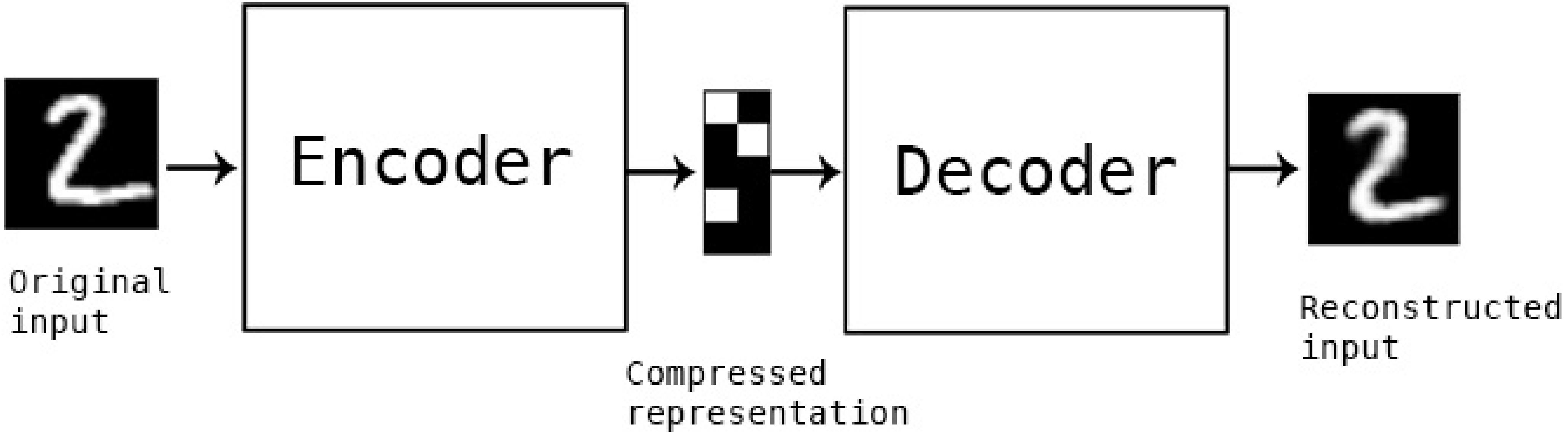
Topics in NLP : Word Embeddings, Sequence models, Attention and Dialogue Systems

Suresh Manandhar
Naamii, Nepal

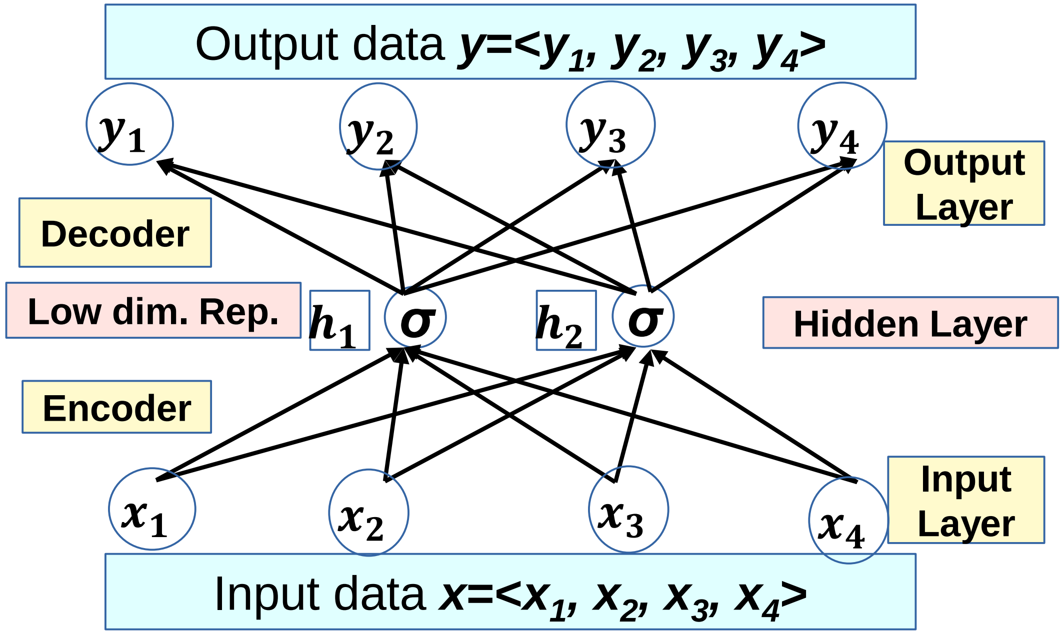
Autoencoders



- Source https://blog.keras.io/building-autoencoders-in-keras.html

Autoencoder

- *Autoencoder* is a generic term used to describe a class of methods for generating low dimensional representations using neural networks.
- There are a large variety of autoencoders (including those based on more complex neural networks such as CNNs, LSTMs, GANs, VAEs etc.)
- The basic structure of an autoencoder is depicted here



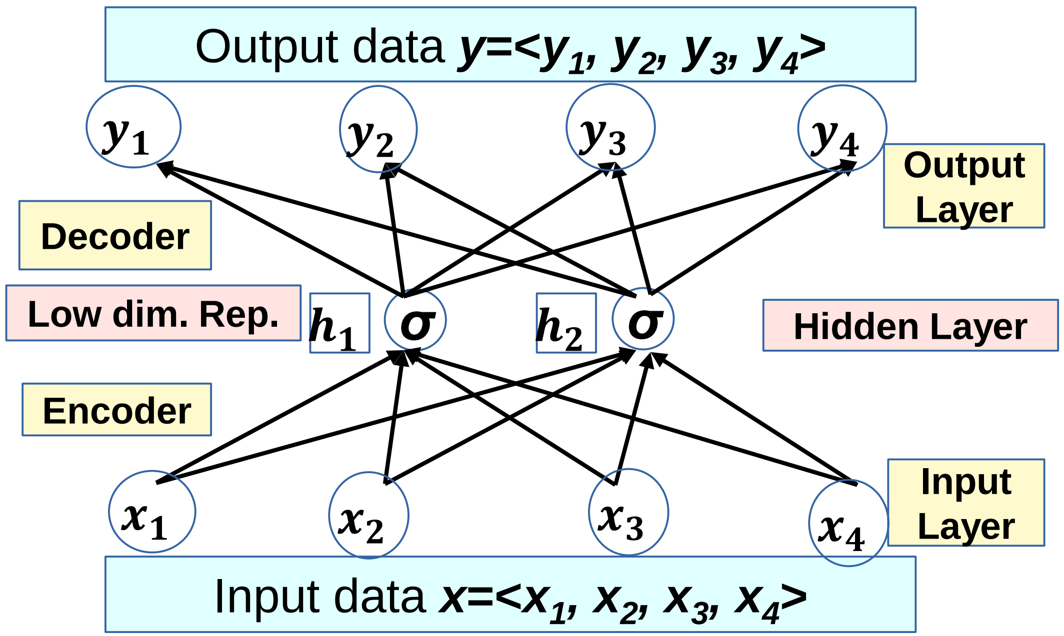
- Common to all autoencoders is that the loss is measured in terms of a **reconstruction error.** In this example, one can use (per example) *squared loss*:

$$L(\theta) = ||y - x||^2 = ||f(x,\theta) - x||^2$$

where ***f(x, θ)*** is the neural network

- The hidden layer, in this example, given by ***h* = <*h*$_1$, *h*$_2$>** **outputs** the ***low dimensional representation***.
- If no sigmoid units are used, then the **linear** low dim. representation will be similar to that given by PCA (without the orthogonality requirement)

Autoencoder



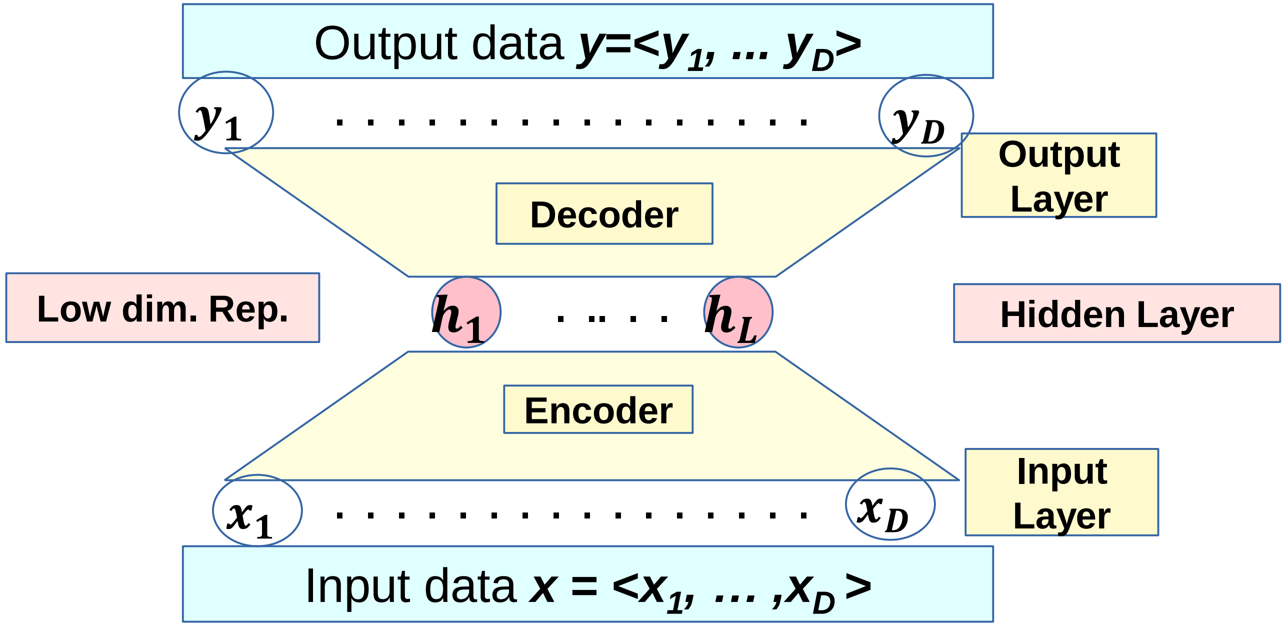
$$L(\theta) = \|y - x\|^2 = \|f(x,\theta) - x\|^2$$
$$f(x,\theta) = y = \langle y_1, y_2, y_3, y_4 \rangle = hW_2$$
$$h = \langle h_1, h_2 \rangle = \sigma(xW_1)$$
$$\theta = \langle W_1, W_2 \rangle$$
$$f(x,\theta) = \sigma(xW_1)W_2$$

where $\boldsymbol{W_1}$ is 4x2 dim. And $\boldsymbol{W_2}$ is 2x4 dim.

- Partial differentiation *wrt* $\boldsymbol{W_1}$ *and* $\boldsymbol{W_2}$ gives the gradient needed for gradient descent algorithm

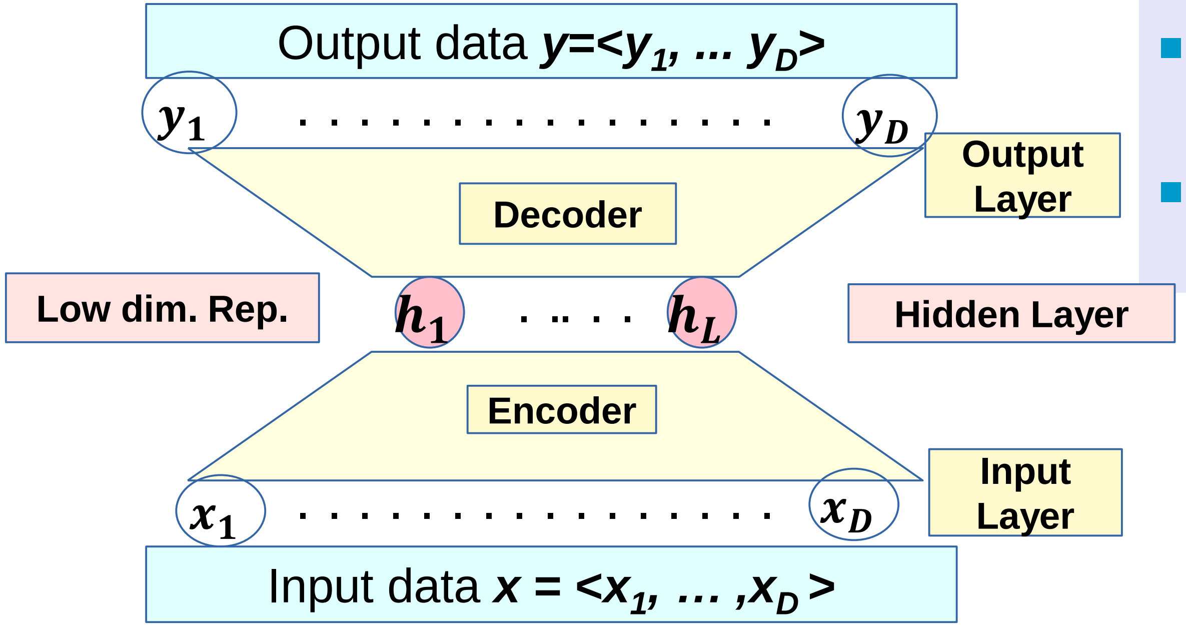
- *Autoencoder* consists of an **encoder** and **decoder**
- The **encoder** transforms the input into a (hidden) low dimensional representation
- The **decoder** attempts to transform the hidden representation back into the input
- The **reconstruction loss** measures the discrepancy between the reconstructed input vs. the original input

Autoencoder general setup



- In general, both encoder/decoder can be any suitably complex function, typically implemented as a neural network
- For image processing applications, the encoder/decoder are typically implemented as CNNs (convolutional neural networks)
- For sequential data (e.g. natural language, speech), the encoder/decoder are typically implemented as LSTMs (long short term memory)
- Many variations exists

Denoising Autoencoder



- In a denoising autoencoder the input is corrupted by adding Gaussian noise.
- The decoder is trained to reconstruct the original uncorrupted input.

- The denoising autoencoder finds applications in image/video restoration (e.g WW1 movies)
- It can also be used to make neural networks more robust to noisy input data

Distributional Semantics

- **Distributional Semantics: (Harris, 1954)**

  Words that occur in similar contexts tend to have similar meanings i.e. the meaning of a word can be defined in terms of its context.

- **Word Space Model (aka Vector Space Model):**

  Meaning of a word can be represented as a co-occurrence vector built from a corpus

Distributional Semantics

- **Example:**

  Freddy is planning to buy a **house** near the city centre

  All the students are having a party in Freddy's **house** in the city centre

	planning	buy	near	city	centre	Freddy	party	having
house	1	1	1	2	2	1	1	1

Distributional Semantics

Distributional Hypothesis (Harris, 1954)

Words that occur in similar contexts tend to have similar meanings i.e. meaning of a word can be defined in terms of its context.

Word Space Model (WSM)

Meaning of a word is represented as a co-occurrence vector built from a corpus

[I went to buy an] apartment [but the price was high] **(5 word context)**

	vector dimensions					
	animal	buy	apartment	price	rent	kill
House	⟨ 30	60	90	55	45	10 ⟩
Hunting	⟨ 90	15	12	20	33	90 ⟩

Instead of using counts we can use other measures

- **Conditional probability**

$$\boldsymbol{p(y|x) = \frac{p(y,x)}{p(x)} = \frac{\#(y,x)}{N}\frac{N}{\#(x)} = \frac{\#(y,x)}{\#(x)}}$$

- Conditional probability gives a measure of directional/asymmetric association
- For window based VSMs, frequent words will have a detrimental effect i.e. if $\boldsymbol{y}$ is frequent
- **Pointwise mutual information** (**PMI**) is a symmetric measure

$$\boldsymbol{pmi(x,y) = \log\left(\frac{p(x,y)}{p(x)p(y)}\right) = log\left(\frac{\#(x,y)}{N}\frac{N}{\#(x)}\frac{N}{\#(y)}\right) = log\left(\frac{\#(x,y)}{\#(x)\#(y)}N\right)}$$

- Insensitive to frequent words but can give negative values
- **Positively shifted PMI** (**PPMI**) gives smoothed positive values:

$$\boldsymbol{ppmi(x,y) = \log\left(1 + \frac{p(x,y)}{p(x)p(y)}\right)}$$

Vector Space Model (VSM) for words

So, we could represent the meaning of a word as a very long vector:

Vocabulary = < animal, buy, apartment, price, rent, kill, …. >

(all words in dict)

House = < 0.1, 0.2, 0.3, 0.16, ….. >

Hunting = < 0.3, 0.07, 0.05, 0.02, ….. >

Apartment = ??

Vector Space Model (VSM) for words

So, we could represent the meaning of a word as a very long vector:

Vocabulary = < animal, buy, apartment, price, rent, kill, …. >

(all words in dict)

House = < 0.1, 0.2, 0.3, 0.16, ….. >

Hunting = < 0.3, 0.07, 0.05, 0.02, ….. >

Which one is more likely?

Apartment = < 0.1, 0.18, 0.32, 0.10, ….. > **---- 1**

Apartment = < 0.31, 0.1, 0.07, 0.05, ….. > **---- 2**

Vector Space Model (VSM) for words

So, we could represent the meaning of a word as a very long vector:

Vocabulary = < animal, buy, apartment, price, rent, kill, …. >

(all words in dict)

House = < 0.1, 0.2, 0.3, 0.16, ….. >

Hunting = < 0.3, 0.07, 0.05, 0.02, ….. >

Given the distributional hypothesis we expect that it is more likely:

Apartment = < 0.1, 0.18, 0.32, 0.10, ….. > **---- 1**

VSM as a meaning representation in vector space

- The VSM is an explicit representation that is high dimensional (~ vocabulary size > 30,000)
- It is also very sparse (with most entries 0). **Why**?

Computing Similarity in meaning between two words

- VSMs can recover the similarity in meaning between words e.g. using cosine similarity or KL/JS divergence

- Thus, we expect $\boldsymbol{cos(book, novel)}$ to be high

$$\boldsymbol{cos(A, B) = \frac{A.B}{|A||B|}}$$

Issues with VSMs

- However, VSMs suffer from sparsity issues and generalise poorly
- **Why**?

Issues with VSMs

- However, VSMs suffer from sparsity issues and generalise poorly
- **Why**?
- To fill all the elements of a high dimensional vector, you will need a huge amount of data.
- So, using VSMs is not an ideal solution.
- **What would be a better solution?**

Issues with VSMs

- However, VSMs suffer from sparsity issues and generalise poorly
- **Why**?
- To fill all the elements of a high dimensional vector, you will need a huge amount of data.
- So, using VSMs is not an ideal solution.
- **What would be a better solution?**
- Ideally would want a lower dimensional representation
- that generalises better (i.e. can work with smaller datasets)

Word/Sentence Embeddings – General ideas

Creating training data using distributional semantics

We can set the problem of learning word/sentence meanings as a machine learning task that **requires some semantic interpretation**:

- **The dog ____ the cat (fill in the blank)**

Word/Sentence Embeddings – General ideas

Creating training data using distributional semantics

We can set the problem of learning word/sentence meanings as a machine learning task that **requires some semantic interpretation**:

- **The dog ___ the cat** **(fill in the blank)**
- **I went to the party wearing a nice ____** **(predict the next word)**

Word/Sentence Embeddings – General ideas

Creating training data using distributional semantics

We can set the problem of learning word/sentence meanings as a machine learning task that **requires some semantic interpretation**:

- **The dog ___ the cat** **(fill in the blank)**
- **I went to the party wearing a nice _____** **(predict the next word)**
- **My neighbours have a dog that is quite scary** **(predict left/right context word)**

Word/Sentence Embeddings – General ideas

Creating training data using distributional semantics

We can set the problem of learning word/sentence meanings as a machine learning task that **requires some semantic interpretation**:

- **The dog ___ the cat** **(fill in the blank)**
- **I went to the party wearing a nice ____** **(predict the next word)**
- **My neighbours have a dog that is quite scary** **(predict left/right context word)**
- **I heated the food → The food got hot** **(entails/contradicts/unrelated)**

Word/Sentence Embeddings – General ideas

Creating training data using distributional semantics

- We can set the problem of learning word/sentence meanings as a machine learning task that requires some semantic interpretation:
  - The dog ___ the cat **(fill in the blank)**
- For each of the tasks we can generate a training dataset containing the correct and incorrect predictions.
- For example, for the **fill in the blank** task we can create training data:
  - [the, dog] [the, cat] → chases **(+ example)** should give class ***1***
  - [the, dog] [the, cat] → bites **(+ example)** should give class ***1***
  - [the, dog] [the, cat] → buy **(- example)** should give class ***0***
- Think of the → as a machine learning model that we train using this data

