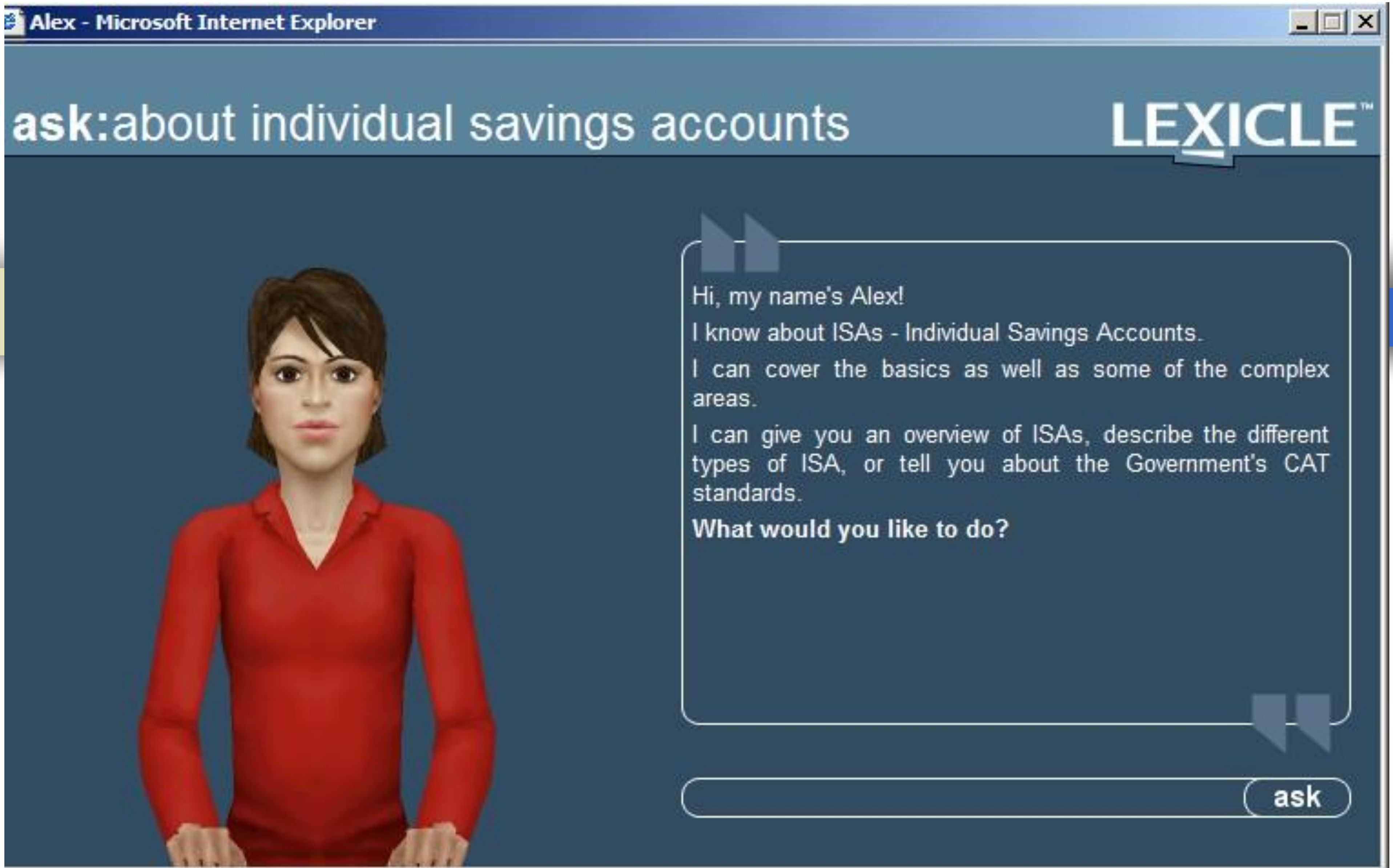
Natural Language Processing

A short tutorial

Suresh Manandhar
suresh@cs.york.ac.uk

Alex – The virtual assistant



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

why join us | contact us | firstdirect.net | awards | media | site map | security | legals | jobs | rates

apply for... fd bank account | smartmortgage | savings | credit card | other services | shopping

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.

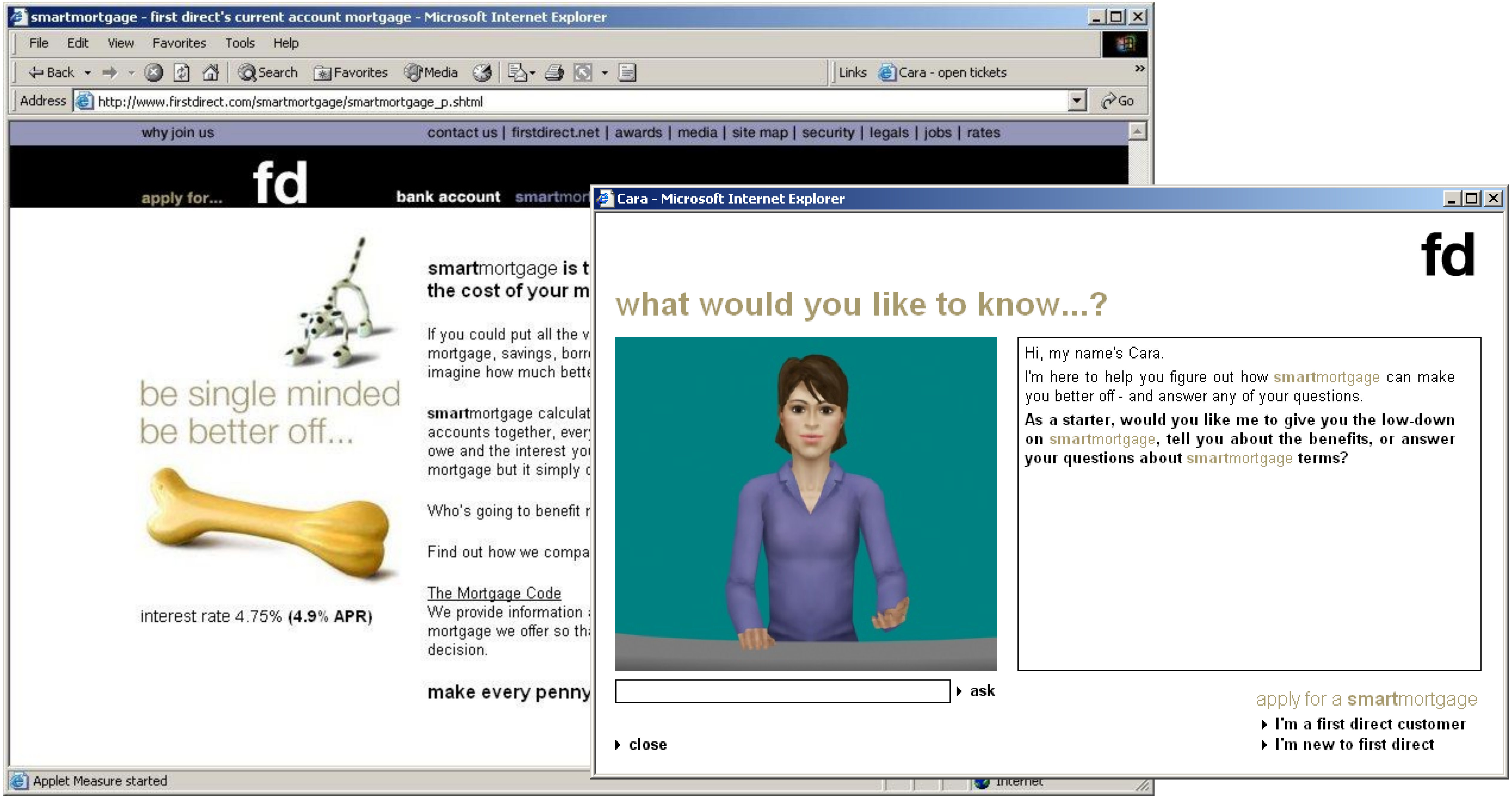
smartmortgage
- the benefits
- how it works
- moving mortgages
- faqs
- ask Cara
- in detail

ask Cara

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

get smartmortgage
- **apply**

Applet Measure started Internet


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
why join us
contact us | firstdirect.net | awards | media | site map | security | legals | jobs | rates
apply for...
fd
bank account smartmor
smartmortgage is t
the cost of your m
If you could put all the v
mortgage, savings, borr
imagine how much bette
smartmortgage calculat
accounts together, every
owe and the interest you
mortgage but it simply c
Who's going to benefit r
Find out how we compa
The Mortgage Code
We provide information a
mortgage we offer so tha
decision.
make every penny
be single minded
be better off...
interest rate 4.75% (4.9% APR)
Applet Measure started
Internet
Cara - Microsoft Internet Explorer
fd
what would you like to know...?
Hi, my name's Cara.
I'm here to help you figure out how smartmortgage can make you better off - and answer any of your questions.
As a starter, would you like me to give you the low-down on smartmortgage, tell you about the benefits, or answer your questions about smartmortgage terms?
ask
close
apply for a smartmortgage
I'm a first direct customer
I'm new to first direct


