ieeta

Departamento de Eletrónica, Telecomunicações e Informática

Complements of Machine Learning

Lecture 8: Language Models

Petia Georgieva
(petia@ua.pt)

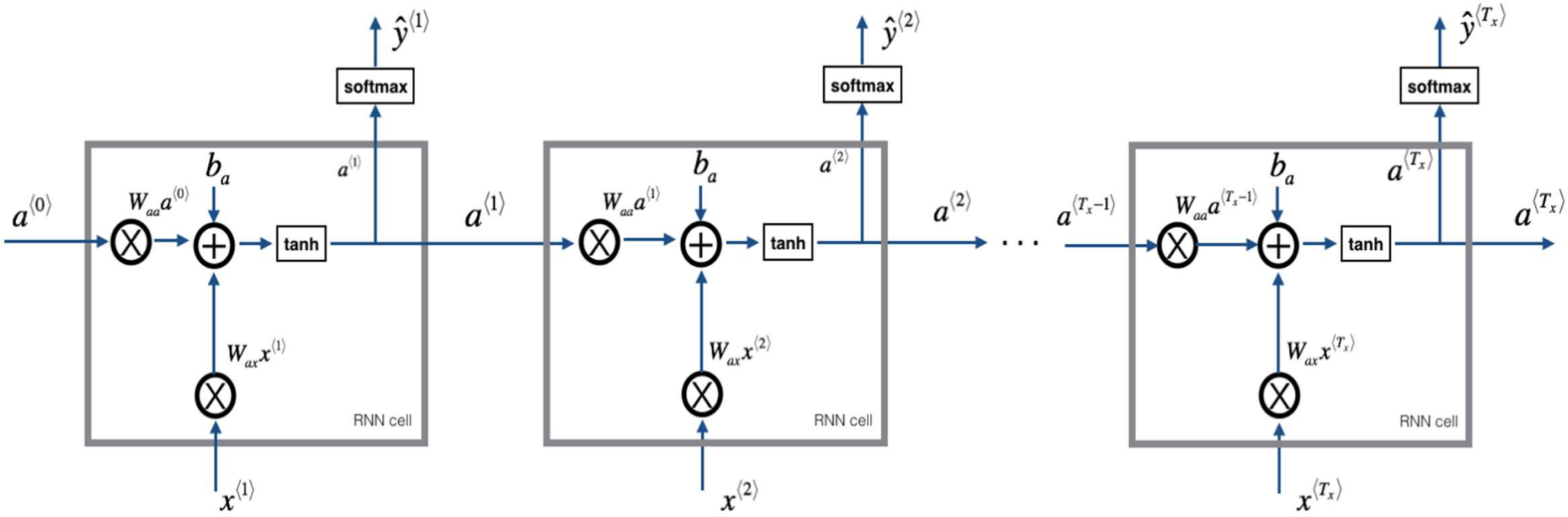
universidade de aveiro

Outline

1. **RNN-based Language Model**
2. **Natural Language Processing (NLP)**
3. **Word Embedding - Cosine Similarity**
4. **Neural Language Model**
5. **Word2Vec - skip grams model**
6. **Negative Sampling, GloVe Word Vectors**
7. **Debiasing Word Embeddings**

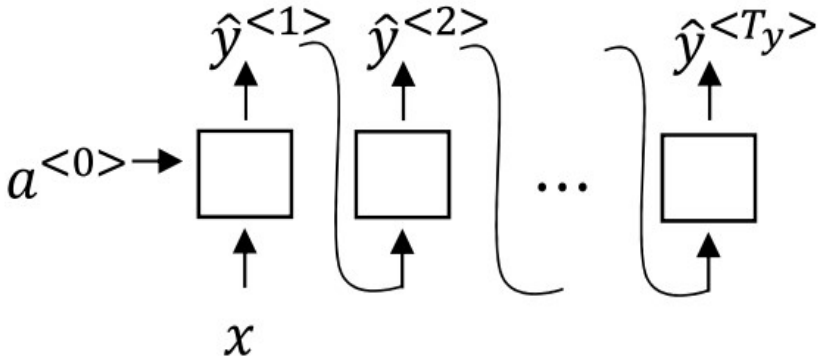
Recurrent Neural Network (RNN)

RNN is a sequence of basic RNN cells.
If input sequence is x=[$x^{<1>}$, $x^{<2>}$,...$x^{<Tx>}$] => RNN cell is copied T_x times.
RNN outputs y=[$y^{<1>}$, $y^{<2>}$,...$y^{<Tx>}$]

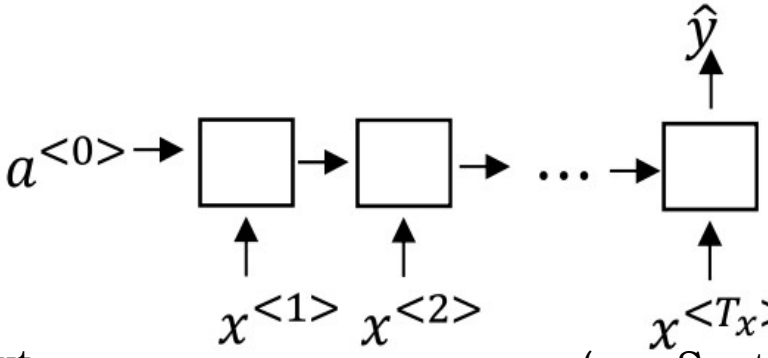


Different Types of RNN

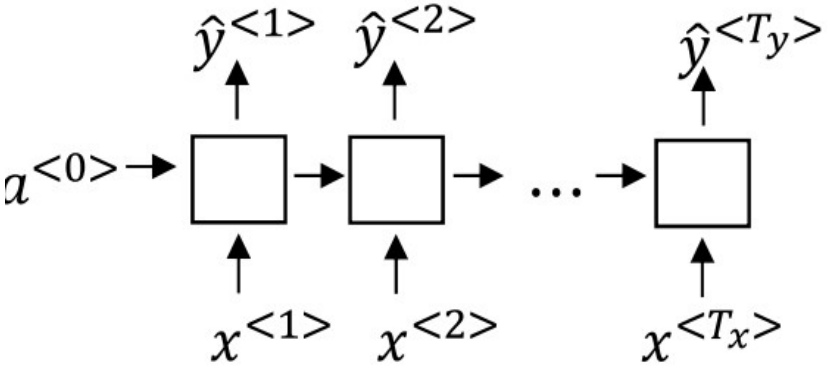
Input X and output Y can be of different types and different lengths.



One to many (e.g. Music/Text generation)

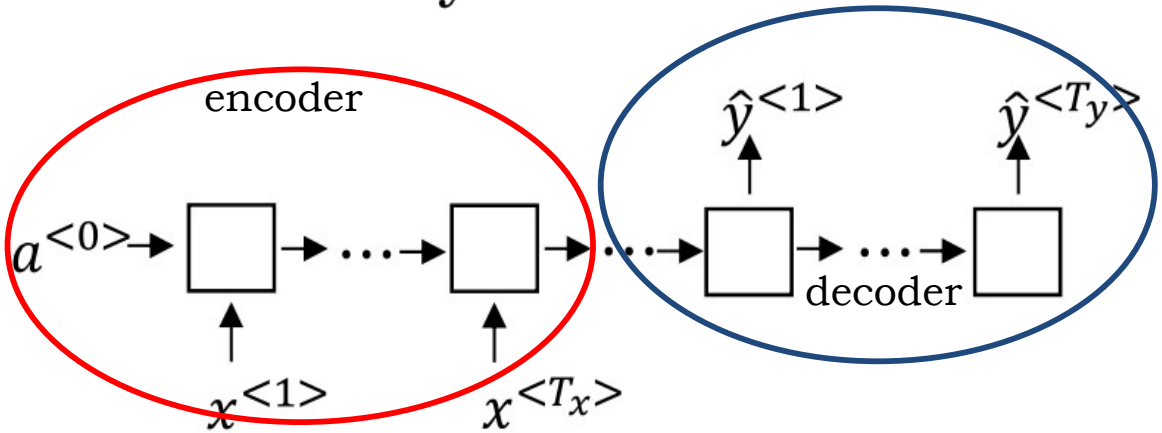


Many to one (e.g. Sentiment analysis)



Many to many

Tx=Ty (e.g. Name entity rec.)