Classwork – create training data

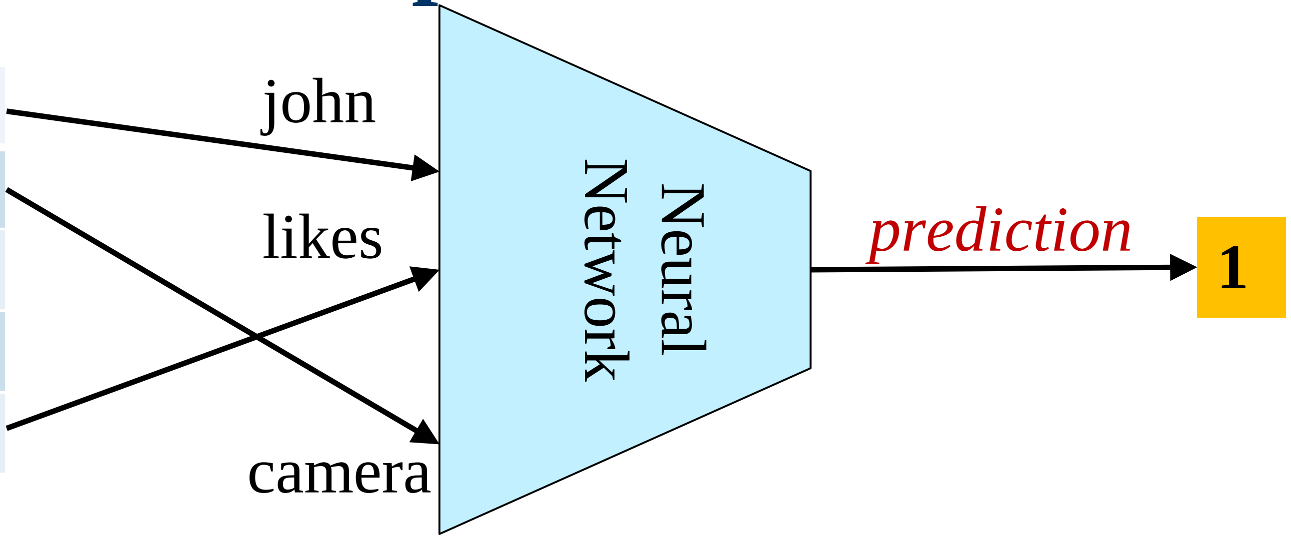
- For the left/right context prediction task:

  My neighbours have a dog that is quite scary (predict left/right context word)

- Create the training data:

Word Embeddings – The setup

john	0.0012	0.0025	….
camera	0.20	…	
……	…		
……			
likes	0.01	0.001	



- Transfer learning using pre-trained embeddings (e.g. word2vec, GloVe)
- Domain specific learning
- Combination

Word2vec

- word2vec is a very popular word embedding learning toolkit
- It can generate several different variants of embeddings depending upon the settings

Skip-gram Embeddings

- Trained to learn the context word prediction task:
  - {big, the, fat, my } dog { like, chases, bites, eats} **(predict left/right context word)**
- Let the training data $\boldsymbol{D} = \{< \boldsymbol{w}, \boldsymbol{c}, \boldsymbol{c_N} >\}_{\boldsymbol{1}}^{|\boldsymbol{D}|}$ where
  - $\boldsymbol{w}$ is the target word
  - $\boldsymbol{c}$ is a context work
  - $\boldsymbol{c_N}$ is a list of negative context words typically sampled randomly
- The context words can be arbitrary e.g. words within a window, words connected by a parse tree
- Each word $\boldsymbol{w}$ is associated with two embeddings – **word embeddings $\overline{\boldsymbol{w}}$**, and its **context embedding $\overline{\boldsymbol{w_c}}$**
- Similarly, each context $\boldsymbol{c}$ is associated with two embeddings – word embeddings $\overline{\boldsymbol{c_w}}$, and its context embedding $\overline{\boldsymbol{C}}$

Skip-gram Embeddings

- The per training example likelihood becomes:

$$p(w, c, C_N) = p(< w, c >) \prod_{c_i \in C_N} \left(1 - p(< w, c_i >)\right)$$

- Per example, log likelihood can be written as:

$$\begin{aligned} log\left(p(w, c, C_N)\right) &= log\left(p(w, c)\right) + \sum_{c_i \in C_N} log\left(\left(1 - p(< w, c_i >)\right)\right) \\ &= log\left(\sigma(\bar{w}.\bar{c})\right) + \sum_{c_i \in C_N} log\left(\left(1 - \sigma(\bar{w}.\bar{c}_i)\right)\right) \\ &= log\left(\sigma(\bar{w}.\bar{c})\right) + \sum_{c_i \in C_N} log\left(\sigma(-\bar{w}.\bar{c}_i)\right) \end{aligned}$$

- [Aside] Derivative of the log of sigmoid:

$$\frac{\delta}{\delta x} log(\sigma(x)) = \frac{\delta}{\delta x} log\left(\frac{1}{1+e^{-x}}\right) = \frac{\delta}{\delta x}[log(1) - log(1+e^{-x})]$$

$$= -\left(\frac{1}{1+e^{-x}}\right)(-e^{-x}) = \frac{e^{-x}}{1+e^{-x}} = \sigma(-x) = 1 - \sigma(x)$$

- Derivative with respect to the word embedding/vector:

$$\frac{\delta}{\delta \overline{w}} log(p(w, c, C_N)) = \frac{\delta}{\delta w}\left[log(\sigma(\overline{w}.\overline{c})) + \sum_{c_i \in C_N} log(\sigma(-\overline{w}.\overline{c_i}))\right]$$

$$= \sigma(-\overline{w}.\overline{c})(\overline{c}) + \sum_{c_i \in C_N} \sigma(\overline{w}.\overline{c_i})(-\overline{c_i})$$

$$= \sigma(-\overline{w}.\overline{c})\overline{c} - \sum_{c_i \in C_N} \sigma(\overline{w}.\overline{c_i})\overline{c_i}$$

- Derivative with respect to the context embedding/vector:

$$\frac{\delta}{\delta\bar{c}} log\big(p(w,c,C_N)\big) = \frac{\delta}{\delta w}\left[log\big(\sigma(\bar{w}.\bar{c})\big) + \sum_{c_i \in C_N} log\big(\sigma(-\bar{w}.\bar{c_i})\big)\right]$$

$$= \sigma(-\bar{w}.\bar{c})(\bar{w})$$

- Derivative with respect to the context embedding/vector for the negative sample:

$$\frac{\delta}{\delta\bar{c_i}} log\big(p(w,c,C_N)\big) = \frac{\delta}{\delta w}\left[log\big(\sigma(\bar{w}.\bar{c})\big) + \sum_{c_i \in C_N} log\big(\sigma(-\bar{w}.\bar{c_i})\big)\right]$$

$$= \sigma(\bar{w}.\bar{c_i})(-\bar{w})$$

Stochastic gradient descent algorithm

- For each input word $\boldsymbol{w}$
- Sample $\boldsymbol{k}$ negative contexts $\boldsymbol{C_N}$ (e.g. sample from ***top k*** most frequent words)
- Repeat for each context word $\boldsymbol{c}$ of $\boldsymbol{w}$:

$$\bar{\boldsymbol{w}} \coloneqq \bar{\boldsymbol{w}} + \boldsymbol{\eta}\left(\boldsymbol{\sigma}(-\bar{\boldsymbol{w}}.\bar{\boldsymbol{c}})\bar{\boldsymbol{c}} - \sum_{\boldsymbol{c_i} \in \boldsymbol{C_N}} \boldsymbol{\sigma}(\bar{\boldsymbol{w}}.\bar{\boldsymbol{c_i}})\bar{\boldsymbol{c_i}}\right)$$

$$\bar{\boldsymbol{c}} \coloneqq \bar{\boldsymbol{c}} + \boldsymbol{\eta}\big(\boldsymbol{\sigma}(-\bar{\boldsymbol{w}}.\bar{\boldsymbol{c}})(\bar{\boldsymbol{w}})\big)$$

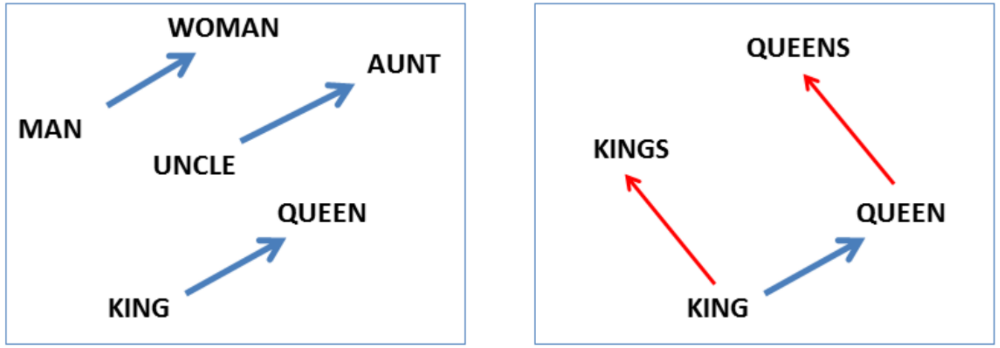
For each $\boldsymbol{c_i} \in \boldsymbol{C_N}$:

$$\bar{\boldsymbol{c_i}} \coloneqq \bar{\boldsymbol{c_i}} - \boldsymbol{\eta}\big(\boldsymbol{\sigma}(\bar{\boldsymbol{w}}.\bar{\boldsymbol{c_i}})(\bar{\boldsymbol{w}})\big)$$

- Move pointer to the next word
- Stop after a fixed number of iterations

Analogy tasks

- Analogy between words:
  - woman – man ≈ queen – king
  - king – man + woman ≈ queen



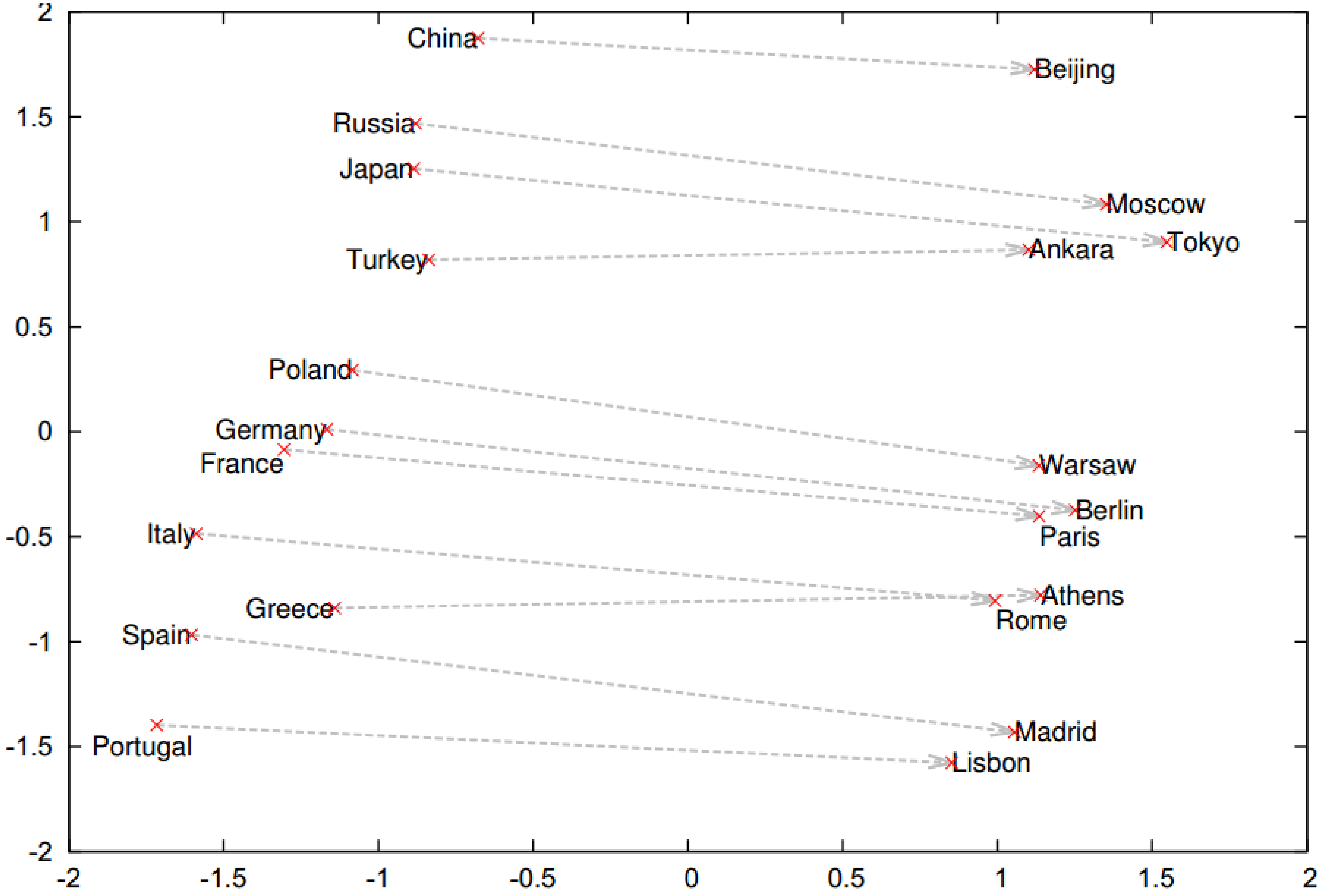
- England – London + Baghdad = ? Iraq
- Equivalently:

$$\arg\max_{B'} \cos(B', \text{England} - \text{London} + \text{Baghdad})$$

- Directional similarity

Slide from Omer Levy

Directional similarity example



Levy and Goldberg's interpretation

- Let $\boldsymbol{D} = \{(\boldsymbol{w}, \boldsymbol{c})\}_{\mathbf{1}}^{\boldsymbol{N}}$ (with $\boldsymbol{w} \in \boldsymbol{W}$ as word vocabulary, and $\boldsymbol{C}$ as context vocabulary) denote the input data
- Let: $\#(\boldsymbol{w}, \boldsymbol{c})$ denote the number of times the pair $(\boldsymbol{w}, \boldsymbol{c})$ appears in $\boldsymbol{D}$
- Let: $\#(\boldsymbol{w}) = \sum_{\boldsymbol{c} \in \boldsymbol{C}} \#(\boldsymbol{w}, \boldsymbol{c})$ is the number of times $\boldsymbol{w}$ appears in $\boldsymbol{D}$
- Similarly: $\#(\boldsymbol{c}) = \sum_{\boldsymbol{w} \in \boldsymbol{W}} \#(\boldsymbol{w}, \boldsymbol{c})$ is the number of times $\boldsymbol{c}$ appears in $\boldsymbol{D}$

$$\boldsymbol{log}\big(\boldsymbol{p}(\boldsymbol{w}, \boldsymbol{c})\big) = \boldsymbol{log}\big(\boldsymbol{\sigma}(\overline{\boldsymbol{w}}.\overline{\boldsymbol{c}})\big) + \sum_{\boldsymbol{c_i} \in \boldsymbol{C_N}} \boldsymbol{log}\big(\boldsymbol{\sigma}(-\overline{\boldsymbol{w}}.\overline{\boldsymbol{c_i}})\big)$$

$$= \boldsymbol{log}\big(\boldsymbol{\sigma}(\overline{\boldsymbol{w}}.\overline{\boldsymbol{c}})\big) + \boldsymbol{k}\ \mathbf{E}_{\boldsymbol{c_i} \sim \boldsymbol{D}}\big[\boldsymbol{log}\big(\boldsymbol{\sigma}(-\overline{\boldsymbol{w}}.\overline{\boldsymbol{c_i}})\big)\big]$$

- Log likelihood of the data $\boldsymbol{D}$ is then given by:

$$\boldsymbol{log}\big(\boldsymbol{p}(\boldsymbol{D})\big) = \sum_{\boldsymbol{w} \in \boldsymbol{W}} \sum_{\boldsymbol{c} \in \boldsymbol{C}} \#(\boldsymbol{w}, \boldsymbol{c}) \boldsymbol{log}\big(\boldsymbol{\sigma}(\overline{\boldsymbol{w}}.\overline{\boldsymbol{c}})\big) + \boldsymbol{k} \sum_{\boldsymbol{w} \in \boldsymbol{W}} \sum_{\boldsymbol{c} \in \boldsymbol{C}} \#(\boldsymbol{w}, \boldsymbol{c})\ \mathbf{E}_{\boldsymbol{c_i} \sim \boldsymbol{D}}\big[\boldsymbol{log}\big(\boldsymbol{\sigma}(-\overline{\boldsymbol{w}}.\overline{\boldsymbol{c_i}})\big)\big]$$

$$= \sum_{\boldsymbol{w} \in \boldsymbol{W}} \sum_{\boldsymbol{c} \in \boldsymbol{C}} \#(\boldsymbol{w}, \boldsymbol{c}) \boldsymbol{log}\big(\boldsymbol{\sigma}(\overline{\boldsymbol{w}}.\overline{\boldsymbol{c}})\big) + \boldsymbol{k} \sum_{\boldsymbol{w} \in \boldsymbol{W}} \#(\boldsymbol{w}) \mathbf{E}_{\boldsymbol{c_i} \sim \boldsymbol{D}}\big[\boldsymbol{log}\big(\boldsymbol{\sigma}(-\overline{\boldsymbol{w}}.\overline{\boldsymbol{c_i}})\big)\big]$$

- The expectation term can be expanded as follows:

$$\mathbf{E}_{c_i \sim D}\left[log\left(\sigma(-\bar{w}.\bar{c_i})\right)\right] = \sum_{c_i \in C} \frac{\#(c_i)}{|D|} log\left(\sigma(-\bar{w}.\bar{c_i})\right)$$

- Log likelihood of the data is then given by:

$$\begin{aligned} log\left(p(D)\right) &= \sum_{w \in W} \sum_{c \in C} \#(w,c) log\left(\sigma(\bar{w}.\bar{c})\right) + k \sum_{w \in W} \sum_{c \in C} \#(w,c) \mathbf{E}_{c_i \sim D}\left[log\left(\sigma(-\bar{w}.\bar{c_i})\right)\right] \\ &= \left[\sum_{w \in W} \sum_{c \in C} \#(w,c) log\left(\sigma(\bar{w}.\bar{c})\right)\right] + k \sum_{w \in W} \#(w) \left[\sum_{c_i \in C} \frac{\#(c_i)}{|D|} log\left(\sigma(-\bar{w}.\bar{c_i})\right)\right] \\ &= \sum_{w \in W} \sum_{c \in C} \left[\#(w,c) log\left(\sigma(\bar{w}.\bar{c})\right) + k\#(w) \frac{\#(c)}{|D|} log\left(\sigma(-\bar{w}.\bar{c})\right)\right] \end{aligned}$$

- Hence per example objective is:

$$log\left(p(w,c)\right) = \#(w,c) log\left(\sigma(\bar{w}.\bar{c})\right) + k\#(w) \frac{\#(c)}{|D|} log\left(\sigma(-\bar{w}.\bar{c})\right)$$

- Hence Letting $x = \bar{w}.\bar{c}$ and setting derivative:

$$\frac{\delta}{\delta x} log\left(p(w,c)\right) = 0$$

Levy and Goldberg’s derivation seems to be unnecessarily more complicated

- We have $\boldsymbol{x} = \ \overline{\boldsymbol{w}}.\overline{\boldsymbol{c}}$ and:

$$\frac{\delta}{\delta x} log\big(p(w,c)\big) = 0$$

$$\frac{\delta}{\delta x} log\big(p(w,c)\big) \ = \frac{\delta}{\delta x}\left[\#(w,c) log\big(\sigma(x)\big) + k\#(w)\frac{\#(c)}{|D|} log\big(\sigma(-x\,)\big)\right]$$

$$= \ \#(w,c)\sigma(-x) - k\#(w)\frac{\#(c)}{|D|}\sigma(x)$$

$$= \ \#(w,c)\left(\frac{1}{1+e^{x}}\right) - k\#(w)\frac{\#(c)}{|D|}\left(\frac{1}{1+e^{-x}}\right) = 0$$

- We have $\boldsymbol{x} = \ \overline{\boldsymbol{w}}.\overline{\boldsymbol{c}}$ and:

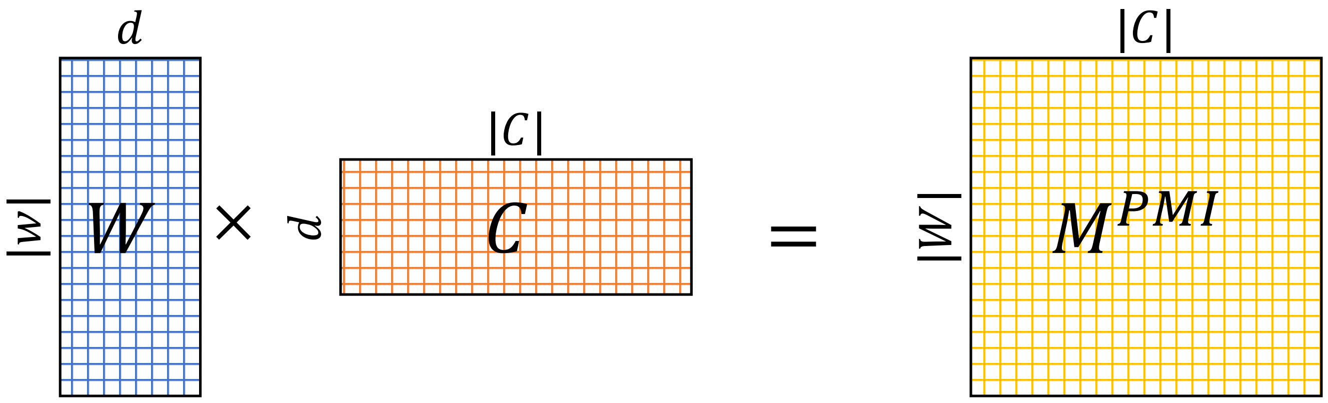
$$\#(w,c) - k\#(w)\frac{\#(c)}{|D|} e^{x} = 0$$

$$e^{x} = \frac{\#(w,c)\ \ |D|}{\#(w)\#(c)\ k}$$

$$x = \ \overline{w}.\overline{c} = log\left(\frac{\#(w,c)\ \ |D|}{\#(w)\#(c)}\right) - log(k) = pmi(w,c) - log(k)$$