SAMSUNG

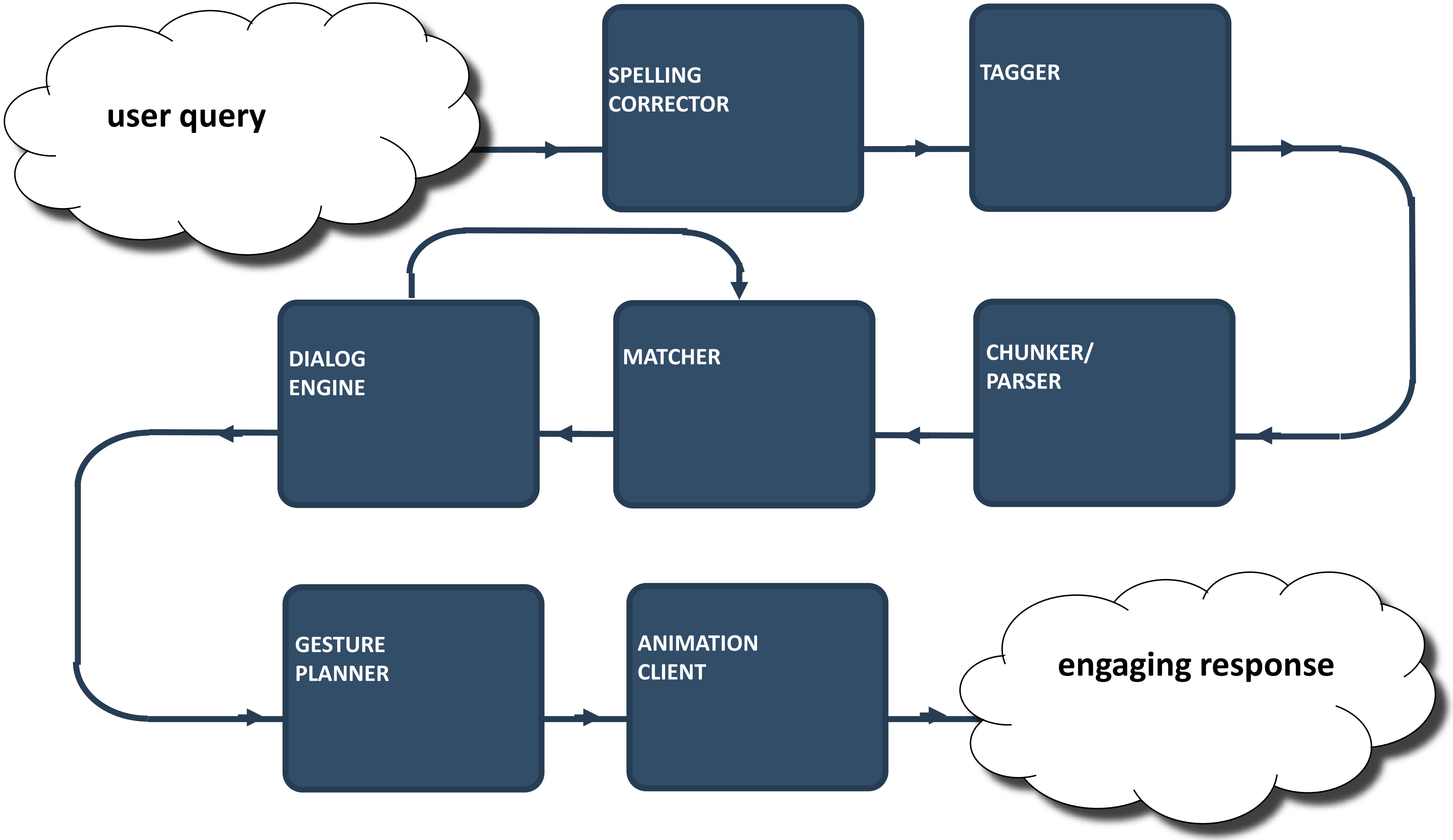
Natural language processing

- Ultimate goal -- build machines that can “*understand*” human language
- Speech vs Language Processing

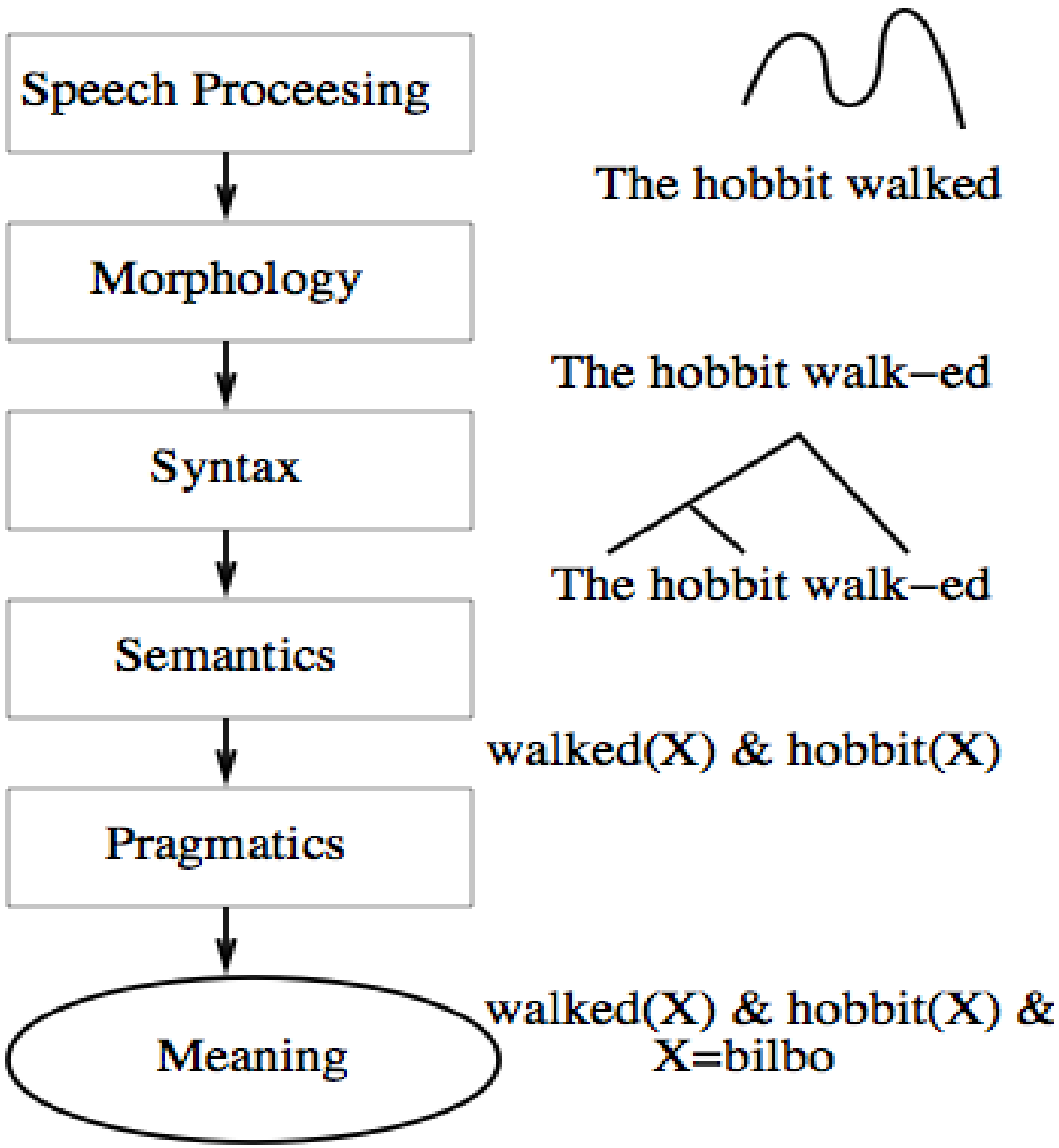
Why is it hard?