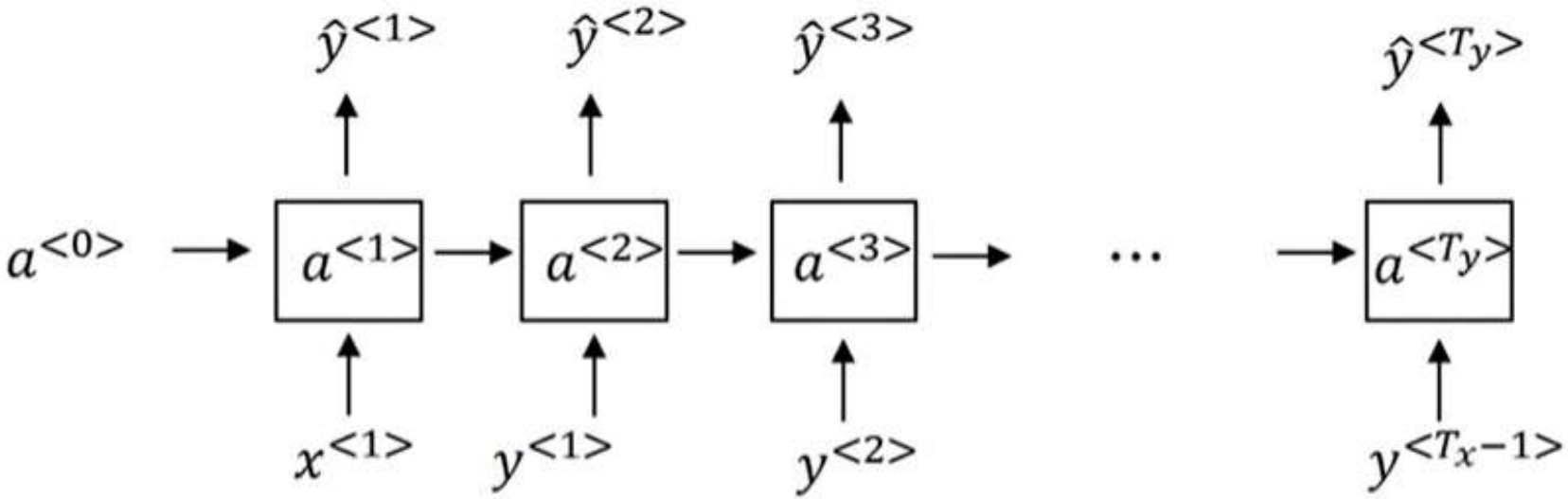


Many to many

Tx different from Ty (e.g. machine translation)

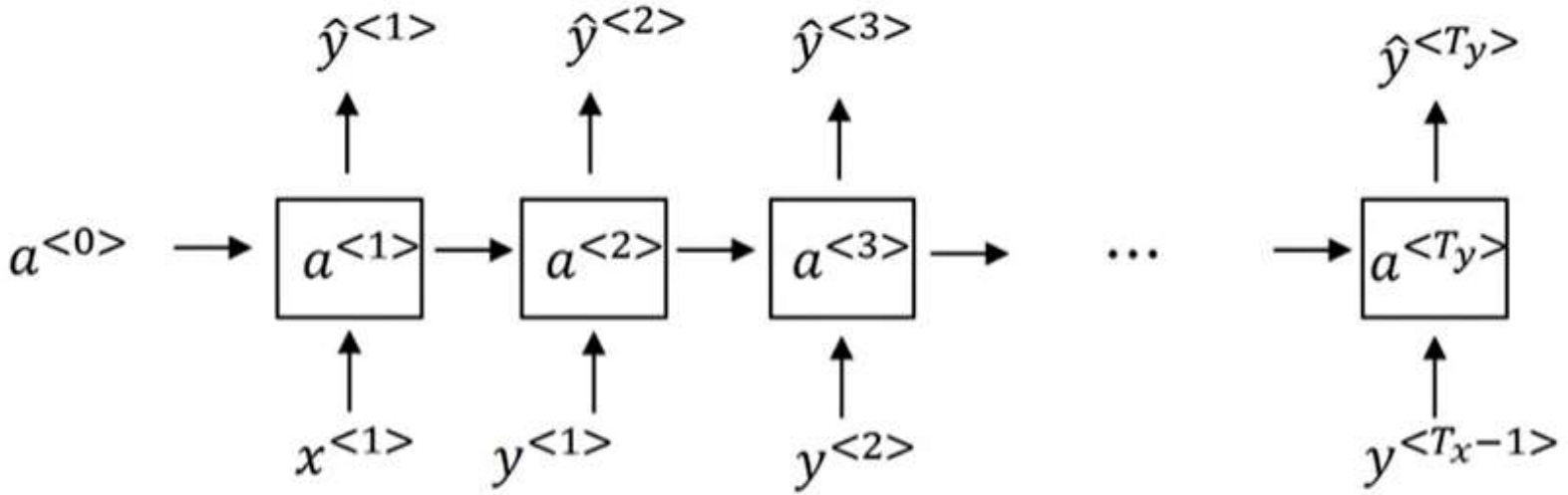
RNN Word-level Language Model

P(sentence) = P($word_1$, $word_2$,…$word_N$) =>
what is the sentence probability ?
P("The apple and pear salad.") or P("The apple and pair salad") ?
Training set : Large corpus of text.



Given the first word in the sentence ($x^{<1>}$) what is the most probable second word ($y^{<1>}$) .
Then given the first two words what is the most probable third word ($y^{<2>}$) , and so on.

RNN Word-level Language Model



3 word sentence:

$P(y^{<1>}, y^{<2>}, y^{<3>}) = P(y^{<1>}) * P(y^{<2>} | y^{<1>}) * P(y^{<3>} | y^{<1>}, y^{<2>})$

Overall Loss function: sum of loss functions at each step. At a certain time *t* if the correct word is $y^{<t>}$ and NN softmax predicted some hat $y^{<t>}$, this is softmax loss function:

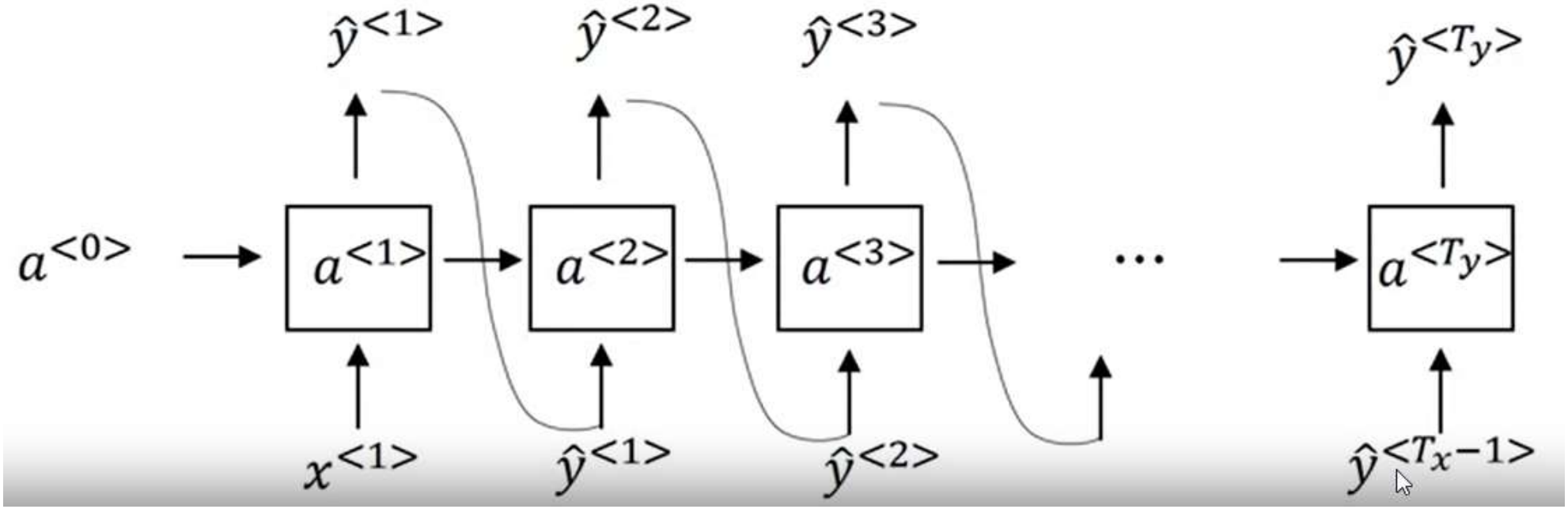
$$\mathcal{L}(\hat{y}^{<t>}, y^{<t>}) = -\sum_i y_i^{<t>} \log \hat{y}_i^{<t>}$$

