Deep Neural Networks και NLP

Ηρακλής Βαρλάμης

Περιεχόμενα

- Το κείμενο ως ακολουθία λέξεων
- NLP
- Deep Learning για κείμενα και ακολουθίες

Γλωσσικά μοντέλα

- Τα γλωσσικά μοντέλα θεωρούν το κείμενο ως μια ακολουθία λέξεων και εξετάζουν κάθε φορά την πιθανότητα εμφάνισης μιας λέξης στα πλαίσια της ακολουθίας
- π.χ. Ο τροχονόμος _____ την κυκλοφορία (ρυθμίζει, κατευθύνει, σταματά, αγοράζει,...)

  P(ρυθμίζει|ο, τροχονόμος)>P(αγοράζει|ο, τροχονόμος)

- Εφαρμογές: Συμπλήρωση μηνυμάτων, οπτική αναγνώριση κειμένου, αναγνώριση φωνής, μετάφραση

n-grams

- Τα n-grams είναι μια ακολουθία n συνεχόμενων λέξεων (ή χαρακτήρων)
- Bigram, Trigrams, n-grams

$$<w_1,w_2,...w_n> \rightarrow w_1^n$$

- Ο κανόνας της αλυσίδας:

$$P(w_1^n)=P(w_1,w_2,...w_n)=P(w_1)*P(w_2|w_1)*P(w_3|w_1,w_2)*...*P(w_n|w_1^{k-1})$$

- Οι πιθανότητες υπολογίζονται συνήθως με τα Maximum Likelihood Estimates σε ένα σώμα C λέξεων.
- P_{MLE}(ο)=c(ο)/C , P_{MLE}(τροχονόμος|ο)=c(ο,τροχονόμος)/c(ο) , P_{MLE}(ρυθμίζει|ο, τροχονόμος)=c(ο,τροχονόμος,ρυθμίζει)/c(ο,τροχονόμος)
- Ολοένα και μικρότερες οι πιθανότητες για μεγάλα n-grams

Υπόθεση Markov

- Bigram language model

  $P(w_1^k) \sim P(w_1|start)*P(w_2|w_1)*P(w_3|w_2)*...(w_k|w_{k-1})$

- Trigram LM

  $P(w_1^k) \sim P(w_1|start_1,start_2)*P(w_2|start_2,w_1)*P(w_3|w_1,w_2)*...(w_k|w_{k-2},w_{k-1})$

- Ακόμη και έτσι πολλά n-grams είναι πολύ απίθανο να βρεθούν στο σώμα κειμένων

Παράδειγμα

Σώμα κειμένων:

- Είμαι καλός μαθητής
- Είμαι ο πρώτος στη σειρά
- Είμαι ο γιος του Κώστα

P(είμαι ο πρώτος στη σειρά)=P(είμαι|<s>)P(ο|είμαι)P(πρώτος|ο)P(στη|πρώτος)P(σειρά|στη)

=3/3*⅔*½*1*1=1/3

P(είμαι πρώτος στη σειρά)=P(είμαι|<s>)P(πρώτος|είμαι)P(στη|πρώτος)P(σειρά|στη)

=3/3 *0/3*1*1 =0

Laplace smoothing

- Για να αποφύγουμε τις μηδενικές πιθανότητες προσθέτουμε 1 σε κάθε πλήθος εμφανίσεων ενός bigram (και το μέγεθος του λεξικού σε κάθε παρονομαστή).
- $P_{Laplace}(w_k|w_{k-1})=[c(w_k,w_{k-1})+1] / [c(w_{k-1}) + V]$, όπου V το μέγεθος του λεξικού.

  P(είμαι πρώτος στη σειρά)=P(είμαι|<s>)P(πρώτος|είμαι)P(στη|πρώτος)P(σειρά|στη)

  =(3+1)/(3+10) *1/13*2/11*2/11=12/(13*13*11*11)

- Add-α smoothing

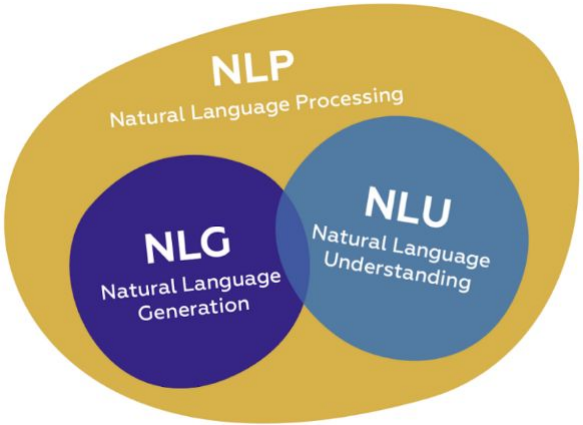
  $P_{Laplace}(w_k|w_{k-1})=[c(w_k,w_{k-1})+\alpha] / [c(w_{k-1}) + \alpha*V]$ με το $0\leq\alpha\leq1$

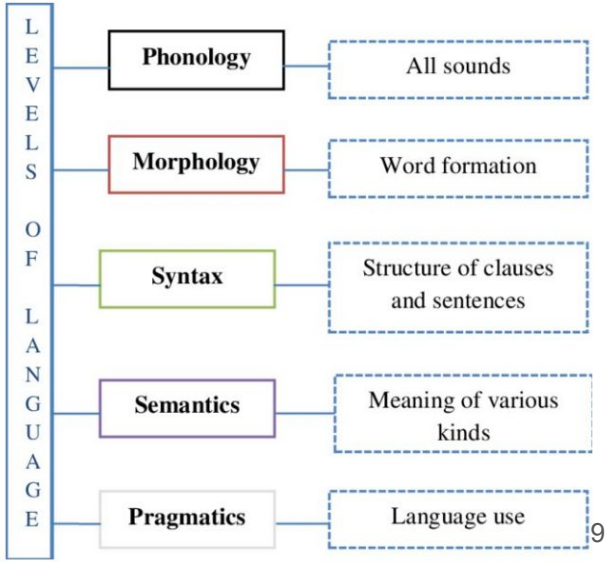
Περιεχόμενα

- Το κείμενο ως ακολουθία λέξεων
- NLP
- Deep Learning για κείμενα και ακολουθίες

Επεξεργασία φυσικής γλώσσας (NLP)

- Περιλαμβάνει όλες τις εργασίες μετατροπής κειμένου (συνήθως) σε δομημένη μορφή που μπορεί να αξιοποιηθεί από αλγορίθμους
- Κατανόηση της φυσικής γλώσσας: αφορά κυρίως την ανάγνωση της φυσικής γλώσσας
- Παραγωγή φυσικής γλώσσας: αφορά τις εργασίες μετατροπής δομημένων δεδομένων σε κείμενο (συνήθως)