- Ambiguity
  - Phonetic
    - I scream
    - Ice cream
  - Syntactic
    - I saw the man with a telescope
    - Flying planes can be dangerous
    - Intelligent men and women
  - Semantic
    - I went to the bank
  - Pragmatic …..

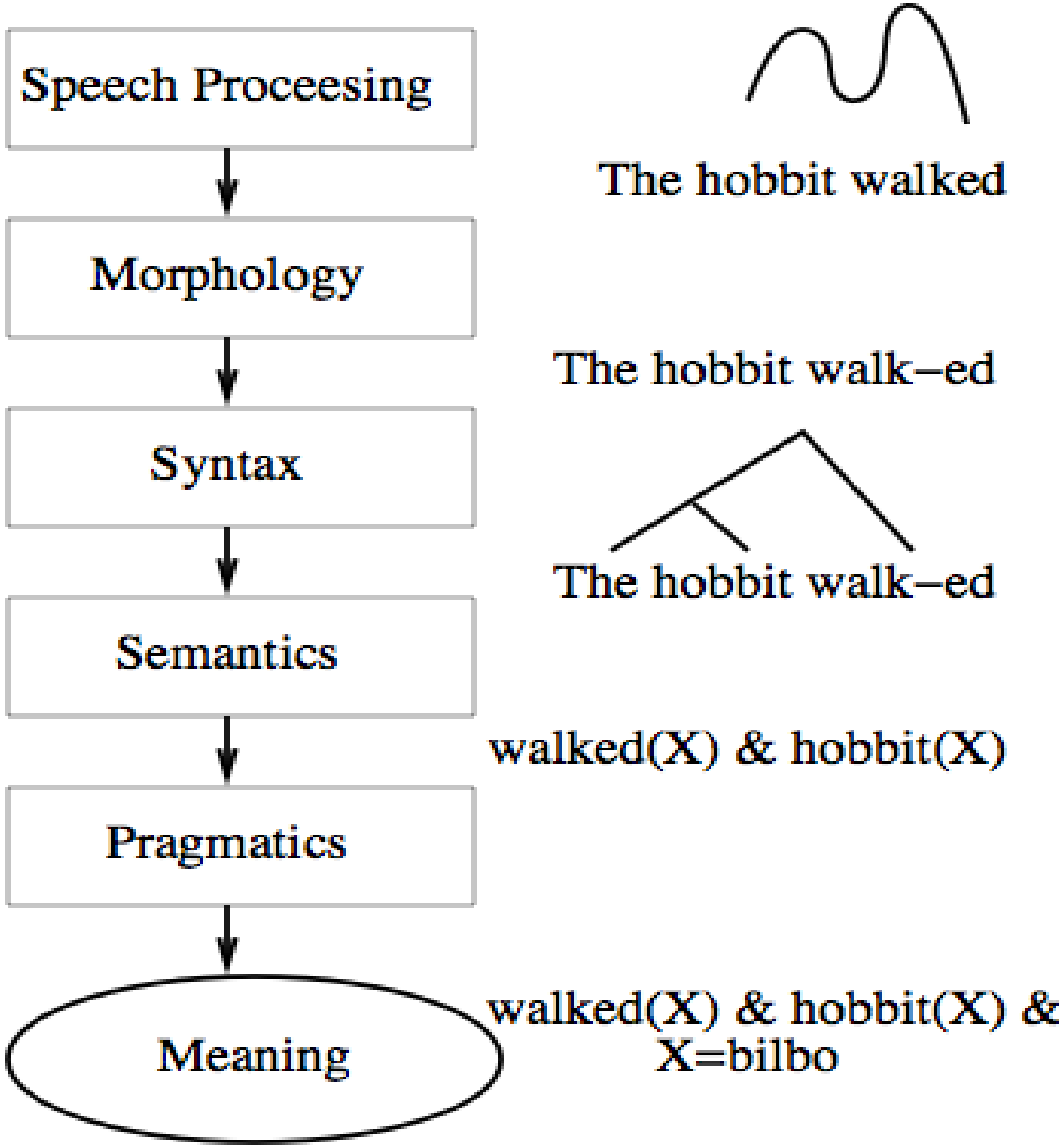
Typical Stages in a NLU + Virtual Agent system



