ΜΥΕ003: Ανάκτηση Πληροφορίας

Διδάσκουσα: Ευαγγελία Πιτουρά

Σημασιολογική Ανάκτηση Πληροφορίας. Ενσωματώσεις (embeddings).

Ακαδημαϊκό Έτος 2024-2025

- Διαβάθμιση βασισμένη σε tf-idf
  Άσκηση

- Embeddings (ενσωματώσεις)

Στάθμιση tf-idf

Term frequency (tf)

$tf_{t,d}$ #εμφανίσεις του όρου t στο έγγραφο d

Inverse document frequency (idf)

$$idf_t = \frac{N}{df_t}$$

$$\text{score}(q,d) = \sum_{t \in q \cap d} w_{t,d}$$

$$w_{t,d} = \left(1 + log_{10} tf_{t,d}\right) log_{10}(idf_t)$$

Έγγραφα	
d_1	a b c
d_2	a d a b
d_3	a c d e c
d_4	b e a b
d_5	a d e

Ερώτημα

q b c

Okapi BM25

$$score(d,q) = \sum_{t \in d \cap q} fdf(t) \frac{tf_{t,d}\,(k_1+1)}{tf_{t,d} + k_1(1 - b + b\frac{|d|}{avgd1}}$$

$$fdf(t) = \ln(\frac{N - df(t) + 0.5}{df(t) + 0.5} + 1)$$

$|d|$ document length in words
$avgd1$ average document length
$k_1$parameter, if no learning $k_1 \in [1.2, 2]$
b parameter, if no learning $b = 0.75$
N number of documents in the collection

Based on the probabilistic retrieval model

Στάθμιση στο διανυσματικό χώρο

1. Αναπαράσταση κάθε εγγράφου ως ένα tf-idf διάνυσμα
2. Αναπαράσταση του ερωτήματος ως ένα tf-idf διάνυσμα
3. Υπολογισμός της (cosine) ομοιότητας για κάθε ζεύγος ερωτήματος, εγγράφου
4. Διάταξη των εγγράφων με βάση την ομοιότητα

Έγγραφα	
d_1	a b c
d_2	a d a b
d_3	a c d e c
d_4	b e a b
d_5	a d e

Ερώτημα

q b c

Διαβάθμιση βασισμένη σε tf-idf
Άσκηση

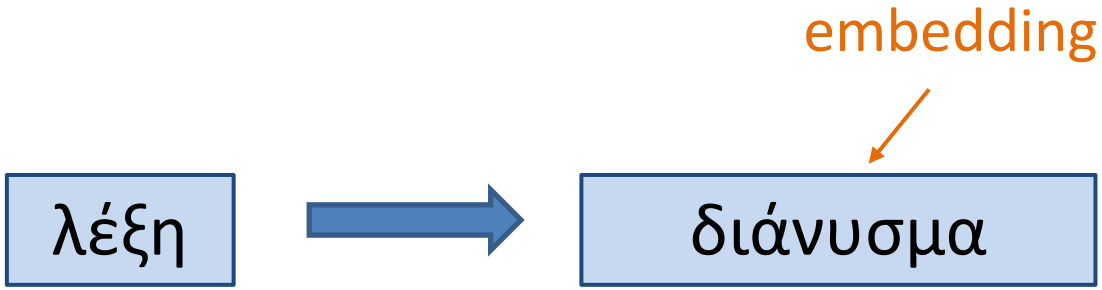
Embeddings (ενσωματώσεις)

Retrieval Augmented Generation

Word embeddings

- Διανυσματική αναπαράσταση λέξεων

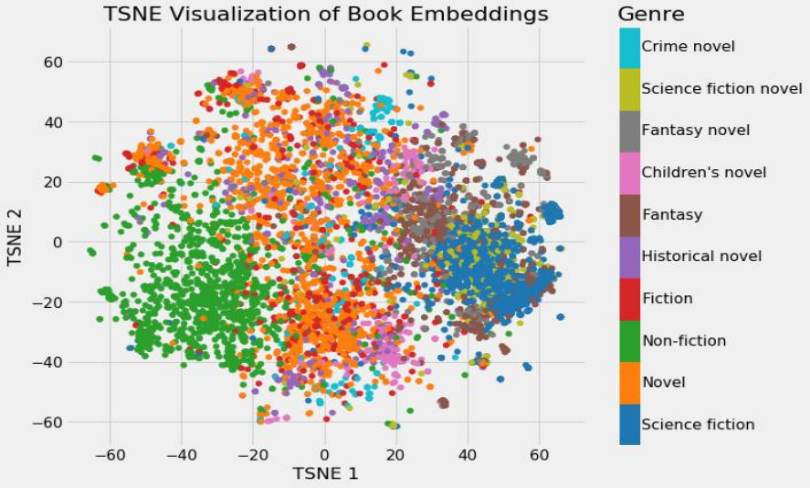
Στόχος: Διανυσματική *αναπαράσταση* (representation) λέξεων – *κατανεμημένη (distributed) αναπαράσταση*



Στόχος: παρόμοιες λέξεις -> παρόμοια διανύσματα

Word embeddings

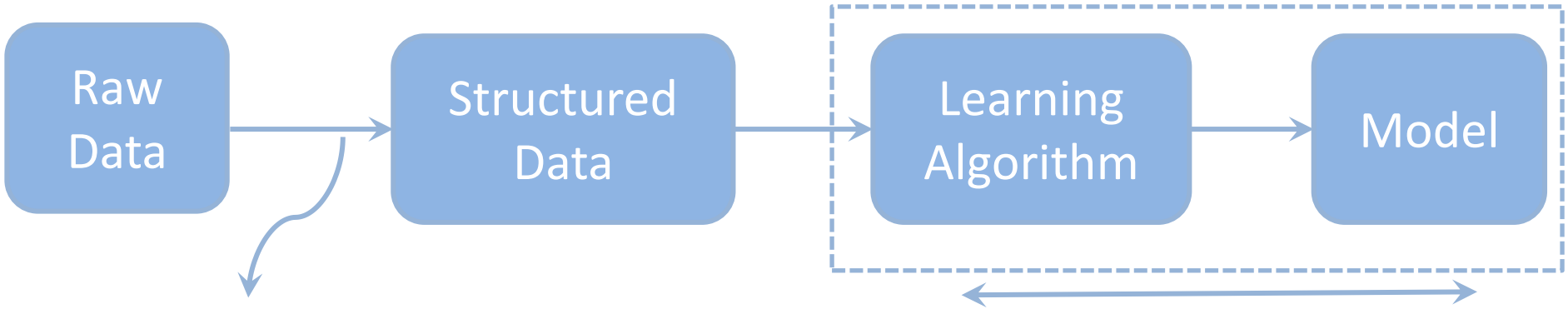
- Διανυσματική αναπαράσταση λέξεων



Books on Wikipedia colored by genre in two dimensions. Source: Koehrsen 2018

Embeddings: why?

Machine learning lifecycle



For words: document occurrences, k-grams, etc

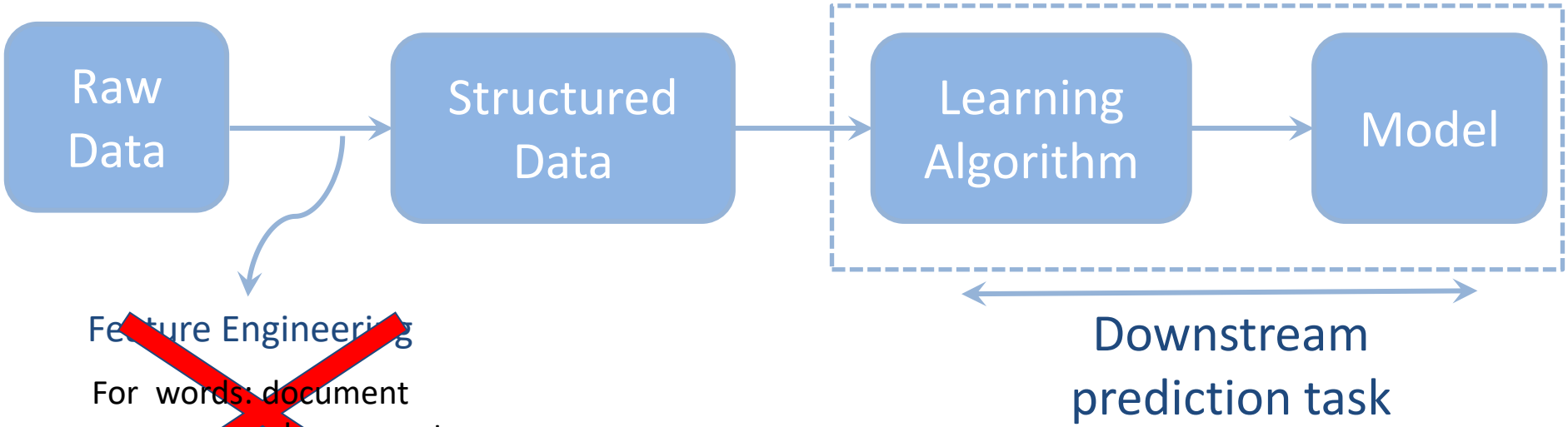
For documents: length, words, etc

Classification
Learning to rank
Clustering

Query answering, machine translation

Embeddings: why?

Machine learning lifecycle



Automatically learn the features (embeddings)

Πως θα πάρουμε αυτά τα διανύσματα;

One-hot vectors

Έστω ότι υπάρχουν |V| διαφορετικές λέξεις (όροι) στο λεξικό μας

- Διατάσσουμε τις λέξεις αλφαβητικά
- Αναπαριστούμε κάθε λέξη με ένα $\mathrm{R}^{|V|x1}$ διάνυσμα που έχει παντού 0 και μόνο ένα 1 στη θέση που αντιστοιχεί στη θέση της λέξης στη διάταξη

$$w^{aardvark} = \begin{bmatrix} \mathbf{1} \\ 0 \\ 0 \\ . \\ . \\ . \\ 0 \end{bmatrix} \quad w^{a} = \begin{bmatrix} 0 \\ \mathbf{1} \\ 0 \\ . \\ . \\ . \\ 0 \end{bmatrix} \quad w^{at} = \begin{bmatrix} 1 \\ 0 \\ \mathbf{0} \\ . \\ . \\ . \\ 0 \end{bmatrix} \quad \cdots \quad w^{zerba} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ . \\ . \\ . \\ \mathbf{1} \end{bmatrix}$$

- Κάθε λέξη ένα διαφορετικό σύμβολο
- Πολλές διαστάσεις (|V| όσες οι λέξεις
- Καμία πληροφορία για ομοιότητα

Term-Document co-occurrence matrix

Έστω ότι υπάρχουν |V| διαφορετικές λέξεις (όροι) στο λεξικό μας και |M| έγγραφα

- Κατασκευάζουμε ένα |V|x M πίνακα με τις εμφανίσεις των λέξεων στα έγγραφα
- Αναπαριστούμε κάθε λέξη με ένα $\mathrm{R}^{|\mathrm{M}|x1}$

Παράδειγμα:

d1: a b c
d2: a d a b
d3: a c d e c a f
d4: b e a b
d5: a b d c a

|V| = 6, |M| =5

	d1	**d2**	**d3**	**d4**	**d5**
a	1	1	1	1	1
b	1	1	0	1	1
c	1	0	1	0	1
d	0	1	1	0	1
e	0	0	1	1	0
f	0	0	1	0	0

Word vector for c

Term-Document co-occurrence matrix

Μπορούμε αντί για 0-1 να έχουμε το tf ή και το tf-idf βάρος

- Πολλές διαστάσεις
- Πρόβλημα κλιμάκωσης με τον αριθμό των εγγράφων

Window-based co-occurrence matrix

- Κατασκευάζουμε ένα $|V|x|V|$ affinity-matrix για τις λέξεις: για δύο λέξεις, μετράμε τον αριθμό των φορών που αυτές οι δύο λέξεις εμφανίζονται μαζί σε έγγραφα
- Συγκεκριμένα, μετράμε τον αριθμό των φορών που κάθε λέξη εμφανίζεται μέσα σε ένα *παράθυρο* συγκεκριμένου μεγέθους W γύρω από τη λέξη ενδιαφέροντος

Παράδειγμα:

d1: a b c
d2: a d a b
d3: a c d e c a f
d4: b e a b
d5: a b d c a

W = 1 (σε απόσταση 1)

	a	b	c	d	e	f
a	0	4	3	1	1	1
b	4	0	1	1	1	0
c	3	1	0	2	1	0
d	1	1	2	0	1	0
e	1	1	1	1	0	0
f	1	0	0	0	0	0

Window-based co-occurrence matrix

Λέξεις όπως apple, orange, mango, κλπ μαζί με λέξεις όπως eat, grow, cultivate, slice, κλπ και το ανάποδο

Παράδειγμα:

d1: I enjoy flying.
d2: I like NLP.
d3: I like deep learning.