Skipgram embeddings ≃ Matrix factorization

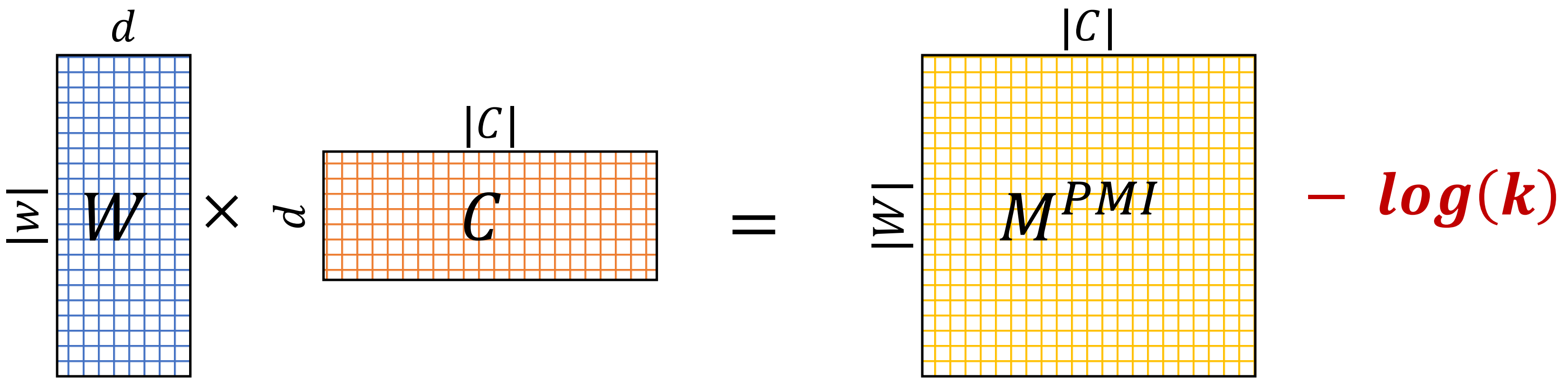
- The skipgram model learns a matrix factorization of the PMI matrix



Slide from Omer Levy

Skipgram embeddings ≃ Matrix factorization

- The skipgram model learns a matrix factorization of the PMI matrix *shifted by a global constant*



Slide from Omer Levy

Embeddings as latent features

- We can replace words with their corresponding embeddings.
- **But how can be encode a variable length sentence into a fixed length vector**

Computing Sentence Representations

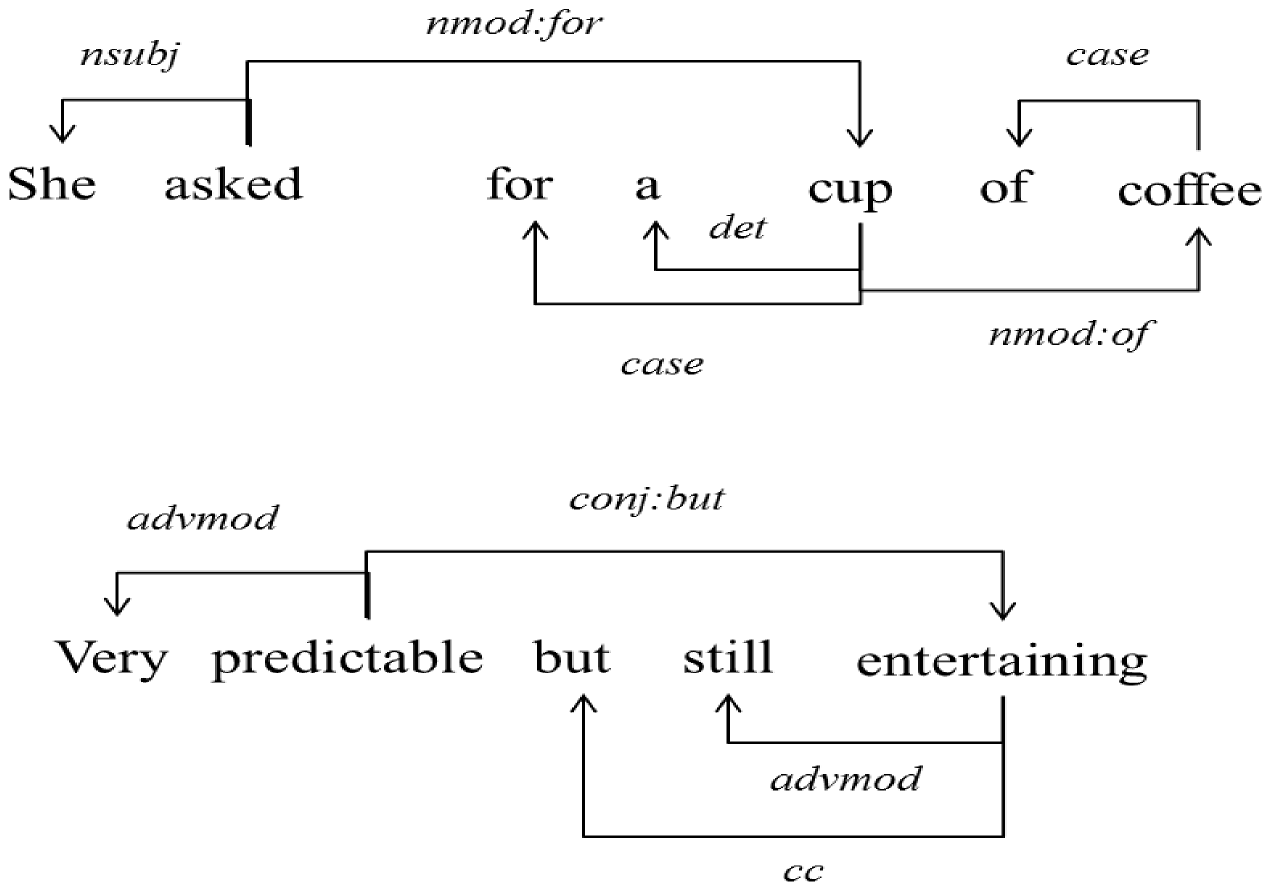
- There are multiple possibilities.
- Sum the vectors and compute average

$$\mathbf{s} = \begin{bmatrix} | & & | \\ \mathbf{w}_1 & \dots & \mathbf{w}_s \\ | & & | \end{bmatrix}$$

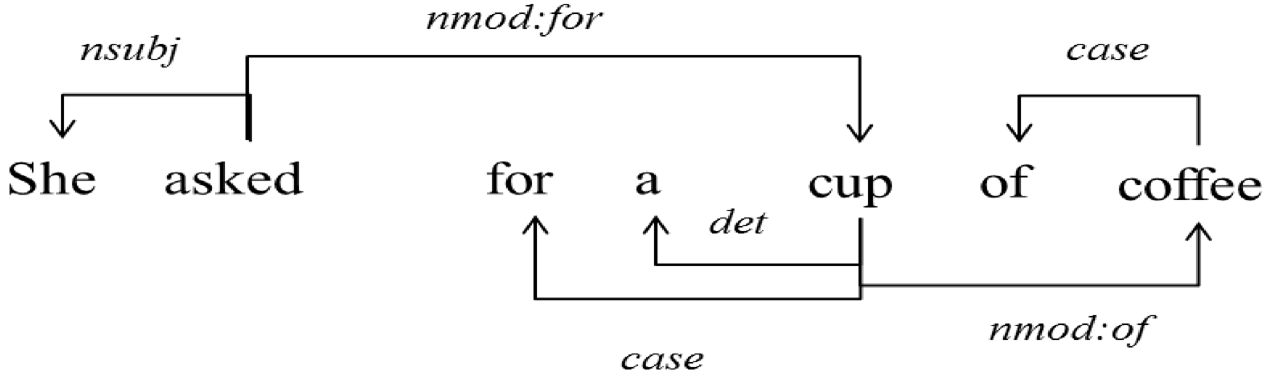
- Compute row wise max

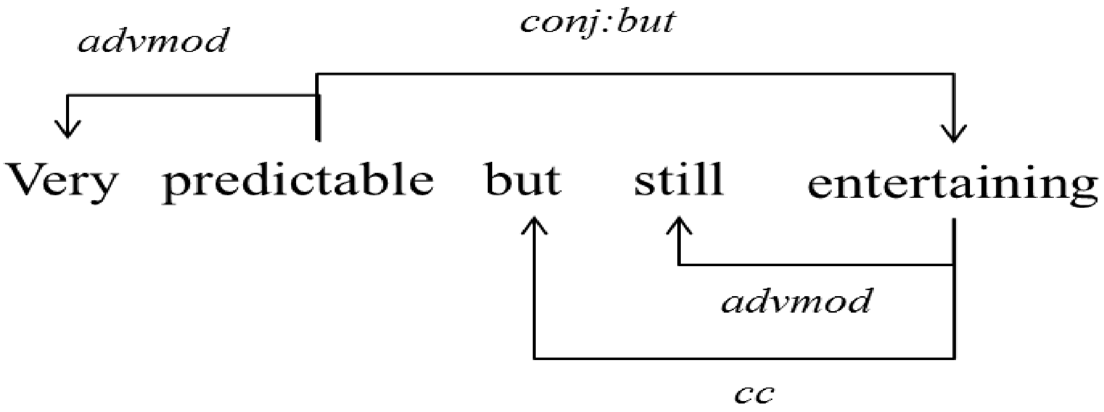
$$\mathbf{c}_{max} = \begin{bmatrix} \max(\mathbf{c}_{1,:}) \\ \vdots \\ \max(\mathbf{c}_{d,:}) \end{bmatrix}$$

Dependency Parsing based Word Embeddings



Dependency based word embeddings





Target - **cup**

Context words :
She, asked, for, a, of, coffee

Syntactic contexts (edges):
for:nmod^{-1}_asked, case_for, det_a, of:nmod_coffee

from [Komninos and Manandhar, 2016]

Sentence level classification tasks in NLP

- **Sentiment analysis** (positive, negative, neutral etc.)

Example: The food was fine but the décor was unimpressive

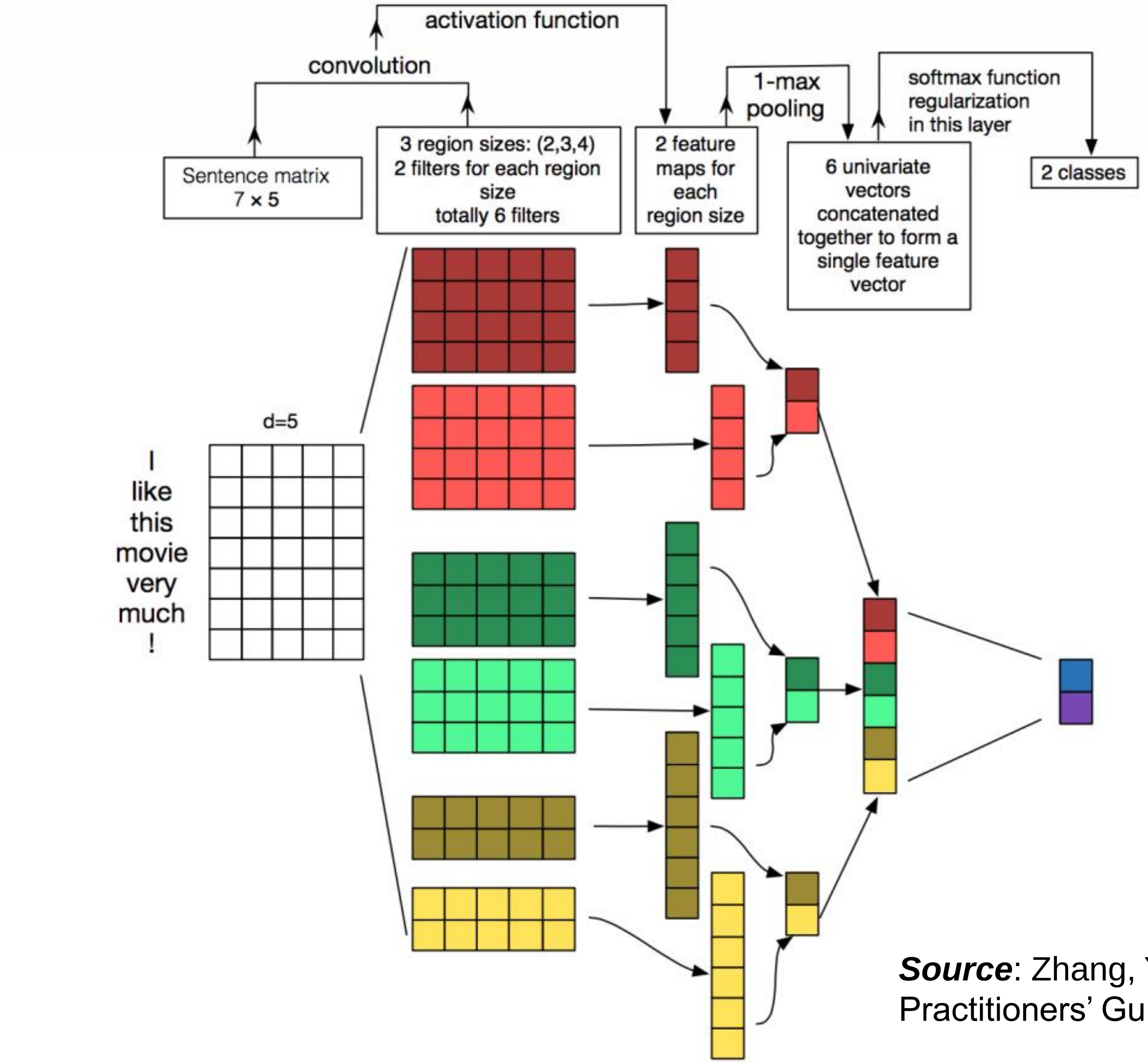
- **Subjectivity classification** (subjective vs objective)

Example: The match today was bit boring

- **Question type classification** (who i.e. person, where i.e. location, which restaurant → restaurant)

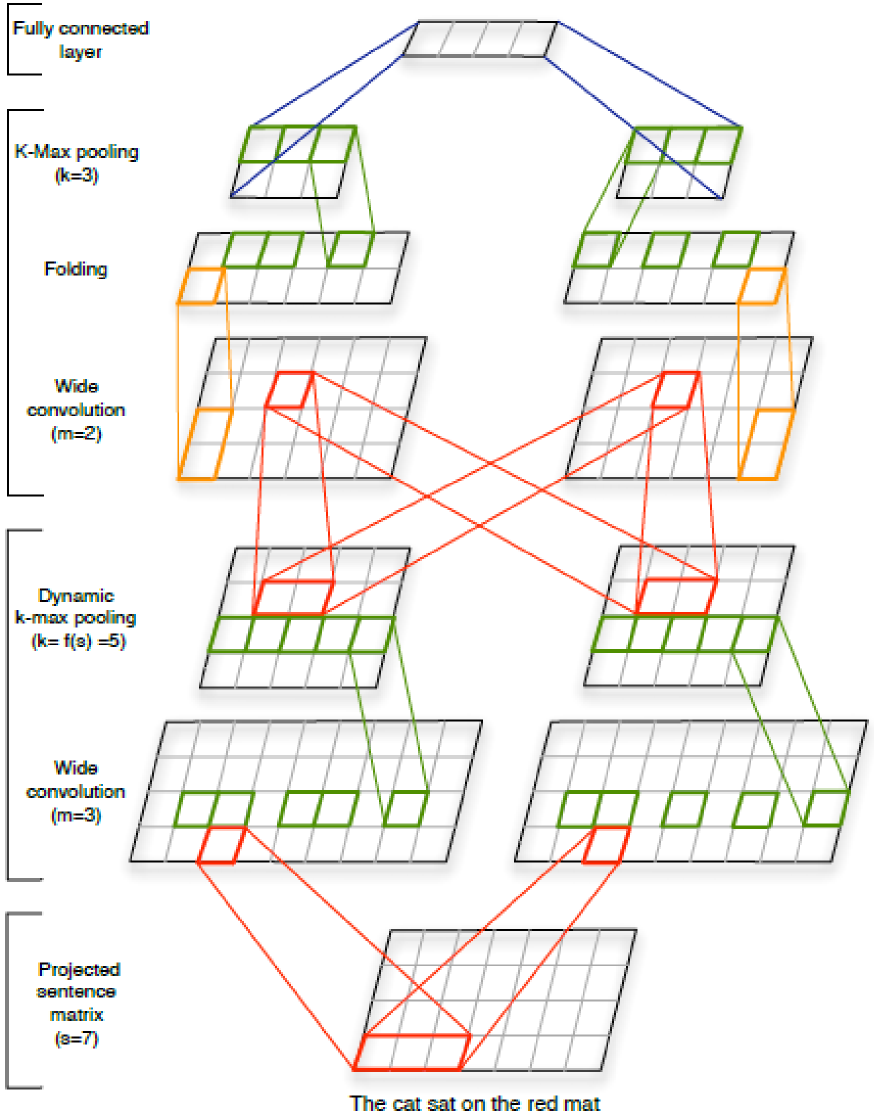
Example: Which hotel is near to the city centre?

CNN models for sentence classification



Source: Zhang, Y., & Wallace, B. (2015). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification.

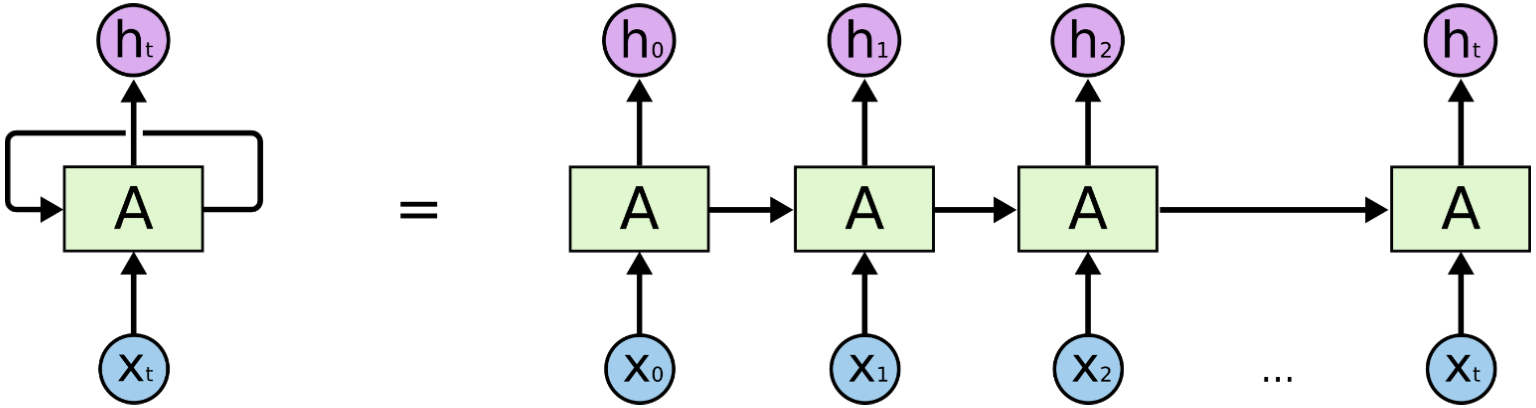
CNN models for sentence classification



Source: Kalchbrenner N., Grefenstette E., Blunsom P. A Convolutional Neural Network for Modelling Sentences.