$$\mathcal{L} = \sum \mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

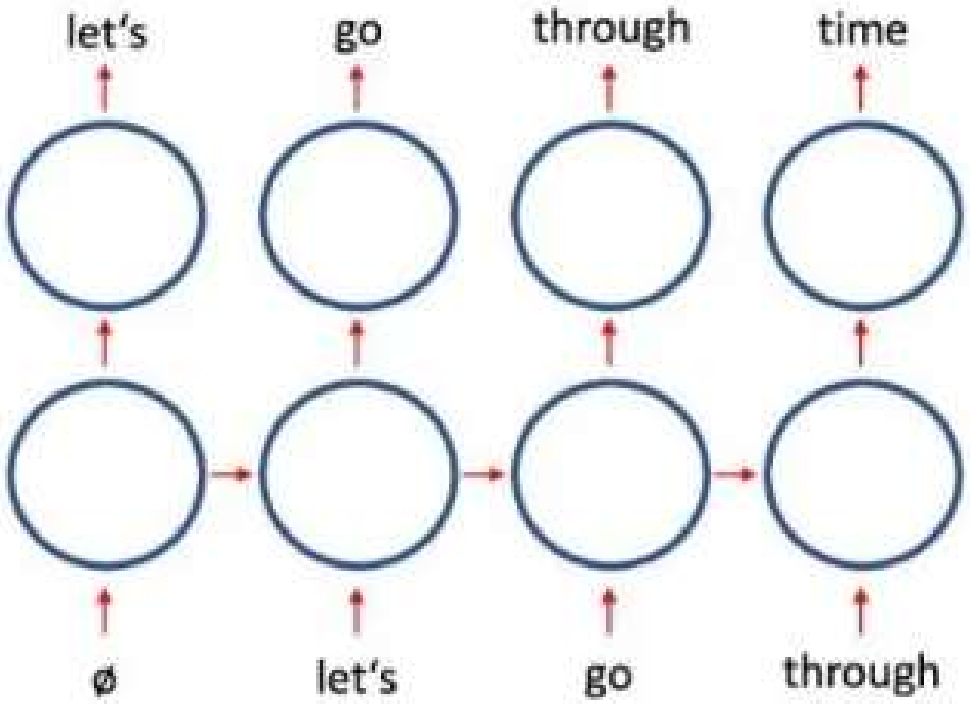
Sequence Generation with RNN

RNN is trained to predict the probability distribution of the next element => it can be used to generate sequences.

Start with empty symbol, then use RNN to generate some output (e.g. “The”)
Then, take this output and put it into the next state’s input.
Go ahead and see that the trained RNN can actually generate whole sequences with some meaning.



Ex. -Sequence Generation with RNN



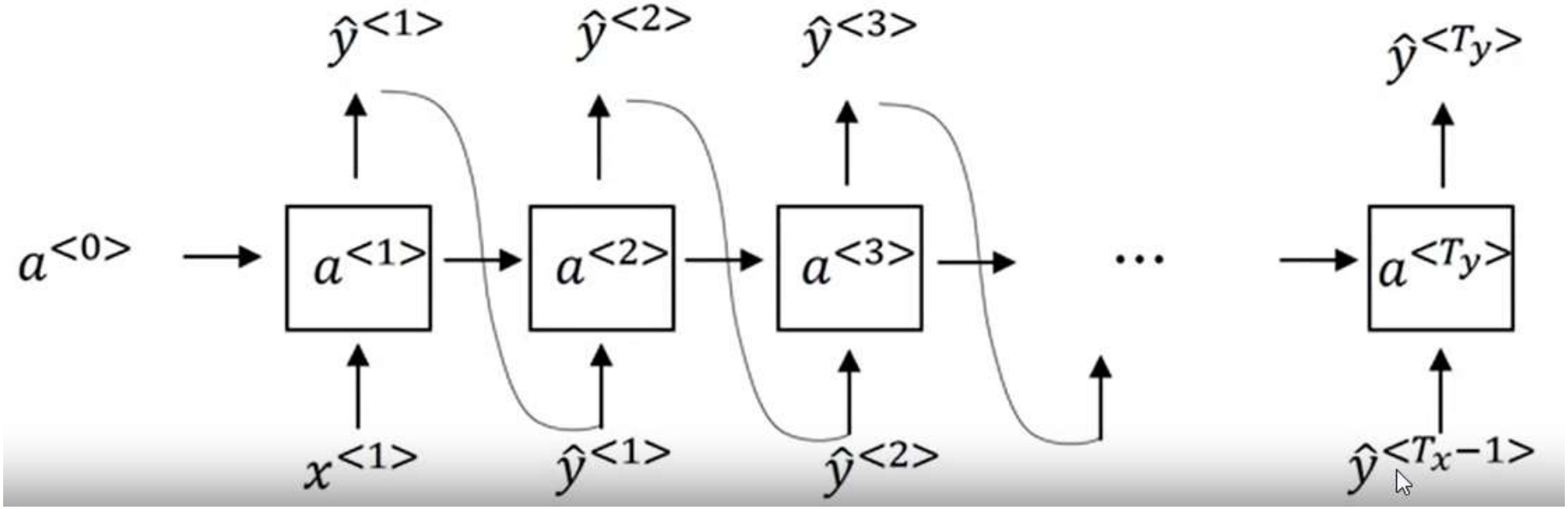
Start with empty symbol => the model generates at the first output = "let's"
If $x^{<1>}$ = "let's" what is the most probable second word [max P(?/ $x^{<1>}$ =let's)].
The model generates at the second output "go"
Then given the first two words what is the most probable third word
[max P(?/ "let's", "go")].
The model generates "through", and so on….

Character-level Language Model

Vocabulary for word-level language model:
V={a, aaron, …zulu, <UNK>} - 10 000 /50 000/400 000 words

Vocabulary for character-level language model:
V= {a, b, c, ….z, ., ; , 0,….9, A,……Z}

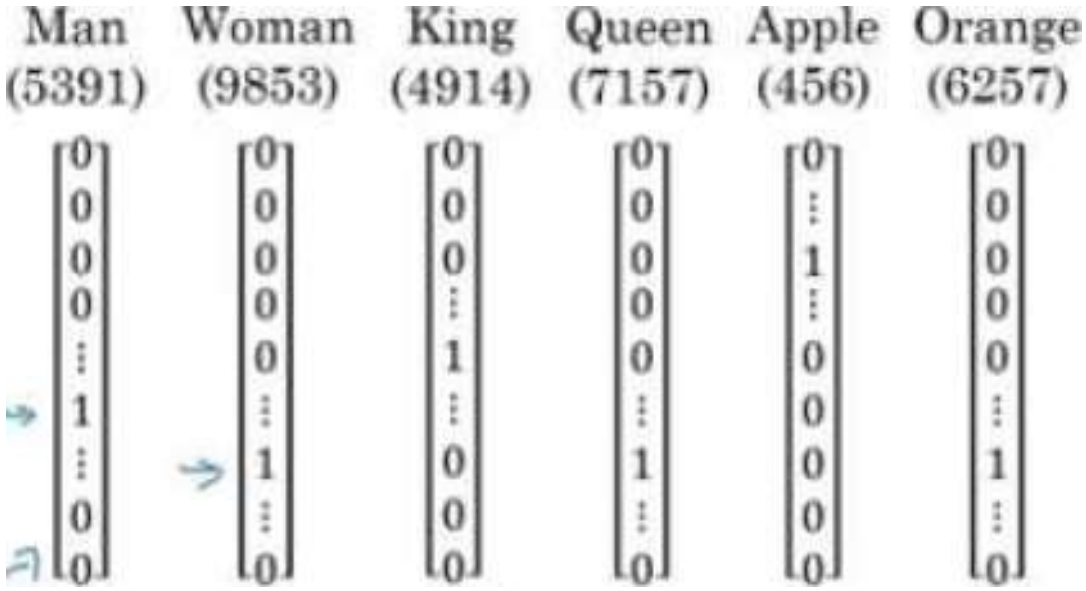
The sequence will be individual characters: **$y^{<1>}$, $y^{<2>}$, $y^{<3>}$…**
Computationally more expensive, more specialised applications to deal with unknown words.