W = 1

$$X = \begin{array}{c} \\ I \\ like \\ enjoy \\ deep \\ learning \\ NLP \\ flying \\ . \end{array} \begin{array}{c} \begin{array}{cccccccc} I & like & enjoy & deep & learning & NLP & flying & . \end{array} \\ \begin{bmatrix} 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

- Πολλές διαστάσεις

Θα μπορούσαμε να χρησιμοποιήσουμε μια τεχνική για να μειώσουμε τις διαστάσεις (dimensionality reduction) (πχ PCA analysis – SVD)

Singular Value Decomposition

From dimension *n* to dimension *r*

$$A = U \quad \Sigma \quad V^T = [\vec{u}_1 \quad \vec{u}_2 \quad \cdots \quad \vec{u}_n] \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix} \begin{bmatrix} \vec{v}_1 \\ \vec{v}_2 \\ \vdots \\ \vec{v}_n \end{bmatrix}$$

$[n \times n] \qquad [n \times n] \qquad [n \times n]$

- $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$: singular values (square roots of eigenvals AA^T, A^TA)
- $\vec{u}_1, \vec{u}_2, \cdots, \vec{u}_n$: left singular vectors (eigenvectors of AA^T)
- $\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_n$: right singular vectors (eigenvectors of A^TA)
- Cut the singular values at some index r (get the largest r such values)

Singular Value Decomposition

A_r best approximation of A
(Frobernius norm)

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} {a_{ij}}^2}$$

$$A = U \quad \Sigma \quad V^T = \begin{bmatrix} \vec{u}_1 & \vec{u}_2 & \cdots & \vec{u}_r \end{bmatrix} \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_r \end{bmatrix} \begin{bmatrix} \vec{v}_1 \\ \vec{v}_2 \\ \vdots \\ \vec{v}_r \end{bmatrix}$$

$$[n \times r]\,[r \times r]\,[r \times n]$$

- **r** : rank of matrix A (the number of the non zero values)
- $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r$: singular values (square roots of eigenvals AA^T, A^TA)
- $\vec{u}_1, \vec{u}_2, \cdots, \vec{u}_r$: left singular vectors (eigenvectors of AA^T) (first r columns, ranked by σ)
- $\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_r$: right singular vectors (eigenvectors of A^TA) (first r rows)

$$A_r = \sigma_1 \vec{u}_1 \vec{v}_1^T + \sigma_2 \vec{u}_2 \vec{v}_2^T + \cdots + \sigma_r \vec{u}_r \vec{v}_r^T$$

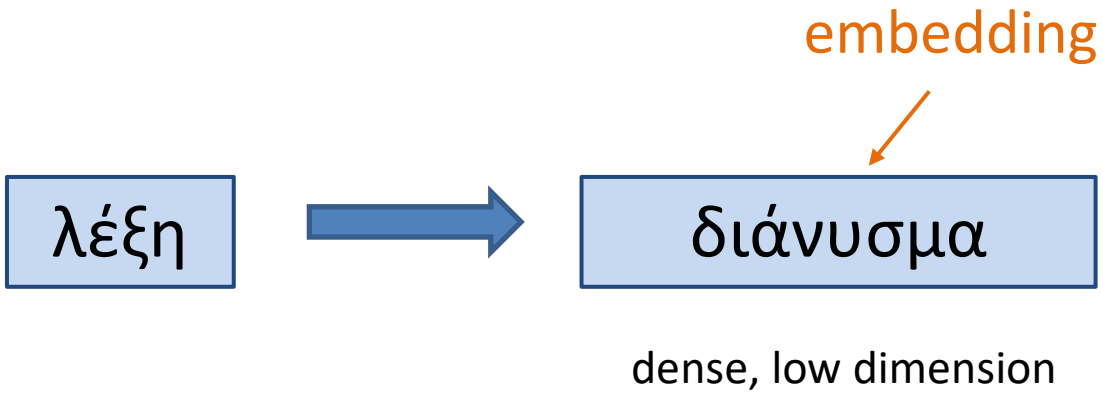
Αλλά

- Δύσκολο να ενημερώσουμε, πχ, αλλάζουν οι διαστάσεις συχνά
- Αραιός πίνακας
- Πολύ μεγάλες διαστάσεις

Θα δούμε μια τεχνική που βασίζεται σε επαναληπτικές μεθόδους –
Θα «μάθουμε» αυτά τα διανύσματα

word2vec

Στόχος: Διανυσματική *αναπαράσταση* (representation) λέξεων – *κατανεμημένη (distributed) αναπαράσταση*



Στόχος: παρόμοιες λέξεις -> παρόμοια διανύσματα

word2vec

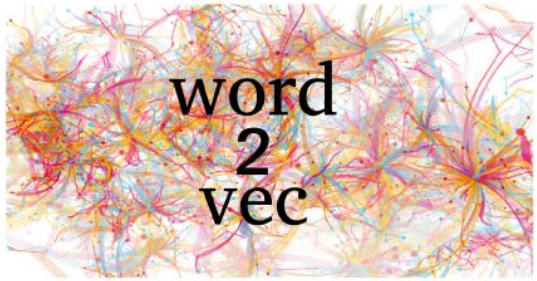
Είσοδος κείμενα, προτάσεις

feed in text

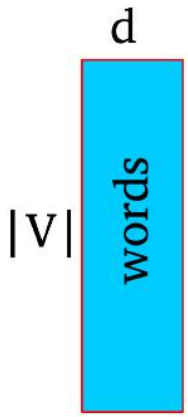
Text

Google news

...



wait a few hours



dog = (0.12,-0.32,0.92,0.43,-0.3 ...)
cat = (0.15,-0.29,0.90,0.39,-0.32 ...)

chair = (0.8,0.9,-0.76,0.29,0.52 ...)

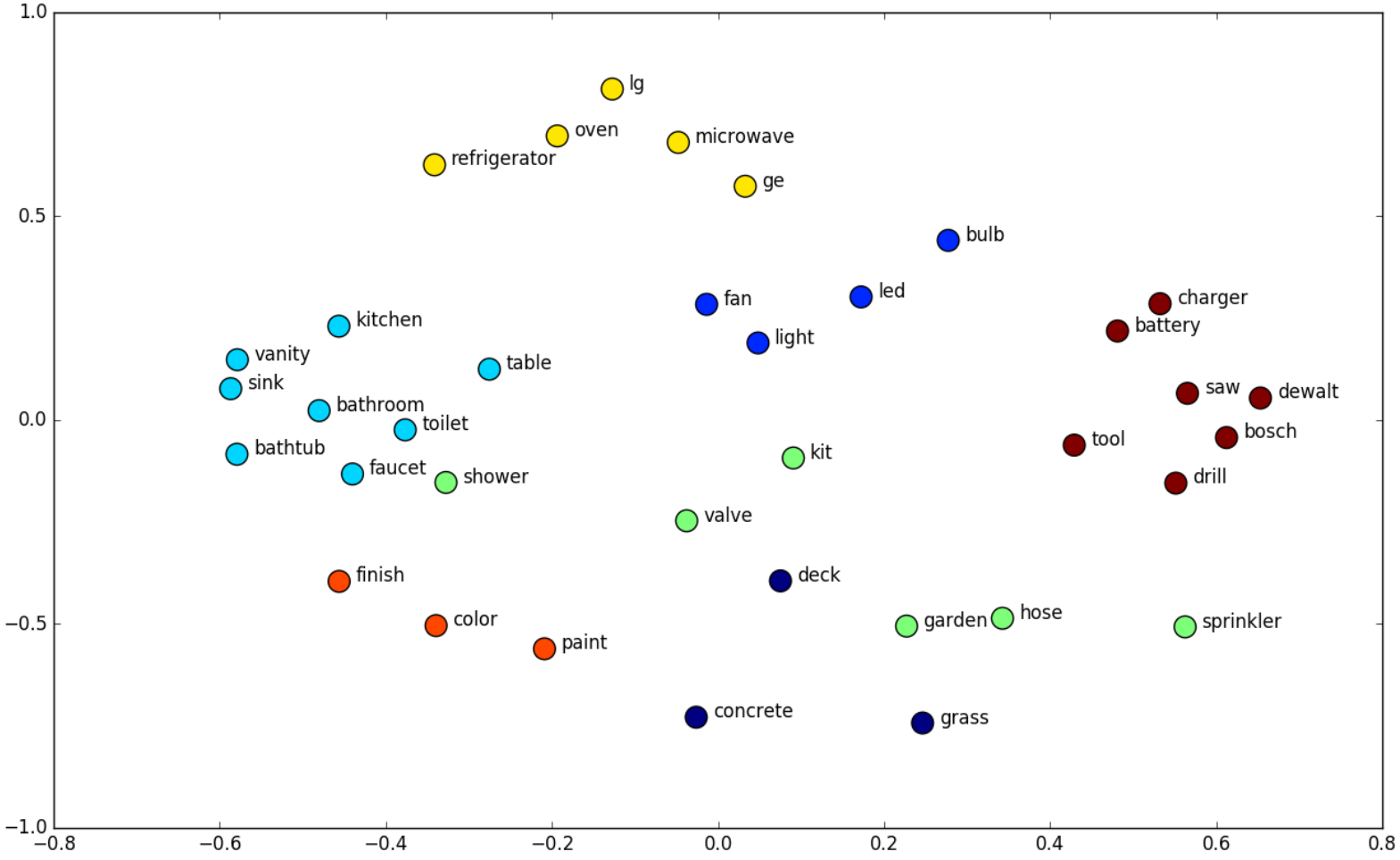
d συνήθως μεταξύ του 50 και του 300

get a |V|xd matrix W where each row is a vector for a word

word2vec

- dog
  - cat, dogs, dachshund, rabbit, puppy, poodle, rottweiler, mixed-breed, doberman, pig
- sheep
  - cattle, goats, cows, chickens, sheeps, hogs, donkeys, herds, shorthorn, livestock
- november
  - october, december, april, june, february, july, september, january, august, march
- jerusalem
  - tiberias, jaffa, haifa, israel, palestine, nablus, damascus katamon, ramla, safed
- teva
  - pfizer, schering-plough, novartis, astrazeneca, glaxosmithkline, sanofi-aventis, mylan, sanofi, genzyme, pharmacia

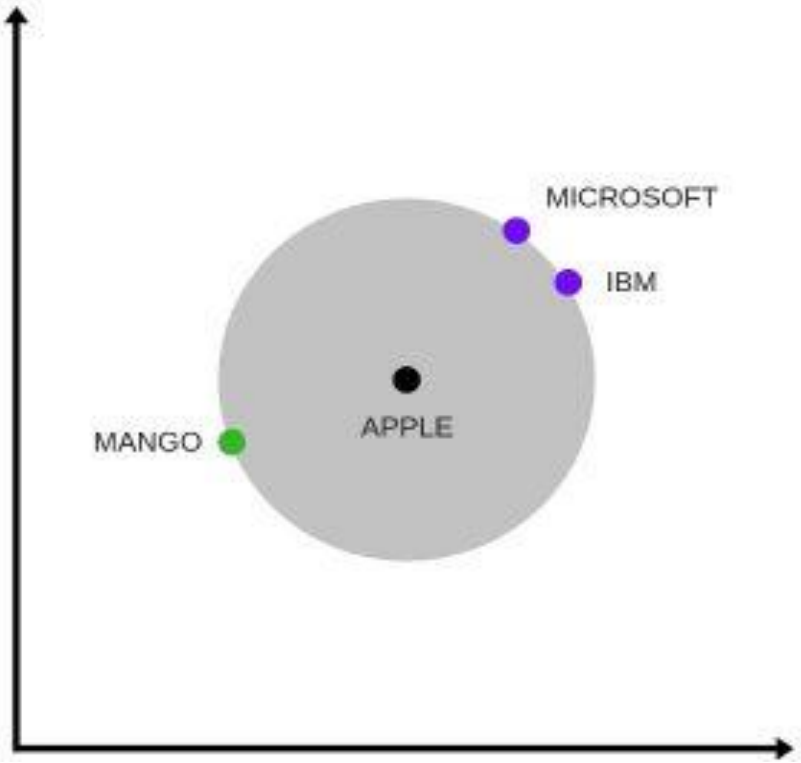
Παράδειγμα: 2-διάστατα embeddings



t-SNE plot

http://suriyadeepan.github.io

Apple: φρούτο και εταιρεία



https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/

Σε προηγούμενα μαθήματα είδαμε

Lemmatization
Stemming

Λέξεις σημασιολογικά κοντινές

Ομοιότητα/απόσταση

- Similarity is calculated using *cosine similarity*:

$$sim(\vec{dog}, \vec{cat}) = \frac{\vec{dog} \cdot \vec{cat}}{||\vec{dog}||\,||\vec{cat}||}$$

- For normalized vectors ($||x|| = 1$), this is equivalent to a dot product:

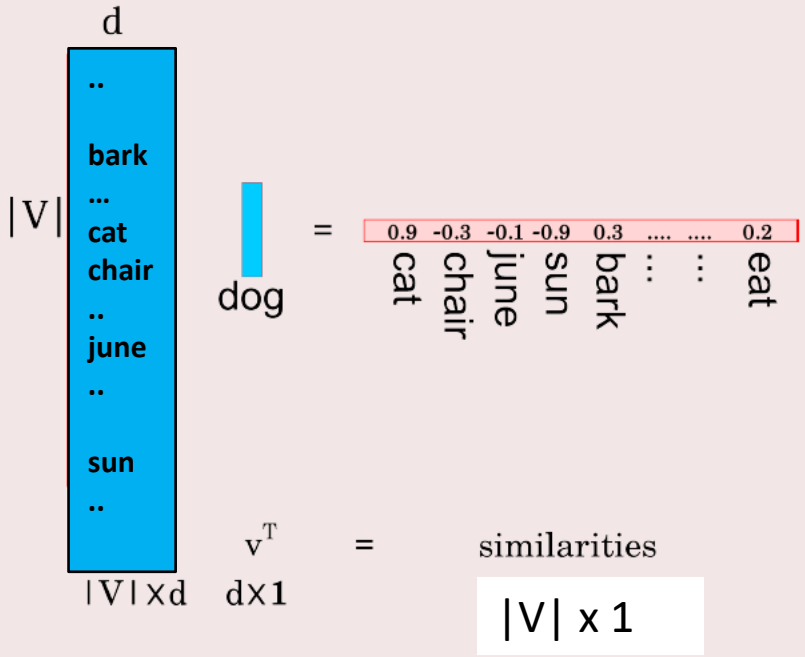
$$sim(\vec{dog}, \vec{cat}) = \vec{dog} \cdot \vec{cat}$$

- **Normalize the vectors when loading them.**

Παράδειγμα: Έστω ότι έχουμε τον πίνακα W με τα embeddings όλων των λέξεων, πως θα βρούμε την ποιο όμοια λέξη με το “dog”?

