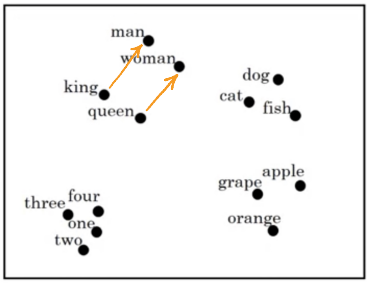
$e_{man} - e_{woman} \approx \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

$e_{king} - e_{queen} \approx \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

THE MAIN DIFFERENCE BETWEEN MAN/WOMAN AND KING/QUEEN IS THEIR GENDER FEATURE.

$e_{man} - e_{woman} \approx e_{king} - e_{queen}$



* you want to find the word (w) that maximizes the similarity between e_w and $e_{king} - e_{man} + e_{woman}$.

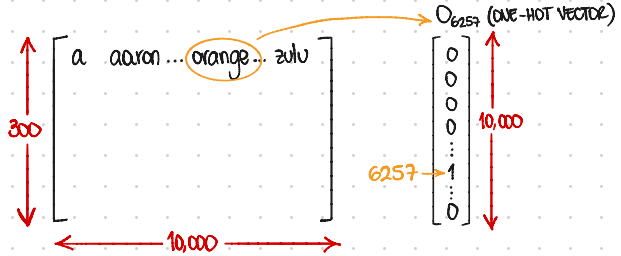
$$\arg\max_w \; \text{Sim}(e_w, e_{king} - e_{man} + e_{woman})$$

* you can also find the similarity between two vectors with the cosine similarity:

IF u AND v ARE VERY SIMILAR, THEN THE INNER PRODUCT (NUMERATOR) WILL BE VERY LARGE

$$\text{Sim}(u,v) = \frac{u^T v}{\|u\| \|v\|}$$

EMBEDDING MATRIX



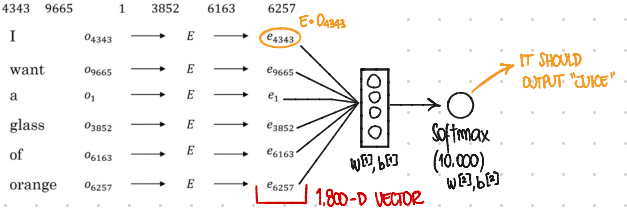
$E_{(300,10K)} \cdot O_{6257}{}_{(10K,1)} = e_{6257}{}_{(300,1)}$

$E \cdot O_j = e_j$

EMBEDDING FOR WORD j

LEARNING WORD EMBEDDINGS

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



I want a glass of orange juice to go along with my cereal.

CONTEXT TARGET CONTEXT

* you can select (#) contexts to feed your (NN) to predict the target
  - → last 4 words
  - → 4 words on the left and right
  - → word before the target
  - → nearby one word (skip gram)

WORD2VEC MODEL

* rather than having the context be always the previous 4 words or the following 4 words, you randomly pick a word to be the context word. Then you pick randomly another word within some window (±5 or ± 10 words from the context word) and that becomes the target.

I want a glass of orange juice to go along with my cereal.

T CONTEXT T T T

* you want to learn the mapping from some word (c) to some target (t)

context c ("orange") → target t ("juice")

6257 4834

X → Y

SKIP-GRAM MODEL

$O_c \rightarrow E \rightarrow e_c \rightarrow O \rightarrow \hat{y}$ (SOFTMAX)

$\theta_c = E_c \cdot O_c$

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

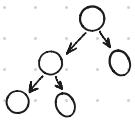
SOFTMAX

θ_t = parameter associated with output t.

LOSS FUNCTION

$$\mathcal{L}(\hat{y}, y) = -\sum_{i=1}^{10,000} y_i \log \hat{y}_i$$

* disadvantages of skip-gram model:
  - computational speed: for the softmax function, you need to carry out a sum over all 10,000 words in the vocabulary. SOLUTION?
    - → HIERARCHICAL SOFTMAX CLASSIFIER:
      * you run the softmax on subsets of the vocabulary until you narrow it down to the target



* how to choose/sample context (c)? → sample uniformly at random from your training corpus.

NEGATIVE SAMPLING

* you sample a context and target word and associate the pair with a label of 1 (they go well together) or a label of 0 (they don't go well together).