NLP – Word Embedding

Word embedding is a way of representing words.
Vocabulary ={a, aaron, …zulu, <UNK>} - e.g.10000 words

One hot encoding vector. This representation does not relate the words. The inner product between one-hot vectors is 0. The distances between them are the same. It cannot generalize across words:
Example: **"Ana wants a glass of orange …?.",** let assume the model learned to predict the next word as **juice**.
A similar example **"Ana wants a glass of apple …?.",** the model won't easily predict juice if it was not trained on it.
The two examples are not related although orange and apple are similar.



Word Embedding (featurized representation)

Instead of one hot vector, each word is represented with a number of features – gender, royal, age, food, ….etc. e.g. 300-dimensional vector. It reduces the size of the input (ex. from 10000 to 300).

Example: Orange and apple now share many similar features which makes it easier for the model to generalize between them.

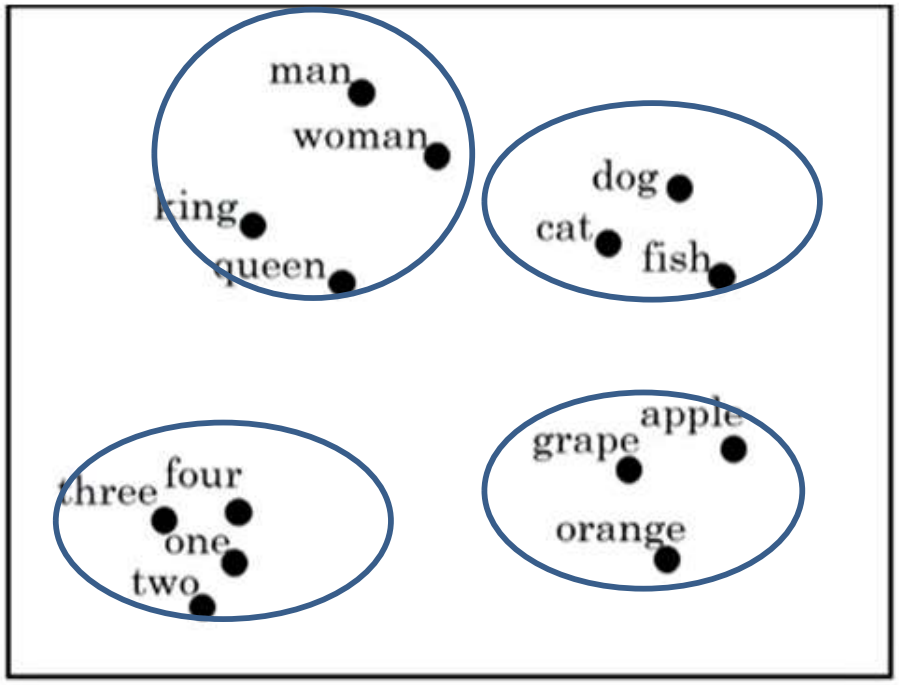
	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	−1	1	-0.95	0.97	0.00	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0.00
Age	0.03	0.02	0.70	0.69	0.03	-0.02
Food	0.09	0.01	0.02	0.01	0.95	0.97

e_{Man} e_{Woman} e_{King} e_{Queen}

Visualizing word embeddings

t-distributed Stochastic neighbour embedding (**t-SNE**) reduces the features to 2D . t-SNE is a complex non-linear mapping.
Related words are closer to each other.

Ex. ***Ana*** wants a glass of **orange** juice.
Mary wants a glass of **apple** ---- juice. => easier to find the similarity.
Tom wants a glass of **durian** -----juice. => easier to find the similarity
even if this word was not in the train set but is in the embedding dictionary



Van der Maaten L, Hinton G. Visualizing data using t-SNE. Journal of ML research. 2008

Transfer Learning & Word Embeddings

1. Learn word embedding from a large corpus (1-100 billion of words) of unlabelled text or download pre-trained embedding (if your data is smaller).

2. Transfer embedding to new task with the smaller training set (e.g. 100k words).

3. Optional. Continue to finetune the word embeddings if your data from p. 2 is relatively big.

Word embeddings has a big effect when for the new task we work with smaller training set.

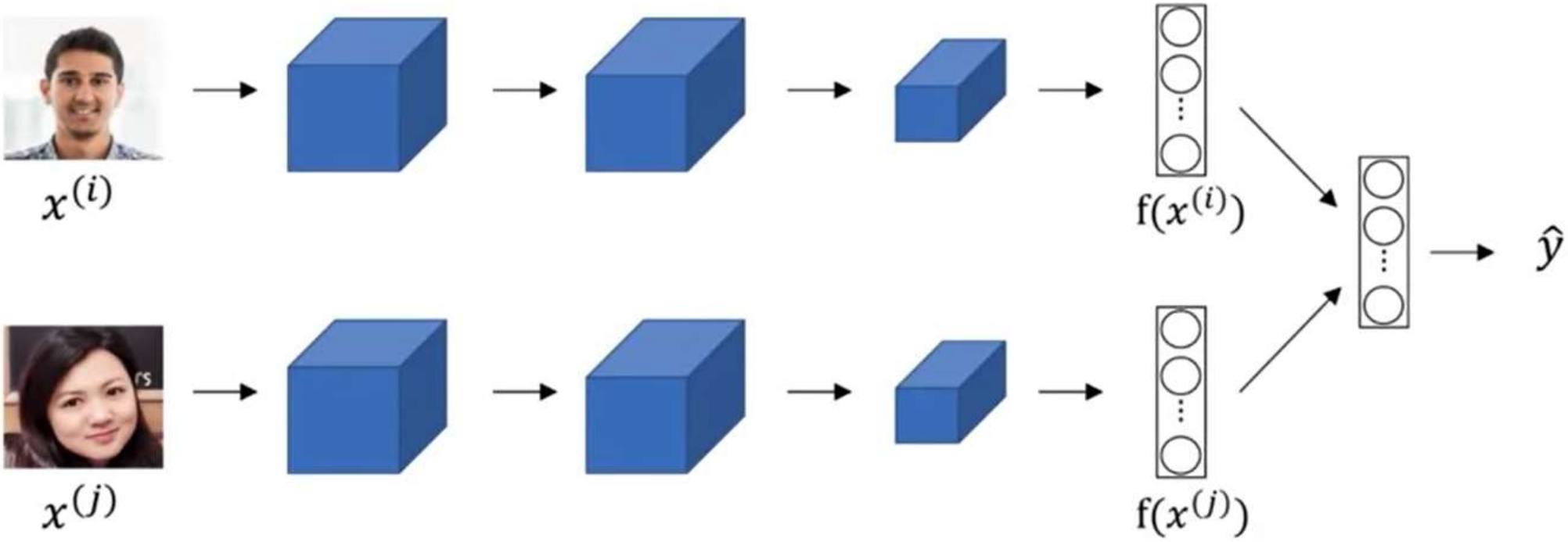
Word embeddings reduce the size of the input, e.g. 10000 one hot sparse encoding compared to 300 feature (embedding) vector.

Relation to face recognition

In Face Recognition we **encode** each face into a feature vector and check how similar are these vectors.

In Word Embedding we also represent each word with a feature vector and search similarity between words.

Image ***encoding*** and word ***embeddings*** have similar meaning.