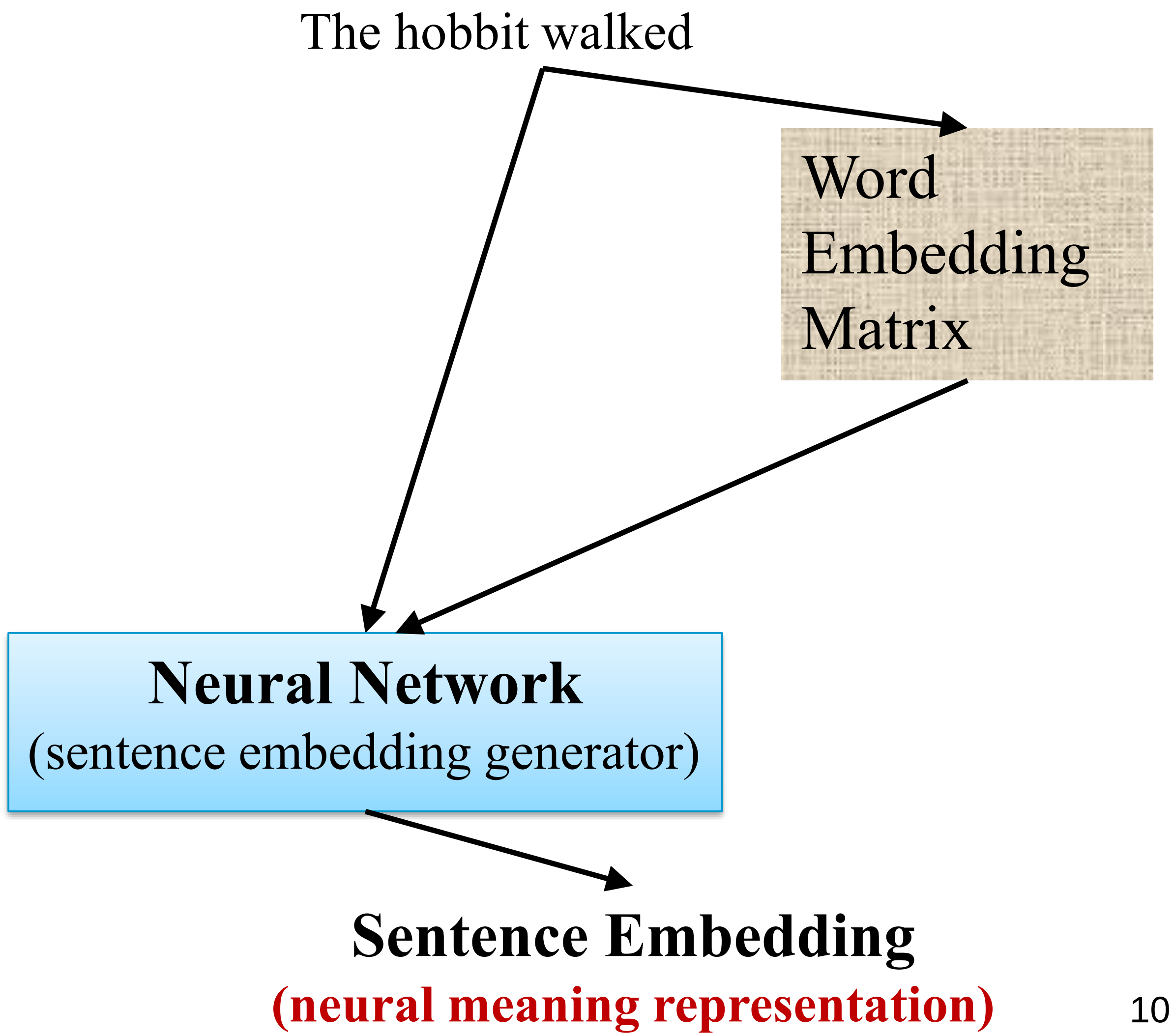
Classical NLP pipeline



Classical NLP pipeline



Deep learning pipeline

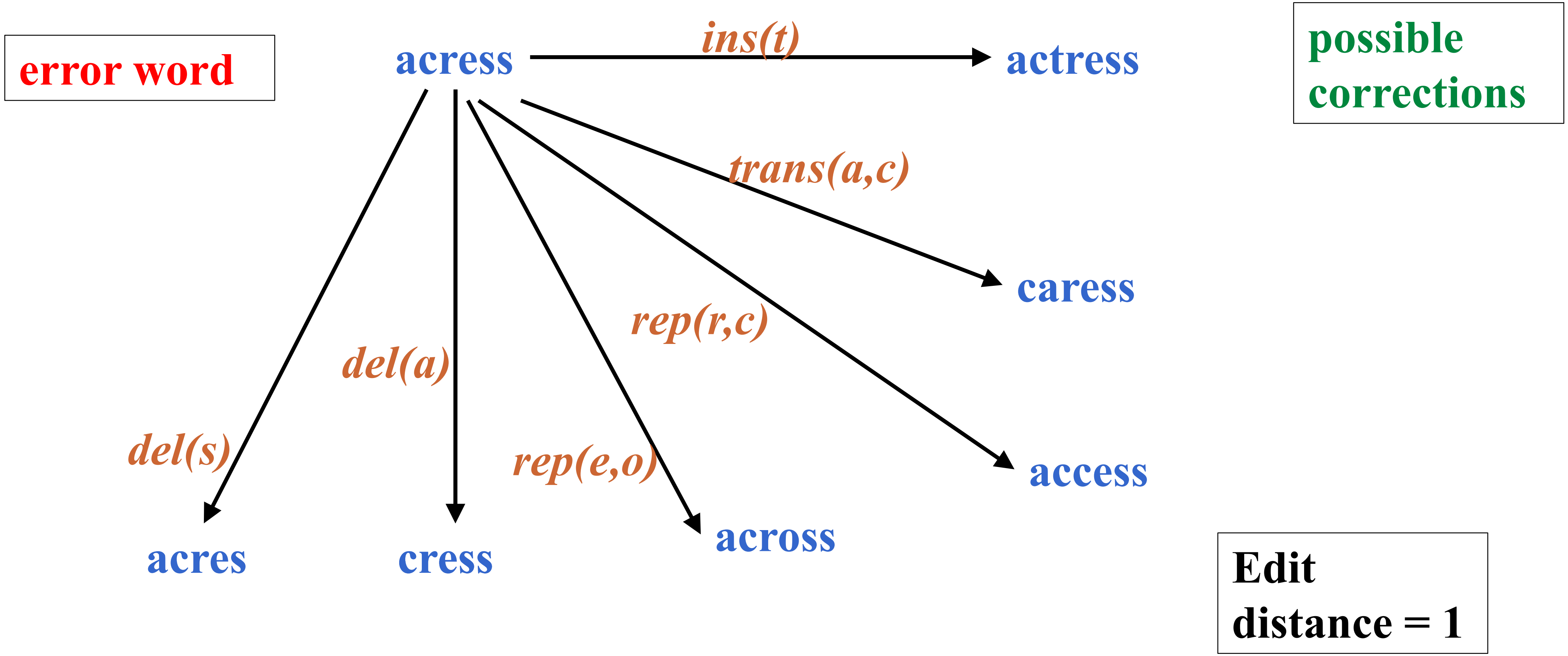


Statistical learning example

Spelling correction

- spelling errors common in user queries
  - *e.g. “Tell me about the benfits?”*
- types of errors:
  - Insertions (beinifits)
  - Deletions (benfits)
  - Transpositions (bnefits)
  - Repetitions (bennifits)
  - Replacements (benifits)
- Simple technique is *edit distance*

Edit distance



Statistical spelling correction

- $\boldsymbol{t}$ is the typo and $\boldsymbol{c}$ is the correct word

$$\boldsymbol{p(c \mid t) = \frac{p(t|c)p(c)}{p(t)}}$$

- Choose correct ***c*** using:

$$\boldsymbol{c^* = \arg\max_c p(c|t) = \arg\max p(t|c) \times p(c)}$$

- $\boldsymbol{p(t|c)}$ is too sparse
- Approximate with ***p(error_type | prev_char)***
- Corrects only single char. errors [Church et.al., 90]
- Extended to multi-error case [Brill & Moore,02]

Statistical spelling correction

- error word : acress

c	p(c)	error type	p(t|c)	p(t|c)p(c)	nearest %
actress	.0000315	p(del(t)|c)	.000117	3.69×10^{-9}	37%
cress	.000000014	p(ins(a)|` `)	.00000144	2.02×10^{-14}	0%
caress	.0000001	p(tr(c)|a)	.00000164	1.64×10^{-13}	0%
access	.000058	p(rep(c)|r)	.000000209	1.21×10^{-11}	0%
across	.00019	p(rep(o)|e)	.0000093	1.77×10^{-9}	18%
acres	.000065	p(ins(s)|e)	.0000321	2.09×10^{-9}	21%
acres	.000065	p(ins(s)|s)	.0000342	2.22×10^{-9}	23%

- estimate probabilities from real data
- incorporate domain data automatically

Statistical spelling correction

- error word : acress

c	p(c)	error type	p(t|c)	p(t|c)p(c)	nearest %
actress	.0000315	p(del(t)|c)	.000117	3.69×10^{-9}	37%
cress	.000000014	p(ins(a)|\` \`)	.00000144	2.02×10^{-14}	0%
caress	.0000001	p(tr(c)|a)	.00000164	1.64×10^{-13}	0%
access	.000058	p(rep(c)|r)	.000000209	1.21×10^{-11}	0%
across	.00019	p(rep(o)|e)	.0000093	1.77×10^{-9}	18%
acres	.000065	p(ins(s)|e)	.0000321	2.09×10^{-9}	21%
acres	.000065	p(ins(s)|s)	.0000342	2.22×10^{-9}	23%

- estimate probabilities from real data
- incorporate domain data automatically
- **Lab** – Implement above algorithm

Vector Space Models / Word Embeddings

Distributional Hypothesis

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 ⟩

Typically replace counts with PMI or PPMI (positive PMI)

Instead of using counts we can use other measures

- **Conditional probability**

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

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

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

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

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

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**?

