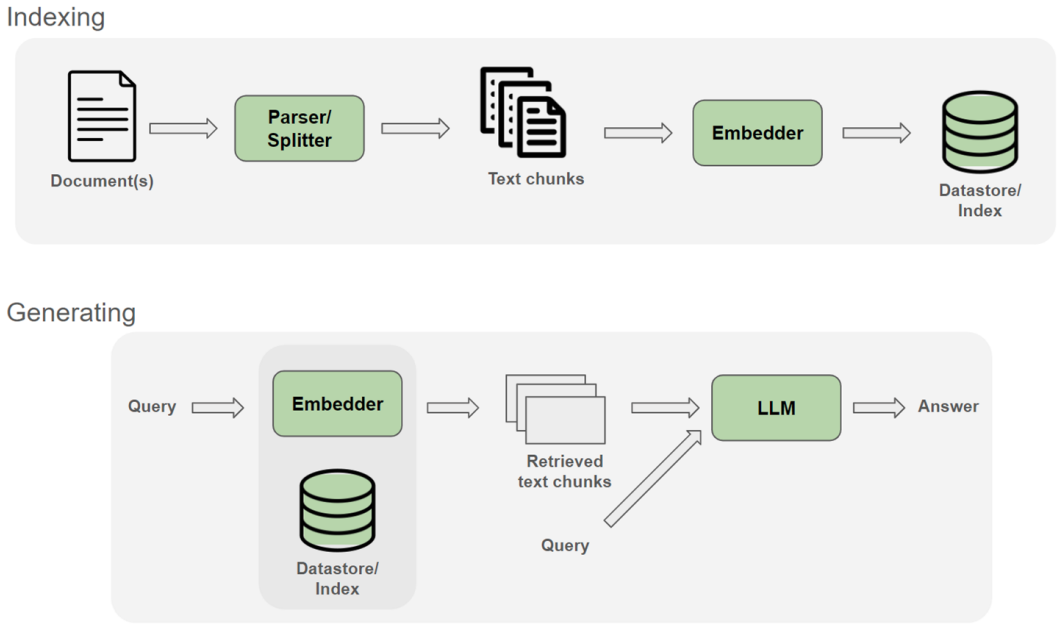
- Γλωσσικό Μοντέλο: Ένα γλωσσικό μοντέλο είναι στατιστικό μοντέλο ή ένα μοντέλο μηχανικής μάθησης το οποίο δίνει πιθανότητας σε μια αλληλουχία από το tokens

- Μεγάλο Γλωσσικό μοντέλο: Ένα γλωσσικό μεγάλο με δις παραμέτρους, εκατοντάδες δις token, πολλά GPUs

- Πιθανά Ζητήματα:
  - Παραισθήσεις (hallucinations)
  - Παλιές πληροφορίες
  - Έλλειψη γνώσεων σε συγκεκριμένους τομείς

RAG

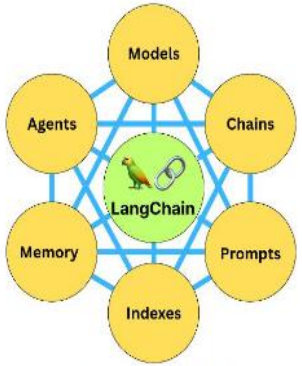
- Αρχικά γίνεται ανάκτηση σχετικών πληροφοριών από έναν μεγάλο αριθμό εγγράφων
- Στη συνέχεια, το γλωσσικό μοντέλο δημιουργεί απαντήσεις με βάση αυτές τις πληροφορίες
- Στην πράξη, συνδέεται μια εξωτερική βάση γνώσεων



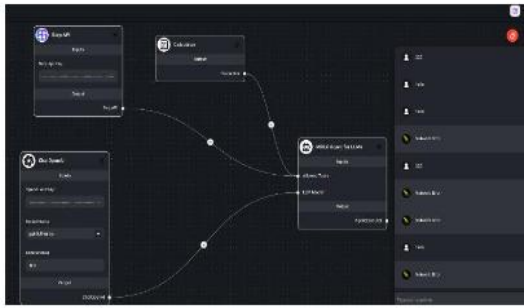
RAG

- Σενάρια για την χρήση RAG:
  - Συχνή ανανέωση δεδομένων
  - Οι απαντήσεις χρειάζονται επιβεβαίωση και ανιχνευσιμότητα
  - Ειδική γνώση