Word embeddings – analogy reasoning

	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	−1	1	-0.95	0.97	0.00	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0.00
Age	0.03	0.02	0.70	0.69	0.03	-0.02
Food	0.09	0.01	0.02	0.01	0.95	0.97
	e_{Man}	e_{Woman}	e_{King}	e_{Queen}		

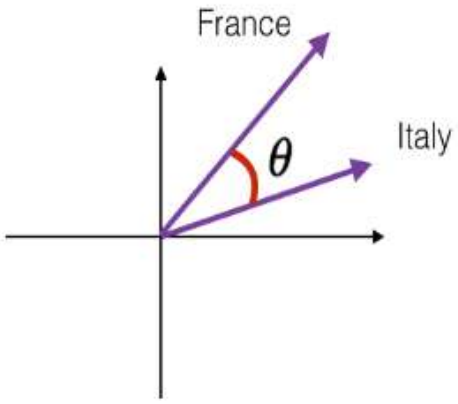
Given the word embeddings table. Can we find the analogy between:
Man => Woman
King => ??
Subtract: $e_{man} - e_{woman}$ =[-2 0 0 0]
Subtract: $e_{king} - e_{queen}$ =[-2 0 0 0]
The two relations are similar, the difference is about the gender.
We formulate the problem: $e_{man} - e_{woman}$ is similar to $e_{king} - e_{???}$
Which is the word $e_{???}$ that maximizes similarity ($e_{???}$, $e_{king} - e_{man} + e_{woman}$) ?

Cosine Similarity

To measure the similarity between two words, you need a way to measure the degree of similarity between two embedding vectors for the two words. Given two vectors u and v, cosine similarity is defined as follows:

$$\text{CosineSimilarity(u, v)} = \frac{u \cdot v}{||u||_2 ||v||_2} = cos(\theta) \quad (1$$

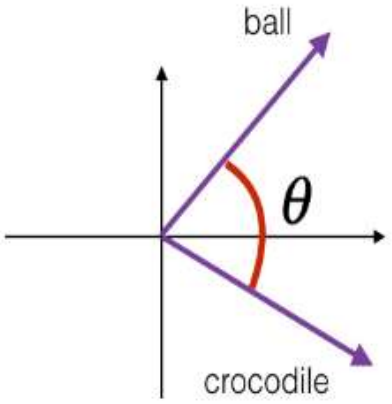
- $u \cdot v$ is the dot product (or inner product) of two vectors
- $||u||_2$ is the norm (or length) of the vector u
- θ is the angle between u and v.
- The cosine similarity depends on the angle between u and v.
  - If u and v are very similar, their cosine similarity will be close to 1.
  - If they are dissimilar, the cosine similarity will take a smaller value.



France and Italy are quite similar

θ is close to 0°

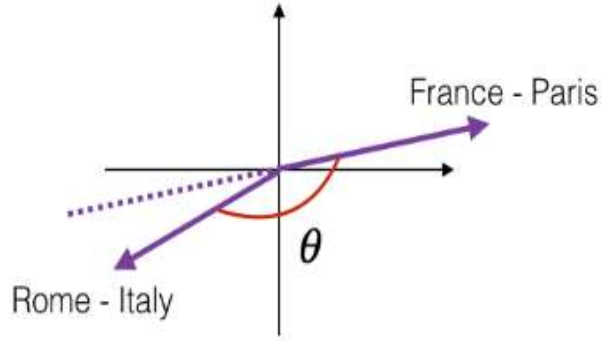
$\cos(\theta) \approx 1$



ball and crocodile are not similar

θ is close to 90°

$\cos(\theta) \approx 0$



the two vectors are similar but opposite
the first one encodes (city - country)
while the second one encodes (country - city)

θ is close to 180°

$\cos(\theta) \approx -1$

Cosine Similarity

$$\text{CosineSimilarity(u, v)} = \frac{u.\,v}{||u||_2||v||_2} = cos(\theta)$$

We can use this equation to calculate the similarity between word embeddings for the analogy problem where :

$$\mathbf{u = e_{???}}\ , \qquad \mathbf{v = e_{king} - e_{man} + e_{woman}}$$

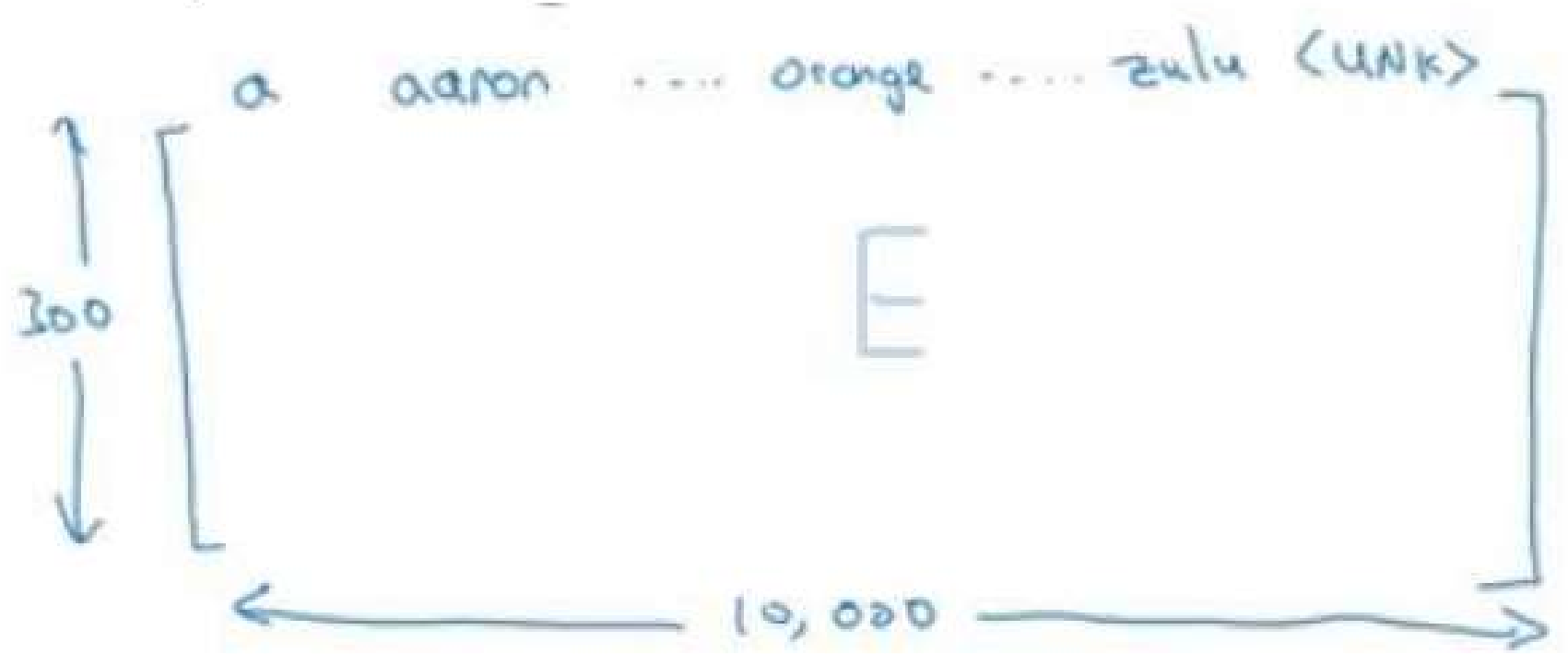
Compute the Cosine Similarities between all words in the dictionaryy and $\boldsymbol{v}$.

$\boldsymbol{e_{queen}}$ is expected to be the best solution for this example.

Euclidian distance can be also used as a similarity (dissimilarity) function.

Embedding matrix

- Suppose we are using 10,000 words as our vocabulary (plus token).
- The algorithm should create a matrix E of the shape (300, 10000) in case we are extracting 300 features.