Sequence models

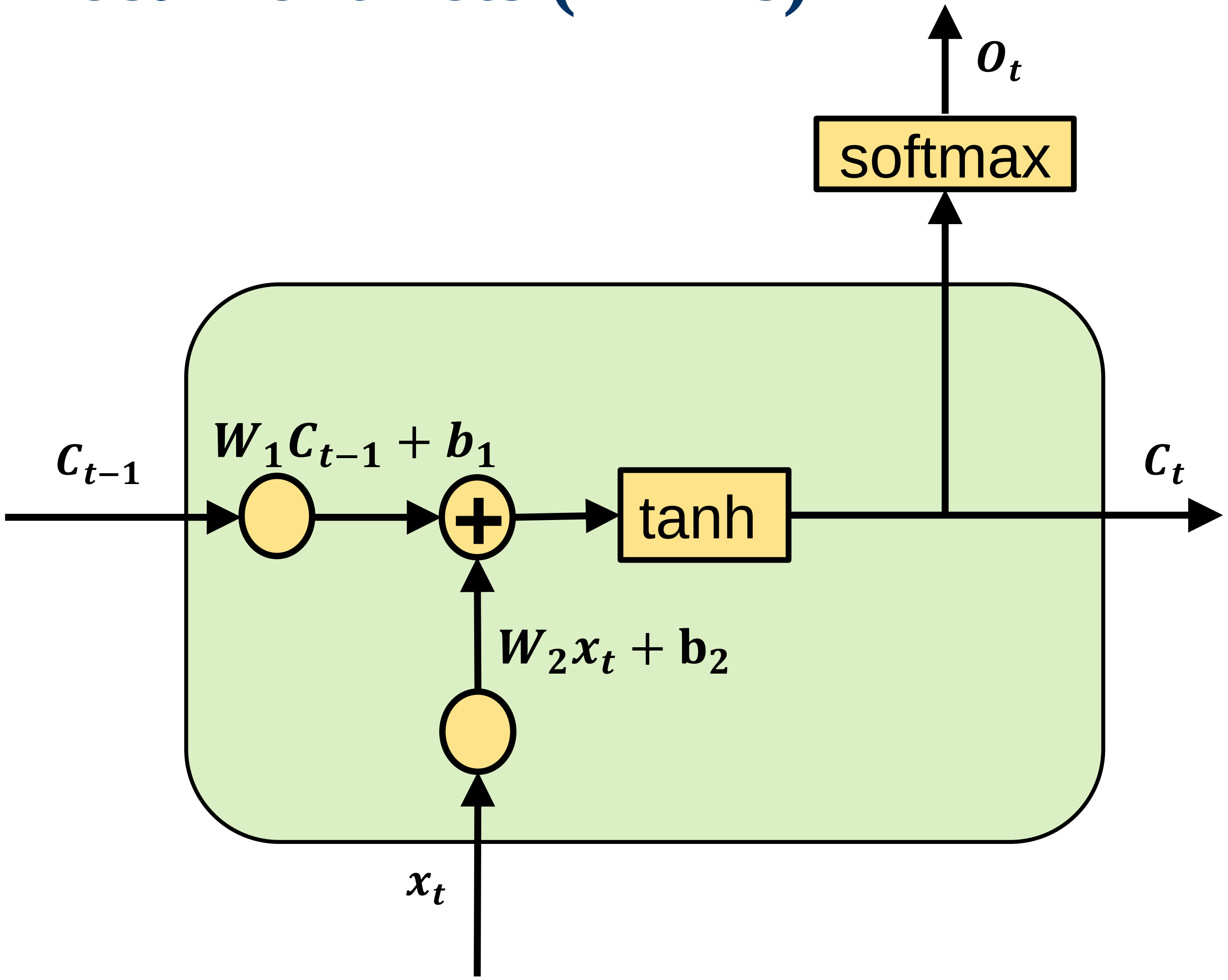
Recurrent nets (RNNs)



- Tremendously popular for many tasks
- Model of choice within NLP for sequence modelling tasks:
- **Shared parameter** (single cell)
- Cell is ***unrolled*** to feed a sequence input
- Each cell can remember some information
- Pass this to the next cell

Source: https://colah.github.io/posts/2015-08-Understanding-LSTMs

Recurrent nets (RNNs)



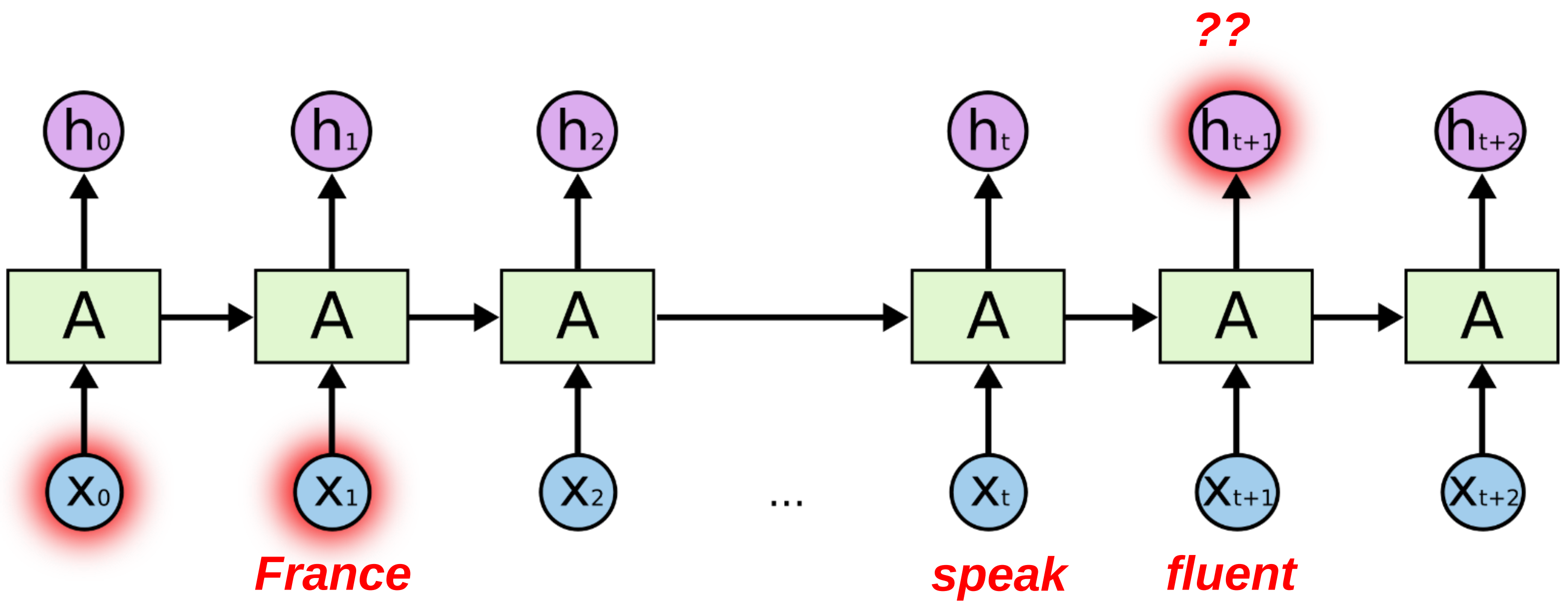
$$C_t = \ tanh(W_1C_{t-1} + W_2x_t)$$

$$O_t = softmax(C_t)$$

- The softmax unit is optional to generate the output O_t
- The hidden state is denoted by C_t
- The RNN is parameterised by two trainable weight parameters W_1 and W_2
- Additional parameter is the (no. of units) to control the dimensionality of C_t
- The other units are deterministic

- Good tutorial demo based on Coursera module is available at: https://github.com/omerbsezer/LSTM_RNN_Tutorials_with_Demo

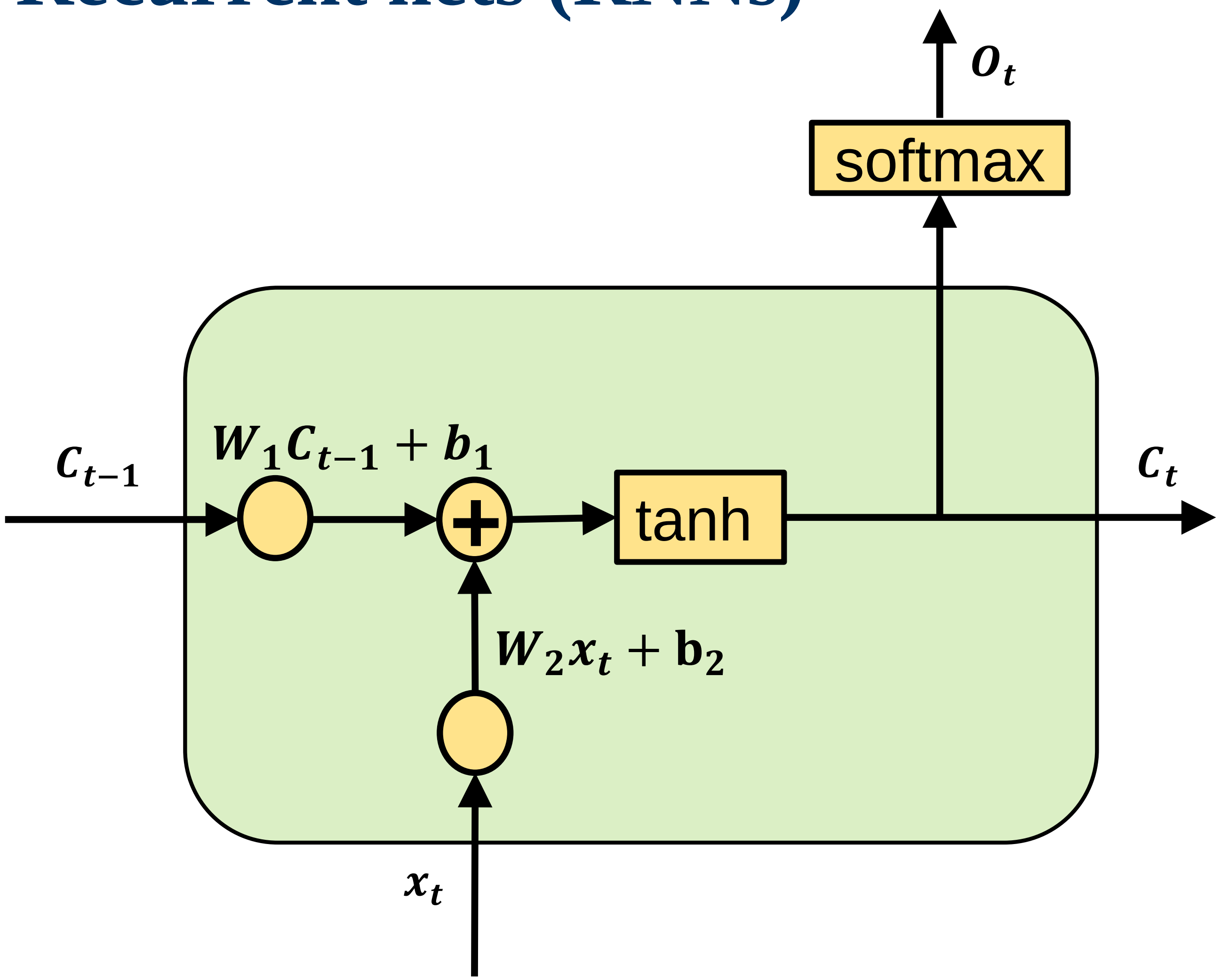
Issues with standard RNNs



- As the item to remember becomes too far
- Standard RNNs have problem keeping this information
- e.g. Language modelling problem *'I grew up in France, I speak fluent ...'*

Source: https://colah.github.io/posts/2015-08-Understanding-LSTMs

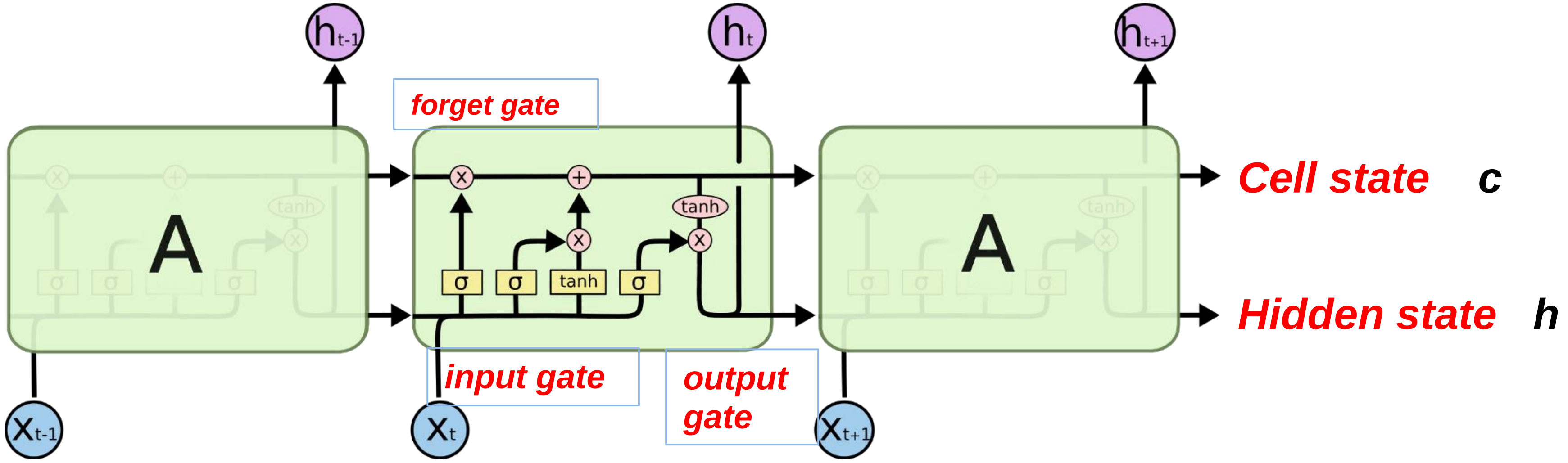
Recurrent nets (RNNs)



- Since the weight parameters W_1 is not dependent on the input x_t but only dependent upon previous cell state C_{t-1}
- the RNN has no robust mechanism for storing long range information that is sensitive to the input
- Simple RNNs also suffer from the vanishing/exploding gradient problem
  - use ReLU instead of tanh (to deal with vanishing gradient)
  - use gradient clipping (to deal with exploding gradient)

- See also: https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/

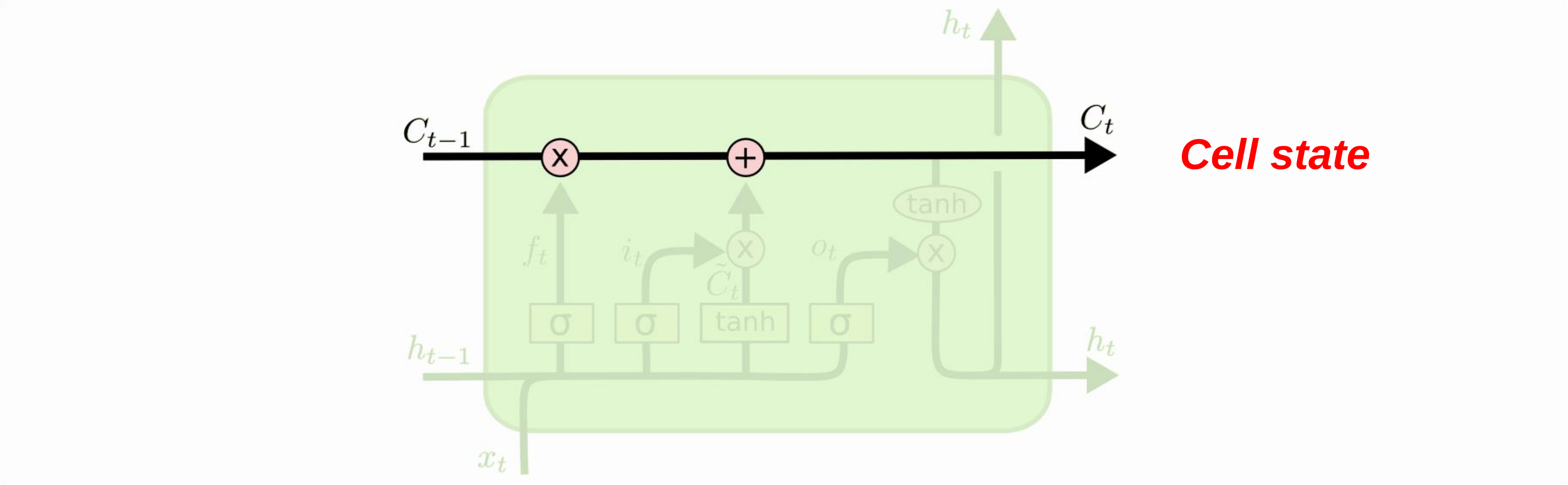
LSTM networks



- A LSTM consists of two internal states
  - **Cell state** (memory to carry forward)
  - **Hidden state** (current state to output)
- And a number of gates
  - **Input gate** (decides how much of previous cell state to carry forward)
  - **Forget gate** (how much of the current hidden state to mix with the previous cell state)
- **Output gate** (how much of the new cell state to output as the new hidden state)

Source: https://colah.github.io/posts/2015-08-Understanding-LSTMs

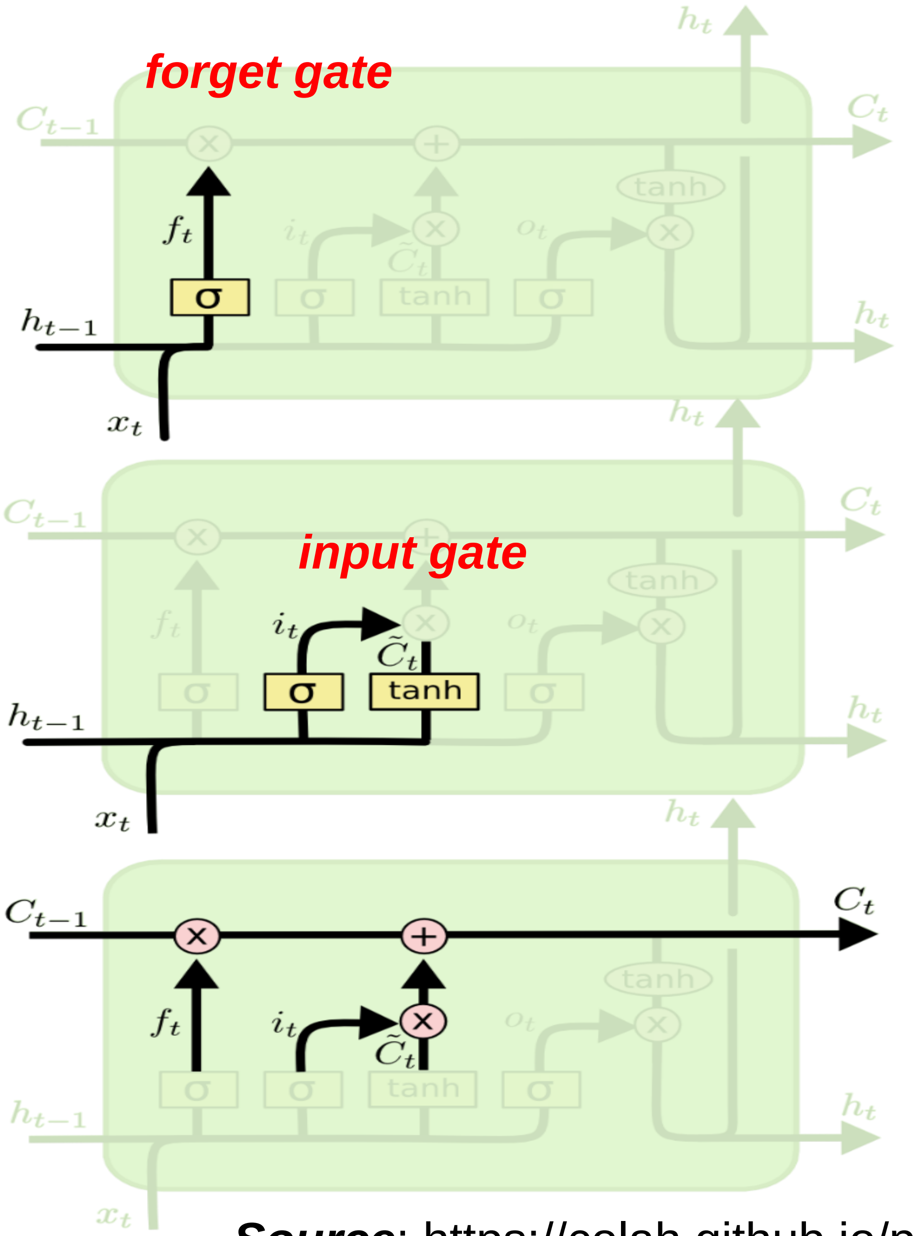
LSTM networks



Cell state

Source: https://colah.github.io/posts/2015-08-Understanding-LSTMs

LSTM networks



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

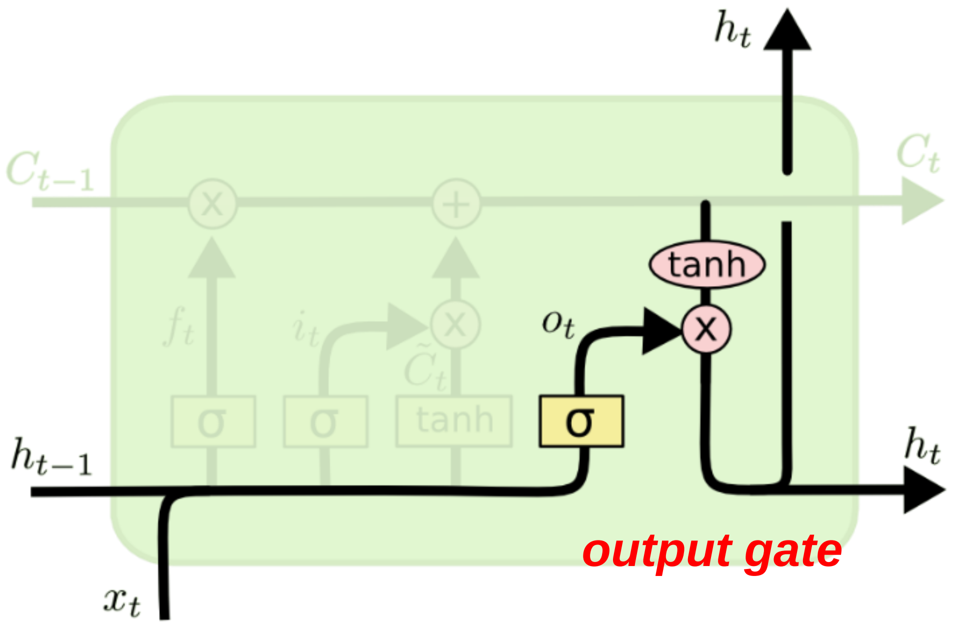
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Cell state ***c***