TIP: Όπου μπορούμε χρησιμοποιούμε πράξεις πινάκων. Γιατί;

- Compute the similarity from word $\vec{v}$ to all other words.
- This is a **single matrix-vector product**: $W \cdot \vec{v}^{\top}$



- Result is a $|V|$ sized vector of similarities.
- Take the indices of the k-highest values.

Λέξη ποιο όμοια σε πολλές άλλες;

- "Find me words most similar to cat, dog and cow".
- Calculate the pairwise similarities and sum them:

$$W \cdot \vec{cat} + W \cdot \vec{dog} + W \cdot \vec{cow}$$

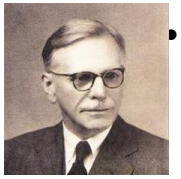
- Now find the indices of the highest values as before.
- Matrix-vector products are wasteful. **Better option:**

$$W \cdot (\vec{cat} + \vec{dog} + \vec{cow})$$

Πως θα πάρουμε αυτόν τον πίνακα;

Basic Idea

- You can get a lot of value by representing a word by means of its neighbors (distributional semantics)
- "You shall know a word by the company it keeps"
  - (J. R. Firth 1957: 11)

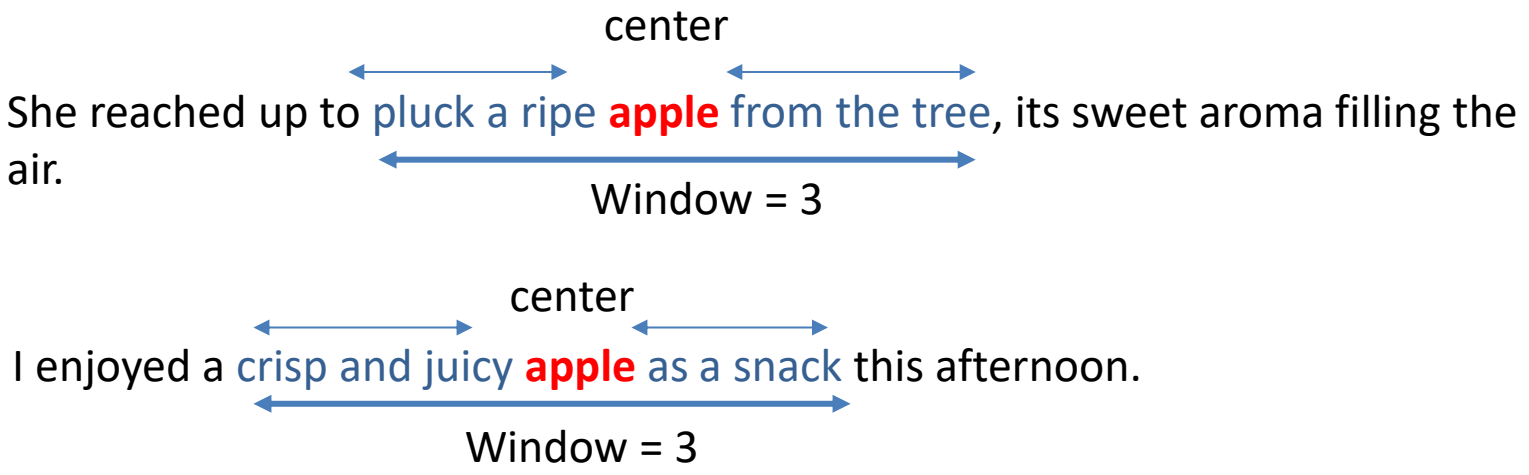


- One of the most successful ideas of modern statistical NLP

Basic Idea

A word is defined by its context

Context: words that appear in a fixed length window around the word



Use the many contexts of w to represent a word

Βασική ιδέα

Κάθε λέξη δύο διανυσματικές αναπαραστάσεις:
(1) center (2) context
Δηλαδή, έχουμε 2 |V| x d πίνακες

- Το *center-διάνυσμα* της λέξης πρέπει να είναι *όμοιο* με τα *context-διανύσματα* (δηλαδή, το άθροισμα των context διανυσμάτων) των λέξεων που εμφανίζονται ως context της
- Το *context-διάνυσμα* της λέξης πρέπει να είναι *όμοιο* με τα *center-διανύσματα* των λέξεων που εμφανίζονται ως κεντρικές της

Learning: παραδείγματα κειμένου και προσπαθούμε να «μάθουμε» αυτά τα διανύσματα (βάρη), δηλαδή να μάθουμε αυτούς τους 2 πίνακες

Word2Vec

Predict between every word and its context words

Two **algorithms**

1. Skip-grams (SG)

   Predict context words given the center word

2. Continuous Bag of Words (CBOW)

   Predict center word from a bag-of-words context

Position independent (do not account for distance from center)

Two **training methods**

1. Hierarchical softmax
2. Negative sampling

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, Jeffrey Dean: *Distributed Representations of Words and Phrases and their Compositionality.* NIPS 2013: 3111-3119

Basic Idea

She reached up to pluck a ripe **apple** from the tree, its sweet aroma filling the air.

Window = 3

pluck a ripe _____ from the tree CBOW

____ ____ ____ apple ____ ____ ____ Skipgram

Συμβολισμοί

One-hot vectors

Έστω ότι υπάρχουν |V| διαφορετικές λέξεις (όροι) στο λεξικό μας

- Διατάσσουμε τις λέξεις αλφαβητικά
- Αναπαριστούμε κάθε λέξη με ένα $\mathrm{R}^{|V|x1}$ διάνυσμα που έχει παντού 0 και μόνο έναν 1 στη θέση που αντιστοιχεί στη θέση της λέξης στη διάταξη
- Aka indicator vector

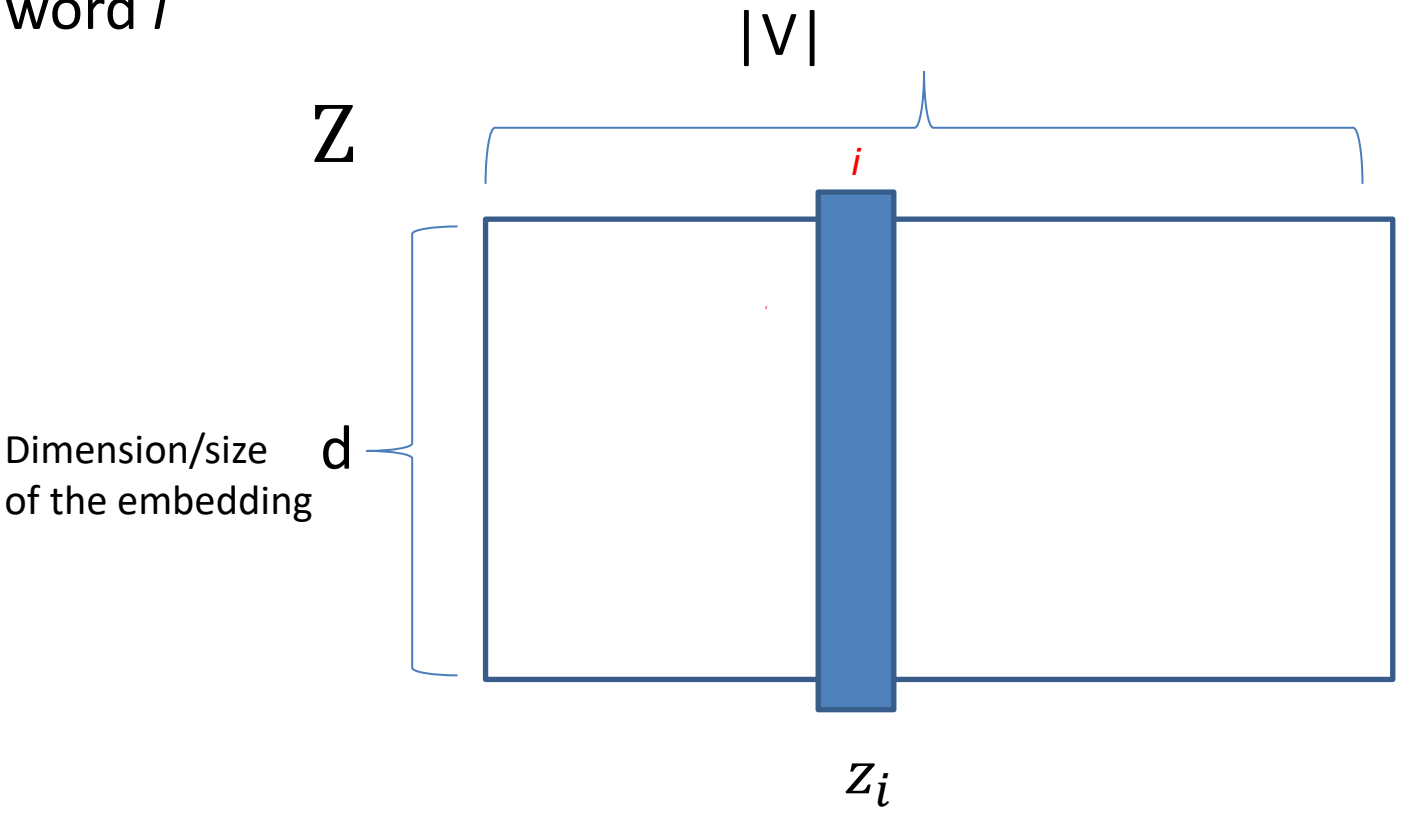
$$w^{aardvark} = \begin{bmatrix} \mathbf{1} \\ 0 \\ 0 \\ . \\ . \\ . \\ 0 \end{bmatrix} \quad w^{a} = \begin{bmatrix} 0 \\ \mathbf{1} \\ 0 \\ . \\ . \\ . \\ 0 \end{bmatrix} \quad w^{at} = \begin{bmatrix} 1 \\ 0 \\ \mathbf{0} \\ . \\ . \\ . \\ 0 \end{bmatrix} \quad \cdots \quad w^{zerba} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ . \\ . \\ . \\ \mathbf{1} \end{bmatrix}$$

|V| number of words
d size of embedding
m size of the window (context)

Look up

Each word is assigned *a d-dimensional vector*

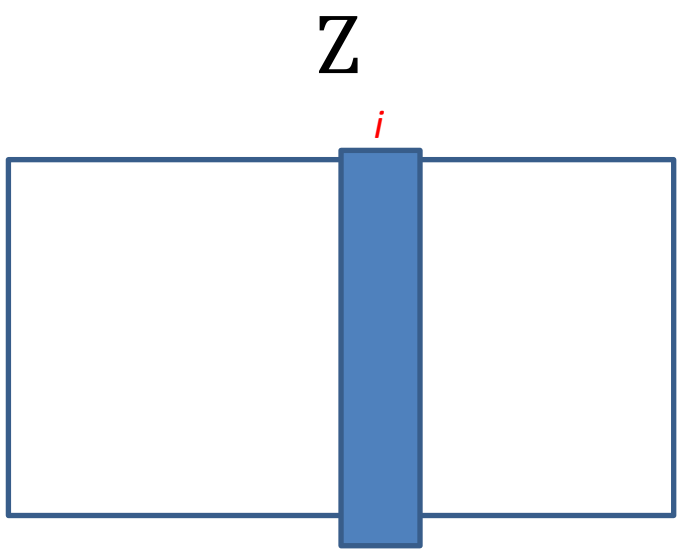
Learn embedding matrix Z: each column *i* is the embedding z_i of word *i*