K = 5-20 SMALL SETS
K = 2-5 LARGE SETS

CONTEXT (c)	WORD (t)	TARGET (y)
orange	juice	1
orange	king	0
orange	book	0
orange	the	0
orange	of	0

(k)

* then you create a supervised learning problem where you input the training pairs along with their labels.

$$P(y=1 \mid c,t) = \sigma(\theta_t^T e_c)$$ we use (σ) for logistic regression

$O_{6257} \rightarrow E \rightarrow e_{6257}$ → juice?

10,000 classification units (binary classification) } less computational power

* how do you sample the negative examples?

- → sample according to the empirical frequency of words in your corpus. DISADVANTAGE: you end up with an overrepresentation of words like "the", "of", "a", etc.
- → sample negative examples uniformly at random. DISADVANTAGE: not very representative of the distributions of English words.
- → THIS WORKS BEST!

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

something in between the previous two examples

GLOVE WORD VECTORS

* global vectors for word representation

X_{ij} = # times (j) appears in the context (i)

j → t

i → c

$$\min \left[\sum_{i=1}^{10{,}000} \sum_{j=1}^{10{,}000} f(X_{ij}) (\theta_i^T e_j + b_i + b'_j - \log X_{ij})^2 \right]$$

WEIGHTING TERM
$f(X_{ij})=0$ IF $X_{ij}=0$

SHOULD BE RANDOMLY INITIALIZED AT THE BEGINNING OF TRAINING

SENTIMENT CLASSIFICATION

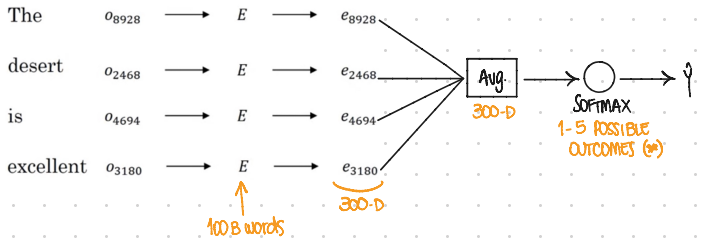
x	y
The dessert is excellent.	★★★★☆
Service was quite slow.	★★☆☆☆
Good for a quick meal, but nothing special.	★★★☆☆
Completely lacking in good taste, good service, and good ambience.	★☆☆☆☆

* you may not have a big training set (10,000 - 100,000 words)

* Simple sentiment classification problem:

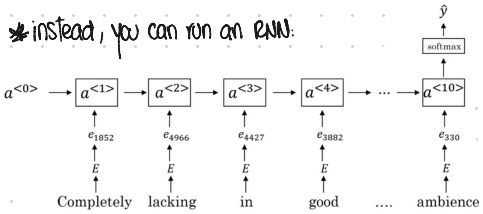
The dessert is excellent ★★★★☆

8928 2468 4694 3180



DISADVANTAGE: it ignores words' order.

* instead, you can run an RNN:



many-to-one architecture

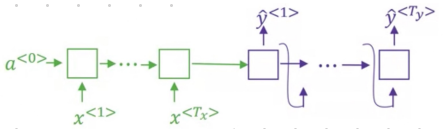
SEQUENCE TO SEQUENCE MODELS

* mostly used for machine translation

$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ $x^{<4>}$ $x^{<5>}$
Jane visite l'Afrique en septembre

⟶ Jane is visiting Africa in September.
$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ $y^{<4>}$ $y^{<5>}$ $y^{<6>}$

DECODER



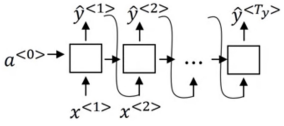
ENCODER

* it can also be used for image captioning

PICKING THE MOST LIKELY SENTENCE

* machine translation ⟶ building a conditional language model

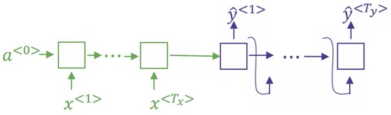
Language model:



* you estimate the probability of a sentence.

$$P(y^{<1>}, \dots, y^{<T_y>})$$

Machine translation:



* "conditional language model": you estimate the probability of an English translation given the French sentence provided as input.

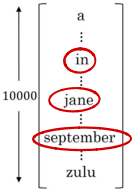
$$P(y^{<1>}, \dots, y^{<T_y>} \mid x^{<1>}, \dots, x^{<T_x>})$$

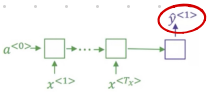
* for machine translation you don't want to sample at random. You want to find the sentence that maximizes the conditional probability.