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

Source: https://colah.github.io/posts/2015-08-Understanding-LSTMs

LSTM networks



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

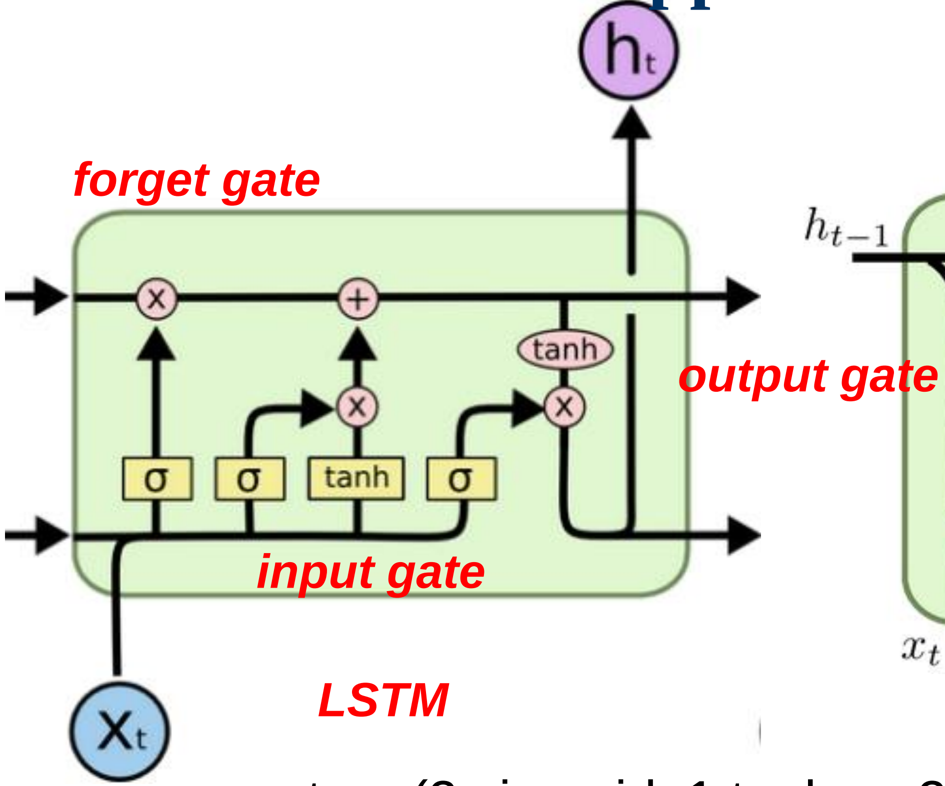
$$h_t = o_t * \tanh\left(C_t\right)$$

Hidden state ***h***

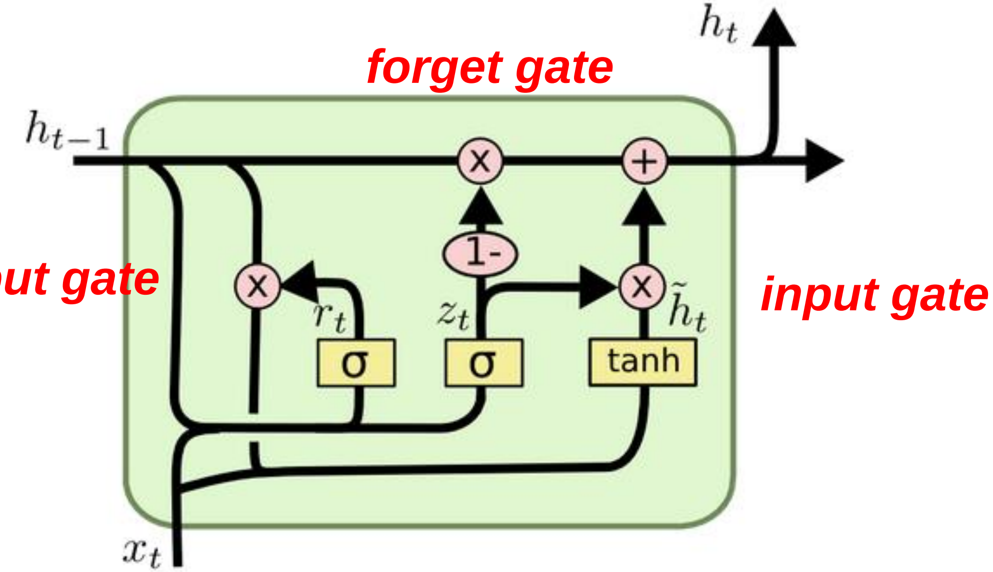
- A LSTM has more precise control of:
  - how much of previous memory (cell state) to keep
  - how much of previous hidden state + current input to store into memory (cell state)
  - how much of the new cell state and combined input + previous hidden state to output as the new hidden state

Source: https://colah.github.io/posts/2015-08-Understanding-LSTMs

Using LSTM networks in NLP applications



LSTM

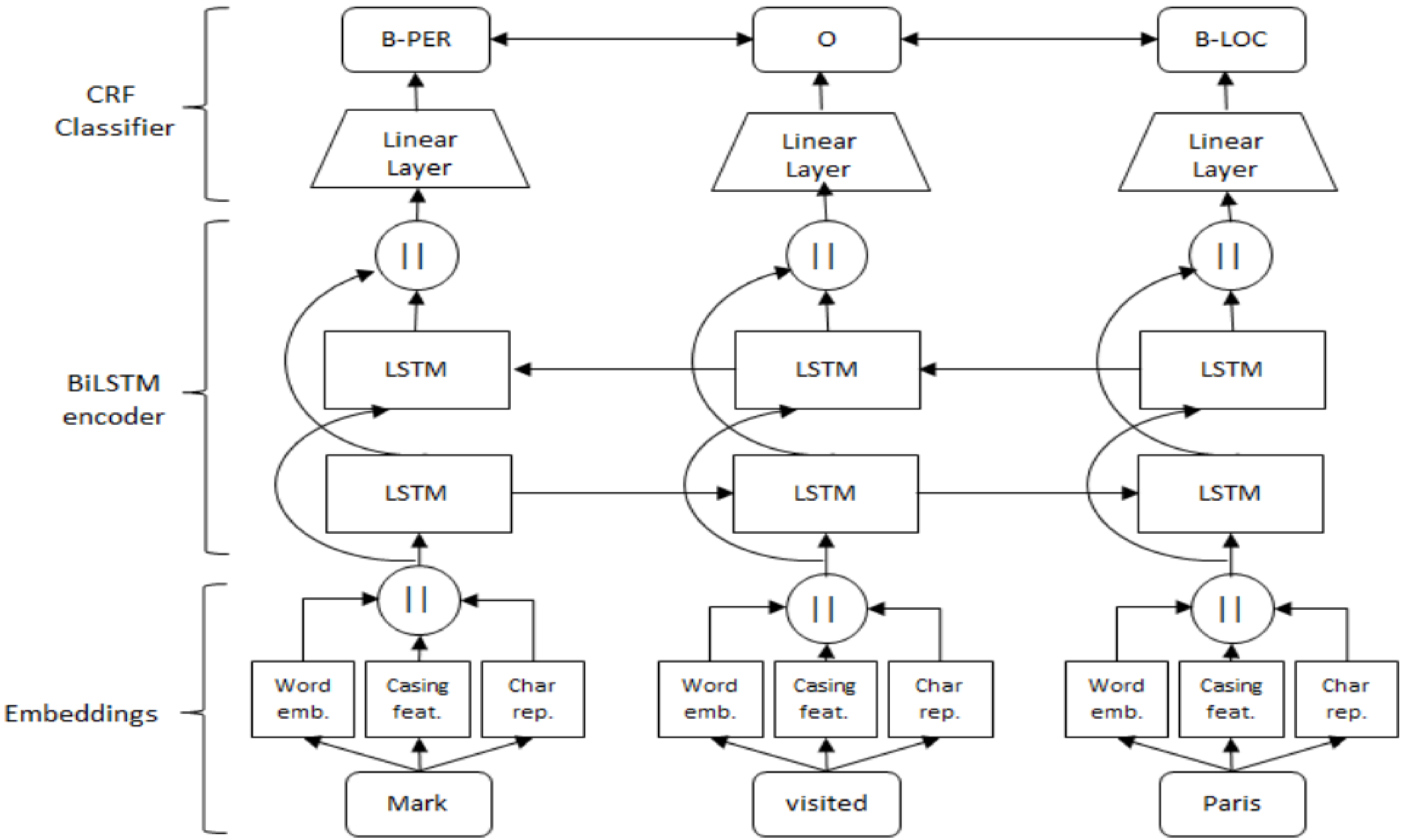


GRU

- A GRU has fewer parameters (2 sigmoid, 1 tanh vs 2 sig, 2 tanh):
  - Input gate is same as before
- *Amount to forget from previous hidden state* **=** *1 – amount to add from new hidden state*
- No cell state. Just hidden state
- Simpler equations

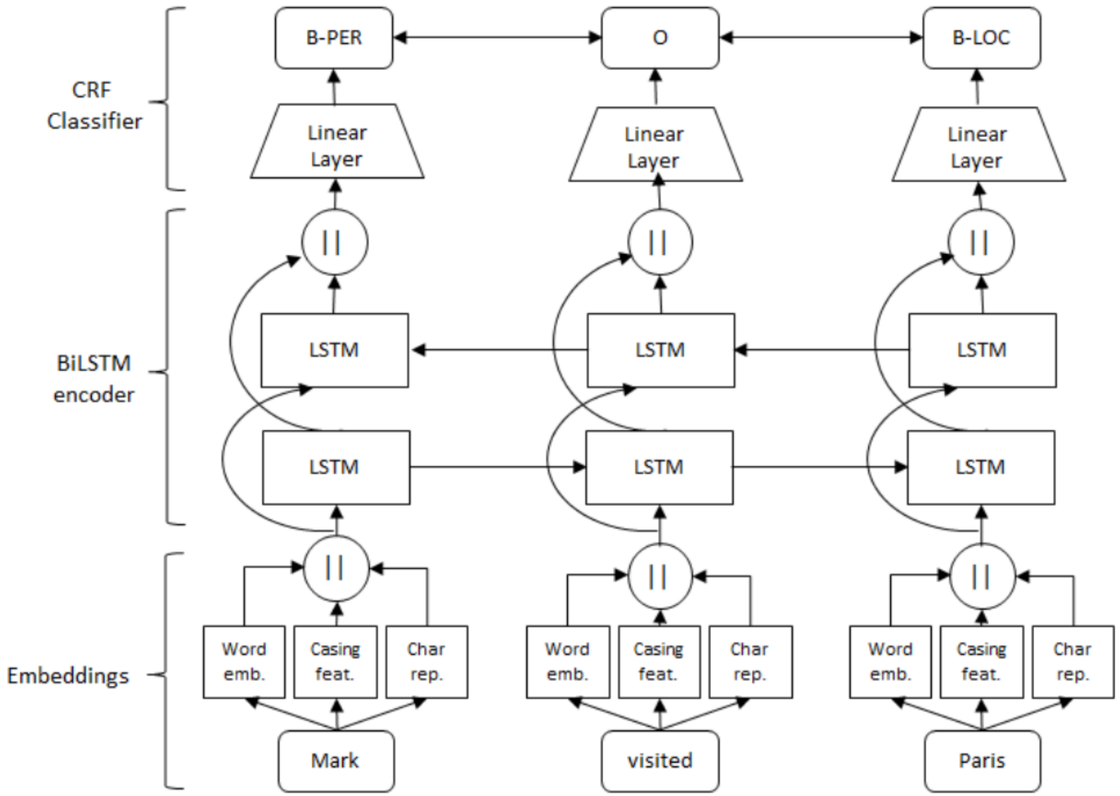
Source: https://colah.github.io/posts/2015-08-Understanding-LSTMs

Using LSTM networks in sequence labelling applications



Source: Nils Reimers and Iryna Gurevych. Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks

Using LSTM networks in sequence labelling tasks



- The BiLSTM architecture is a popular architecture for sequence labelling problems such as:
  - PoS tagging, NER (Named Entity Recognition), sentiment analysis tasks

Source: Nils Reimers and Iryna Gurevych. Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks

Reimers and Gurevych BiLSTM results

Task	Dataset	Training sentences	Test sentences	#tags
POS	WSJ	500	5459	45
Chunking	ConLL 2000 (WSJ)	8926	2009	23
NER	CoNLL 2003 (Reuters)	13862	3420	9
Entities	ACE 2005	15185	674	15
Events	TempEval3	4090	279	3

Dataset	Le. Dep.	Le. BoW	GloVe1	GloVe2	GloVe3	Komn.	G. News	FastText
POS	6.5%	0.0%	0.0%	0.0%	0.0%	**93.5%**	0.0%	0.0%
$\Delta Acc.$	-0.39%	-2.52%	-4.14%	-4.97%	-2.60%		-1.95%	-2.28%
Chunking	**60.8%**	0.0%	0.0%	0.0%	0.0%	37.1%	2.1%	0.0%
ΔF_1		-0.52%	-1.09%	-1.50%	-0.93%	-0.10%	-0.48%	-0.75%
NER	4.5%	0.0%	22.7%	0.0%	**43.6%**	27.3%	1.8%	0.0%
ΔF_1	-0.85%	-1.17%	-0.15%	-0.73%		-0.08%	-0.75%	-0.89%
Entities	4.2%	7.6%	0.8%	0.0%	6.7%	**57.1%**	21.8%	1.7%
ΔF_1	-0.92%	-0.89%	-1.50%	-2.24%	-0.80%		-0.33%	-1.13%
Events	12.9%	4.8%	0.0%	0.0%	0.0%	**71.8%**	9.7%	0.8%
ΔF_1	-0.55%	-0.78%	-2.77%	-3.55%	-2.55%		-0.67%	-1.36%
Average	17.8%	2.5%	4.7%	0.0%	10.1%	**57.4%**	7.1%	0.5%

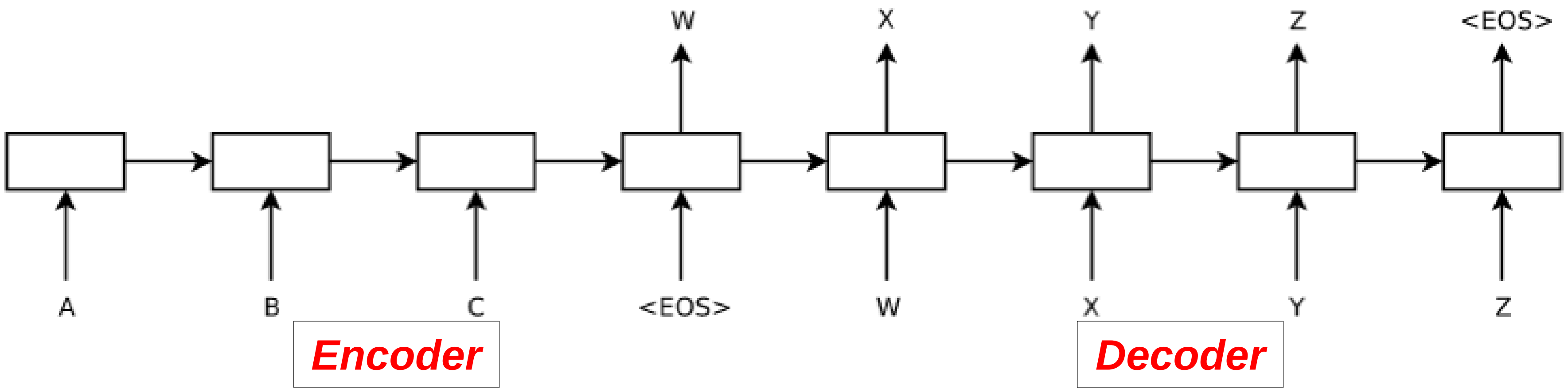
Training data sizes: **GloVe3 840B** **[Komninos and Manandhar, 2016] 2B**

- ***Nils Reimers and Iryna Gurevych* (2017), see arxiv:**
  - Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks
  - Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging

Classwork – Design LSTM word embedding model

- For example:
  - [the, dog] [the, cat] → chases **(+ example)** should give ***high*** probability
  - [the, dog] [the, cat] → bites **(+ example)** should give ***high*** probability
  - [the, dog] [the, cat] → buy **(- example)** should give ***low*** probability

Encoder-Decoder Architectures

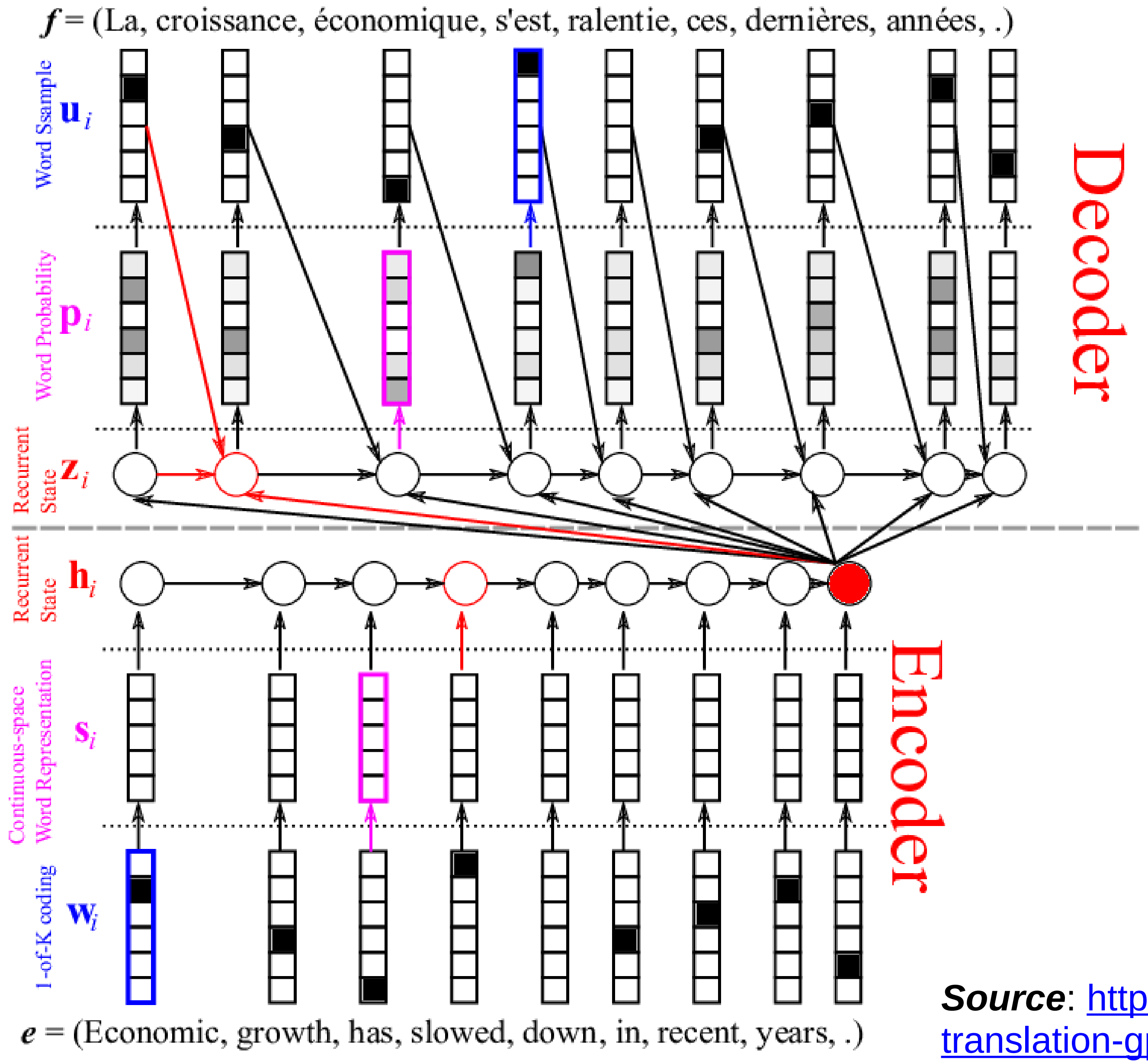


- The encodes the whole sentence into *a compressed representation* ***w***
- The decoder starts decoding ***w***
- At each step the decoder is fed the previous word to generate the next word
- The decoding stops once the *End of Sentence* (**EOS**) token is generated.
- This simple architecture does a good job for ***machine translation***.
- By training the decoder to generate the input sentence itself this architecture can be used to ***learn a sentence representations***

Also see Keras quick 10 minute tutorial on sequence models

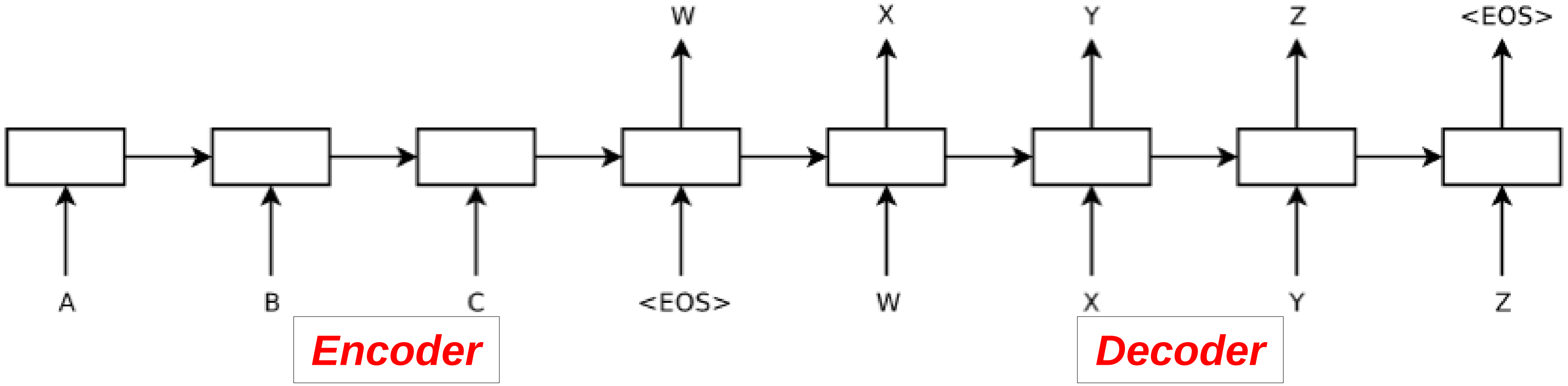
Source: Sutskever, Vinyals and Le. Sequence to Sequence Learning with Neural Networks