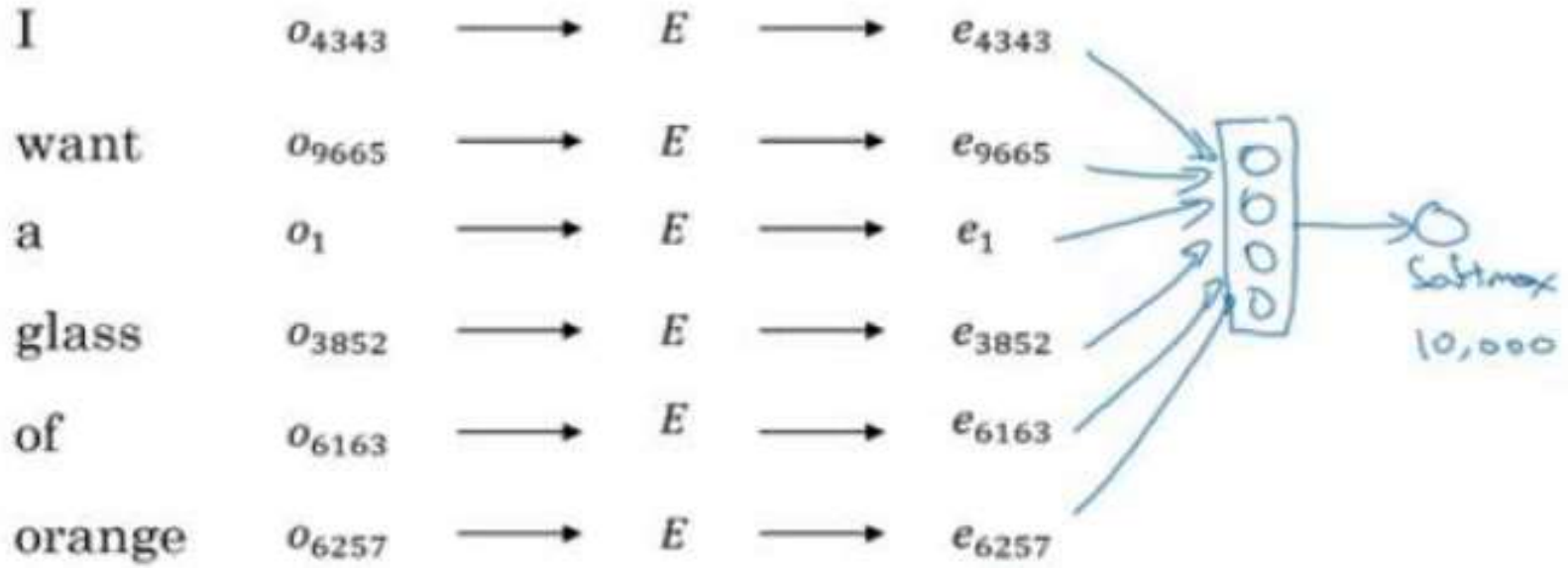


- If O_{6257} is the one hot encoding of the word **orange** of shape (10000, 1), then *np.dot(E ,*O_{6257}*)* = e_{6257} which shape is (300, 1).
- Generally *np.dot(E ,* O_j*)* = e_j

Neural Language Model (2003)

I	want	a	glass	of	orange	_____.
4343	9665	1	3852	6163	6257	

- We want to build a language model so that we can predict the next word.
- So we use this neural network to learn the language model



Shallow ANN (input, hidden, softmax layers):
- e_j =*np.dot(E, o_j)* => NN input dimension is (300*6,1) if window size is 6 previous words
- hidden layer with (***W1, b1)*** parameters
- softmax layer with ***(W2, b2)*** parameters
- **Start with random values for E (embedding matrix) and layers parameters.**

Optimize them to maximise the likelihood to predict the next word given the context (previous words).

Neural Language Model (context-target)

Example: *"I want a glass of orange juice to go along with my cereal"*

To learn ***juice***, there are different choices of **context:**

1) **Last 4 words** (4 is a hyperparameter) : "a glass of orange" =>
try to predict the next word from it

2) **4 words on the left & on the right**: "a glass of orange" &
"to go along with"

3) **Last 1 word**: "orange"

4) **Nearby 1 word** (this is a skip grams model) : "glass"

Summary: Language modelling is a ML problem where the input is the **context** (e.g. last few words) and the output is to predict some **target words.**

Word2Vec - skip grams model

Example: *"I want a glass of orange juice to go along with my cereal"*
Randomly pick up words as context and target words
(e.g. within -10/+10 words of the context word)

CONTEXT (c)	**=>**	**TARGET (t)**
"orange" (vocab index 6257)	=>	"juice" (vocab index 4834)
"orange"	=>	"glass"
"orange"	=>	"my"

Vocab size = 10000 words

Input e_c *=np.dot(E,* o_c*)* => *softmax* => *output* y^{hat}

Softmax model: probability of target word given the context word.
What is the chance of a particular word ***t*** to be the label.
θ_t - parameters associated with output word ***t***.

$$p(t|c) = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10,000} e^{\theta_j^T e_c}}$$

Softmax loss function:

$$\mathcal{L}(\hat{y}^{<t>}, y^{<t>}) = -\sum_i y_i^{<t>} \log \hat{y}_i^{<t>}$$

$$\mathcal{L} = \sum \mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

universidade de aveiro

Problem with softmax model

$$p(t|c) = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10,000} e^{\theta_j^T e_c}}$$

For big vocab size (e.g. 1 mln words) the computations become slow.
Solution: Hierarchical softmax classifier (works as a tree classifier).

The choice of the context words:

Too frequent (common) words: the, of, a, and, to, …

The context words are not chosen uniformly random, instead there are some heuristics to balance the common words and the non-common words.

Negative Sampling

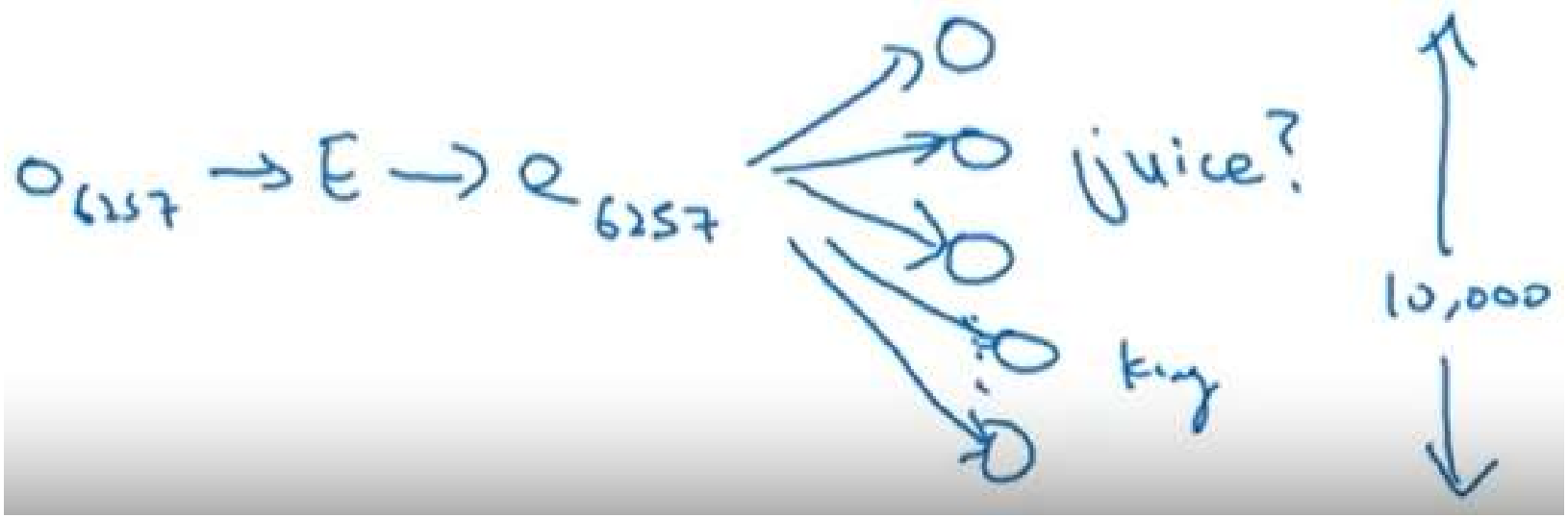
Generate a training dataset:
Take one context word, create one positive context pair,
then pick ***k*** random words from the dictionary but as negative context examples.
k=5-20 examples for small datasets
k=2-5 for larger datasets

CONTEXT(c)	**word (t)**	**=>**	**TARGET (y) ?**	
"orange"	"juice"		1	(only one positive pair)
"orange"	"king"		0	
"orange"	"book"		0	
"orange"	"the"		0	
	.....		0	
"orange"	"of"		0	

Supervised learning, e.g. logistic regression model:
P(y=1 | c, t) = logistic regression (θ_t , e_c)

Negative Sampling

We have 10000 binary classification problems, but we only train k+1 classifier in each iteration.