Similarity in meaning between two words

- VSMs can recover the similarity in meaning between words e.g. using cosine similarity or KL/JS divergence
- Can be used instead of WordNet
- Thus, we expect $\boldsymbol{cos(book, novel)}$ to be high

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

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

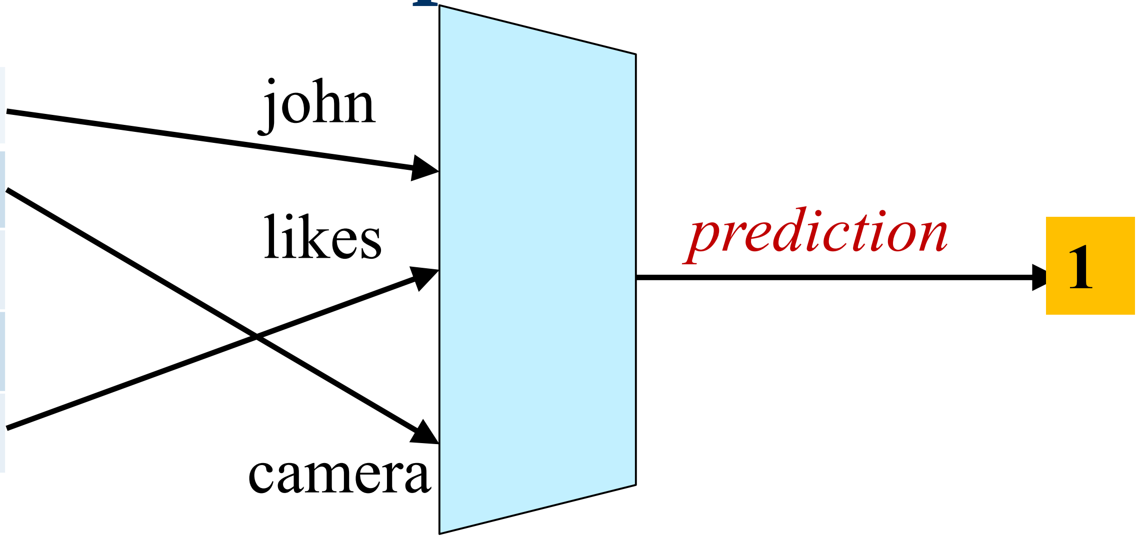
- We can set the problem of learning word/sentence meanings as a machine learning task that requires some semantic interpretation:
  - The dog ___ the cat? **(fill in the blank)**
  - I went to the party wearing a nice ____ **(predict the next word)**
  - {big, the, fat, my } dog { like, chases, bites, eats} **(predict left/right context word)**
  - I heated the food / The food got hot **(entails/contradicts/unrelated)**

Word/Sentence Embeddings – General ideas

- 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)**
  - {big, the, fat, my } dog { like, chases, bites, eats} **(predict left/right context word)**
  - I heated the food / The food got hot **(entails/contradicts/unrelated)**
- For each of these tasks we can generate a training dataset containing the correct and incorrect predictions.
- 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
- Think of the → as a machine learning model that we train using our 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

Classwork – Lets design some word embedding models

- 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

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:

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

- Per example, log likelihood can be written as:

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

- [Aside] Derivative of the log of sigmoid:

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

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

- 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 $\boldsymbol{w}$:

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

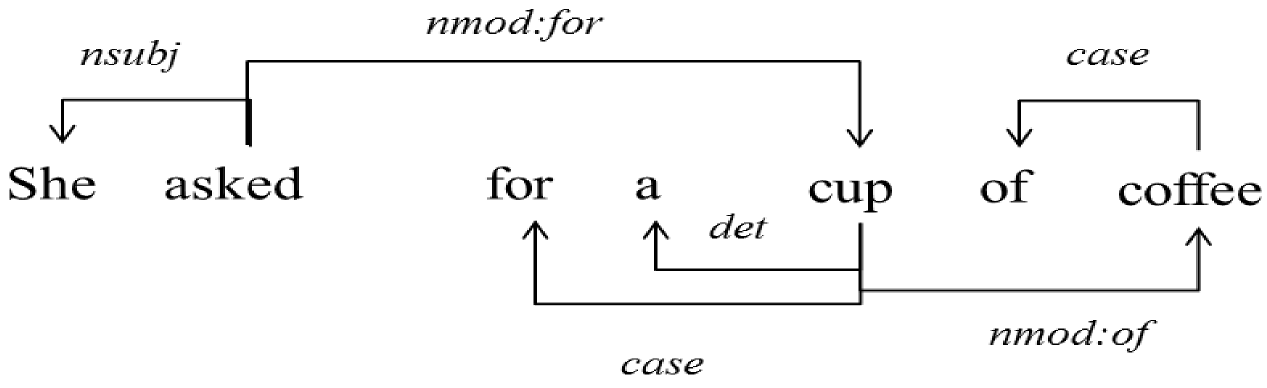
$$\bar{\boldsymbol{c}} \coloneqq \bar{\boldsymbol{c}} + \boldsymbol{\eta}\big(\boldsymbol{\sigma}(-\bar{\boldsymbol{w}}.\bar{\boldsymbol{c}})(\bar{\boldsymbol{w}})\big)$$

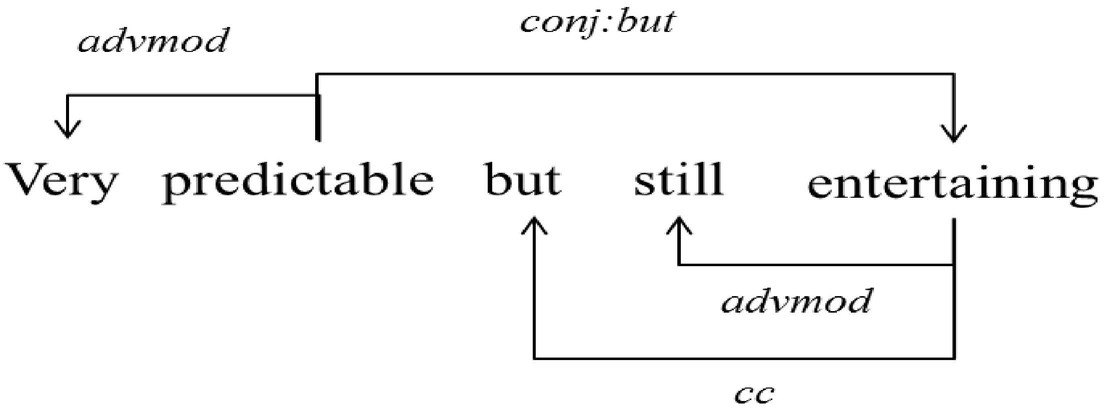
For each $\boldsymbol{c_i} \in \boldsymbol{C_N}$:

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

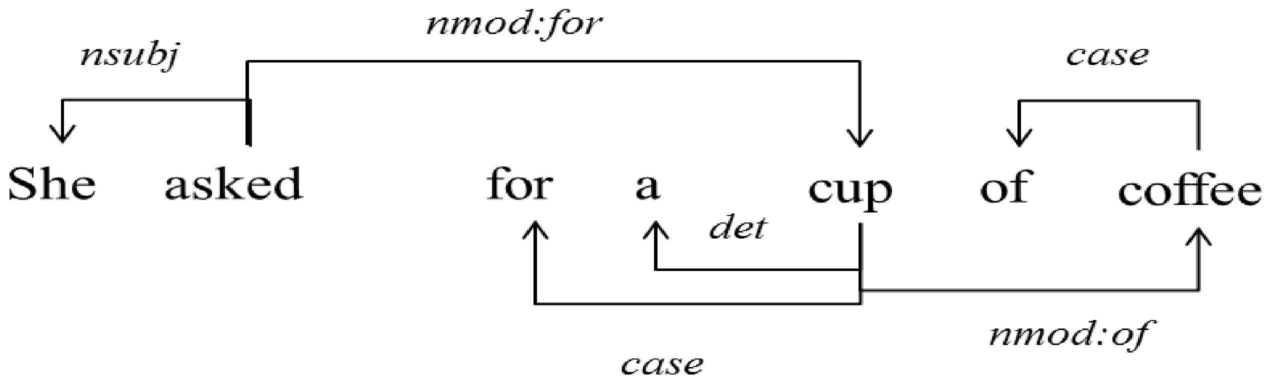
- Move pointer to the next word
- Stop after a fixed number of iterations