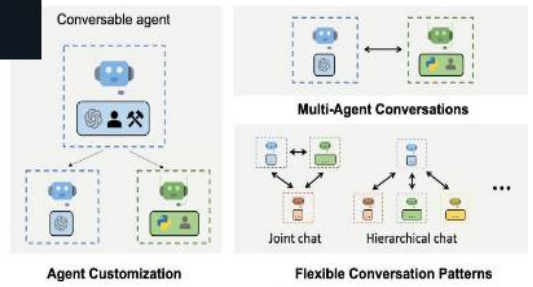
- Tech Stack
  - Langchain
  - LlamaIndex
  - FlowiseAI
  - AutoGen



LangChain


FlowiseAI


LlamaIndex



AutoGen

Agentic AI

- Φανταστείτε έναν ρομποτικό βοηθό
- Όταν θέλουμε καφέ, θα λέγα “Θα ήθελα έναν καφέ”
- Ο βοηθός **καταλαβαίνει** το αίτημα επειδή μπορεί να κατανοήσει φυσική γλώσσα
- Ο βοηθός **λαμβάνει** την εντολή και αρχίζει να **σκέφτεται** και να **σχεδιάζει**
- Πρέπει να αποφασίσει ποια βήματα χρειάζονται για να ολοκληρώσει την εργασία.
- Τα βήματα που σχεδιάζει είναι να πάει στην κουζίνα, να χρησιμοποιήσει τη μηχανή καφέ, να φτιάξει τον καφέ, να φέρει τον καφέ πίσω

Agentic AI

- Τα βήματα που σχεδιάζει είναι να πάει στην κουζίνα, να χρησιμοποιήσει τη μηχανή καφέ, να φτιάξει τον καφέ, να φέρει τον καφέ πίσω
- Αφού σχεδιάσει, ο βοηθός εκτελεί τις ενέργειες
  - Παράδειγμα εργαλείου
    - Μηχανή του καφέ
      - Η μηχανή χρησιμοποιείται για να φτιάξει τον καφέ
- Αν ο βοηθός καταφέρει να εκτελέσει όλο το πλάνο του, έχει **κατανοήσει** τις οδηγίες, έχει **σχεδιάσει** ενέργειες, **χρησιμοποίησε** εργαλεία και **παρέδωσε** αποτέλεσμα

Agentic AI

- Ένας Agent είναι ένα σύστημα που χρησιμοποιεί ένα μοντέλο AI για να αλληλεπιδρά με το περιβάλλον του ώστε να επιτύχει έναν στόχο που ορίζει ο χρήστης
- Συνδυάζει: Reasoning (λογική σκέψη), Planning (σχεδιασμό), Execution (Εκτέλεση ενεργειών), Tools (Χρήση εργαλείων)
- Ένας Agent αποτελείται από τον “εγκέφαλο” και το “σώμα” του
  - Εγκέφαλος=Γλωσσικό μοντέλο: reasoning, planning
  - Σώμα=tools: Τα διαθέσιμα εργαλεία

Agentic AI

- Επίπεδο Agent
  - Επίπεδο 0: Η έξοδος του δεν επηρεάζει τη ροή του προγράμματος
  - Επίπεδο 1: Η έξοδος του καθορίζει τη ροή
  - Επίπεδο 2: Η έξοδος του επιλέγει λειτουργίες
  - Επίπεδο 3: Ο agent ελέγχει πολλαπλά βήματα
  - Επίπεδο 4: Agents που ενεργοποιούν άλλους agents

Agentic AI

- Τα LLMs μπορούν να παράγουν κείμενο, αλλά όχι να αλληλεπιδρούν άμεσα με συστήματα.
- Για αυτό χρησιμοποιούνται Tools (εργαλεία).
- Τα εργαλεία επιτρέπουν στον agent να:
  - Δημιουργεί εικόνες
  - Αναζητά στο διαδίκτυο
  - Στέλνει email
  - Εκτελεί κώδικα