Look up

Encoder is an embedding lookup

$$ENC(i) = Z\ I_i$$



z_i

Embedding matrix

d x|V|

One hot vector I_i

$$\begin{bmatrix} 0 \\ . \\ 1 \\ . \\ . \\ . \\ 0 \end{bmatrix}$$

One-hot or indicator vector, all 0s but position *i*

|V| x 1

=

$$\begin{bmatrix} w_1 \\ . \\ . \\ . \\ w_d \end{bmatrix}$$

Embedding of the *i*-th word

d x 1

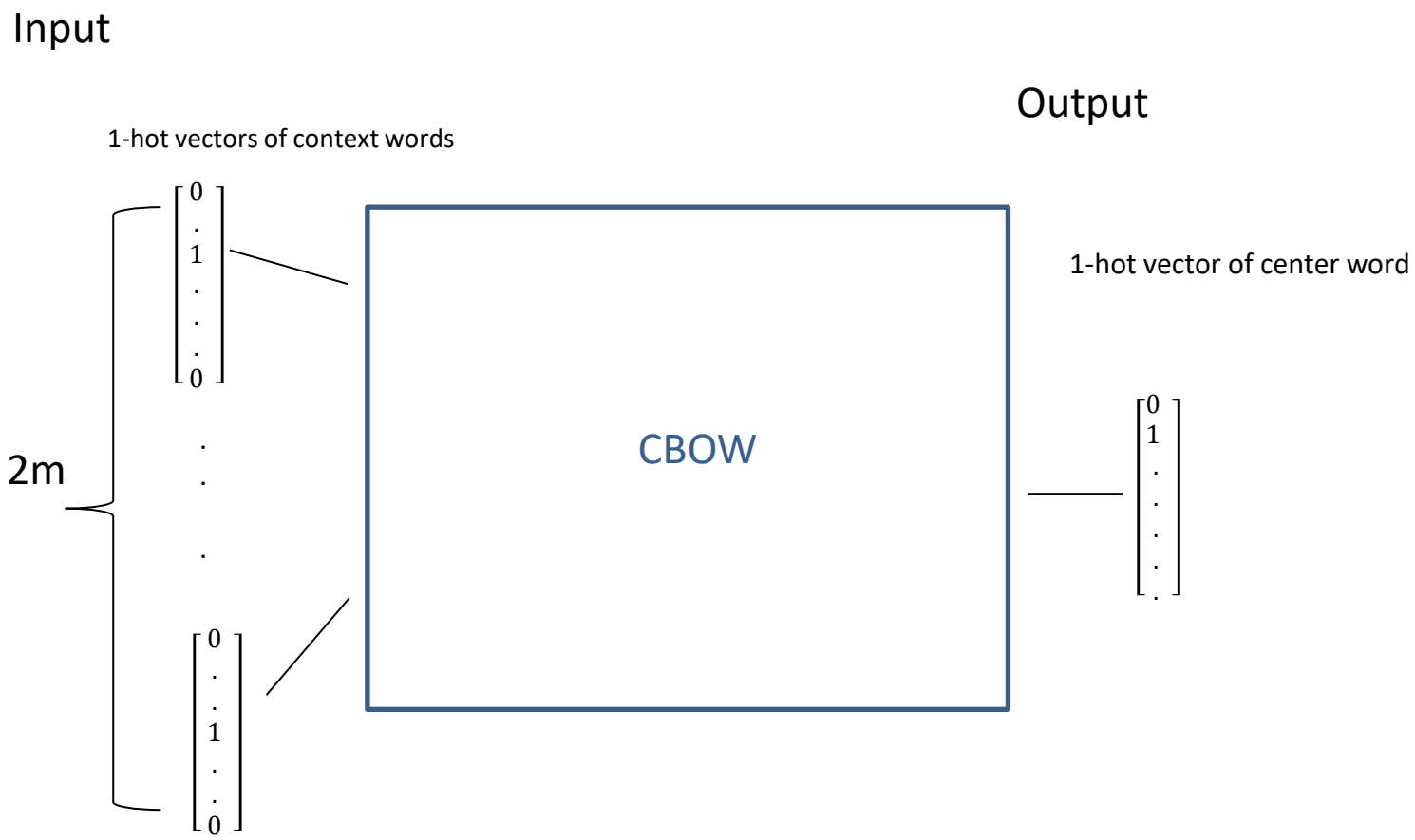
CBOW

CBOW

$|V|$ number of words
d size of embedding
m size of the window (context)

Use a window of context words to predict the center word

Input: $2m$ context words
Output: center word
each represented as a one-hot vector



Εκπαίδευση (learning)

- Δίνουμε *δεδομένα εκπαίδευσης*: (κείμενο) δηλαδή συμφραζόμενες λέξεις για τις οποίες ξέρουμε την κεντρική
- και μαθαίνουμε αυτά τα διανύσματα (βάρη) (δηλαδή, τους 2 πίνακες) ώστε να *ελαχιστοποιούν το «λάθος»* στα δεδομένα εκπαίδευσης, δηλαδή να «βρίσκουν» την κεντρική λέξη

CBOW

Learns two matrices (two embeddings per word, one when context, one when center)

word

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ . \\ . \\ . \\ w_d \end{bmatrix}$$

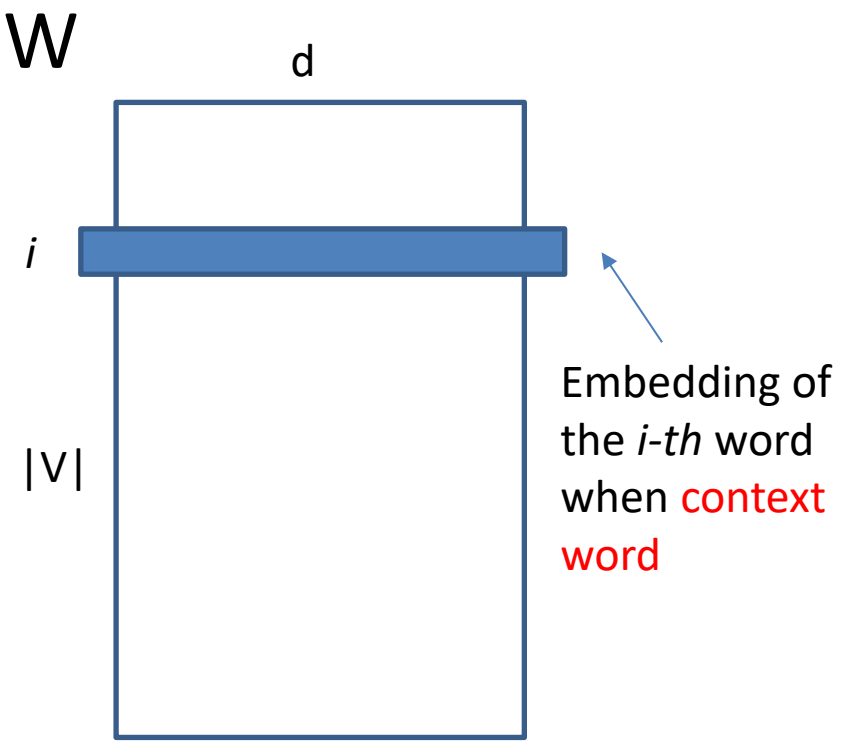
Center

$$\begin{bmatrix} w'_1 \\ w'_2 \\ w'_3 \\ . \\ . \\ . \\ w'_d \end{bmatrix}$$

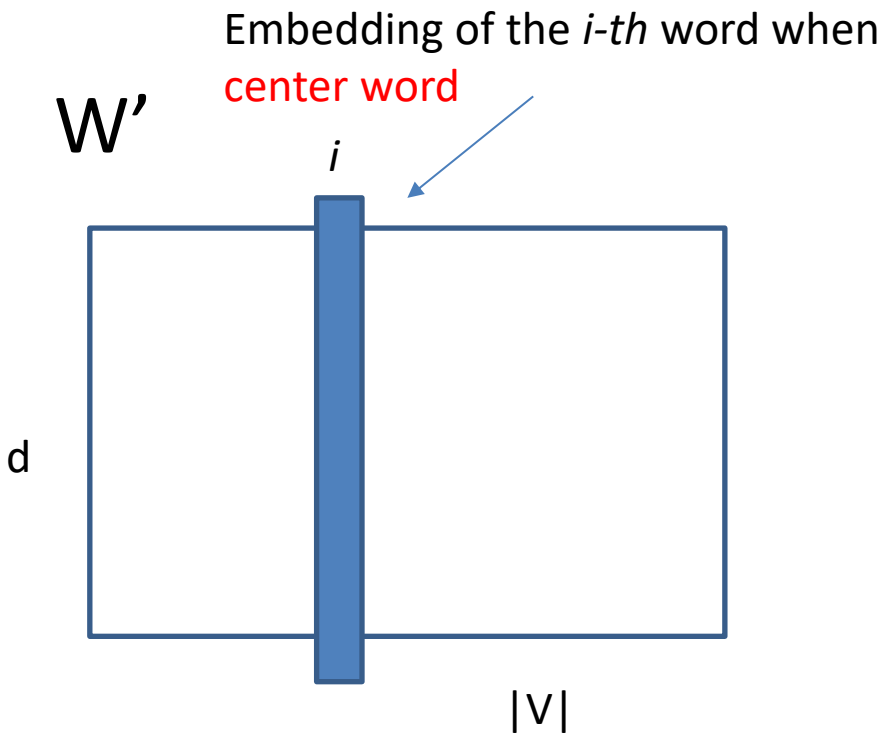
Context

CBOW

Learns two matrices (two embeddings per word, one when context, one when center)



|V| x d context embeddings when *input*



d x |V| center embeddings when *output*

CBOW

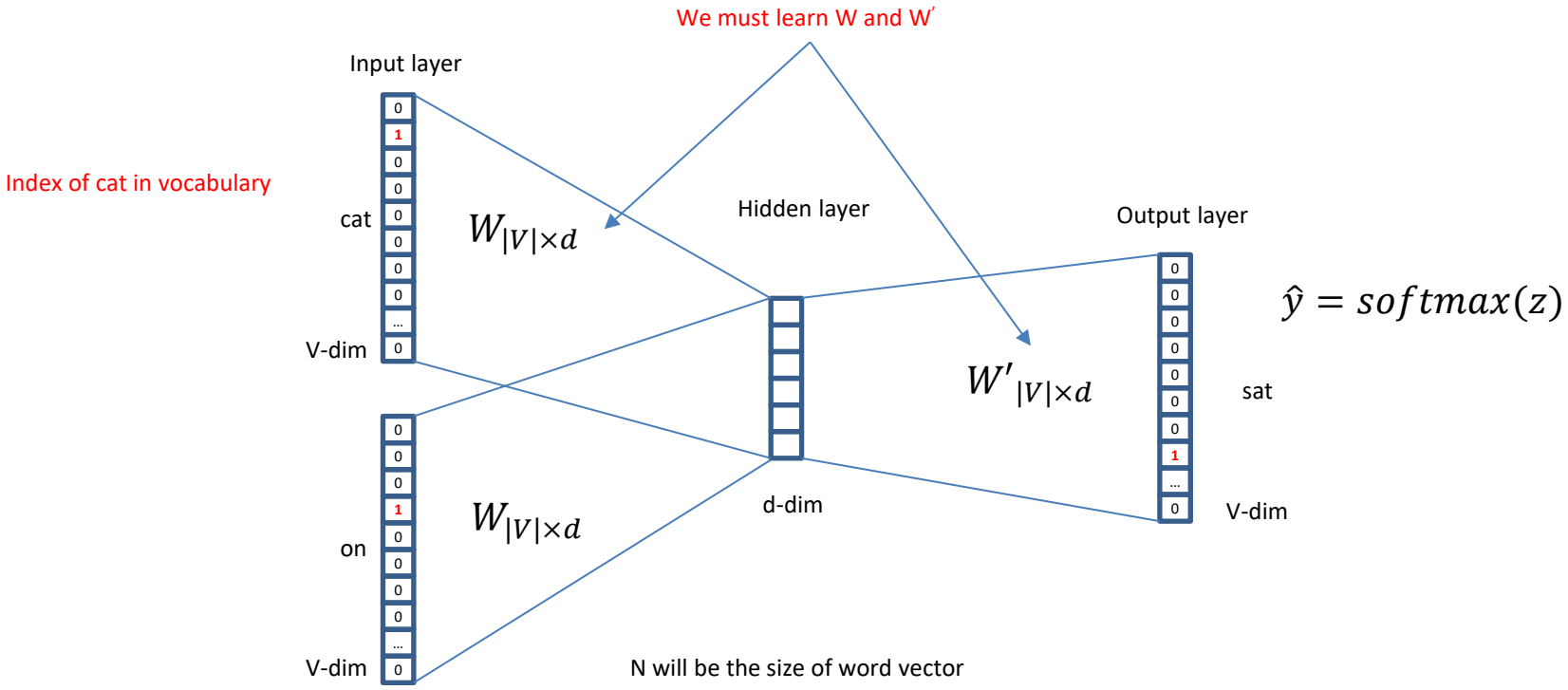
Intuition

The W'-embedding of the *center word* should be *similar* to the (average of) the W-embeddings of its *context words*

- For similarity, we will use cosine (dot product)
- We will take the average of the W-embeddings of the context word

We want similarity close to one for the center word and close to 0 for all other words

cat sat on window size = 1



CBOW

Given window size *m*

$x^{(c)}$ one hot vector for context words, *y* one hot vector for the center word