Encoder-Decoder Architectures



Source: https://devblogs.nvidia.com/introduction-neural-machine-translation-gpus-part-2/

Encoder-Decoder Architectures

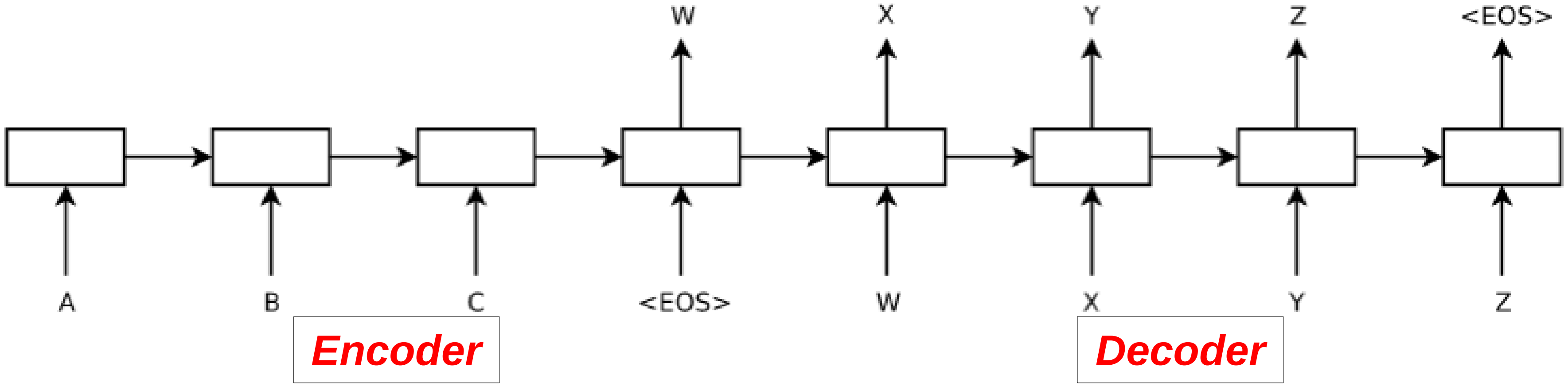


$$p(y_1, \dots, y_m | x_1, \dots, x_n) = \prod_{t=1}^{m} p(y_t | w, y_1, \dots, y_{t-1})$$

- The decoder probability is conditioned on all previous words generated
- ***w*** is the final hidden state from the encoder
- The length ***m*** is variable length dependent upon when EOS is output
- For each output the above probability is calculated
- The sentence chosen as the translation is the arg max of the above

Source: Sutskever, Vinyals and Le. Sequence to Sequence Learning with Neural Networks

Encoder-Decoder Architectures



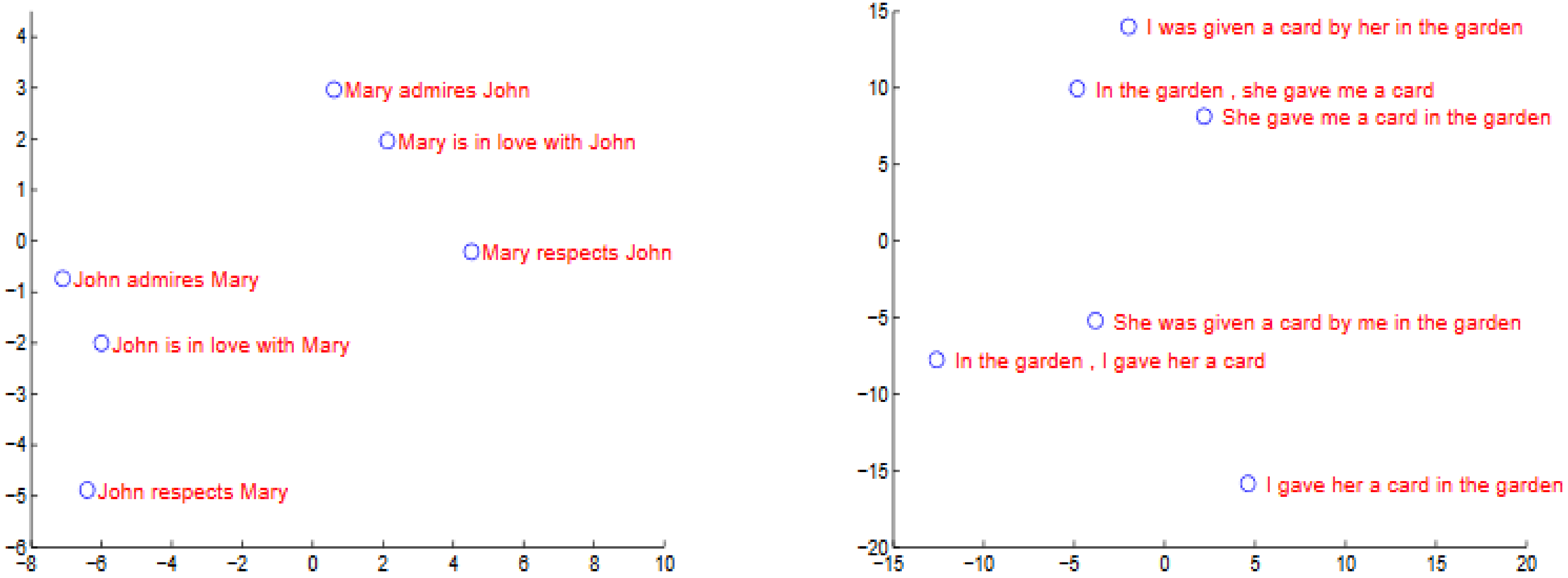
$$p(y_1, \ldots, y_m | x_1, \ldots, x_n) = \prod_{t=1}^{m} p(y_t | w, y_1, \ldots, y_{t-1})$$

$$y_1^*, \ldots, y_m^* = arg\,max_{y_1,\ldots,y_m}\; p(y_1, \ldots, y_m | x_1, \ldots, x_n)$$

- To find the arg max sequence $y_1^*, \ldots, y_m^*$ a beam search is used also known as Viterbi decoding (from HMM literature)

Source: Sutskever, Vinyals and Le. Sequence to Sequence Learning with Neural Networks

Encoder-Decoder Architectures



- ***2D*** projection of the latent encoding of sample sentences
- shows that semantically related sentences have similar representations

Source: Sutskever, Vinyals and Le. Sequence to Sequence Learning with Neural Networks

Encoder-Decoder Architectures

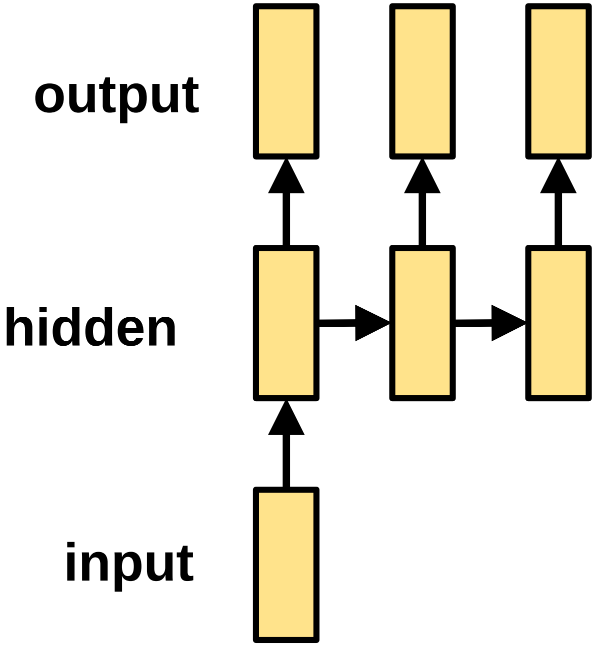
Type	Sentence
Our model	Ulrich UNK , membre du conseil d' administration du constructeur automobile Audi , affirme qu' il s' agit d' une pratique courante depuis des années pour que les téléphones portables puissent être collectés avant les réunions du conseil d' administration afin qu' ils ne soient pas utilisés comme appareils d' écoute à distance .
Truth	Ulrich Hackenberg , membre du conseil d' administration du constructeur automobile Audi , déclare que la collecte des téléphones portables avant les réunions du conseil , afin qu' ils ne puissent pas être utilisés comme appareils d' écoute à distance , est une pratique courante depuis des années .
Our model	" Les téléphones cellulaires , qui sont vraiment une question , non seulement parce qu' ils pourraient potentiellement causer des interférences avec les appareils de navigation , mais nous savons , selon la FCC , qu' ils pourraient interférer avec les tours de téléphone cellulaire lorsqu' ils sont dans l' air " , dit UNK .
Truth	" Les téléphones portables sont véritablement un problème , non seulement parce qu' ils pourraient éventuellement créer des interférences avec les instruments de navigation , mais parce que nous savons , d' après la FCC , qu' ils pourraient perturber les antennes-relais de téléphonie mobile s' ils sont utilisés à bord " , a déclaré Rosenker .
Our model	Avec la crémation , il y a un " sentiment de violence contre le corps d' un être cher " , qui sera " réduit à une pile de cendres " en très peu de temps au lieu d' un processus de décomposition " qui accompagnera les étapes du deuil " .
Truth	Il y a , avec la crémation , " une violence faite au corps aimé " , qui va être " réduit à un tas de cendres " en très peu de temps , et non après un processus de décomposition , qui " accompagnerait les phases du deuil " .

- Sample translations compared with ground truth showing good translation performance for long sentences

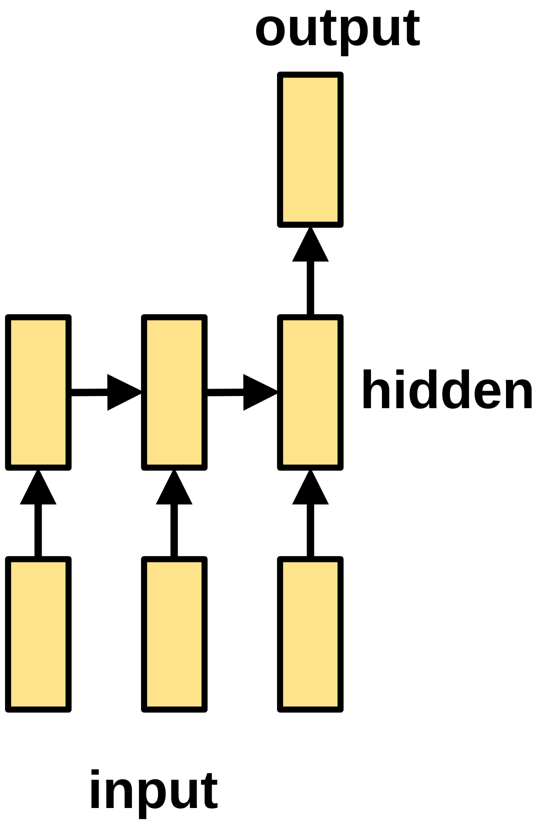
Source: Sutskever, Vinyals and Le. Sequence to Sequence Learning with Neural Networks

Sequence Architectures

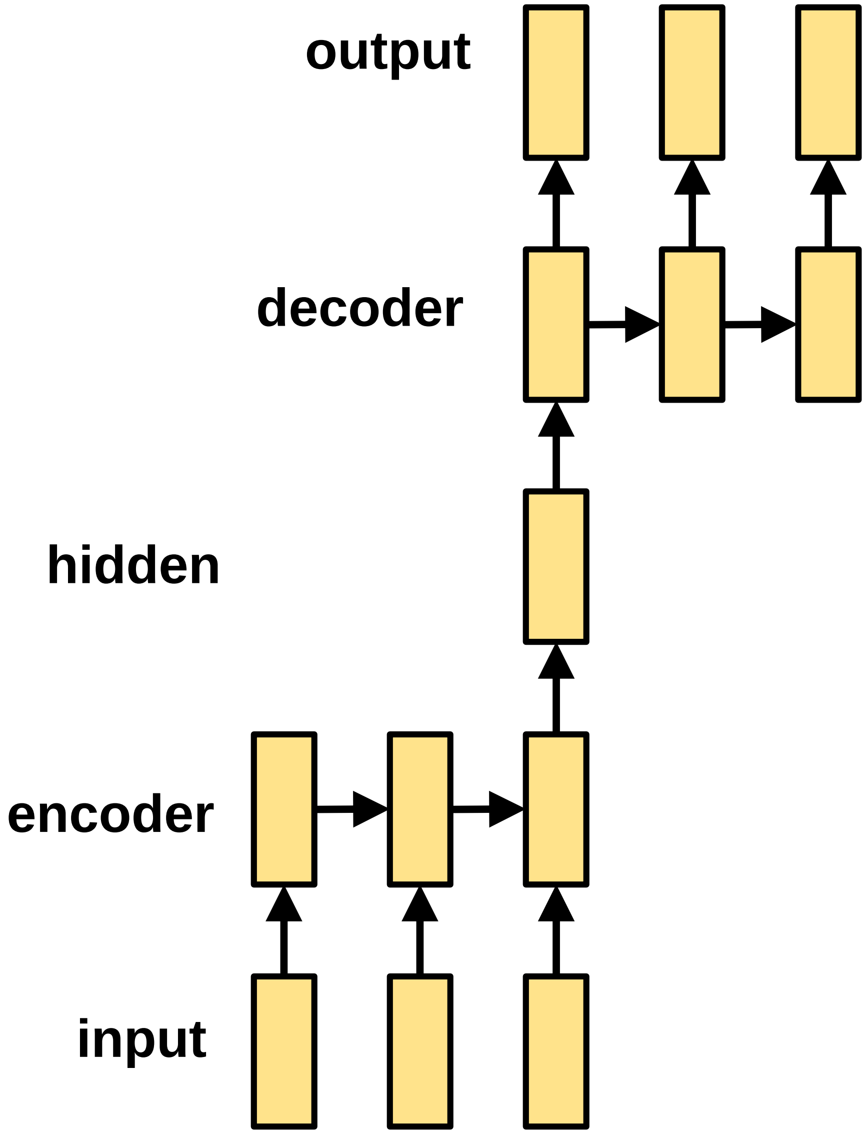
One to many

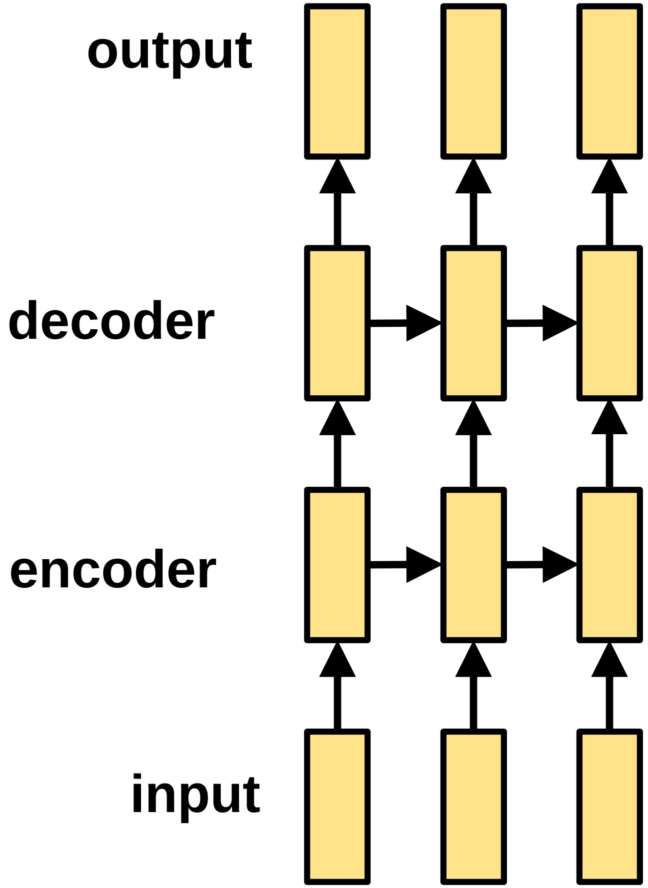


Many to one



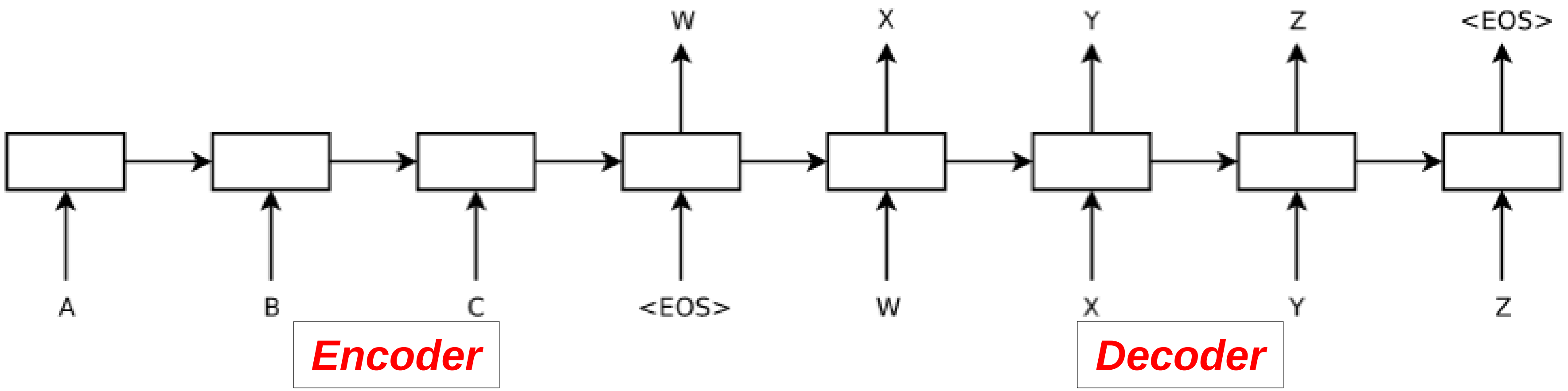
Sequence Architectures





Attention based models

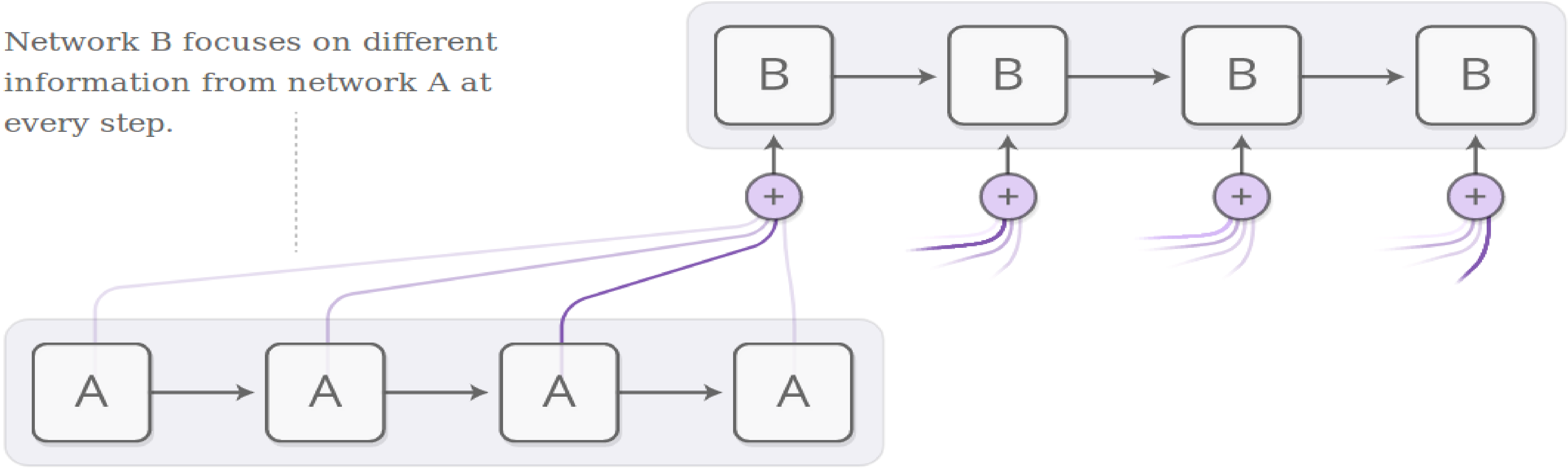
Issues with Encoder-Decoder Architectures



- The encoder needs to summarise the whole sentence into a single vector

Source: Sutskever, Vinyals and Le. Sequence to Sequence Learning with Neural Networks