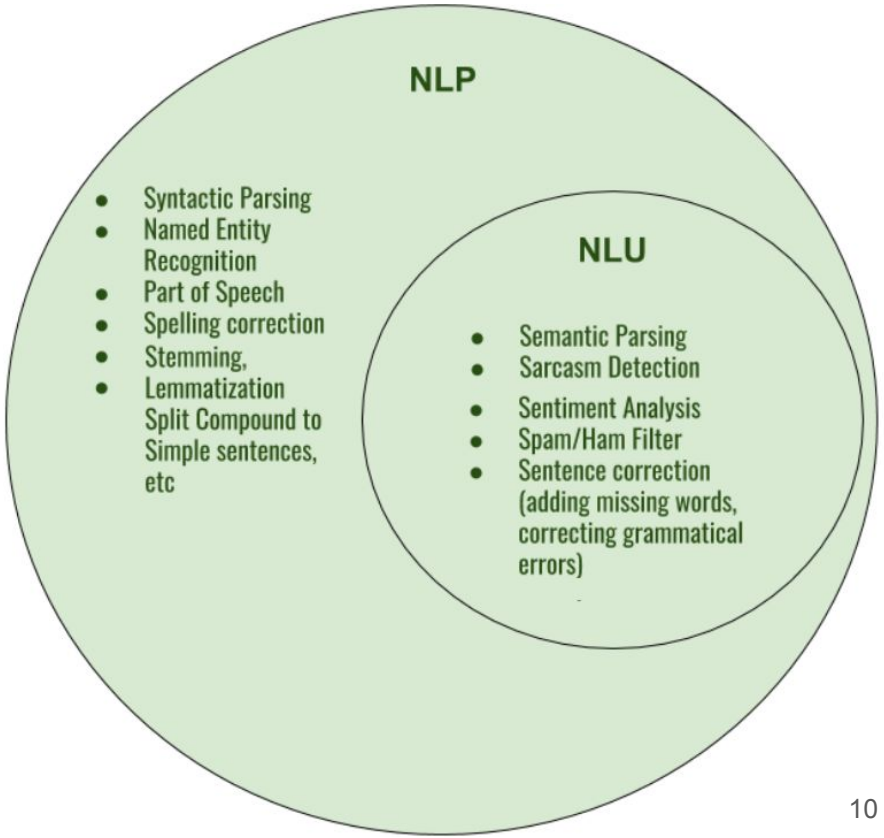




Βασικές κατηγορίες εφαρμογών NLU

- Κατηγοριοποίηση περιεχομένου
- Ανάλυση συναισθήματος
- Σημασιολογική αναζήτηση
- Αυτόματη αναγνώριση λόγου

- Παραγωγή περιλήψεων (+NLG)
- Μηχανική μετάφραση (+NLG)
- Question answering (+NLG)



Συνδυασμός NLP και Computer Vision (+audio)

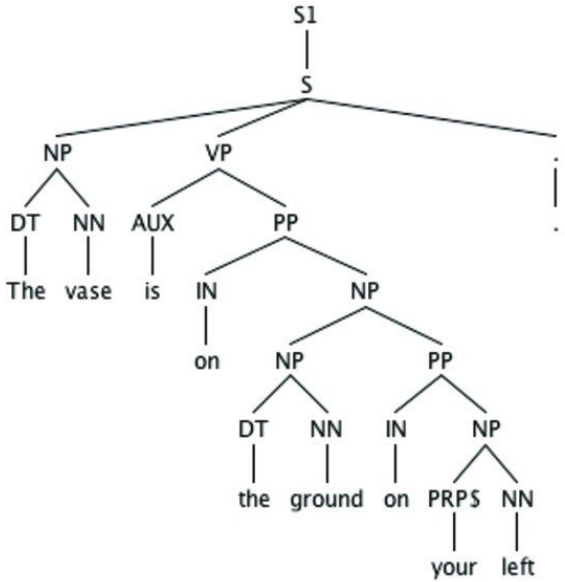
- Περιγραφή ιατρικών (και άλλων) εικόνων
- Λεκτική περιγραφή video (σε ανθρώπους με προβλήματα όρασης)
- Μετατροπή λεκτικών περιγραφών σε εικόνες ή βίντεο

- Text-to-speech, speech-to-text
- Μετατροπή νοηματικής σε λόγο ή κείμενο και αντίστροφα

- Νέοι τρόποι για τη διάδραση ανθρώπου-μηχανής

Συντακτική ανάλυση γλώσσας

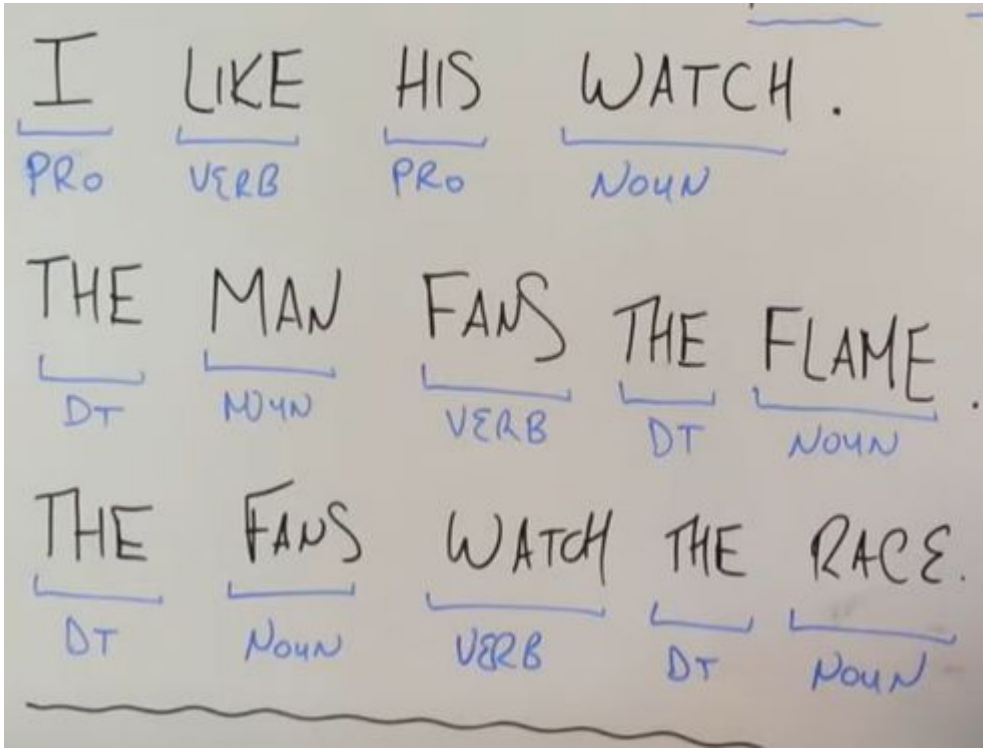
- **Part-of-speech tagging** και parse tree
- Αναθέτει μέρη του λόγου στις λέξεις του κειμένου και δομεί κάθε πρόταση ως ένα δέντρο



Penn Treebank Tagset

1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential there
5.	FW	Foreign word
6.	IN	Preposition or subordinating conjunction
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker
11.	MD	Modal
12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NP	Proper noun, singular
15.	NPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PP	Personal pronoun
19.	PP$	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol
25.	TO	to
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VBN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present
33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WP$	Possessive wh-pronoun
36.	WRB	Wh-adverb