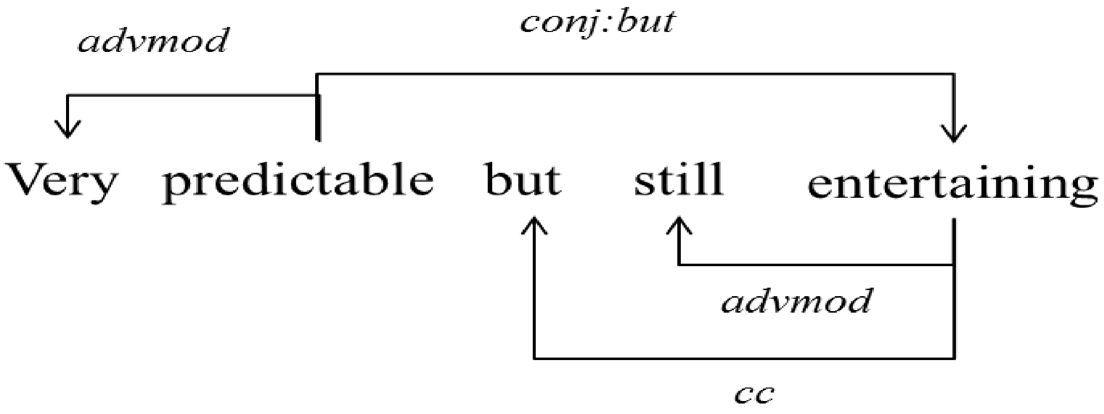
Dependency Parsing





Dependency based word embeddings





Target - **cup**

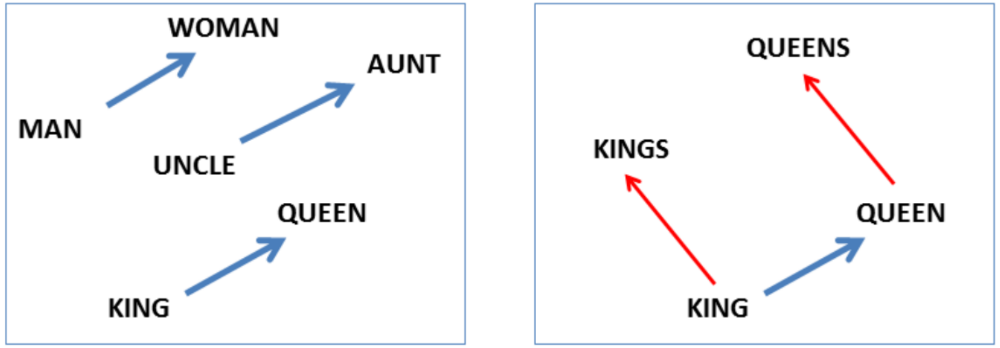
Context words :
She, asked, for, a, of, coffee

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

from [Komninos and Manandhar, 2016]

Analogy tasks

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



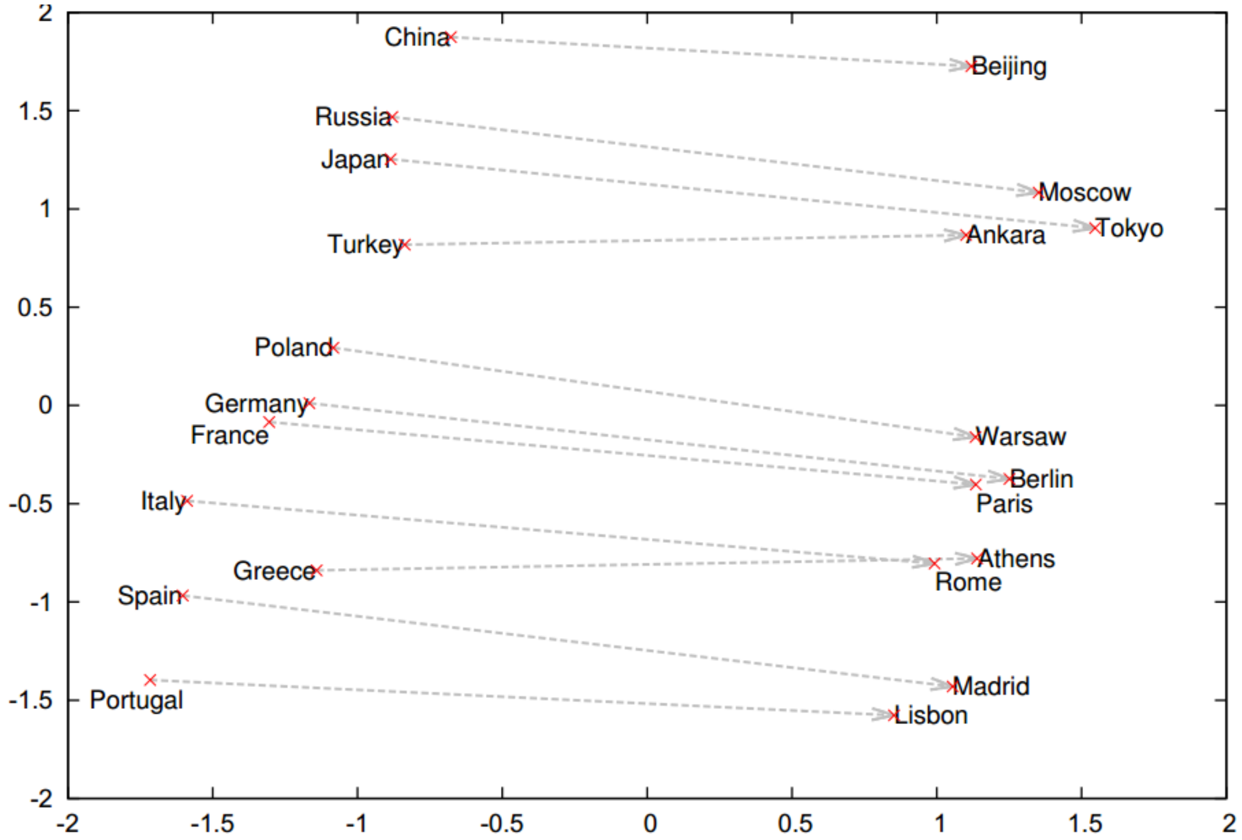
- England − London + Baghdad = ? Iraq
- Equivalently:

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

- Directional similarity

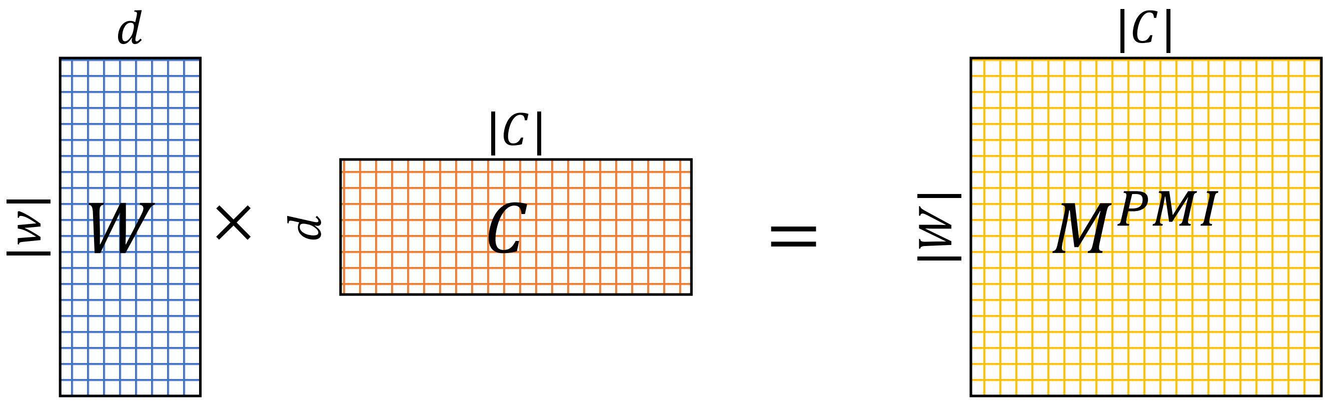
Slide from Omer Levy

Directional similarity example



Skipgram embeddings ≃ Matrix factorization

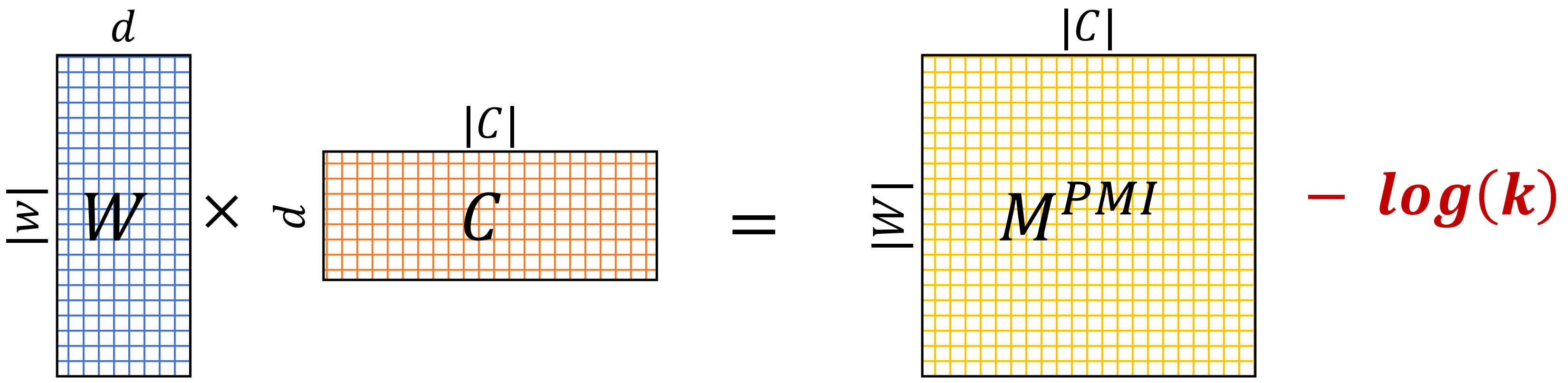
- The skipgram model learns a matrix factorization of the PMI matrix



Slide from Omer Levy

Skipgram embeddings $\simeq$ Matrix factorization

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



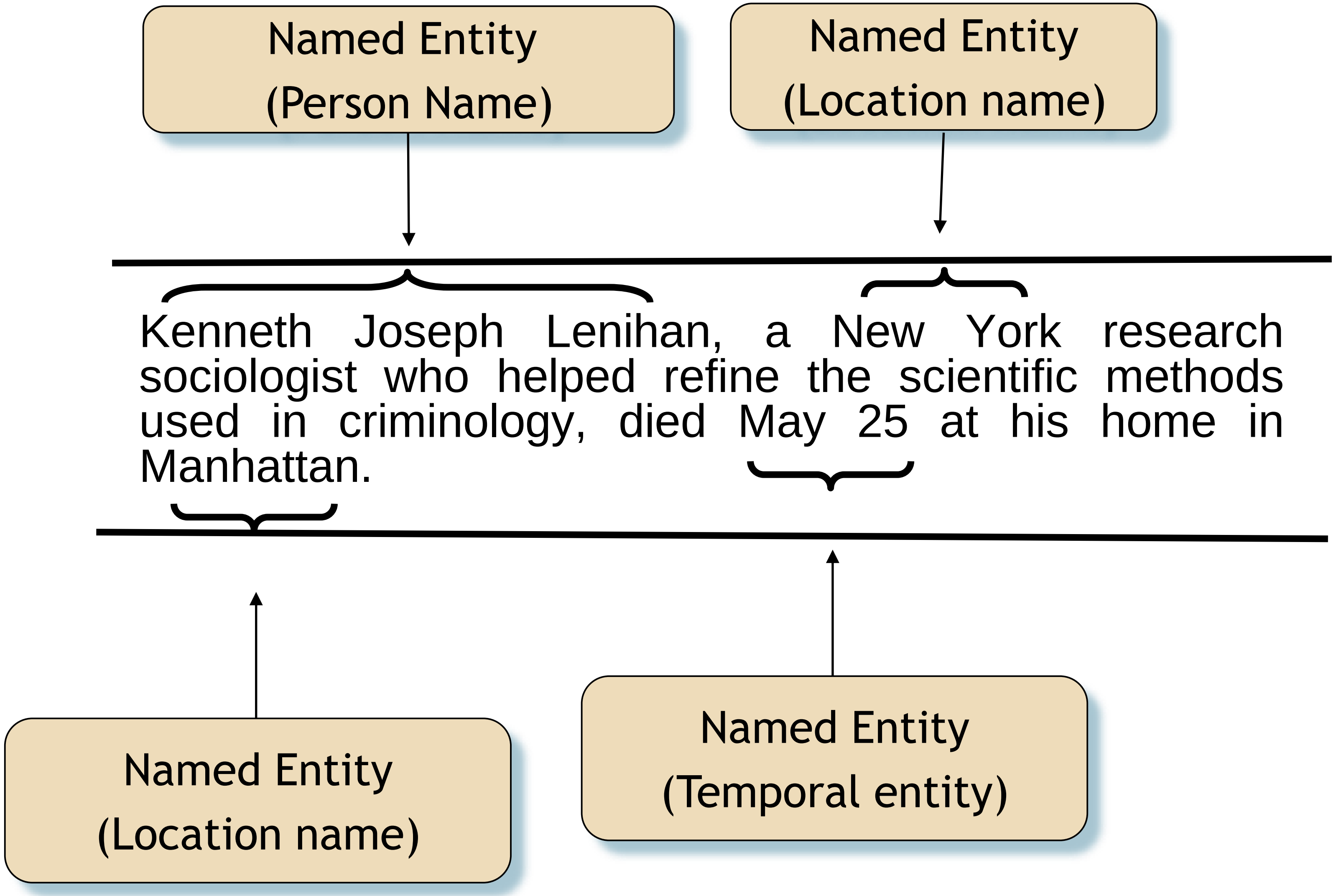
Slide from Omer Levy

Sequence Classification tasks in NLP

Typically want to classify a whole sentence or each word of a sentence.

- Named entity recognition
- Relation extraction
- Part of Speech tagging
- Sentiment analysis

Named entity recognition



Relation Extraction

Source: http://news.bbc.co.uk/1/hi/uk/222225.stm

30 September 2008: **Norwich City** [ORG.SPO] v **Birmingham City** [ORG.SPO]: Twenty Birmingham fans [PER.Group] sprayed rival supporters with CS gas [WEA] and attacked them with bar stools in a pub.