BEAM SEARCH

* steps for the beam search algorithm

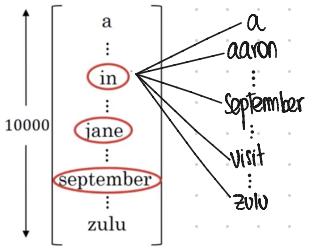
STEP (1)

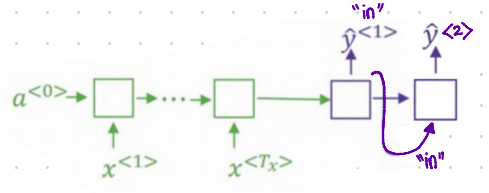




* you run the input French sentence through the encoder network and the first step will decode the network. Then, it will output overall 10,000 possibilities. It will output "in", "Jane", and "September" as the three more likely choice of the first word.

Step II





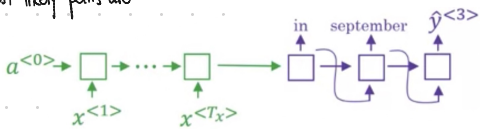
$P(y^{<2>} | x, \text{"in"})$

* in general, we want to find $P(y^{<1>}, y^{<2>} | x) = P(y^{<1>} | x) * P(y^{<2>} | x, y^{<1>})$

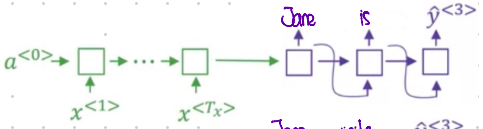
Step III

* you found that the most likely pairs are:

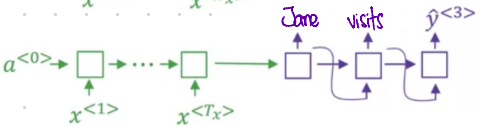
in september



jane is



jane visits



Beam Search Refinement

(#1). length normalization:

- $\arg\max \prod_{t=1}^{T_y} P(y^{<t>} | x, y^{<1>}, \dots, y^{<t-1>})$ → because some probabilities may be almost 0, we implement log for the algorithm to work better.
- $\arg\max \sum_{t=1}^{T_y} \log P(y^{<t>} | x, y^{<1>}, \dots, y^{<t-1>})$ → one other thing that may improve the algorithm is to normalize it by the number of words in the translation. It also reduces the penalty for outputting long translations.
- $\frac{1}{T_y} \sum_{t=1}^{T_y} \log P(y^{<t>} | x, y^{<1>}, \dots, y^{<t-1>})$

(#2) how to set B?

- large B: better results, but slower
- small B: worse results, but faster

} commonly (10)

ERROR ANALYSIS ON BEAM SEARCH

Jane visite l'Afrique en septembre.

Human: Jane visits Africa in September.

Algorithm: Jane visited Africa last September.

} we try to see why this ≠ in translation took place. Was it the RNN or the Beam search?

* in this case, increasing Ⓑ might not influence at all

#1. CASE 1 → $P(y^*|x) > P(\hat{y}|x)$

→ Beam search chose $\hat{y}$, but y^* attains higher $P(y|x)$.

→ Beam search is at fault!!

#2. CASE 2 → $P(y^*|x) \leq P(\hat{y}|x)$

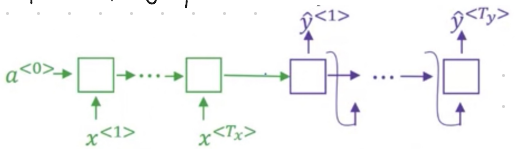
→ y^* is a better translation than $\hat{y}$, but the RNN predicted $P(y^*|x) < P(\hat{y}|x)$.

→ RNN model is at fault !!

Human	Algorithm	$P(y^*\|x)$	$P(\hat{y}\|x)$	At fault?
Jane visits Africa in September.	Jane visited Africa last September.	2×10^{-10}	1×10^{-10}	BEAM!
		1×10^{-10}	2×10^{-10}	RNN