How to select negative examples ?
The following formula is proposed to select the negative examples according to empirical frequences of the words $f(w_i)$ but to sample not too much the more frequent words like the, of, and, etc. :

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=1}^{10,000} f(w_j)^{3/4}}$$

GloVe Word Vectors

GloVe- Global vectors for word representations.

From the training corpus of words we count:

X_{ij} = # of times word *i (target)* appears of context to word *j (context)*.

$X_{ij} = X_{ji}$ (if the context word is within proximity of +/- 10 words of target).

If the context is a few words before the target word, X_{ij} is NOT = X_{ji} .

Loss function represents the difference between the inner product ($\theta_i^T \boldsymbol{e_j}$) as a predictor of how often two words occur together.
Gradient descent optimizes parameters θ_i (associated with target ***i***) and the embedding $\boldsymbol{e_j}$ of the context word j.

$f(X_{ij})$ – weighting term

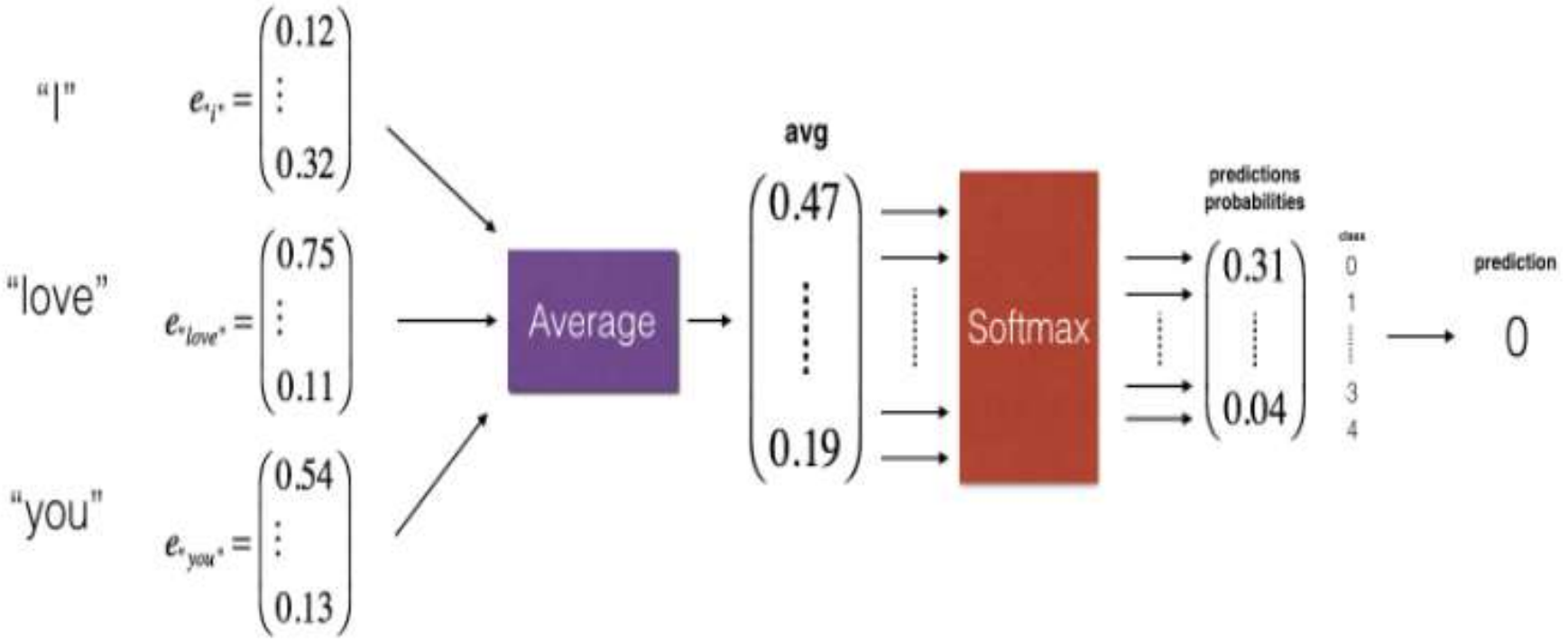
$$\text{minimize} \sum_{i=1}^{10,000} \sum_{j=1}^{10,000} f(X_{ij})(\theta_i^T e_j + b_i - b_j' - \log X_{ij})^2$$

NLP - Sentiment Classification

The dessert is excellent.	
Service was quite slow.	
Good for a quick meal, but nothing special.	
Completely lacking in good taste, good service, and good ambience.	

We may not have a huge labeled training data, but using word embeddings can help.
The common data size varies from 10 000-100 000 words.

Sentiment Classification – simple model



Transform each word of the sentence from one-hot encoding to embedding vector.

Embedding matrix ***E*** trained on 1 – 100 billion words. (in lab 400 000 words)

Number of features in word embedding say 300 (in lab 50 dimensions)

Sum or average the embedding representations of all words of the sentence, then pass to a softmax classifier.

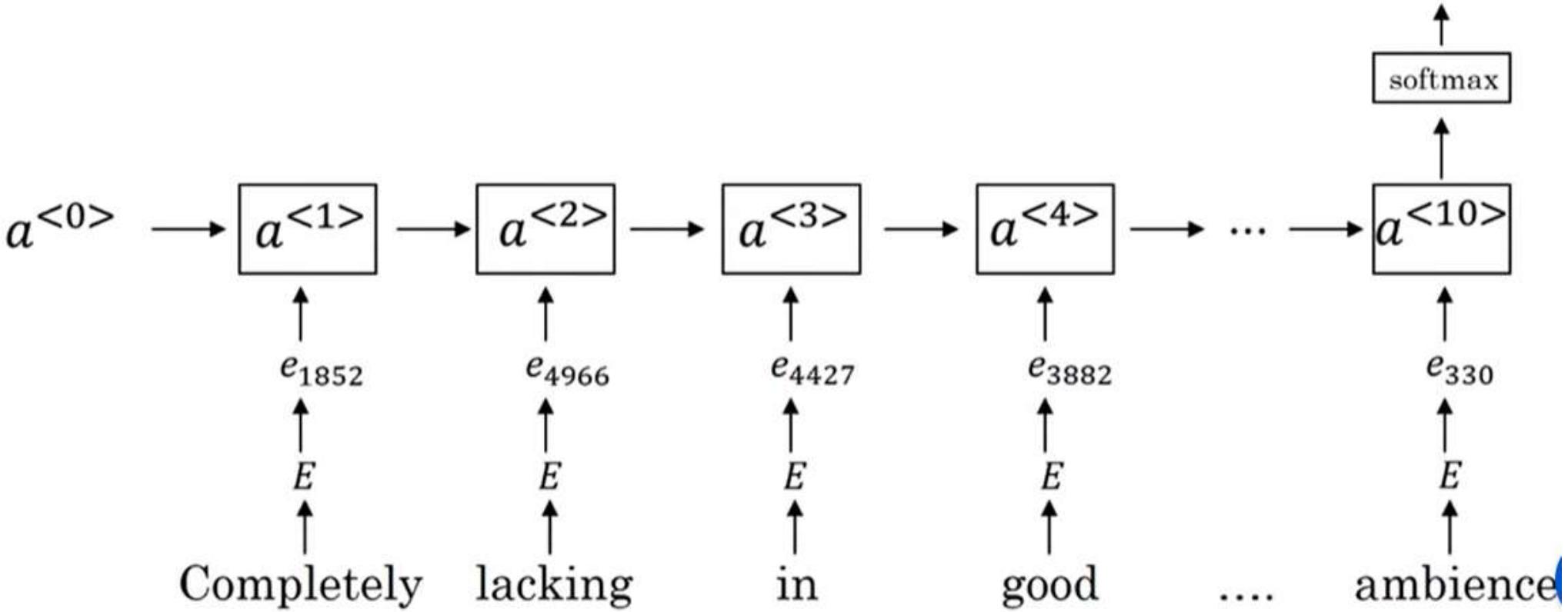
This classifier works for short or long sentences.

Problem: the model ignores words order.