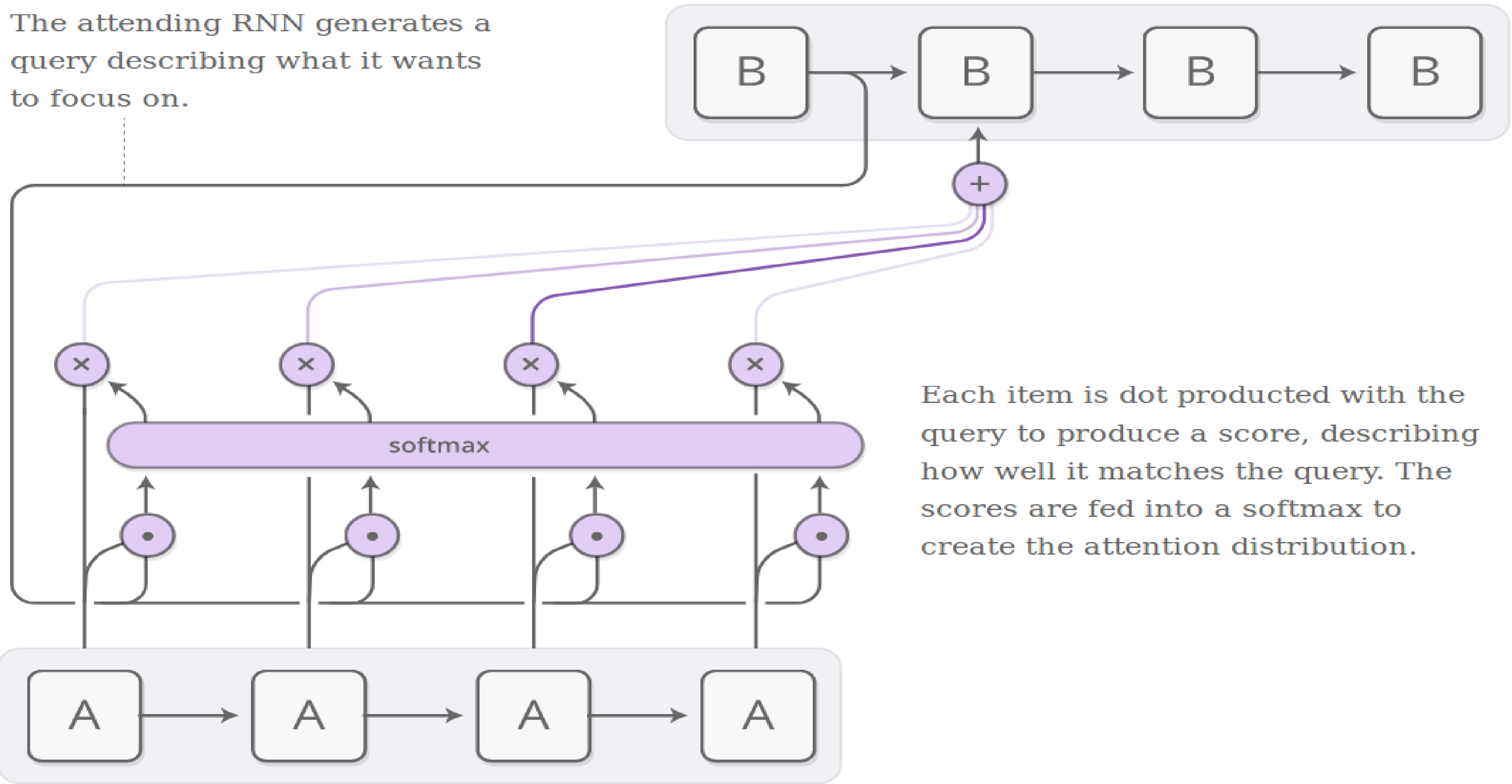
Attention based models



- **Network B** is your output network (here a RNN)
- **Network A** is the input network
- The input to B is now a weighted combination of the output from A

Source: Olah and Carter https://distill.pub/2016/augmented-rnns/

Attention based models



- At each step, ***similarity*** between the **hidden output from B** and the output from A is computed
- The similarity scores are fed to a softmax unit to find the most similar items from A
- Multiply gate is used to generate a linear combination of **most relevant outputs from A**

Source: Olah and Carter https://distill.pub/2016/augmented-rnns/

Attention based models

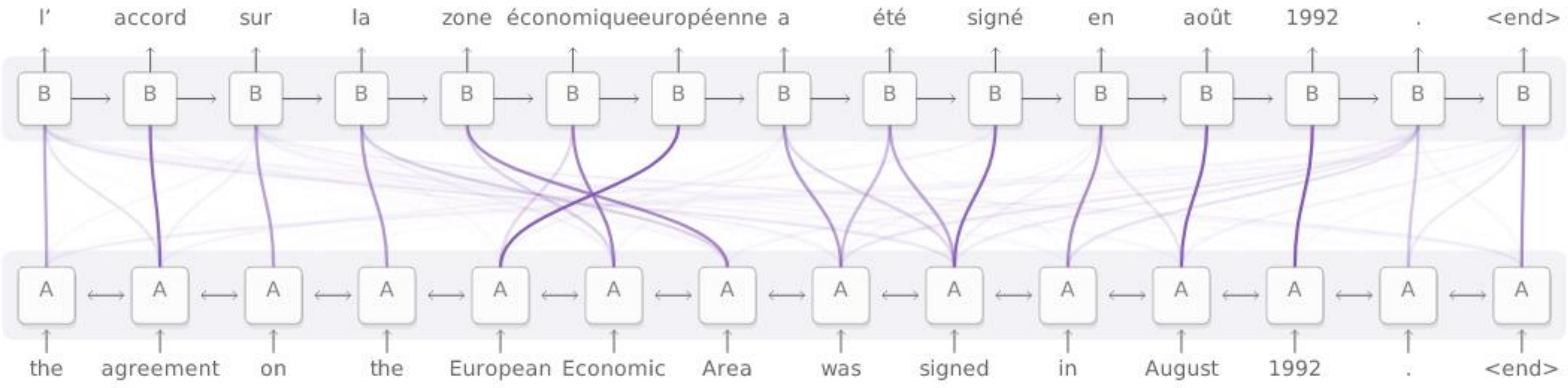
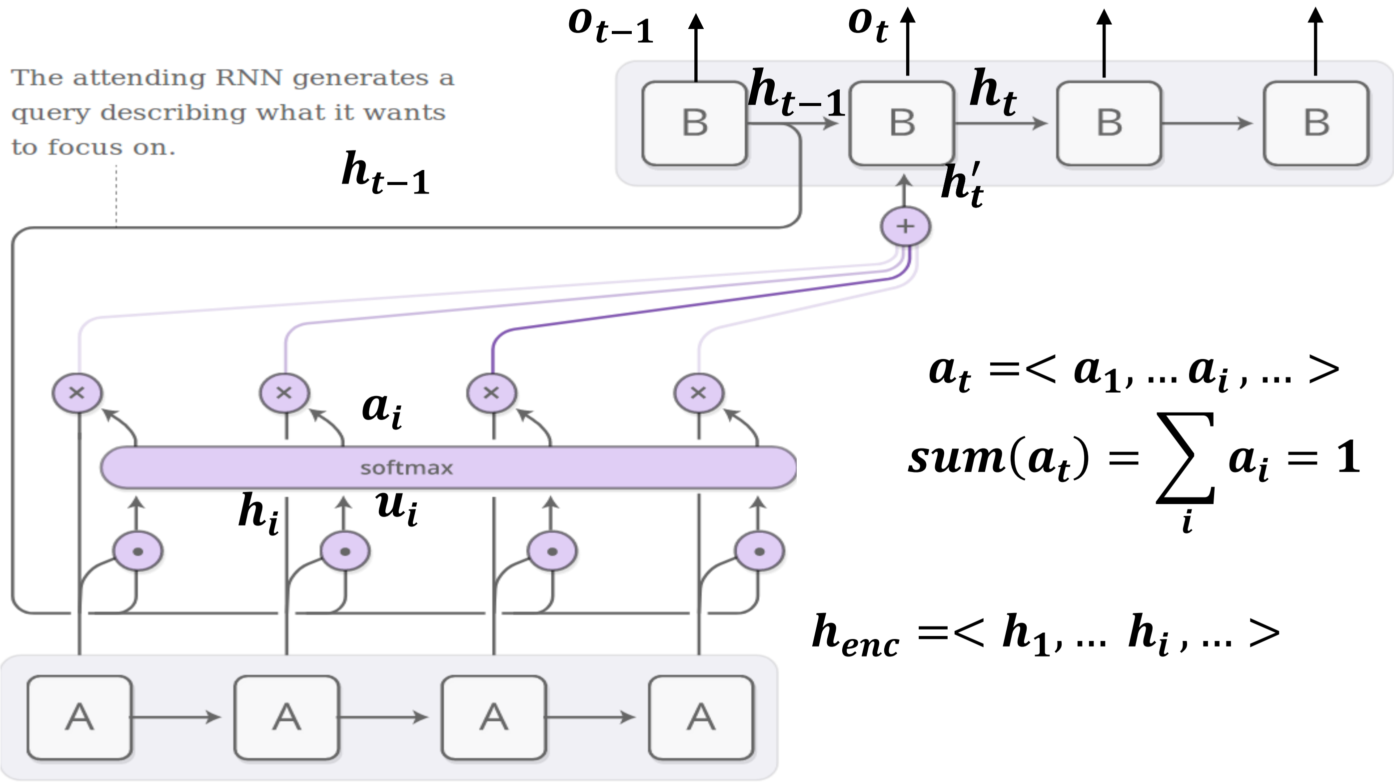


Diagram derived from Fig. 3 of Bahdanau, *et al.* 2014

- The attention mechanism generates a simpler architecture compared to the vanilla encoder-decoder setup
- In the vanilla encoder-decoder setup, the encoder has to summarise the whole sentence into a single vector
- In the above architecture, there is a closer connection between the input and the output
- This results in better gradient flow and hence better performance on machine translation tasks

Source: Olah and Carter https://distill.pub/2016/augmented-rnns/

Attention based models



$$a_t = < a_1, \dots a_i, \dots >$$

$$sum(a_t) = \sum_i a_i = 1$$

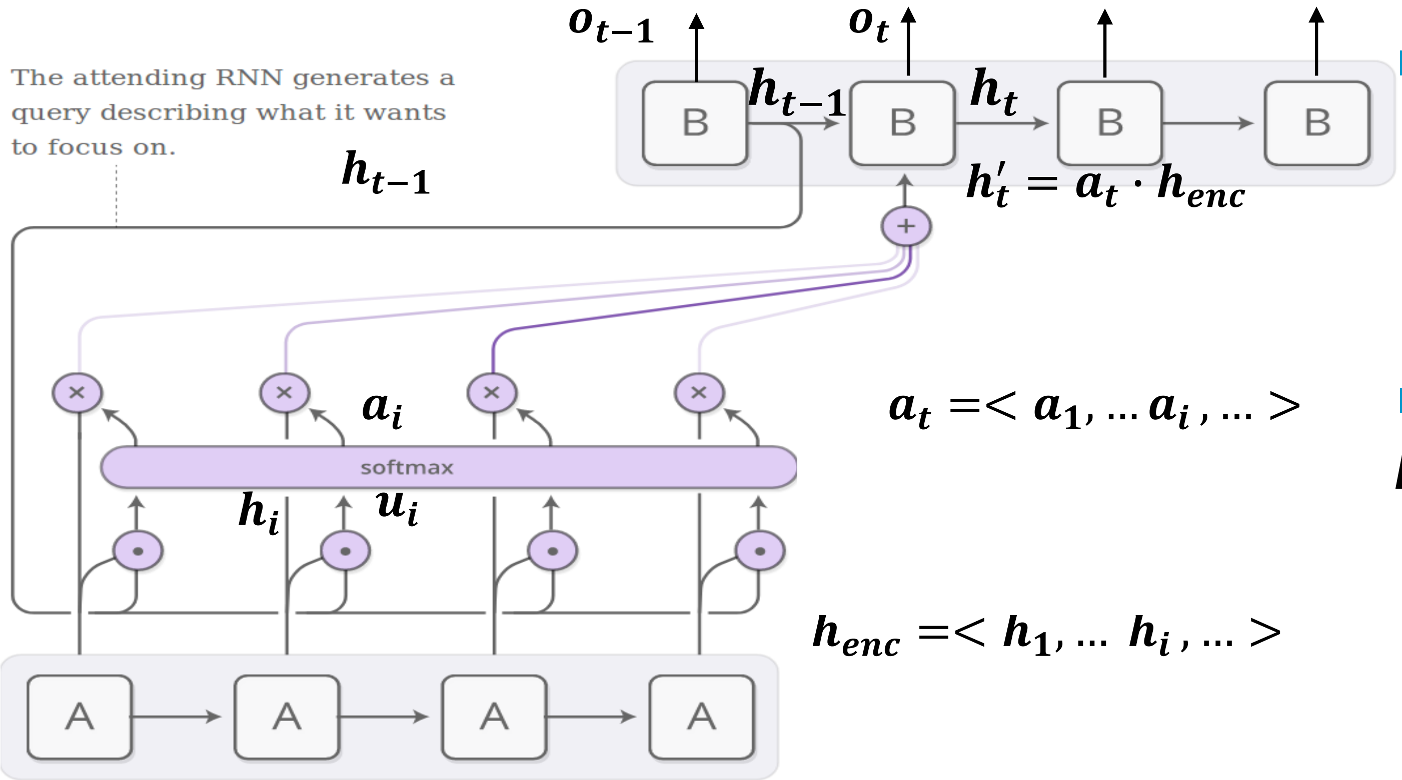
$$h_{enc} = < h_1, \dots h_i, \dots >$$

- In this particular model:

$$u_i = \dots$$
$$a_t = \cdots$$
$$h'_t = \cdots$$

Source: Olah and Carter https://distill.pub/2016/augmented-rnns/

Attention based models



- In this particular model:

$$u_i = h_i \cdot h_{t-1}$$
$$a_t = softmax(u_t)$$
$$h'_t = a_t \cdot h_{enc}$$

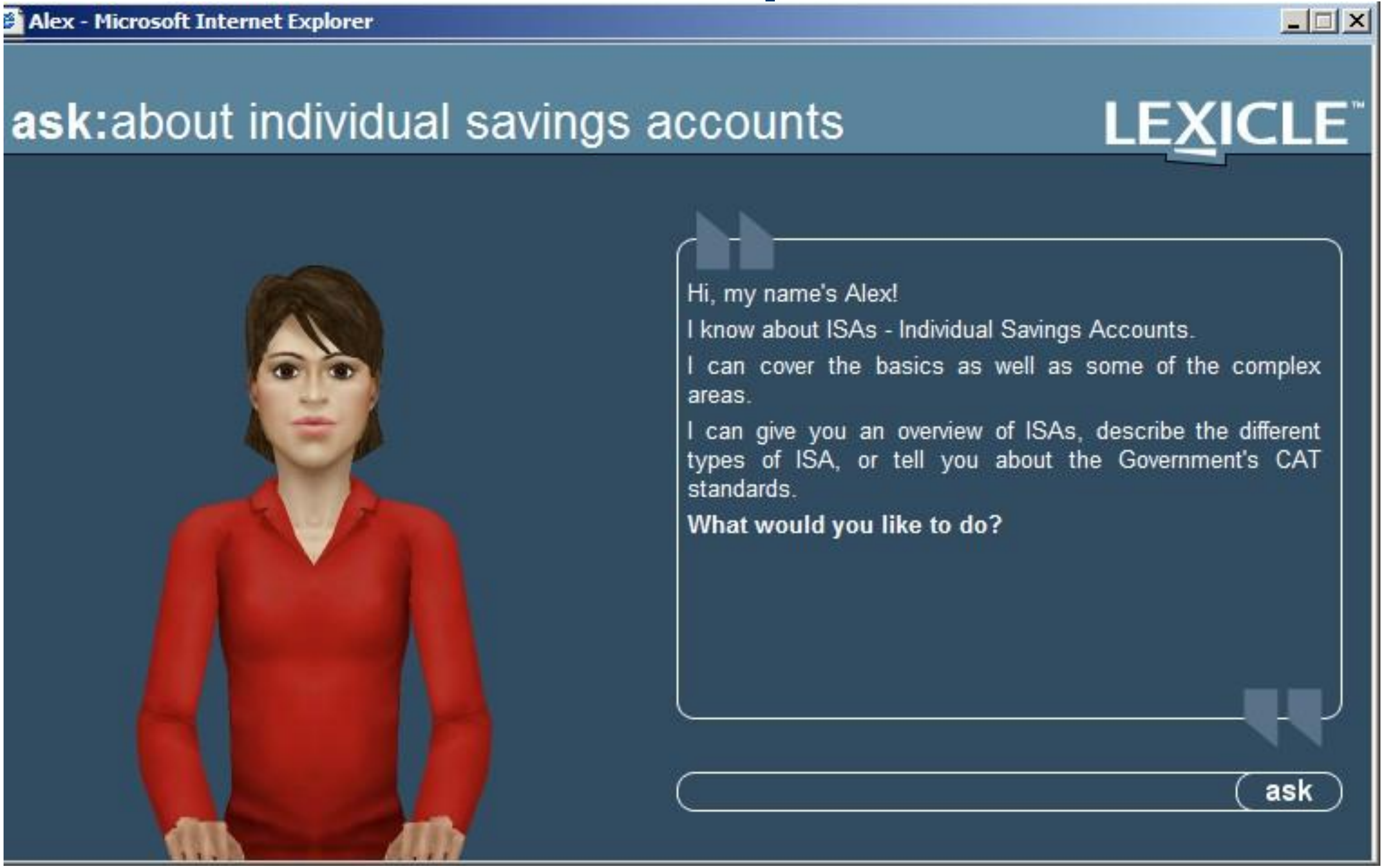
- $h_i \cdot h_{t-1}$ and $a_t \cdot h_{enc}$ denotes ***dot products***

- In general $u_i = f(h_i, h_{t-1}, y_{t-1})$ can be an arbitrary (neural network) function
- **Classwork –** Design your own function f

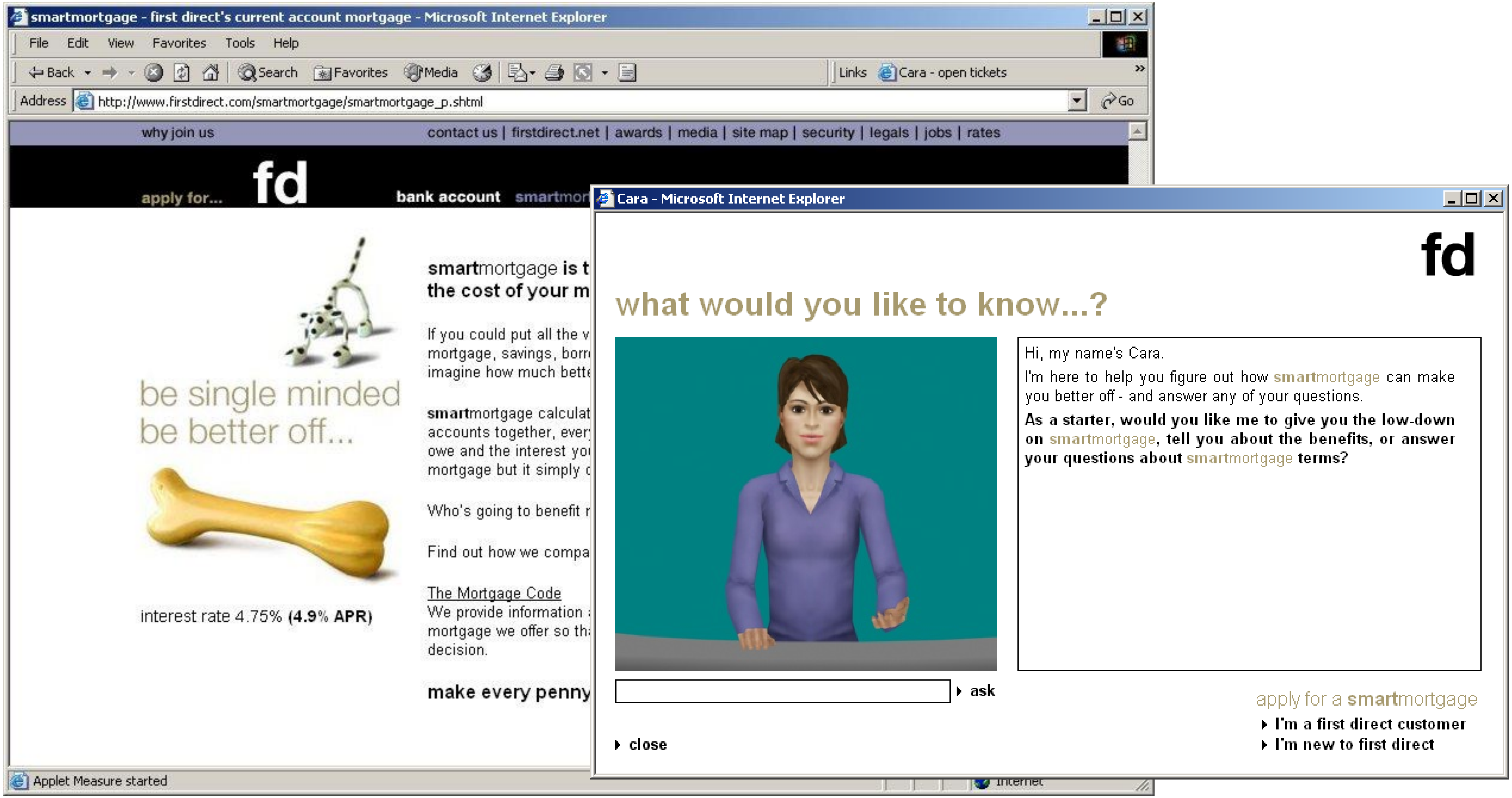
Source: Olah and Carter https://distill.pub/2016/augmented-rnns/

Dialogue models

Virtual assistant example



Source: Lexicle Ltd UK



Source: Lexicle Ltd UK

smartmortgage - first direct's current account mortgage - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Search Favorites Media Links Cara - open tickets

Address http://www.firstdirect.com/smartmortgage/smartmortgage_p.shtml Go

be single minded
be better off...

interest rate 4.75% **(4.9% APR)**

smartmortgage **is the simple way to reduce the cost of your mortgage**

If you could put all the value of your money together - mortgage, savings, borrowing and current account - imagine how much better off you'd be.

smartmortgage calculates all the interest on your accounts together, everyday, to reduce the amount you owe and the interest you pay. It's a straightforward mortgage but it simply costs you less.

Who's going to benefit most from paying less? You are.

Find out how we compare against our competitors.