1. **INPUT:** the *one hot vectors* for the *2m* context words

$x^{(c-m)}, \ldots, x^{(c-1)}, x^{(c+1)}, \ldots, x^{(c+m)}$

2. *GET THE EMBEDDINGS of the context words*

$v_{c-m} = Wx^{(c-m)}, \ldots, v_{c-1} = Wx^{(c-1)}, v_{c+1} = Wx^{(c+1)}, \ldots, v_{c+m} = Wx^{(c+m)}$

3. *TAKE THE SUM* these vectors (average)

$\hat{v} = \frac{v_{c-m} + v_{c-m+1} + \cdots v_{c+m}}{2m}$, $\hat{v} \in R^d$

4. *COMPUTE SIMILARITY: dot product W' (all center vectors) and context $\hat{v}$ (generate score vector z)*

z = W' $\hat{v}$

5. Turn the *score vector to probabilities*

$\hat{y}$ = *softmax(z)*

We want this to be close to 1 for the center word

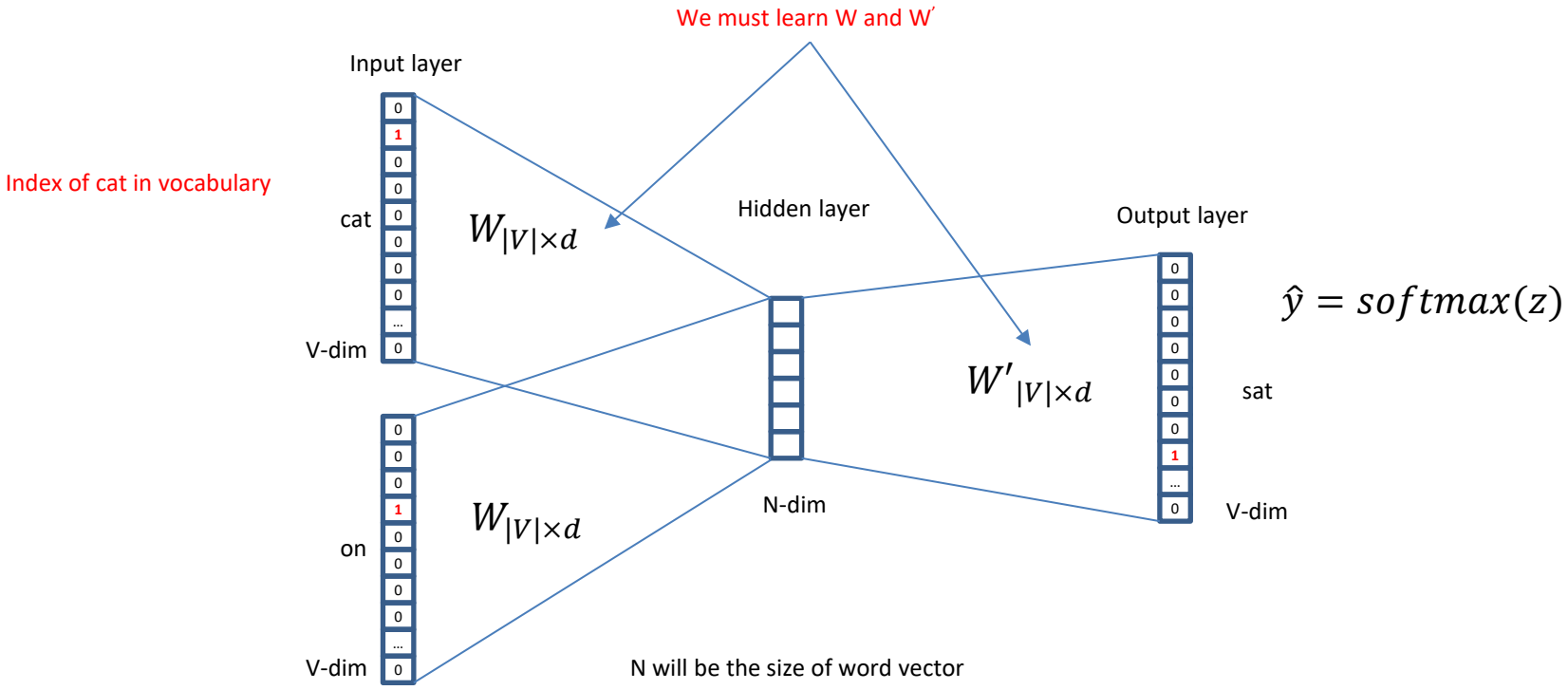
Απεικόνιση κατανομών οποιοδήποτε τιμών σε κατανομές πιθανοτήτων

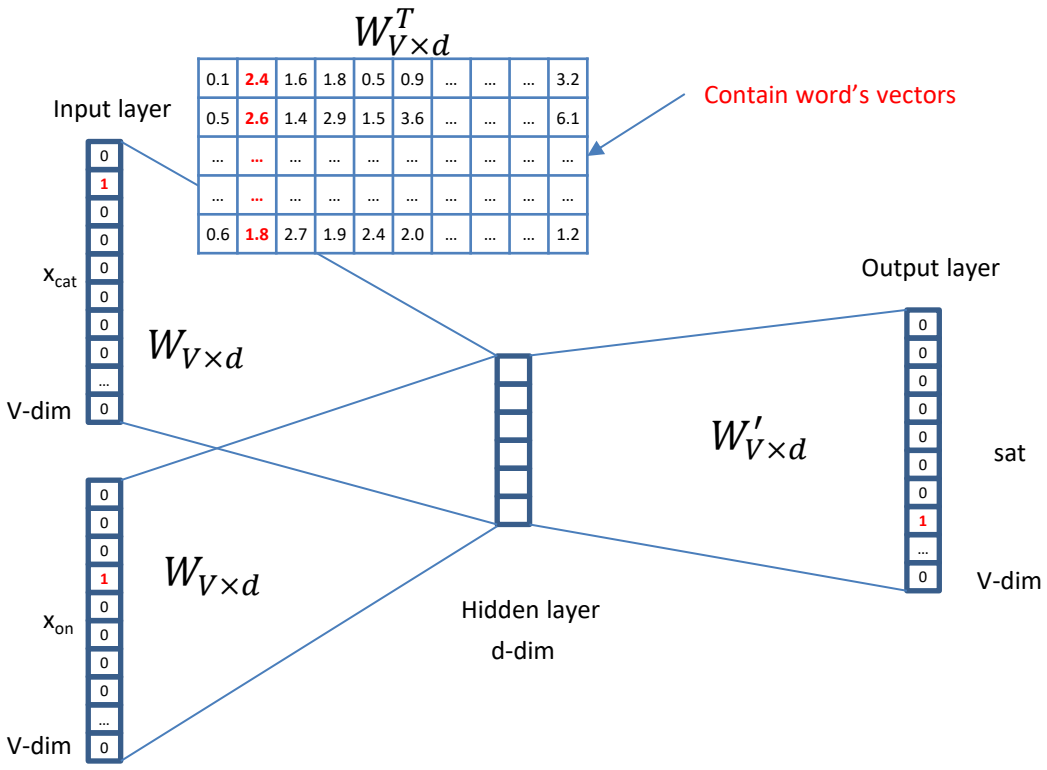
$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

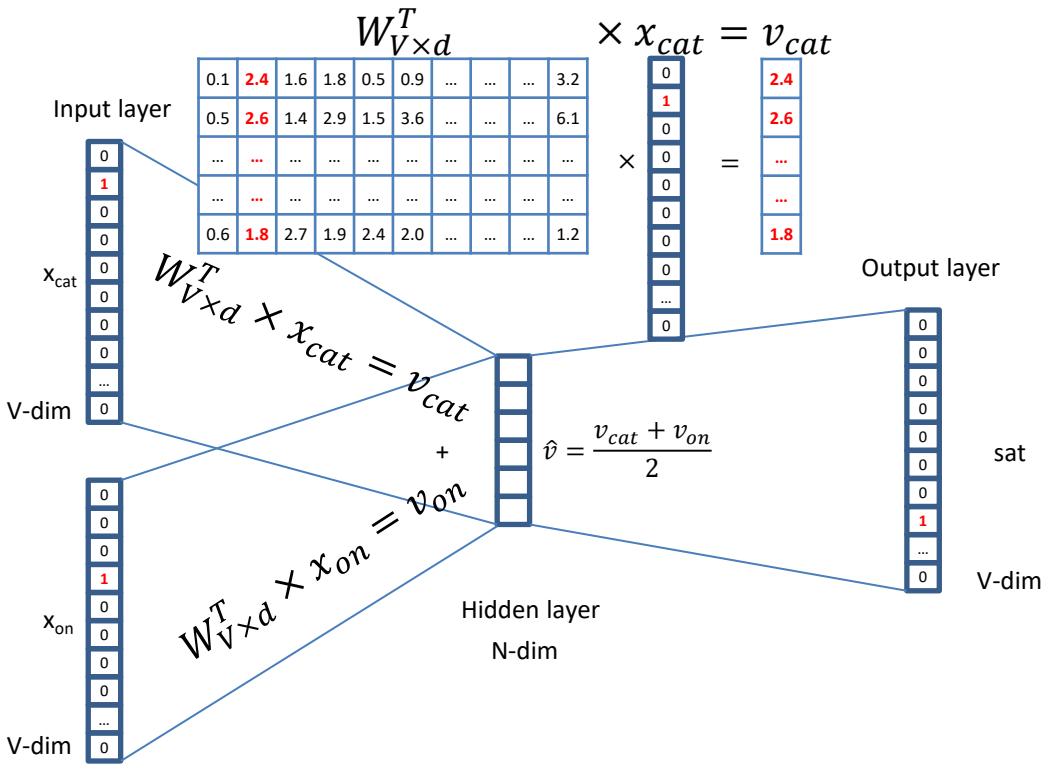
- Εκθετικό για να είναι θετικό
- Ενισχύει την πιθανότητα της μεγαλύτερης τιμής (max)
- Κάποια πιθανότητα και στις μικρότερες τιμές (soft)

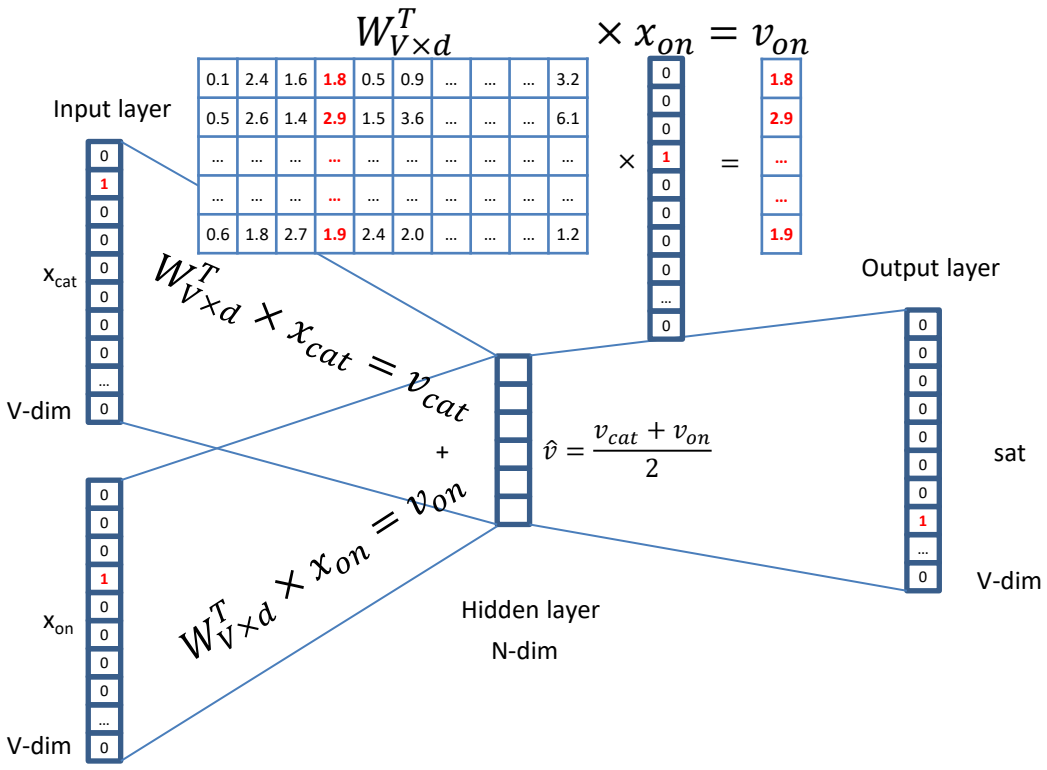
```
>>> import numpy as np
>>> a = [1.0, 2.0, 3.0, 4.0, 1.0, 2.0, 3.0]
>>> np.exp(a) / np.sum(np.exp(a))
array([0.02364054, 0.06426166, 0.1746813, 0.474833,
0.02364054, 0.06426166, 0.1746813])
```