Ασάφειες - Ambiguity resolution



Rule based - Linguistics

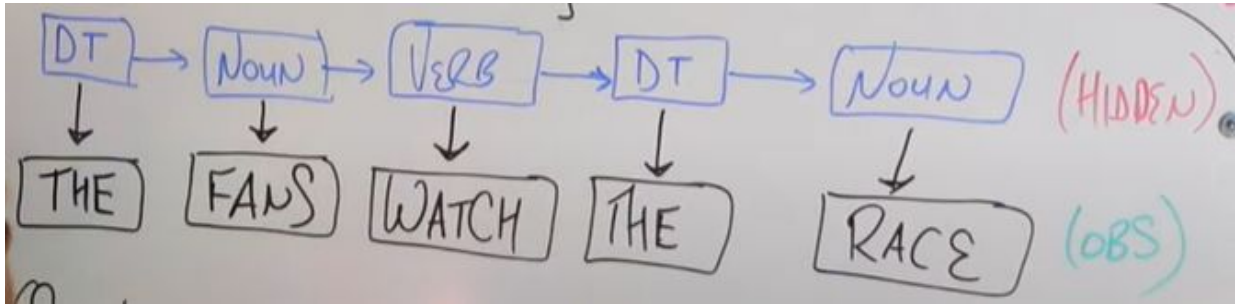
Probability based - HMM

Τα κείμενα ως ακολουθίες

- Ψάχνουμε για ετικέτες (μέρος του λόγου-pos) για κάθε λέξη.
- Στην ουσία ψάχνουμε για ακολουθίες ετικετών (pos-tags) που να είναι συχνές
- Ψάχνουμε για την πιο πιθανή ακολουθία ετικετών με δεδομένη την ακολουθία λέξεων
- Χρησιμοποιούμε Hidden Markov Models

Transition probabilities

Emission probabilities



Ψάχνουμε την ακολουθία hidden states (POS tags) που μεγιστοποιεί την πιθανότητα των observed states

Viterbi algorithm

1) Initialization: Για την πρώτη λέξη υπολογίζουμε την πιθανότητα κάθε POS tag με βάση τα emission probabilities P(word|tag)
2) Recursion: Για κάθε επόμενη λέξη, βρες το πιο πιθανό tag λαμβάνοντας υπόψη:
   a) Τη μέγιστη πιθανότητα του προηγούμενου tag
   b) Τη μέγιστη transition probability από το προηγούμενο tag στο τρέχον $P(tag_i \mid tag_{i-1})$
   c) Την emission probability της λέξης με βάση το νέο tag $P(word_i|tag_j)$
3) Backtracking: μετά και την τελευταία λέξη διασχίζουμε προς την αρχή για να βρούμε την πιο πιθανή ακολουθία tags

$$P(w \mid t) = \frac{\text{Count}(t \to w)}{\sum_{w'} \text{Count}(t \to w')}$$

Τύποι

Initialization: Η πιθανότητα να ξεκινήσω με το tag_j με δεδομένη την 1η λέξη

$$\delta_1(j) = P(\text{tag}_j) \cdot P(\text{word}_1|\text{tag}_j)$$

Recursion:

$$\delta_i(j) = \max_k \left[\delta_{i-1}(k) \cdot P(\text{tag}_j|\text{tag}_k) \cdot P(\text{word}_i|\text{tag}_j)\right]$$

Backtracking: Βρίσκουμε το πιθανότερο tag για την τελευταία λέξη και κάνουμε backtrack προς τα πίσω επιλέγοντας το POS με τη μεγαλύτερη πιθανότητα

$$\hat{z}_n = \arg\max_j \delta_n(j)$$

$$\hat{z}_i = \text{backpointer}_i(\hat{z}_{i+1})$$

Named entity recognition

- Εντοπισμός επωνύμων οντοτήτων στα κείμενα:
  - Άνθρωποι (person), τοποθεσίες (location), οργανισμοί (organization), γεωπολιτικές οντότητες (GPE), υποδομές (facility), μάρκες οχημάτων (vehicle), γονίδια (gene) κλπ.
  - Μπορεί να αποτελούνται από περισσότερες λέξεις
- Επίλυση ασαφειών: Paris (η πόλη ή η Paris Hilton)
- Πιο σπάνια: εντοπισμός χρονικών εκφράσεων, γεγονότων
- Μπορεί να λυθεί ως πρόβλημα κατηγοριοποίησης ή ως πρόβλημα sequence labeling

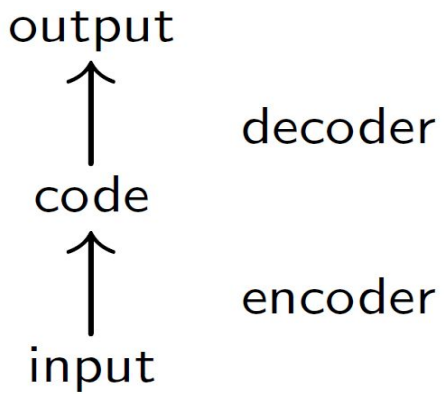
Words	IOB Label	IO Label
American	B-ORG	I-ORG
Airlines	I-ORG	I-ORG
,	O	O
a	O	O
unit	O	O
of	O	O
AMR	B-ORG	I-ORG
Corp.	I-ORG	I-ORG
,	O	O
immediately	O	O
matched	O	O
the	O	O
move	O	O
,	O	O
spokesman	O	O
Tim	B-PER	I-PER
Wagner	I-PER	I-PER
said	O	O
.	O	O

Περιεχόμενα

- Το κείμενο ως ακολουθία λέξεων
- NLP
- Deep Learning για κείμενα και ακολουθίες

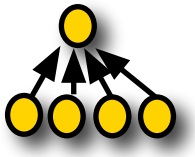
Γιατί deep learning NN;

- Είναι στη φύση τους ιεραρχικά μοντέλα και επιτρέπουν την επεξεργασία σε πολλαπλά επίπεδα
- Μας επιτρέπουν να μάθουμε αναπαραστάσεις/μετασχηματισμούς των δεδομένων που μπορούν στη συνέχεια να γίνουν είσοδοι σε έναν ταξινομητή
- Επιτρέπουν την εκπαίδευση end-to-end που έχει λιγότερη ανάγκη για κατασκευή γνωρισμάτων
- Pre-training σε μη επιβλεπόμενα δεδομένα και fine-tuning
- Encoder-Decoders και ενδιάμεσες αναπαραστάσεις Κατορθώνουν να ανασκευάσουν την είσοδο