Agentic AI

- Δημιουργούμε έναν agent ο οποίος είναι “Agent Καιρού”
- Ο χρήστης ρωτά: “Τι καιρό έχει τώρα στην Ξάνθη;”
- Ο Agent πρέπει να απαντήσει χρησιμοποιώντας ένα weather API tool
  - Σκέφτεται: “Ο χρήστης ζητά τον τρέχοντα καιρό στη Ξ’ανθη.Έχω ένα εργαλείο που μπορεί να πάρει δεδομένα καιρού. Πρέπει πρώτα να καλέσω το weather API.
  - Ο agent σπάει το πρόβλημα σε βήματα
    - Ενέργεια: Ο agent χρησιμοποιεί το εργαλείο get_weather.
      - { “action”: “get_weather”, “input”: {“location”: “Ξάνθη”}
      - Ο agent δηλώνει ποιο εργαλείο θα χρησιμοποιήσει και ποια δεδομένα θα στείλει
    - Παρατήρηση: Μετά την εκτέλεση της ενέργειας, ο agent λαμβάνει αποτέλεσμα
      - 15°C, 60% υγρασία
      - Το πάνω αποτέλεσμα προστίθεται στο context
    - Νέα σκέψη: “Τώρα έχω τα δεδομένα καιρού για την Ξάνθη. Μπορώ να δημιουργήσω την απάντηση για τον χρήστη.”
    - Απάντηση: “Ο τρέχων καιρός στην Ξάνθη είναι μερικώς συννεφιασμένος με θερμοκρασία 15°C και υγρασία 60%.”

Failure modes και safety: πού αποτυγχάνουν τα LLMs

Hallucinations
απαντήσεις με σιγουριά αλλά χωρίς επαρκή στήριξη

Mitigation: grounding, citations, abstention

Bias / fairness
αναπαραγωγή προκαταλήψεων των δεδομένων

Mitigation: evals ανά ομάδα, policy filters

Prompt injection
κακόβουλο περιεχόμενο που χειραγωγεί agent/tool use

Mitigation: tool policies, isolation, least privilege

Context dilution
σημαντικές οδηγίες χάνονται σε πολύ μεγάλο context

Mitigation: prompt structure, retrieval quality, summaries

Αρχή σχεδιασμού
Ο σχεδιασμός ενός γλωσσικού μοντέλου πρέπει να λαμβάνει υπόψη ότι (θα) είναι ένα σύστημα με guardrails, monitoring, logs, evals και ανθρώπινο fallback όπου χρειάζεται.

Τάσεις 2025–2026 που αξίζει να παρακολουθούμε

Reasoning-first μοντέλα
περισσότερος “thinking time”, καλύτερα code/math/science workflows

Πολυτροπικότητα
κείμενο + εικόνα + έγγραφα + ήχος γίνονται standard input surface

Agentic tool use
ενσωματωμένα εργαλεία, web/file search, code execution, MCP connectors

Μικρότερα αποδοτικά μοντέλα
ισχυρά small models για χαμηλότερο latency/cost και edge/private deployments

Interoperability & governance MCP, eval stacks, observability, policy layers και πιο ώριμα deployment patterns

Κατεύθυνση της αγοράς: όχι απλώς “μεγαλύτερα μοντέλα”, αλλά καλύτερα συστήματα με reasoning, tools, multimodality και διαλειτουργικότητα.

Recap και ερωτήσεις για συζήτηση

Σύνοψη

- Το NLP είναι η βάση, η αρχιτεκτονική Transformer τα θεμέλια και τα LLMs η συνέχεια
- Οι αναπαραστάσεις κειμένου εξελίχθηκαν από sparse features σε contextual token-level states.
- Ο transformer έγινε καταλύτης επειδή κλιμακώνεται καλά σε

data και compute.

- LLM. Είναι πιθανοτικός γεννήτορας με ισχυρές δυνατότητες αλλά και failure modes.
- Prompting, RAG, fine-tuning και tool use είναι διαφορετικοί μοχλοί για διαφορετικά προβλήματα.
- Η πραγματική αξία έρχεται όταν σχεδιάζουμε συστήματα με evals, guardrails και σωστό deployment model.

Αν κατανοούμε σωστά το μονοπάτι NLP → embeddings → attention → transformers → LLM systems, μπορούμε να παίρνουμε πολύ πιο ώριμες τεχνικές και επιχειρησιακές αποφάσεις.

DEMO

https://shorturl.at/upepD