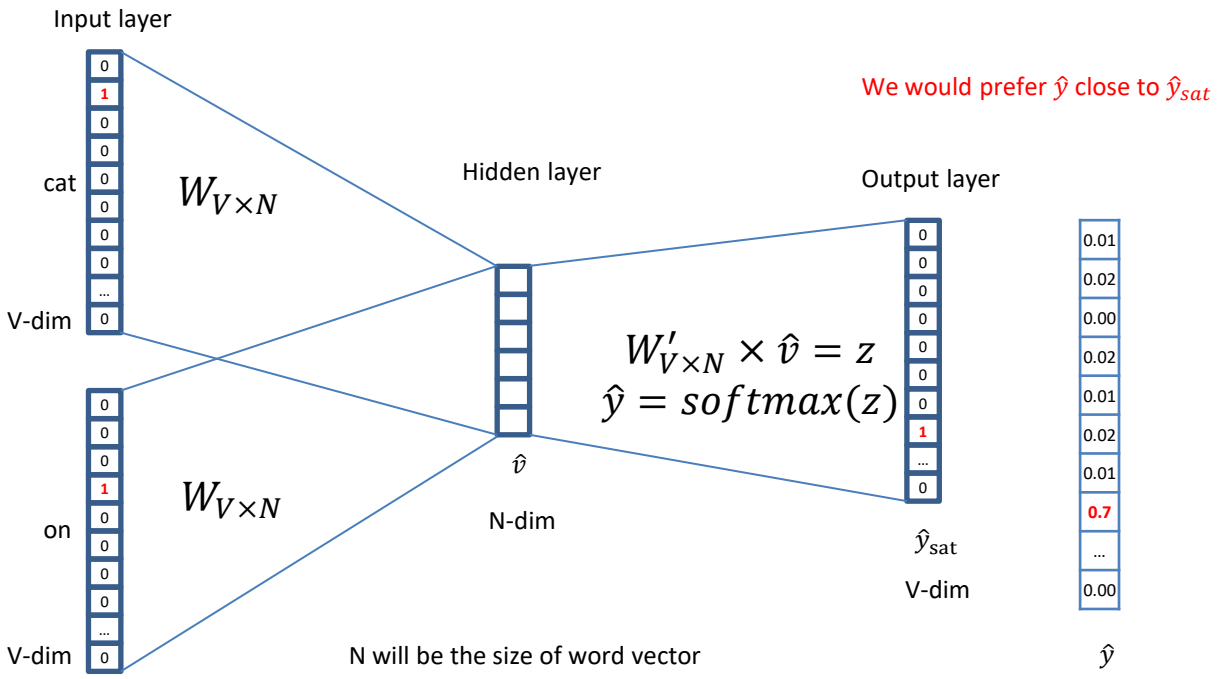
cat sat on window size = 1




$W^T_{V\times d}$
Contain word's vectors
Input layer
x_{cat}
V-dim
$W_{V\times d}$
x_{on}
V-dim
$W_{V\times d}$
Hidden layer
d-dim
Output layer
$W'_{V\times d}$
sat
V-dim


$W^T_{V\times d}$
$\times x_{cat} = v_{cat}$
Input layer
x_{cat}
V-dim
$W^T_{V\times d} \times x_{cat} = v_{cat}$
+
$W^T_{V\times d} \times x_{on} = v_{on}$
x_{on}
V-dim
$\hat{v} = \frac{v_{cat} + v_{on}}{2}$
Hidden layer
N-dim
Output layer
sat
V-dim


$W^T_{V\times d}$
$\times x_{on} = v_{on}$
Input layer
x_{cat}
V-dim
$W^T_{V\times d} \times x_{cat} = v_{cat}$
+
$W^T_{V\times d} \times x_{on} = v_{on}$
x_{on}
V-dim
$\hat{v} = \frac{v_{cat} + v_{on}}{2}$
Hidden layer
N-dim
Output layer
sat
V-dim


Input layer
cat
V-dim
on
V-dim
$W_{V\times N}$
$W_{V\times N}$
Hidden layer
$\hat{v}$
N-dim
$W'_{V\times N} \times \hat{v} = z$
$\hat{y} = softmax(z)$
Output layer
$\hat{y}_{\text{sat}}$
V-dim
We would prefer $\hat{y}$ close to $\hat{y}_{sat}$
$\hat{y}$
N will be the size of word vector

- We can consider either W (context) or W’ (center) as the word’s representation.
  - Or even take the average.

- Use the model to predict a missing word

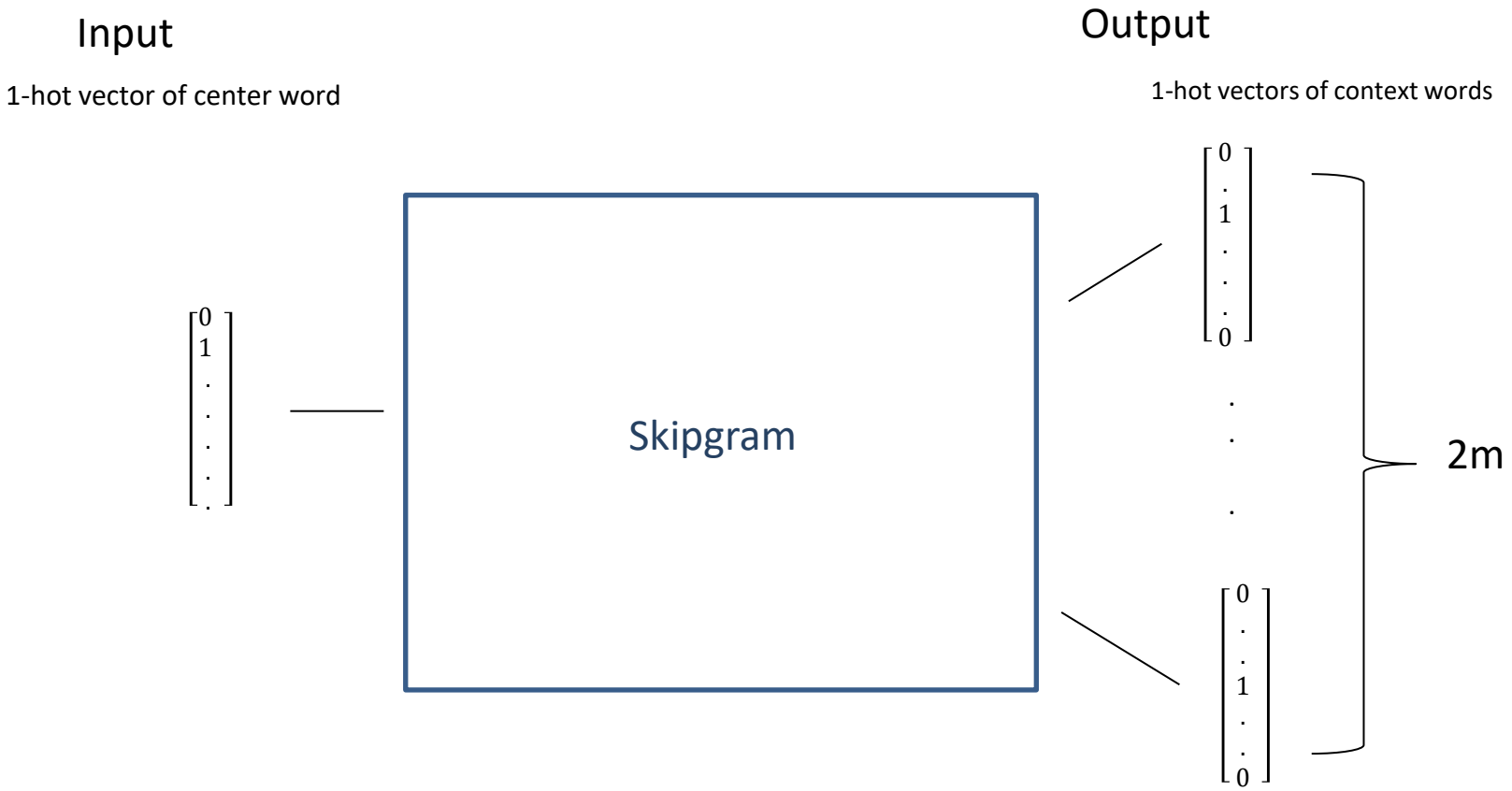
SKIPGRAM

Skipgram

Given the center word, predict (or, generate) the context words

Input: center word
Output: 2m context word
each represented as a one-hot vector

Learn two matrices
W: d x |V|, input matrix, word representation as center word
W': |V| x d, output matrix, word representation as context word



Εκπαίδευση (learning)

Δίνουμε δεδομένα εκπαίδευσης (κείμενο, δηλαδή κεντρικές λέξεις για τις οποίες ξέρουμε τις συμφραζόμενες λέξεις)

Μαθαίνουμε αυτά τα διανύσματα (βάρη) (δηλαδή, τους 2 πίνακες) ώστε να ελαχιστοποιούν το «λάθος» στα δεδομένα εκπαίδευσης, δηλαδή να «βρίσκουν» τις συμφραζόμενες λέξεις

Skipgram

Given the center word, predict (or, generate) the context words

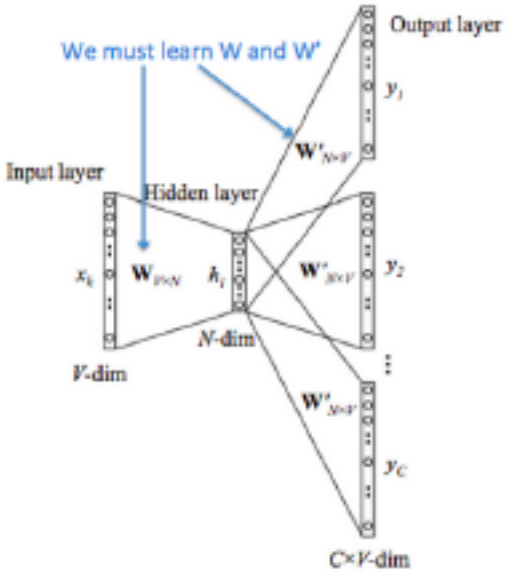
$y^{(j)}$ one hot vector for context words

1. Input: *one hot vector* of the center word
x

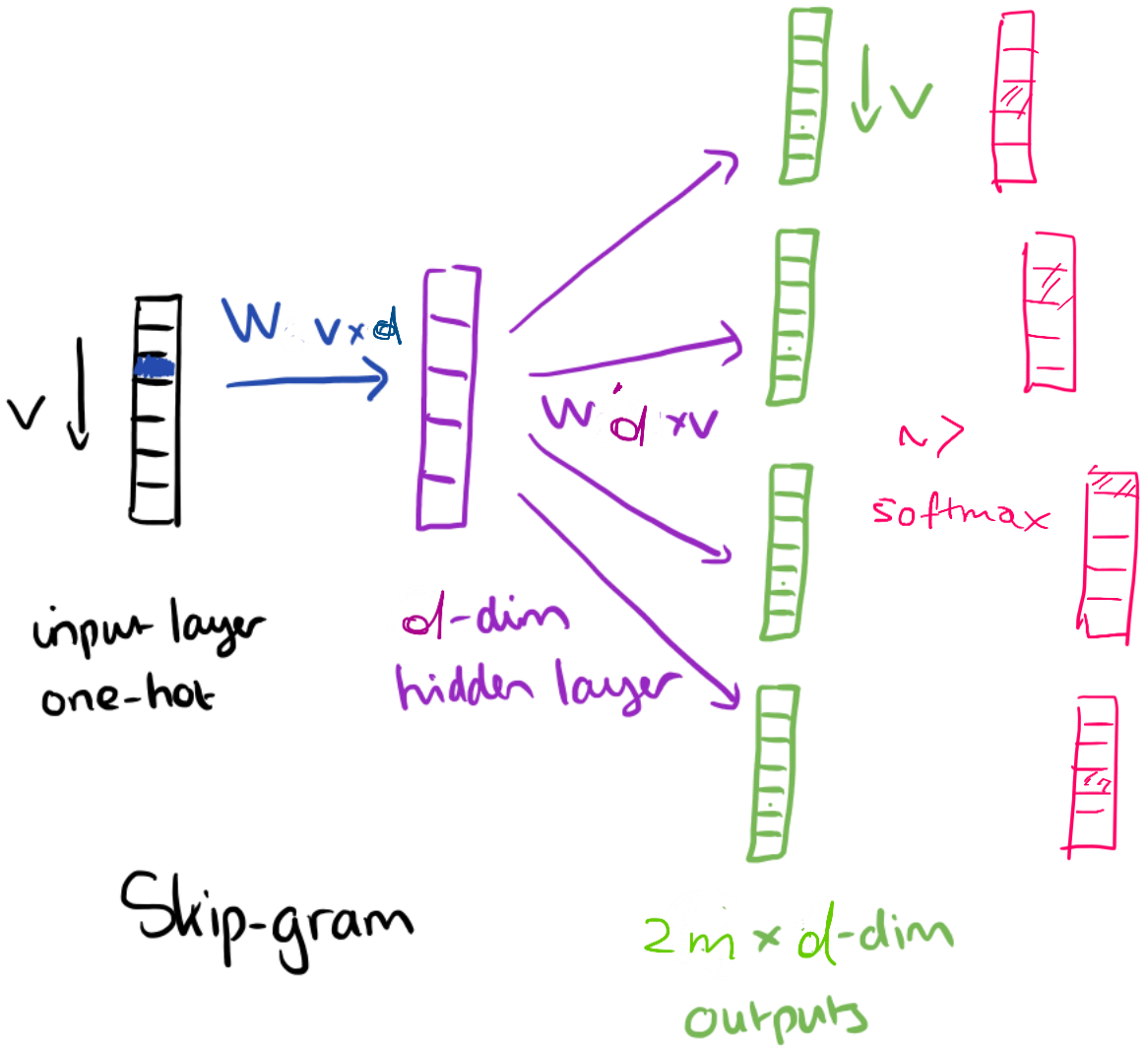
2. Get the *embedding of the center word*
$v_c = W\ x$