Single vs Multi layer NNs

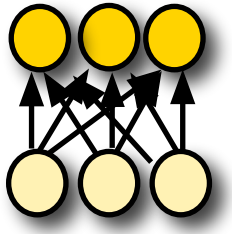
Binary classification



Output unit: scalar *y*

if y > 0.5 return 1
else return 0

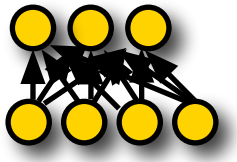
Input layer: vector **x**



Output layer: vector **y**

Hidden layer: vector $\mathbf{h_n}$

Multiclass classification

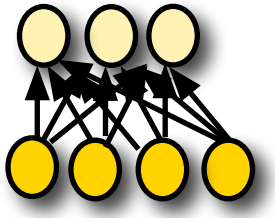


Output layer: vector **y**

$$\text{argmax}_i(y_i)$$

$$y_i = P(i) = \text{softmax}(z_i) = \exp(z_i)/\sum_{k=o..K} \exp(z_k)$$

Input layer: vector **x**

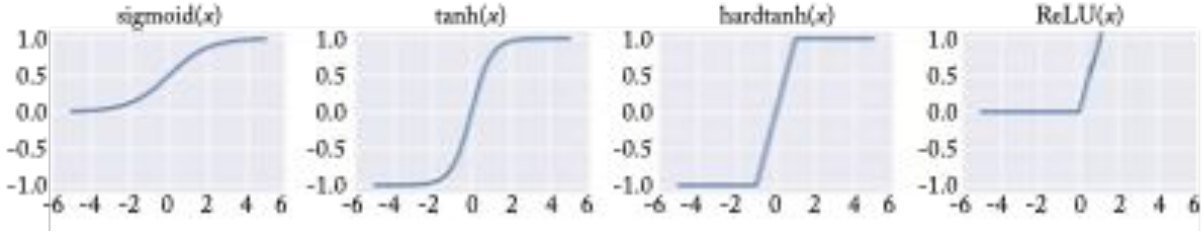


Hidden layer: vector $\mathbf{h_1}$

Input layer: vector **x**

Υπενθύμιση

Μη γραμμικές συναρτήσεις ενεργοποίησης



Sigmoid (logistic function): $\sigma(x) = 1/(1 + e^{-x})$

Χρήσιμη για το επίπεδο εξόδου (πιθανότητες) [0,1] range

Hyperbolic tangent: $\tanh(x) = (e^{2x} - 1)/(e^{2x}+1)$

Χρήσιμη για εσωτερικά επίπεδα: [-1,1] range

Hard tanh (approximates tanh)

htanh(x) = -1 για $x < -1$, 1 για $x > 1$, αλλιώς x

Rectified Linear Unit: $\text{ReLU}(x) = \max(0, x)$

Χρήσιμη για εσωτερικά επίπεδα

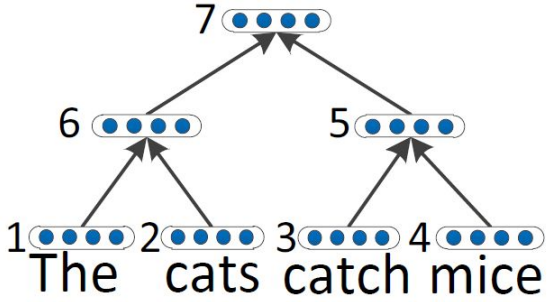
Μαθαίνοντας Sequence-Based γνωρίσματα

- Τα bigrams είναι γνωρίσματα που κατασκευάζονται χειροκίνητα και διατηρούν κάποια πληροφορία σχετικά με τη σειρά των λέξεων.
- Μπορούμε να κάνουμε το μοντέλο να μάθει να κατασκευάζει τα δικά του γνωρίσματα που περιέχουν πληροφορίες σχετικά με τη σειρά των λέξεων;
- Αυτό έχουν σχεδιαστεί να κάνουν τα αναδρομικά μοντέλα (Recurrent models)

Recurrent NNs

Recurrent neural networks (RNNs) are a family of neural networks for processing sequential data

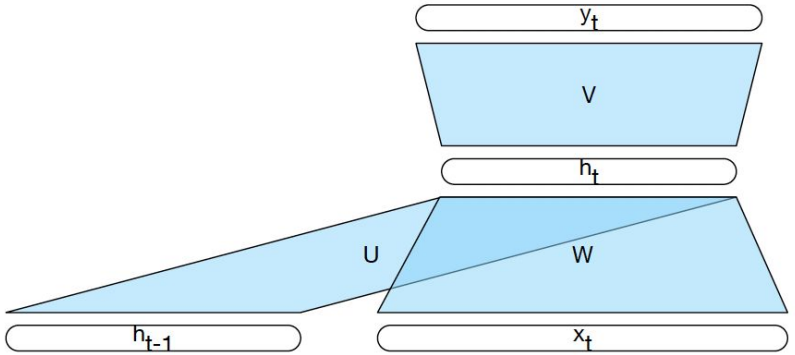
- Μπορούν να χειριστούν εισόδους μεταβλητού μήκους
- Έχουν δενδρική δομή που τη μαθαίνουν από τα κείμενα
- Κάθε κόμβος έχει έναν auto-encoder για να μάθει μια εσωτερική αναπαράσταση