The Mortgage Code
We provide information about the different types of mortgage we offer so that you can make an informed decision.

make every penny count. everyday.

smartmortgage
- the benefits
- how it works
- moving mortgages
- faqs
- ask Cara
- in detail

ask Cara

use our calculators
- **how much could I save**
- **how much can I borrow**

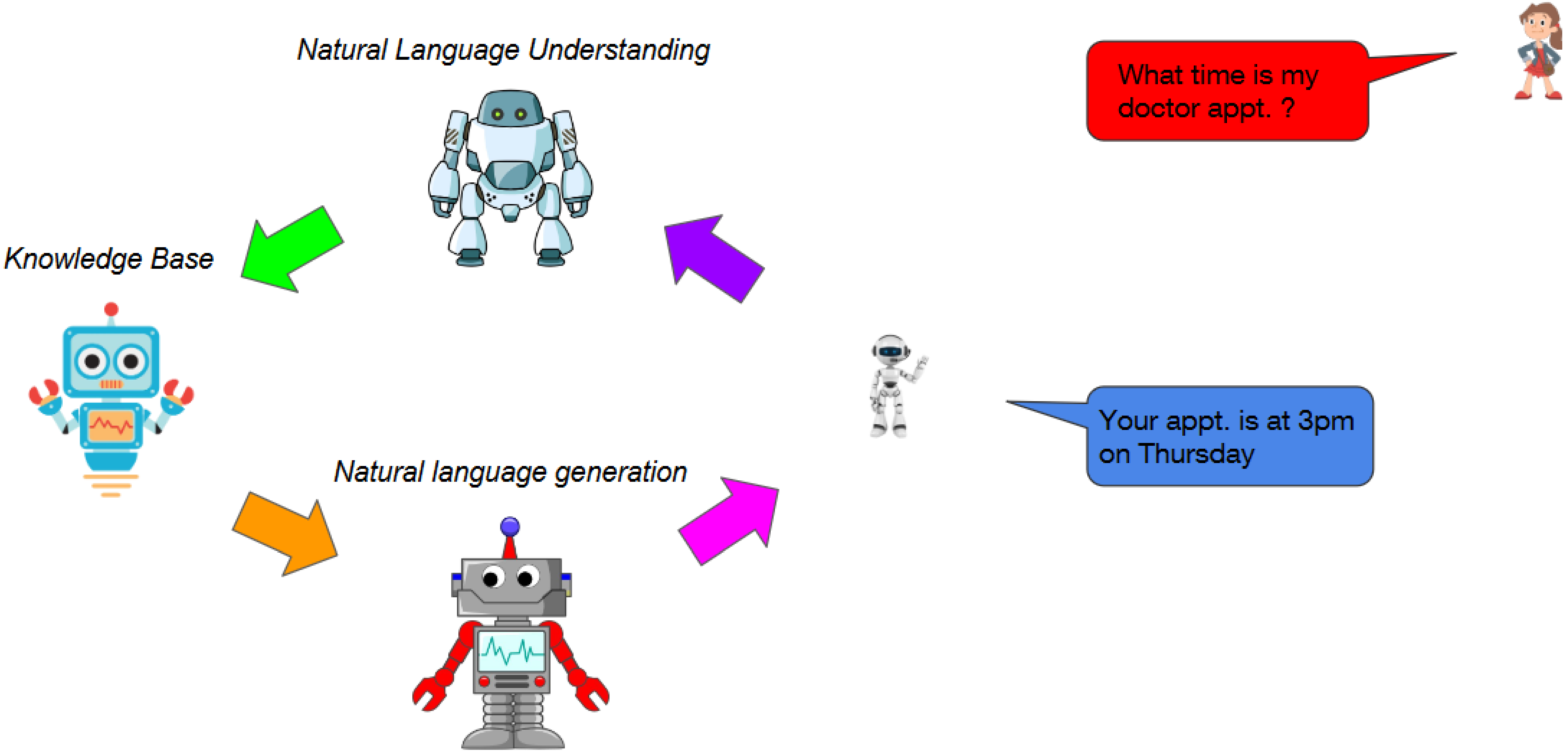
get smartmortgage
- **apply**

Applet Measure started Internet

Virtual assistant example

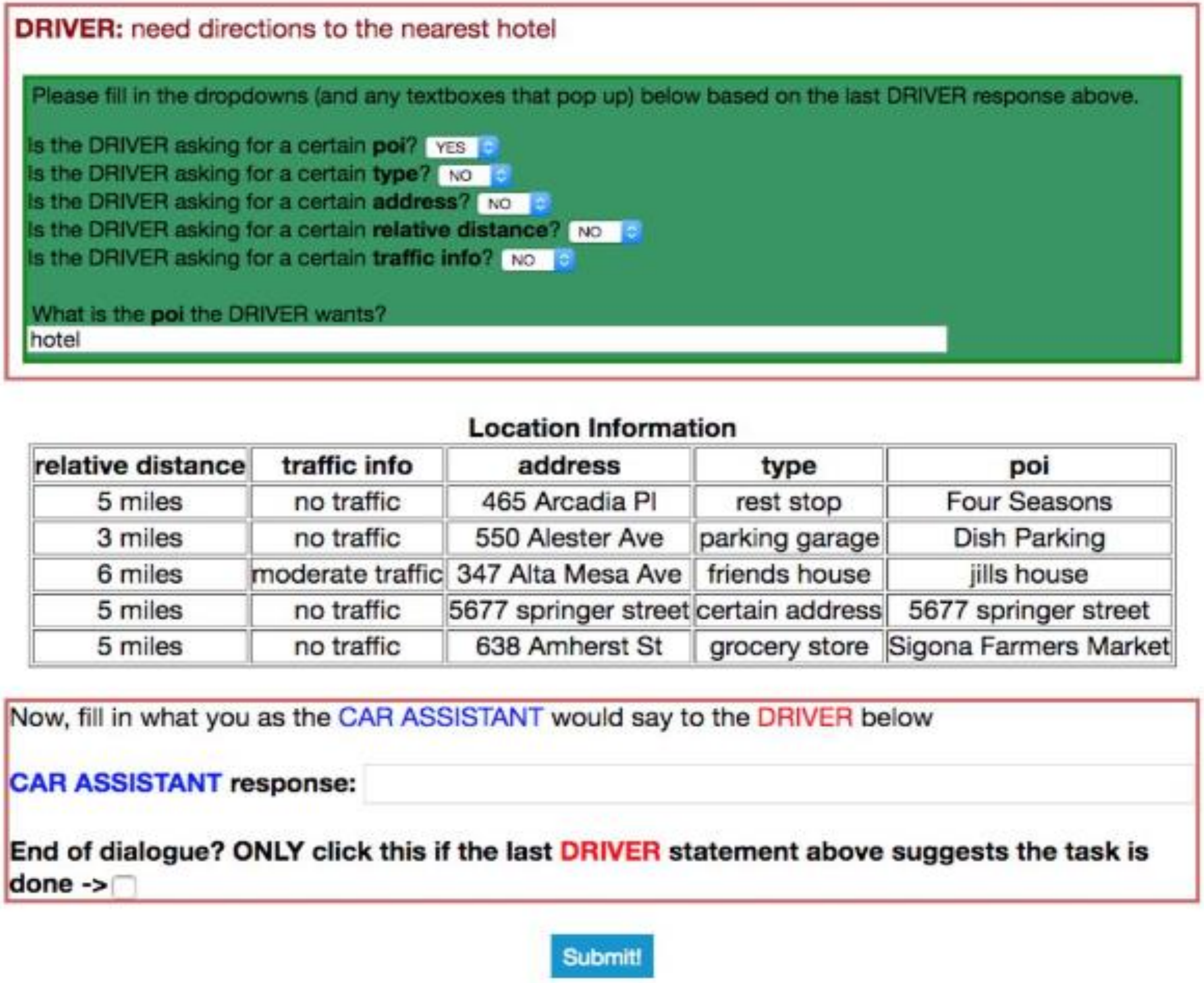
- **Ultimate goal** -- build machines that can "***understand***" human language (without human supervision)
- Current progress:
  - Large scale robust but **shallow understanding of text**
  - Large scale unsupervised learning using automatically labelled data (self supervision)
  - Mapping between knowledge graphs and natural language
  - Question answering from text and knowledge graphs
  - Dialogue systems querying knowledge graphs and text databases (e.g. Wikipedia)
  - Large scale robust (minimally supervised) machine translation

Key-Value Retrieval Networks for Task-Oriented Dialogues



Source: SIGDIAL slides. Key-Value Retrieval Networks for Task-Oriented Dialogues. Mihail Eric, Lakshmi Krishnan,Francois Charette, and Christopher D. Manning

Key-Value Retrieval Networks for Task-Oriented Dialogues



DRIVER: need directions to the nearest hotel

Please fill in the dropdowns (and any textboxes that pop up) below based on the last DRIVER response above.

Is the DRIVER asking for a certain **poi**? YES
Is the DRIVER asking for a certain **type**? NO
Is the DRIVER asking for a certain **address**? NO
Is the DRIVER asking for a certain **relative distance**? NO
Is the DRIVER asking for a certain **traffic info**? NO

What is the **poi** the DRIVER wants?
hotel

Location Information

relative distance	traffic info	address	type	poi
5 miles	no traffic	465 Arcadia Pl	rest stop	Four Seasons
3 miles	no traffic	550 Alester Ave	parking garage	Dish Parking
6 miles	moderate traffic	347 Alta Mesa Ave	friends house	jills house
5 miles	no traffic	5677 springer street	certain address	5677 springer street
5 miles	no traffic	638 Amherst St	grocery store	Sigona Farmers Market

Now, fill in what you as the CAR ASSISTANT would say to the DRIVER below

CAR ASSISTANT response:

End of dialogue? ONLY click this if the last DRIVER statement above suggests the task is done ->

Submit!

- Data collection method
- Use human volunteers to generate the potential dialogues given current knowledge base

Key-Value Retrieval Networks for Task-Oriented Dialogues

- Sample dialogue

Event	Time	Date	Party	Agenda
opt. appt.	10am	Thursday	sister	-
dinner	8pm	the 13th	Ana	-
opt. appt.	7pm	the 20th	Jeff	-
opt. appt.	4pm	the 13th	Alex	-
...	...	...	...	...

DRIVER: I need to find out the time and parties attending my optometrist appointment.
CAR: I have 3 appointments scheduled, with Alex, your sister, and Jeff. Which are you referring to?
DRIVER: I want to know about the one that Alex is joining me at
CAR: That optometrist appointment is at 4 pm.
DRIVER: Thanks
CAR: no problem

Reference: Key-Value Retrieval Networks for Task-Oriented Dialogues.
Mihail Eric, Lakshmi Krishnan,Francois Charette, and Christopher D. Manning

Example dialog

POI	Category	Traffic Info
Civic Center	Parking garage	Car collision
Valero	Gas station	Road block
Webster Garage	Parking garage	Car collision
Trader Joes	Grocery Store	Heavy
Mandarin Roots.	Chinese rest.	Moderate

DRIVER: I am looking for a gas station near me with the shortest route from me.

CAR: *The closest gas station is valero but, but there is a road block nearby.*

DRIVER: What is the next nearest gas station?

CAR: *Valero is the only gas station that I can find*

DRIVER: Thanks

CAR: *Here to serve*

Source: SIGDIAL slides. Key-Value Retrieval Networks for Task-Oriented Dialogues.
Mihail Eric, Lakshmi Krishnan,Francois Charette, and Christopher D. Manning

Modifications to the Knowledge base

Original knowledge base

location	distance	POI type	Traffic info	address
chevron	5 miles	gas station	moderate traffic	783 arcadia pl

Modified knowledge base

(chevron, distance, 5_miles)
(chevron, poi_type, gas_station)
(chevron, traffic_info, moderate_traffic)
(chevron, address, 783_arcadia_pl)

Key-Value Embedding Table

Triple knowledge base

Subject	Relation	Object
dinner	time	8pm
dinner	date	the 13th
dinner	party	Ana
dinner	agenda	-

Key-Value Knowledge base

Key	Value
dinner_time	8pm
dinner_date	the 13th
dinner_party	Ana
dinner_agenda	-

Key-Value Embedding Table

Triple knowledge base

Key	Value
dinner_time	8pm
dinner_date	the 13th
dinner_party	Ana
dinner_agenda	-

Key-Value Memory

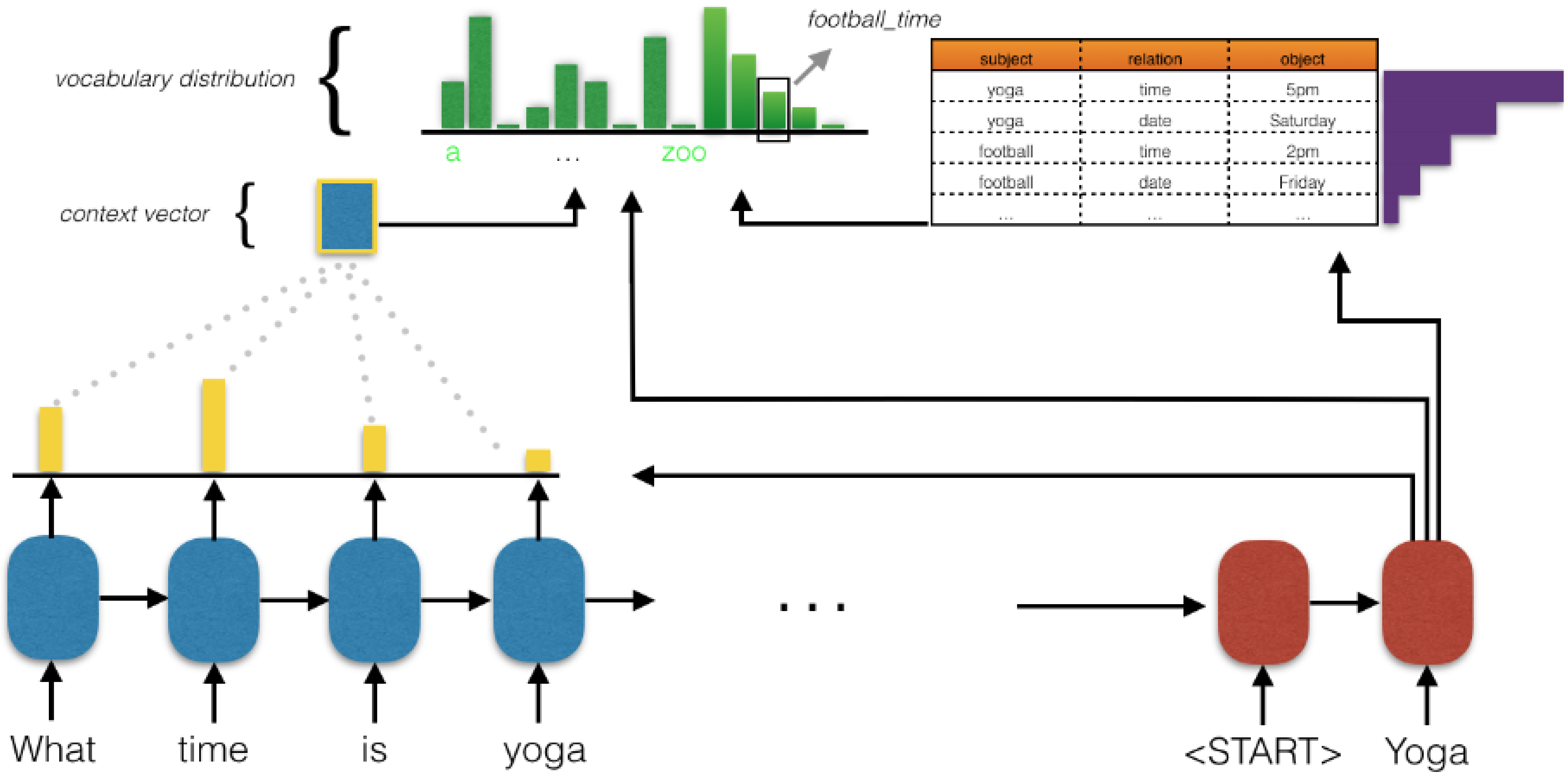
Key	Value
emb(dinner)+ emb(time)	8pm
emb(dinner)+ emb(date)	the 13th
emb(dinner)+ emb(party)	Ana
emb(dinner)+emb(agenda)	-

Modifications to the training data

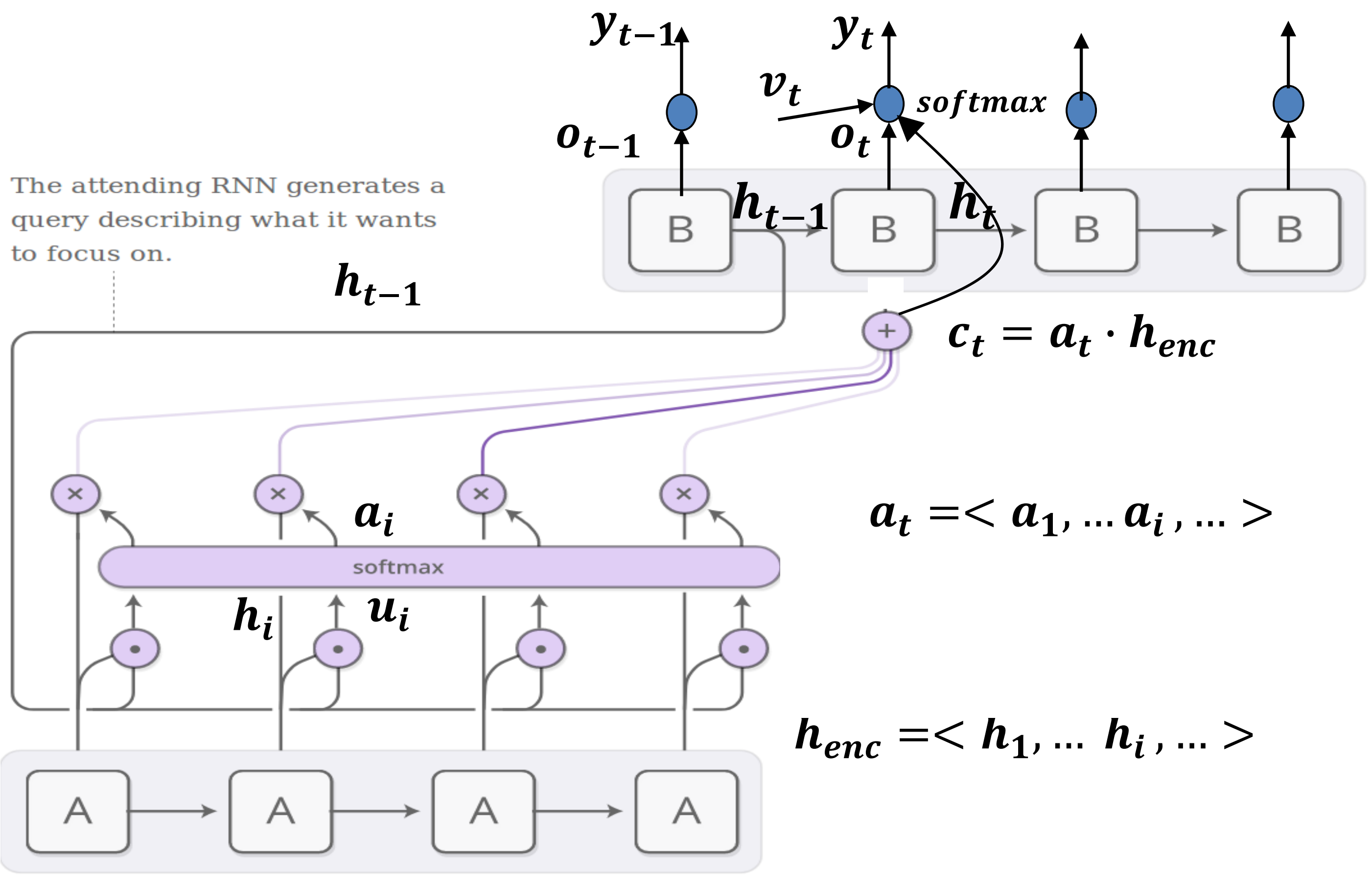
Original data	Modified data
1 what gas_stations are here there is a chevron ['chevron']	1 what gas_stations are here there is a chevron ['chevron']
2 that s good please pick the quickest route to get there and avoid all heavy_traffic taking you to chevron ['chevron']	2 that s good please pick the quickest route to get there and avoid all heavy_traffic taking you to chevron ['chevron']
3 what is the address 783_arcadia_pl is the address for chevron gas_station ['783_arcadia_pl', 'chevron', 'gas_station']	3 what is the address chevron_address is the address for chevron gas_station ['783_arcadia_pl', 'chevron', 'gas_station']
4 perfect thank you you re welcome happy to help []	4 perfect thank you you re welcome happy to help []

- **chevron_address** is now a key whose value is **783_arcadia_pl** in the knowledge base

System architecture



Changes to accommodate key-value knowledge base



$v_t =< v_1, \ldots v_j , \ldots >$

k_j

Key	Value
emb(dinner)+ emb(time)	8pm
emb(dinner)+ emb(date)	the 13th
emb(dinner)+ emb(party)	Ana
emb(dinner)+emb(agenda)	-

$v_j = f(k_j, h_{t-1})$

- To accommodate knowledge base keys, they add attention over knowledge base keys:

$$v_j = w^T tanh(w_2\, tanh(w_1[k_j , h_{t-1}]))$$
$$y_t = softmax([o_t , c_t, v_t])$$

- Now, the output ranges over both the vocabulary words and the knowledge base keys

Thank you!