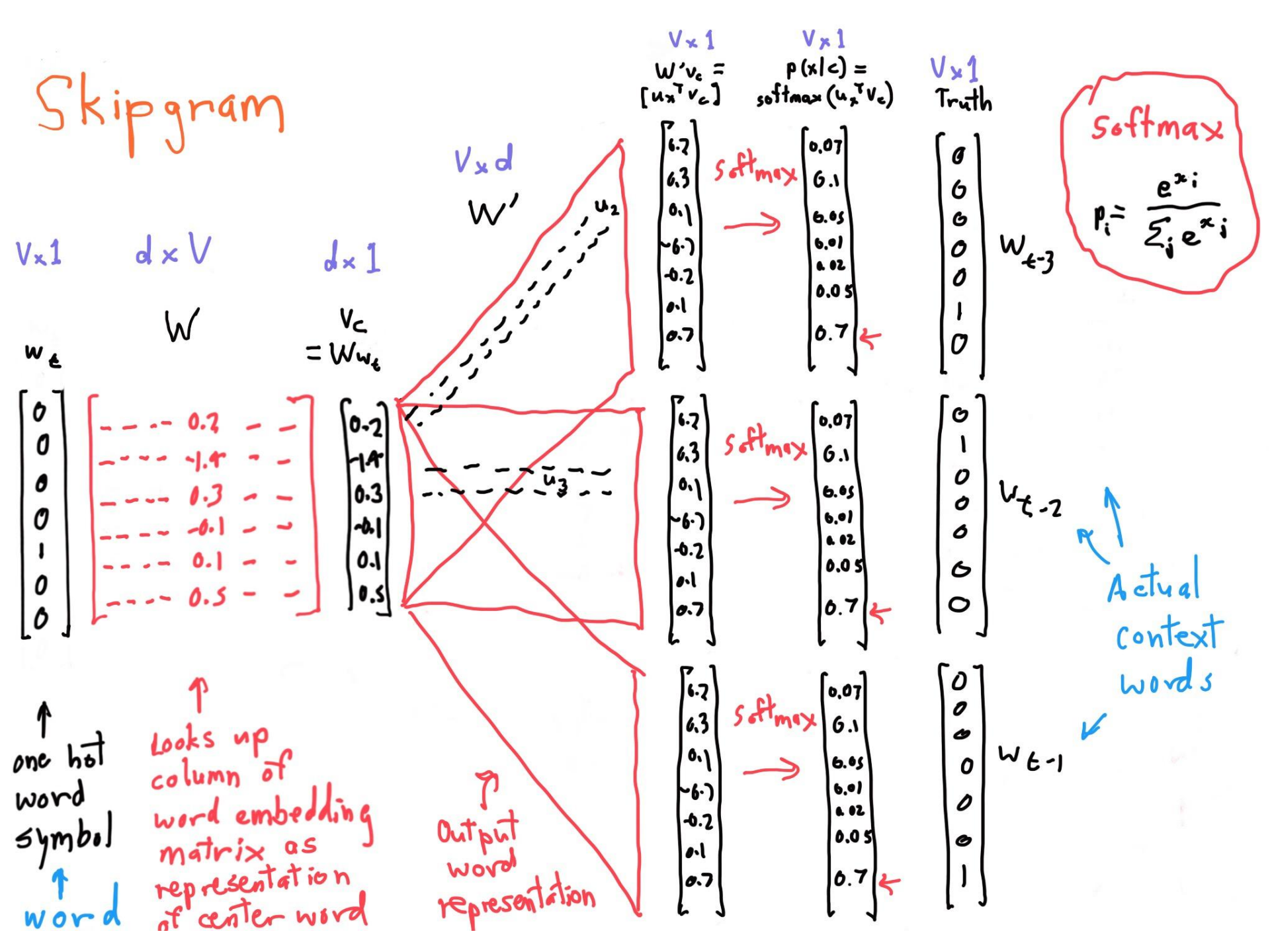
3. Generate a *score vector for each context word*
z = *W'* v_c

5. Turn the *score vector into probabilities*
$\hat{y}$ = *softmax(z)*



We want this to be close to 1 for the context words


V
W V×d
W' d×V
V
softmax
input layer
one-hot
d-dim
hidden layer
Skip-gram
2m × d-dim
outputs


Skipgram
V×1
d×V
d×1
V×d
W'
W
w_t
$v_c = W w_t$
u_2
u_3
V×1
$W' v_c = [u_x^T v_c]$
V×1
$p(x|c) = \text{softmax}(u_x^T v_c)$
V×1
Truth
Softmax
$p_i = \frac{e^{x_i}}{\sum_i e^{x_i}}$
Softmax
W_{t-3}
W_{t-2}
W_{t-1}
Actual context words
one hot word symbol
word
Looks up column of word embedding matrix as representation of center word
Output word representation

Αποτελέσματα

These representations are *very good* at encoding similarity and dimensions of similarity!

- Analogies

A is to B what C is to D

Analogies testing dimensions of similarity can be solved quite well just by doing vector subtraction in the embedding space

Syntactically

- $x_{apple} - x_{apples} \approx x_{car} - x_{cars} \approx x_{family} - x_{families}$
- Similarly for verb and adjective morphological forms

Semantically

- $x_{shirt} - x_{clothing} \approx x_{chair} - x_{furniture}$
- $x_{king} - x_{man} \approx x_{queen} - x_{woman}$

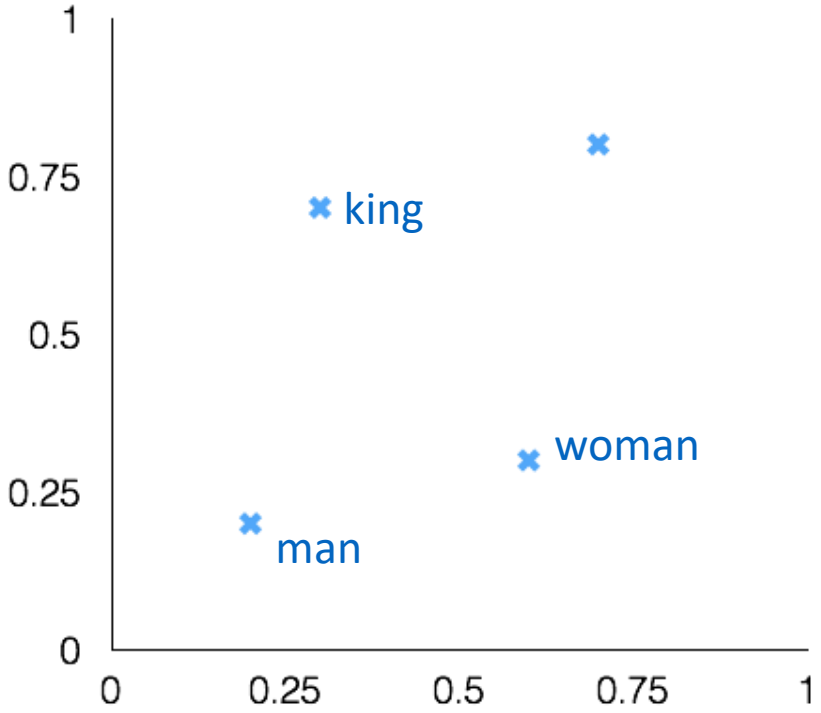
Test for linear relationships, examined by Mikolov et al.

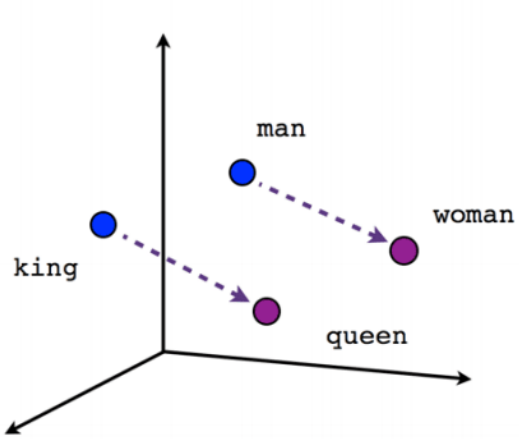
a:b :: c:?

$$d = \arg\max_x \frac{(w_b - w_a + w_c)^T w_x}{||w_b - w_a + w_c||}$$

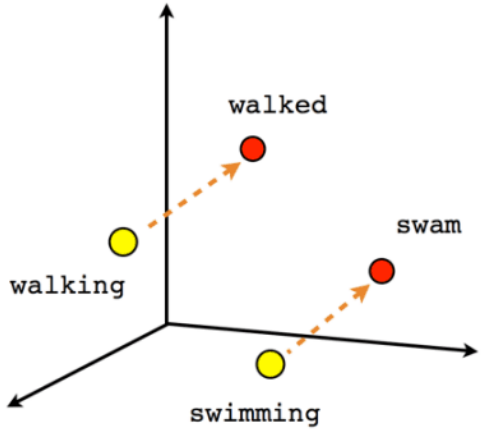
man:woman :: king:?

+	king	[0.30 0.70]
−	man	[0.20 0.20]
+	woman	[0.60 0.30]
	queen	[0.70 0.80]

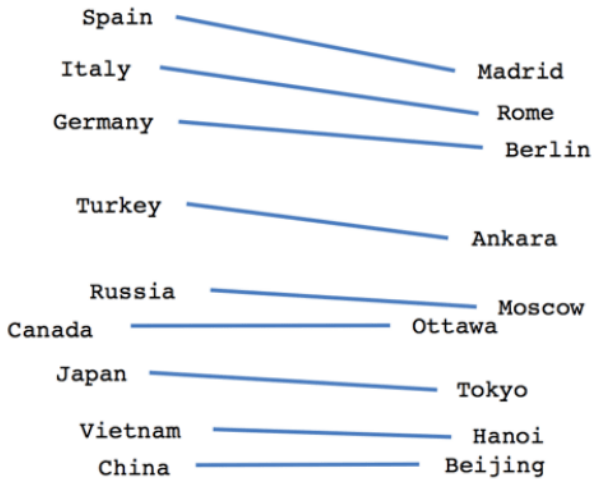




Male-Female



Verb tense



Country-Capital

Κάποιες εφαρμογές

Global vs. local embedding

global	local
cutting	tax
squeeze	deficit
reduce	vote
slash	budget
reduction	reduction
spend	house
lower	bill
halve	plan
soften	spend
freeze	billion

Πάνω *σε ποια συλλογή* (corpus) φτιάχνουμε τα embeddings;
Προτάσεις από ποια κείμενα θα χρησιμοποιήσουμε;

[Diaz 2016] Terms similar to 'cut' for a word2vec model trained on a general news corpus and another trained only on documents related to 'gasoline tax'.