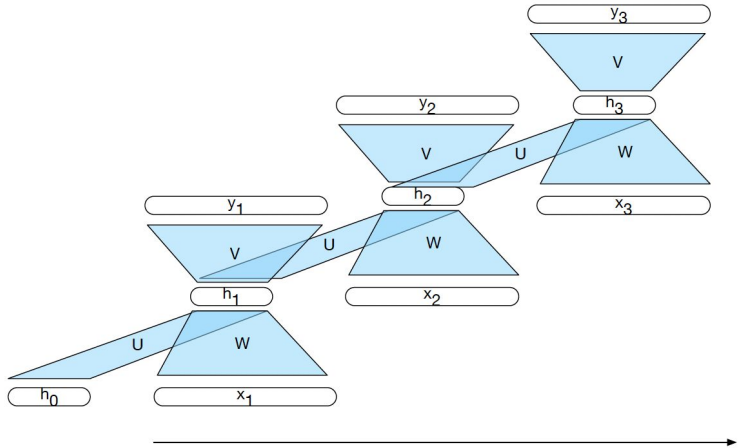


- Paraphrase detection
- Sentiment analysis
- Parsing

Recurrent NNs

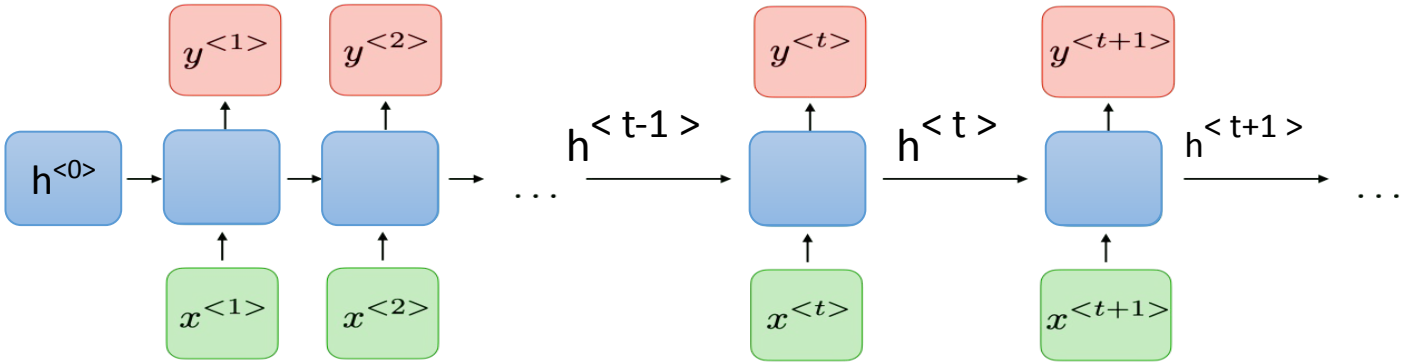
- Επεξεργάζονται μια ακολουθία βήμα προς βήμα.
- Οι μονάδες ενός αναδρομικού επιπέδου λαμβάνουν πληροφορίες τόσο από τα προηγούμενα βήματα όσο και από το τρέχον βήμα και συνδυάζουν αυτές τις πληροφορίες για να υπολογίσουν την έξοδό τους.
- Σε σύγκριση με τις μονάδες SimpleRNN, οι μονάδες LSTM έχουν ακόμη μεγαλύτερη ικανότητα να διατηρούν πληροφορίες από τα προηγούμενα βήματα και από παλαιότερα βήματα στο παρελθόν.





Recurrent NNs

Τα *RNNs* χρησιμοποιούν τις προηγούμενες εξόδους ως εισόδους και έχουν και hidden units:



Για κάθε βήμα *t*, το hidden state $h^{<t>}$ και έξοδος $y^{<t>}$ είναι:

$$\mathbf{h^{<t>} = g_1(W_{hh}h^{<t-1>} + W_{hx}x^{<t>} + b_h)} \text{ and } \mathbf{y^{<t>} = g_2(W_{yh}h^{<t>} + b_y)}$$

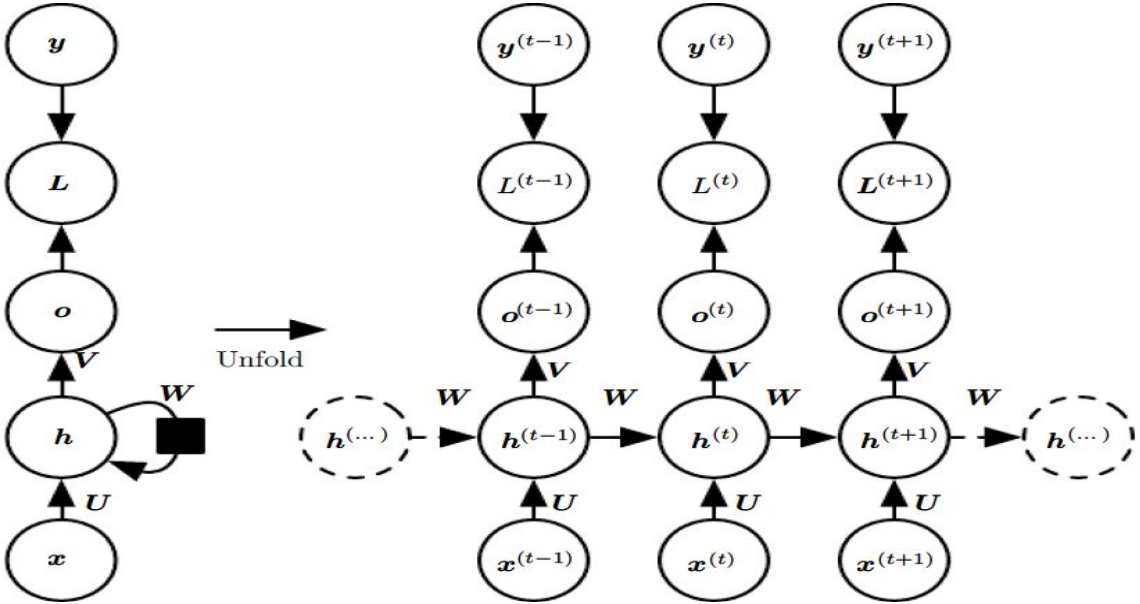
Hidden state

- Το RNN μαθαίνει να χρησιμοποιεί το hidden state ως μια περίληψη των προηγούμενων βημάτων
- Μια ακολουθία προηγούμενων εισόδων $(\mathbf{x}^{<t>}, \mathbf{x}^{<t-1>}, \mathbf{x}^{<t-2>}, \ldots, \mathbf{x}^{<2>}, \mathbf{x}^{<1>})$ αντιστοιχεί σε ένα διάνυσμα σταθερού μήκους $\mathbf{h}^{<t>}$

- Υπάρχουν διάφορες παραλλαγές των RNN ανάλογα με τη συνδεσμολογία και το ποιος τροφοδοτεί το hidden layer

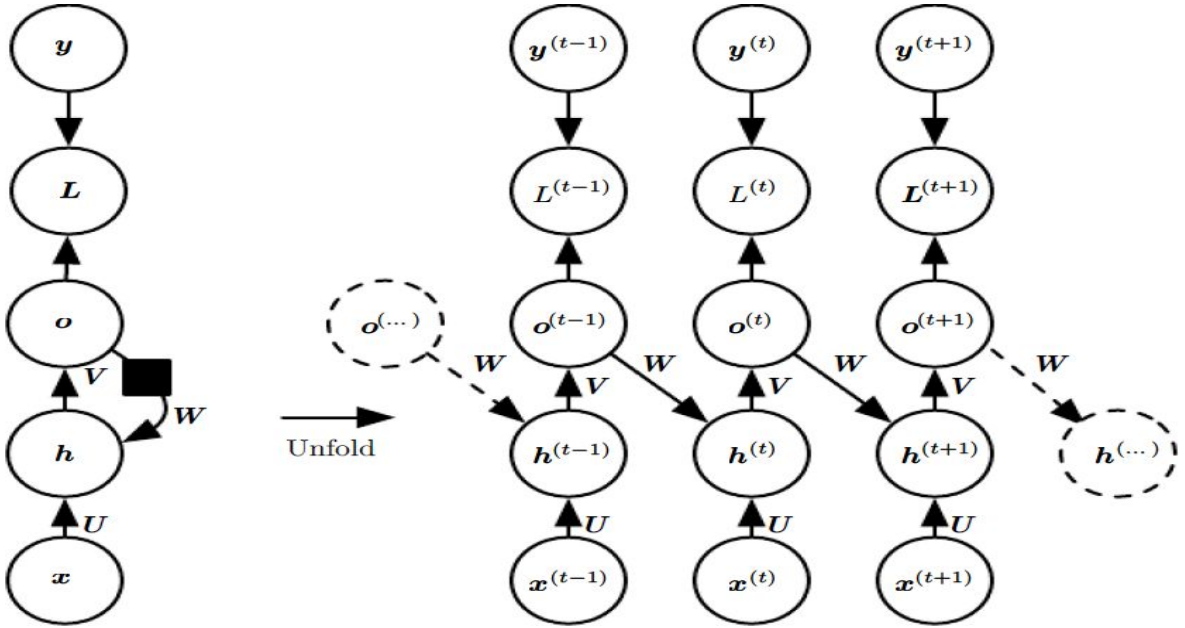
RNN με αναδρομικές συνδέσεις στα hidden units