Ex. "*Completely lacking in good taste, good service, and good ambience*"

3 times the word "good" but it's a negative review.

Sentiment Classification with RNN



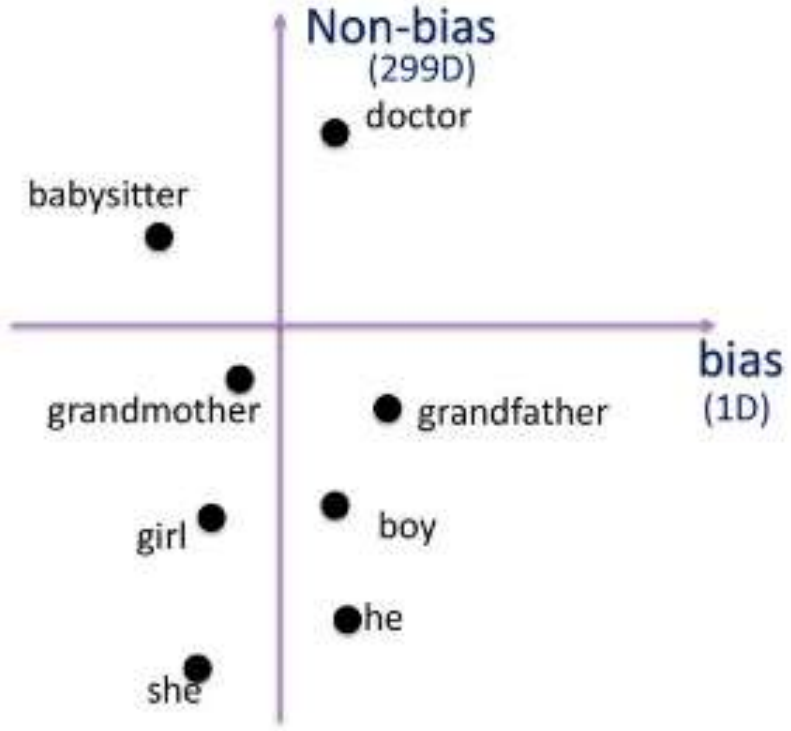
Many-to-one RNN : better generalization than the baseline averaging model.

Ex. "*Completely* ***absent*** *in good taste, good service, and good ambience*"
If the word "**absent**" was not in the small labeled training data specific for sentiment classification but was in the big corpus (1 – 100 billion words) used to train the word embeddings, most probably it will get the right meaning.

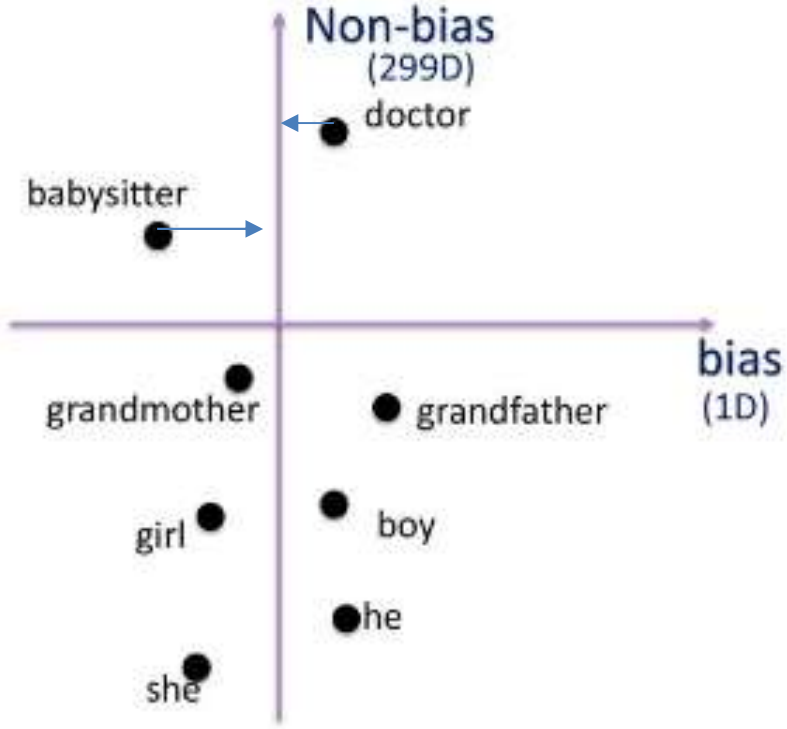
Debiasing Word Embeddings
(ongoing research field)

- Man:Woman as King : Queen (OK)
- Man: Computer programmer as Woman:Homemaker (**Wrong**)
- Father:Doctor as Mother:Nurse (**Wrong, unhealty gender stereotypes**)

Word embedding can reflect gender, ethnicity, age, sexual orientation, and other biases of text used to train the model.



Debiasing Word Embeddings



Steps to solve gender bias problem (valid for any type of bias):

1. Identify the bias direction => calculate some ***k*** differences between gender related words and average them:
$e_{he} - e_{she}$; $e_{male} - e_{female}$, ...etc.

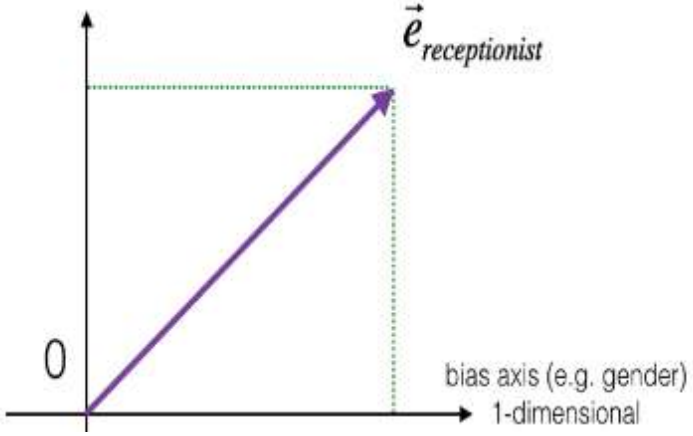
2. Neutralize: for every word that is not definitional (e.g. gender neutral such as doctor, babysitter) project on the non-bias subspace.

3. Equalize pairs to be equaly distant from the non-bias subspace, so that they will be equaly distant from the projected words (e.g. doctor, babysitter):

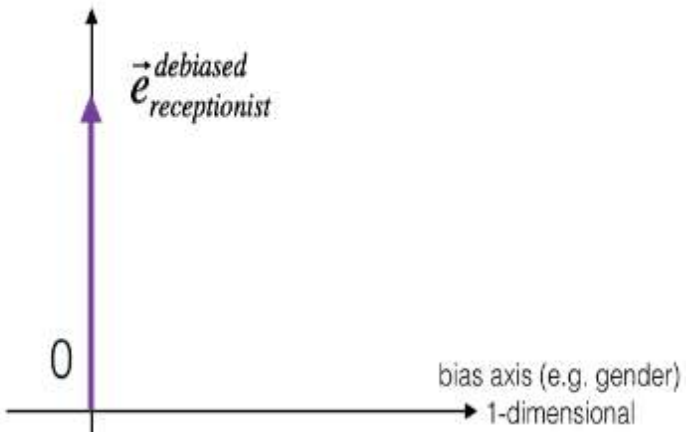
grandfather – grandmother
he-she;
boy-girl

Debiasing Word Embeddings

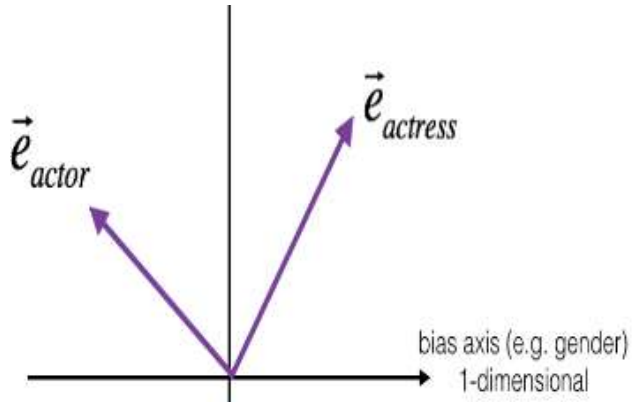
before neutralizing



after neutralizzing



before equalizing



after equalizing