1. Train and create embeddings based on a local collection

Python implementation in gensim

https://radimrehurek.com/gensim/models/word2vec.html

Tensorflow

https://www.tensorflow.org/tutorials/text/word_embeddings

2. Use pretrained embeddings

Pretrained embeddings for 157 languages

https://fasttext.cc/docs/en/crawl-vectors.html

Google

https://code.google.com/archive/p/word2vec/

Hugging Face

Finding the degree of similarity between two words.

```
model.similarity('woman','man')
0.73723527
```

Finding odd one out.

```
model.doesnt_match('breakfast cereal dinner lunch';.split())
'cereal'
```

Amazing things like woman+king-man =queen

```
model.most_similar(positive=['woman','king'],negative=['man'],top
n=1)
queen: 0.508
```

Probability of a text under the model

```
model.score(['The fox jumped over the lazy dog'.split()])
0.21
```

ανεκτική ανάκτηση: (1) επέκταση ερωτήματος ή/και (2) context-dependent διόρθωση λάθους, όπου θα μπορούσαμε να χρησιμοποιήσουμε και το query log και γενικά query suggestions

Improve language translation

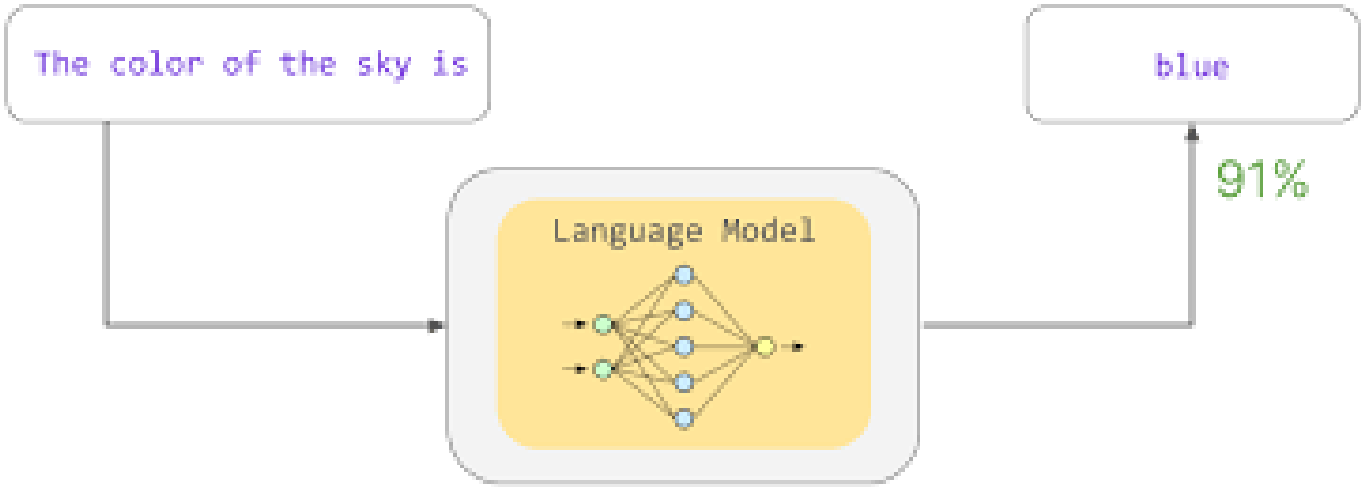
bilingual embedding with chinese in green and english in yellow

By aligning the word embeddings for the two languages

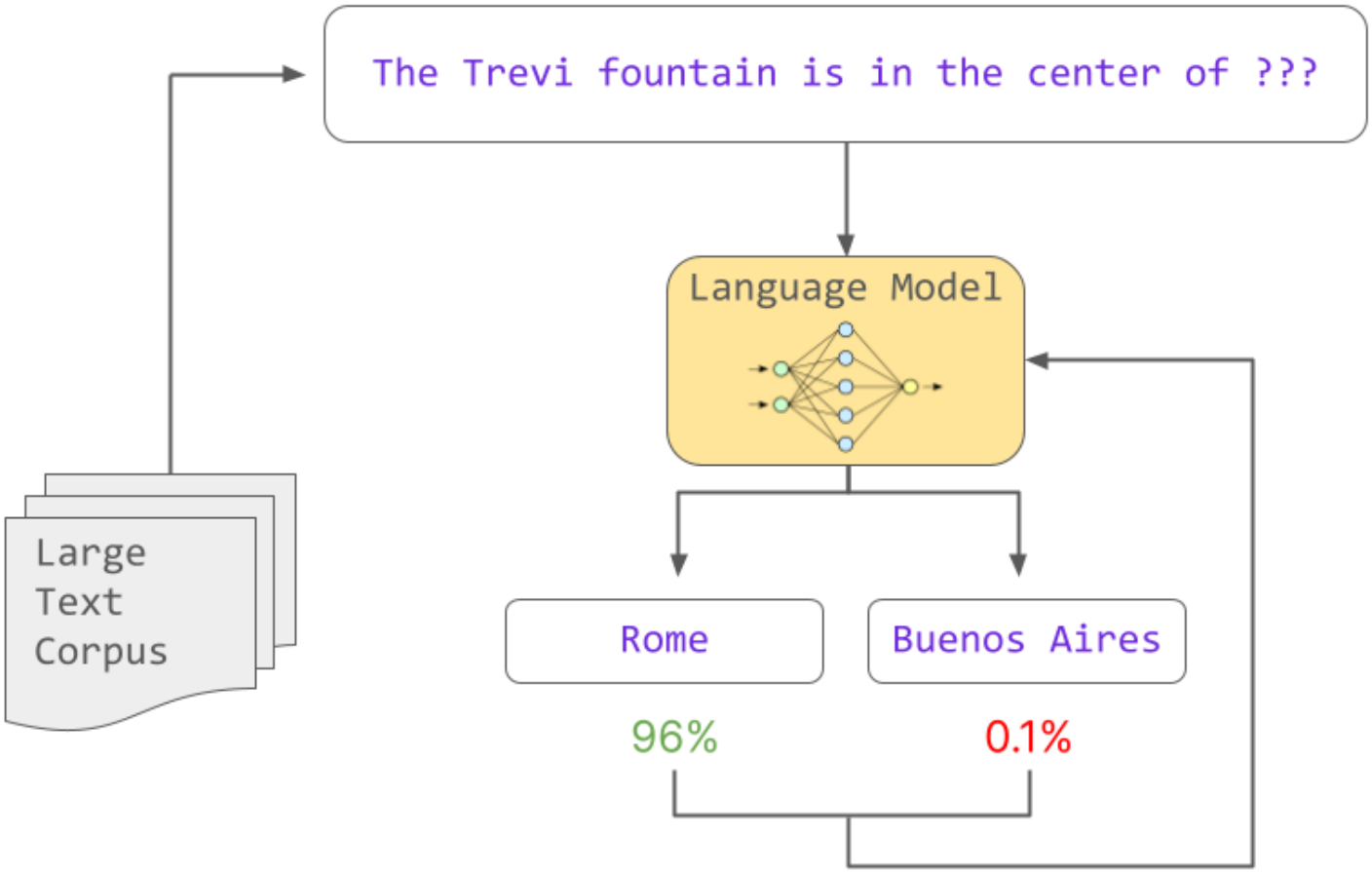
Διαβάθμιση βασισμένη σε tf-idf
Άσκηση

Embeddings

Retrieval Augmented Generation



Given context (prompt), predict the next word (the most probable one)


The Trevi fountain is in the center of ???
Language Model
Large
Text
Corpus
Rome
Buenos Aires
96%
0.1%

Transformers

- A class of *deep learning* models
- An instance of the *decoder-encoder* model
- First introduced in the "Attention is All you Need" paper by Google researchers in 2017
- Original designed for *language translation*, particularly from English to German. But, generalized to other language tasks

Attention Mechanisms
allow the model to weigh the importance of different words or phrases in the text.

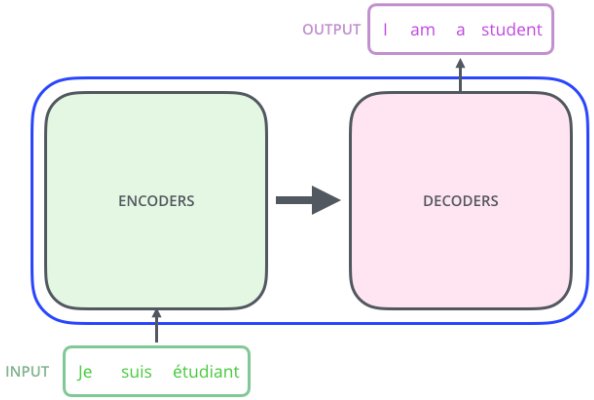
Encoder/decoder architecture

- Encoder: maps an *input sequence* to a sequence of continuous *representations*, which is then fed into a decoder.
- Decoder: receives the *output of the encoder* together with the *decoder output at the previous time step*, to generate an *output sequence*

Jointly trained to minimize the conditional log-likelihood.

Once trained, the encoder/decoder can

- generate an output given an input sequence, or
- can score a pair of input/output sequences.



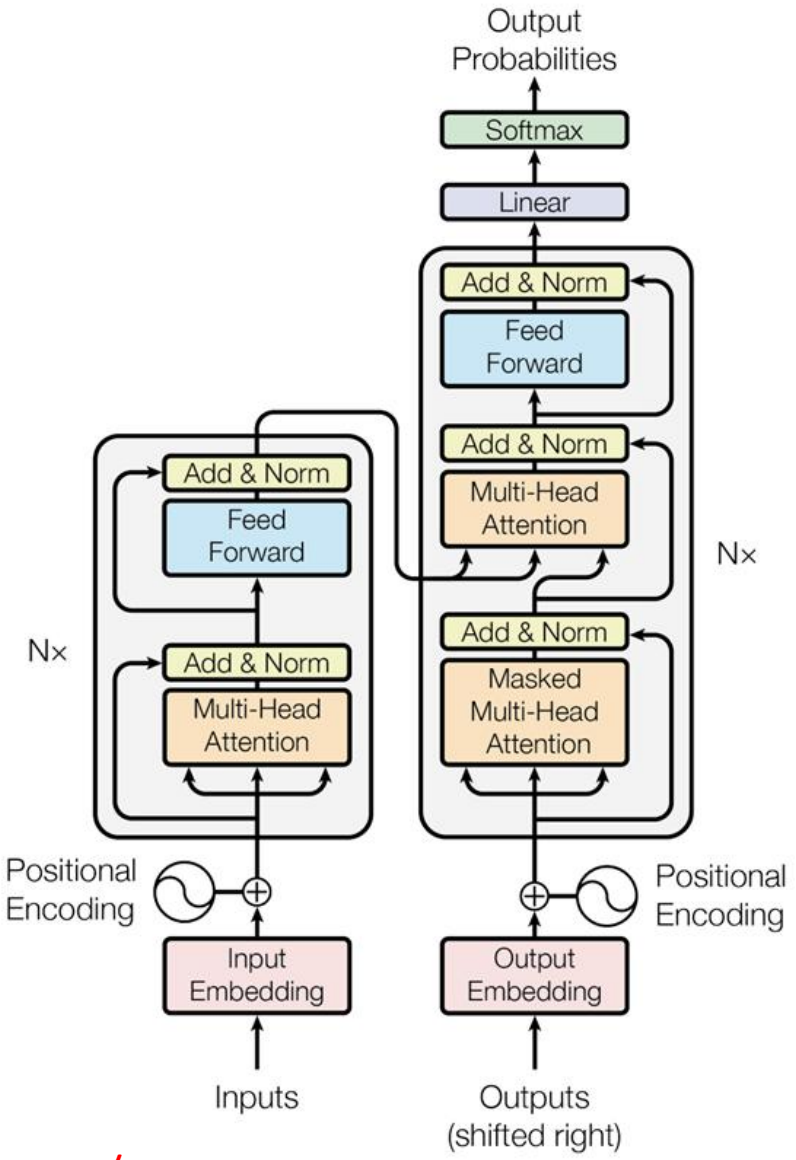
Encoder/decoder architecture

Initially, N = 6 layers
the Encoder has two sub layers:

- a multi-head attention layer, and
- a simple feed forward network.

The Decoder adds a third sublayer: another multi-head attention layer over the output of the Encoder.

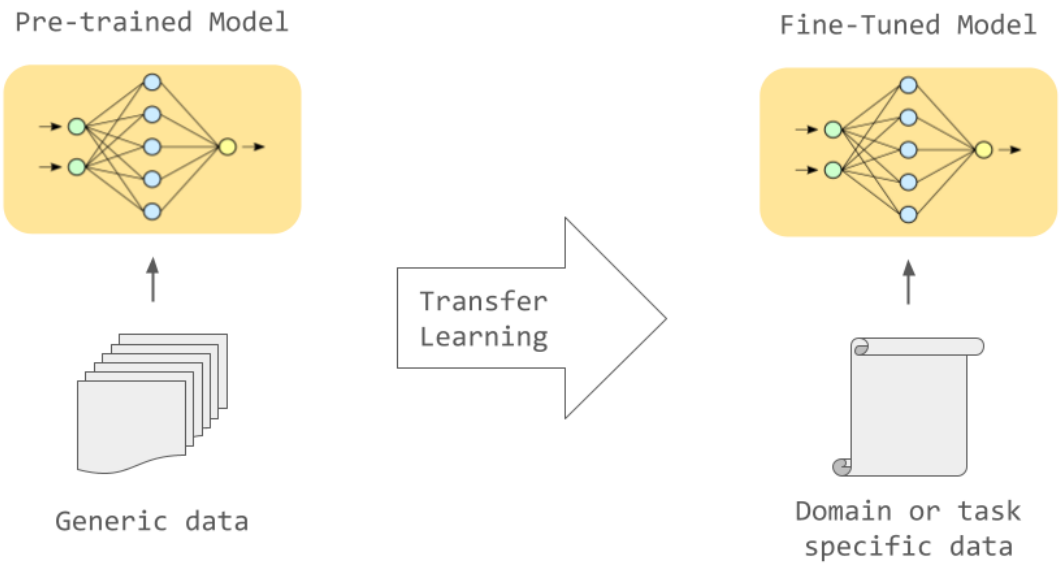
- masked to prevent attention to subsequent positions.
- positional encodings



https://jalammar.github.io/illustrated-transformer/

Transfer learning

- Pretrained Transformer models can adapt to tasks they have not been trained on
- *Re-use* the pretrained model on a new task, by adapting it with a much smaller data set (fine tuning)



Additional Links

- Integrated into the main AI frameworks (e.g., Pytorch, Tensorflow)
- Huggingface, a startup commercializing an open-source Transformers library.
- Llama

What is generative AI and how does it work? The Turing Lectures with Mirella Lapata

https://www.youtube.com/watch?app=desktop&v=_6R7Ym6Vy_I

Pretrained transformer for Greek language

https://huggingface.co/nlpaueb/bert-base-greek-uncased-v1

Embeddings for Greek Language

http://archive.aueb.gr:7000/resources/