Αντιστοιχούν την ακολουθία εισόδου σε μια ακολουθία ίσου μήκους στην έξοδο



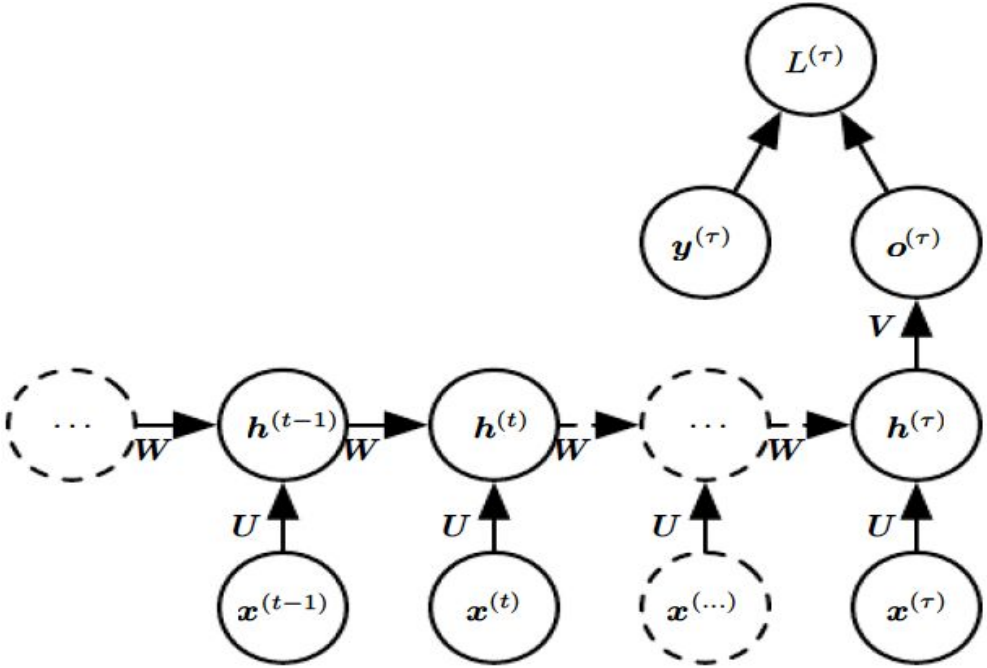
RNN που στέλνουν την προηγούμενη έξοδο στο hidden unit

Είναι πιο εύκολο να εκπαιδευτούν αλλά είναι λιγότερο αποτελεσματικά



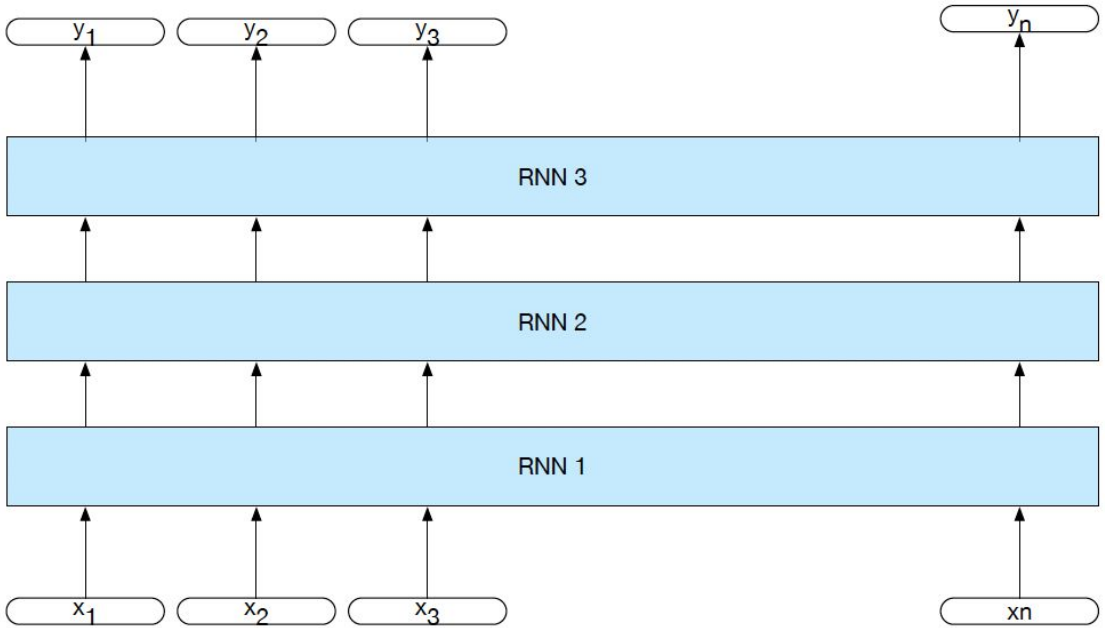
RNN που για μια ακολουθία παράγουν μία έξοδο

Χρησιμοποιούνται για τη δημιουργία περιλήψεων σταθερού μήκους, ή για κατηγοριοποίηση