ATTENTION MODEL INTUITION

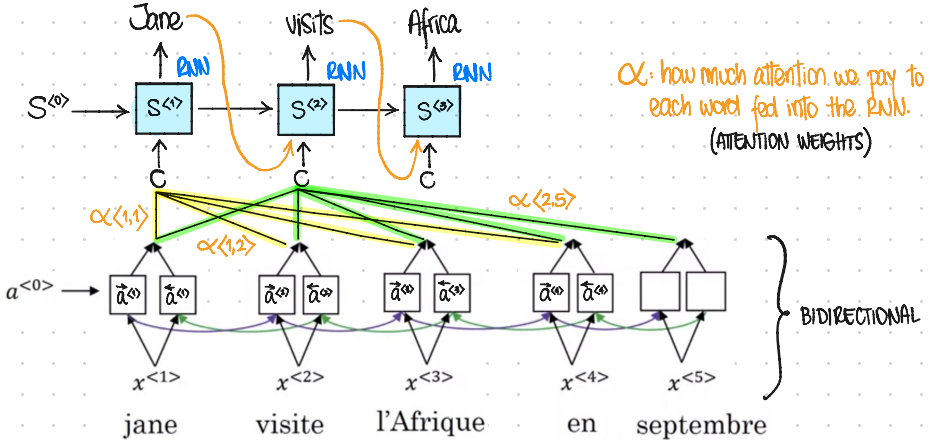
* it helps solve the problem of long sequences



Jane s'est rendue en Afrique en septembre dernier, a apprécié la culture et a rencontré beaucoup de gens merveilleux; elle est revenue en parlant comment son voyage était merveilleux, et elle me tente d'y aller aussi.

* it translates sentences more like humans do: a part of the sentence at a time.

* it pays attention to part of the sentence while generating a translation

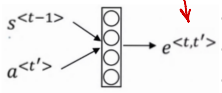


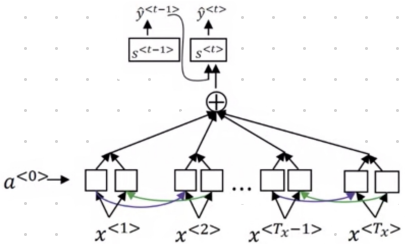
α: how much attention we pay to each word fed into the RNN. (ATTENTION WEIGHTS)

* the attention weights satisfy the condition $\sum_{t'} \alpha^{<1,t'>} = 1$
* the context element (C) satisfies the condition $C = \sum_{t'} \alpha^{<1,t'>} a^{<t'>} \rightarrow a^{<t'>} = (\vec{a}^{<t'>}, \overleftarrow{a}^{<t'>})$

amount of "attention" $y^{<t>}$ should pay to $a^{<t'>}$

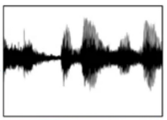
$$\alpha^{<t,t'>} = \frac{\exp(e^{<t,t'>})}{\sum_{t'=1}^{T_x} \exp(e^{<t,t'>})}$$



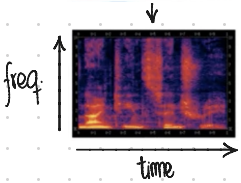


* the model runs in cuadratic cost
* it also works for image captioning

SPEECH RECOGNITION

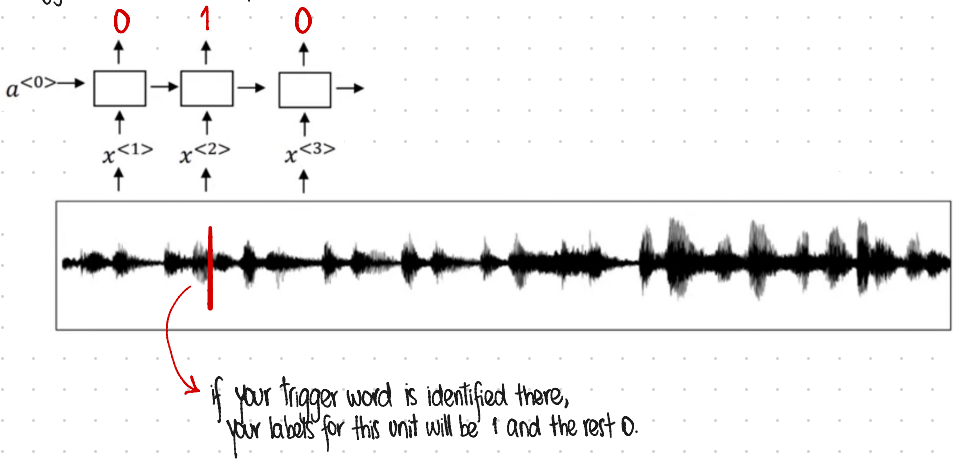


* reasonable size of data: 3,000 - 30,000 hours



SPECTOGRAM → colors: amount of energy

- Trigger work detection system:



- downsize → it creates an imbalanced dataset with more 0 than 1.