Stacked RNNs

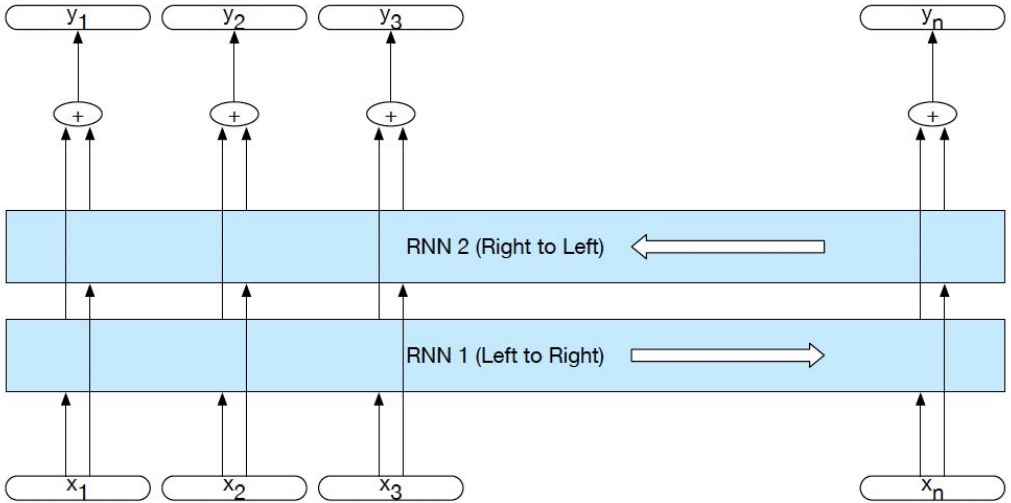
Αυτά τα RNN έχουν “βάθος” σε κάθε βήμα



Bidirectional RNNs

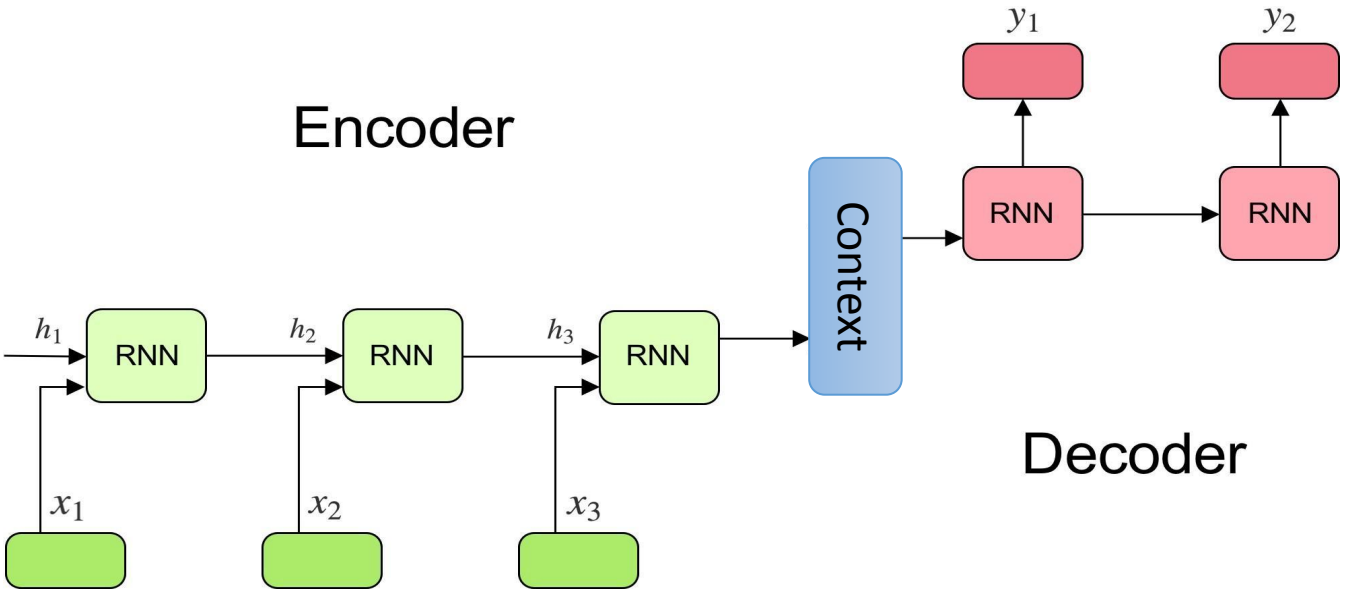
Σε προβλήματα που δεν παράγουν ακολουθία λέξεων μπορούμε να δούμε την είσοδο και προς τις δύο κατευθύνσεις.

Έτσι το κρυφό επίπεδο μαθαίνει διαφορετικό περιεχόμενο κάθε φορά.



Encoder-Decoder (seq2seq)

Ένα μοντέλο sequence to sequence (Seq2Seq) μετατρέπει μια ακολουθία στην είσοδο σε μια ακολουθία στην έξοδο (όχι απαραίτητα ίδιου μήκους)



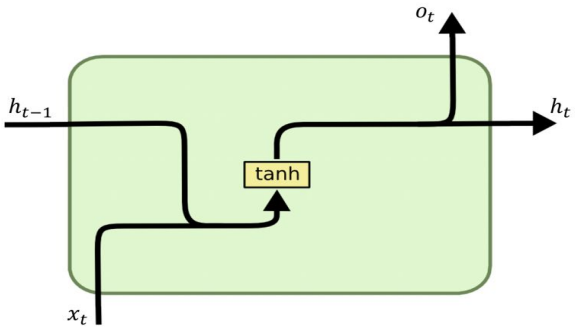
Encoder-Decoder (seq2seq)

- Encoder: Επεξεργάζεται την είσοδο (sequence-to-vector RNN)
- Context: η έξοδος του encoder
  - Μια απλή συνάρτηση του τελικού hidden state
  - Περιέχει την πληροφορία όλων των δεδομένων εισόδου για να βοηθήσει τον decoder να προβλέψει με ακρίβεια
- Decoder: βλέπει το context και παράγει την ακολουθία στην έξοδο
  - Το context είναι η αρχική hidden state του decoder μέρους του μοντέλου
  - Παράγει την έξοδο σε κάθε βήμα

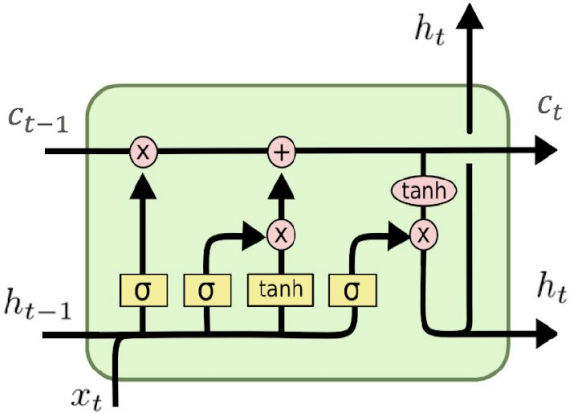
LSTM και εξαρτήσεις σε απόσταση

- Το πρόβλημα του vanishing gradient και των exploding gradients: μικρές διορθώσεις τείνουν να εξαφανιστούν (μεγάλες διορθώσεις μπορεί να επικρατήσουν) μετά από λίγες λέξεις, έτσι τα RNN δεν μπορούν να θυμούνται πολλά βήματα πίσω.
- Τα LSTM δίκτυα και τα GRU δίκτυα είναι μια λύση

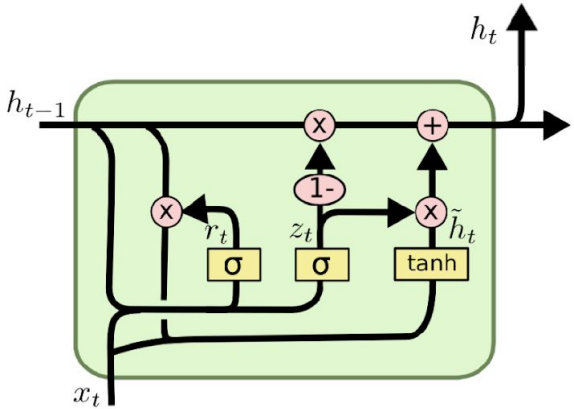
Σύγκριση



Vanilla RNN



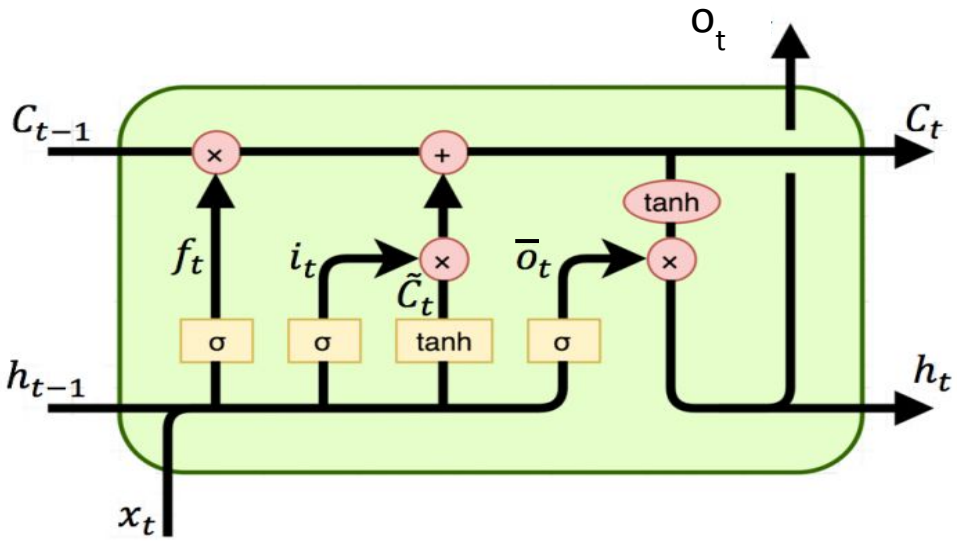
LSTM
(Long-Short Term Memory)



GRU
(Gated Recurrent Unit)

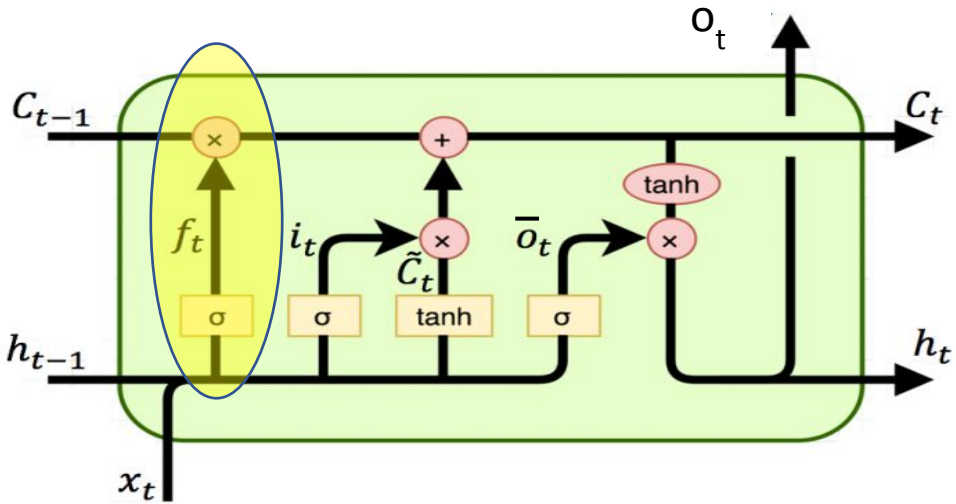
LSTM

An internal cell state is maintaining the global information at each time



LSTM

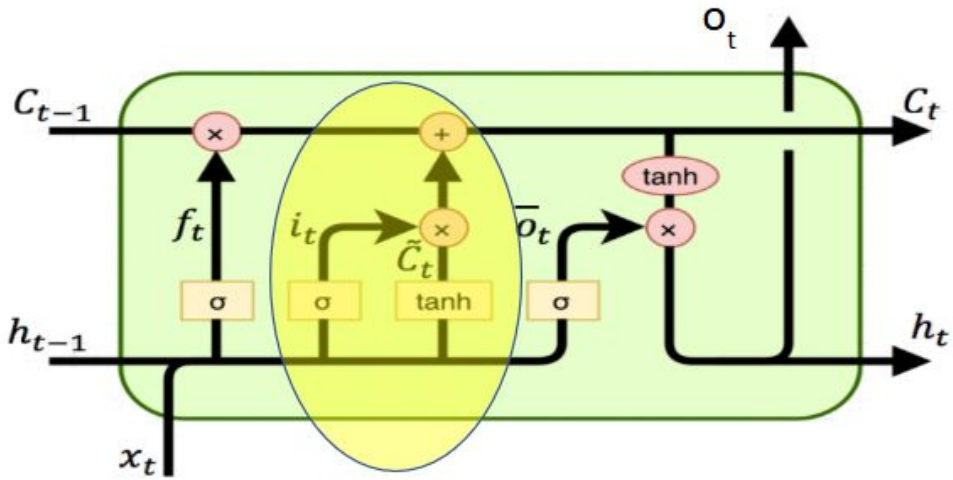
The forget gate is used to remove irrelevant information from the cell state as new input is encountered



$f_t = \sigma (W_f * [h_{t-1}, x_t] + b_f$

LSTM

The input logic adds new information in the cell state



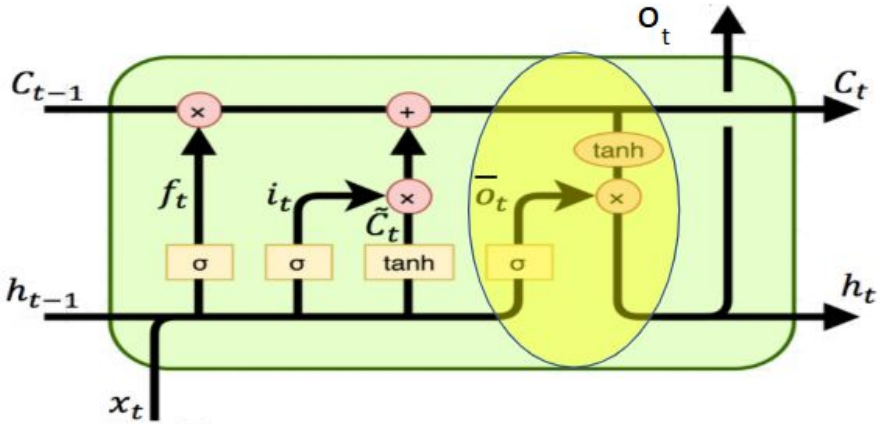
$i_t = \sigma (W_i * [h_{t-1}, x_t] + b_i$ ← decides which values to retain

$C_t = \tanh (\tilde{W}_c * [h_{t-1}, x_t] + b_c$ ← new candidate values

$$C_t = f_t \odot C_{t-1} + \tilde{C}_t \odot i_t$$

LSTM

The output gate computes the new hidden state based on the updated cell state and the output



$$\bar{o}_t = \sigma\,(W_o * [h_{t-1}, x_t] + b_o$$

$$h_t = \bar{o}_t \odot \tanh(C_t)$$

Εφαρμογές των RNN

- Text Generation
- Text Summarization
- Report Generation
- Timeseries anomaly detection
- Speech Recognition
- Call Center Analysis
- Handwriting Recognition
- Music composition
- Grammar Learning
- Sequence Prediction/Forecasting
- Conversational Interfaces and chatbots
- Machine Translation
- Visual Search, Face detection, OCR Applications
- Semantic Search
- Sentiment Analysis
- Human action recognition
- Protein Homology Detection

Sequential Data

- text data
- protein/gene sequences
- sequential patterns: e.g., sequences of purchases
- timeseries
  - stock prices
  - amount of precipitation
  - ECG data
  - music
  - audio data
  - trajectories
  - video