Relation Extraction

Identified relations

R1: [Twenty Birmingham fans, Birmingham City FC] – Membership

R2: [Twenty Birmingham fans, CS gas] – Chemical weapon possession

R3: [Twenty Birmingham fans, bar stools] – Weapon possession

Sequence Classification tasks in NLP

Coarse grained sentiment analysis

Sentiment class labels:
Negative Positive

Examples:
The food was not that bad : Positive

The food was great: Positive

The food was OK : Positive

I hated the curry : Negative

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

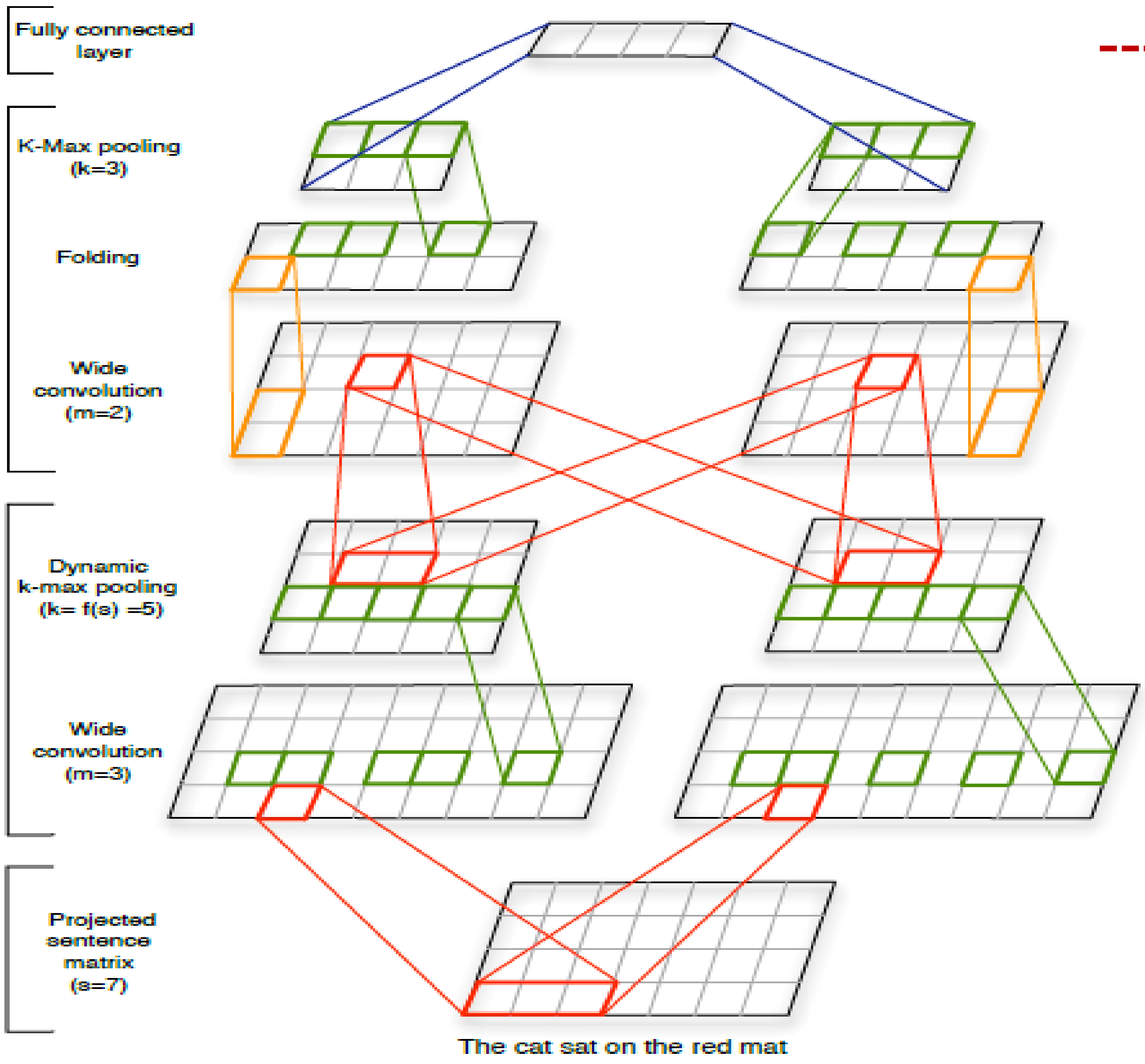
$$\mathbf{s} = \begin{bmatrix} | & & | \\ \mathbf{w}_1 & \cdots & \mathbf{w}_s \\ | & & | \end{bmatrix}$$

- Compute row wise max

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

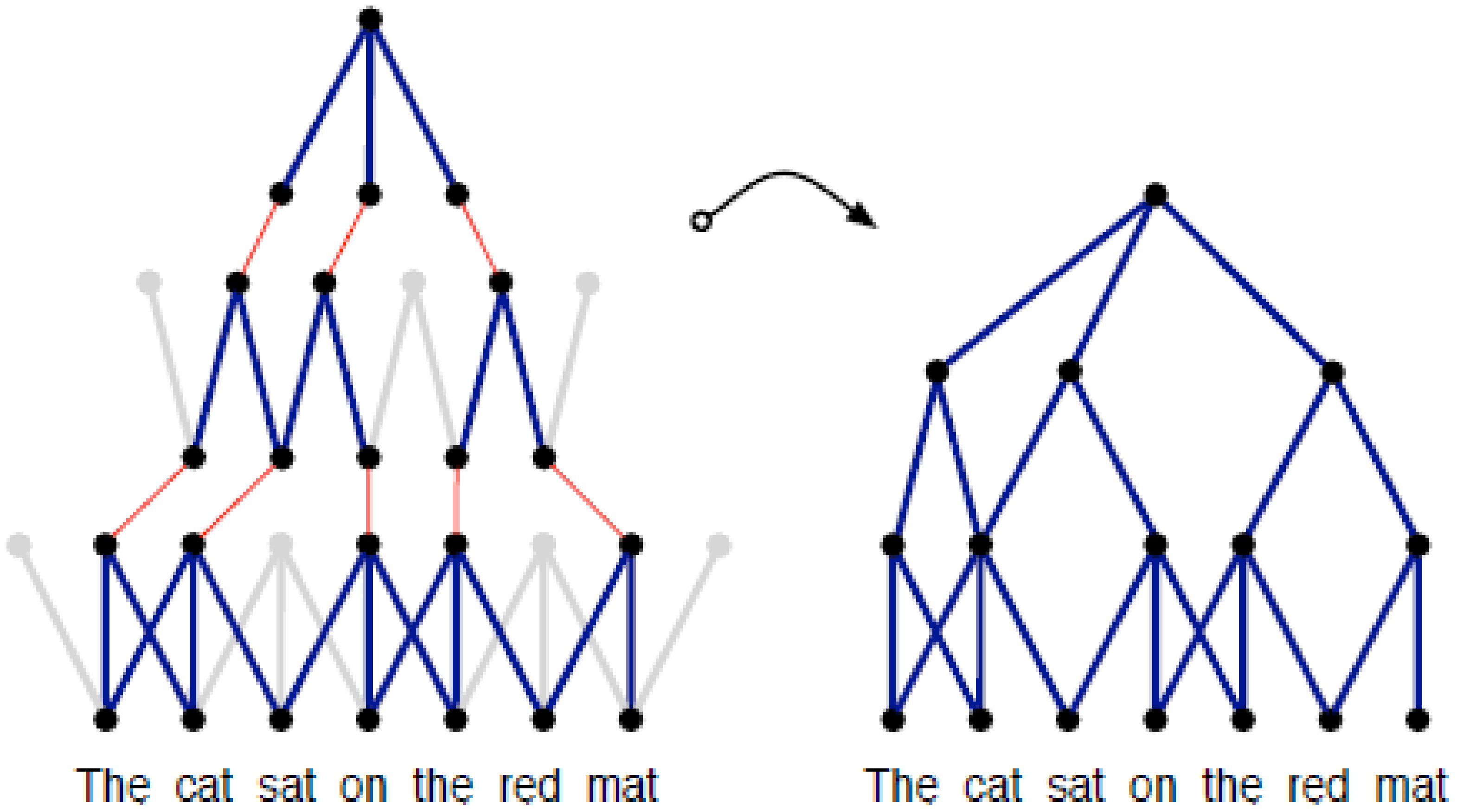
Task specific representation CNN architecture

--- for classification problems



- CNN architecture from [*Kalchbrenner et. al. 2014*]

CNN example



- The convolution and pooling layers learn the correct syntactic structure and how these need to be combined (from [Kalchbrenner et. al. 2014])

Coarse grained sentiment analysis

Embeddings	SVM	CNN	LSTM
Win5 Words	80.1	83.5	76.1
Win5 AvgE	79.5	83.2	76.9
Win5 ConcE	80.3	82.9	77.6
LG Words	78.5	84.5	77.2
LG Dep	76.0	76.8	69.1
LG Wavg	78.9	82.0	78.6
LG Conc	79.8	82.7	79.7
EXT Words	80.5	84.1	77.6
EXT Dep	77.7	77.2	69.6
EXT Wavg	**80.6**	**84.6**	75.7
EXT Conc	**80.6**	83.5	**79.8**
CNN-multichannel	**88.1**		

Compositional Semantics – Long distance dependencies

- Lord of the Rings, I read.
- Lord of the Rings, I managed to read.
- Lord of the Rings, I believe John managed to read.
- I believe Lord of the Rings, John managed to read.

- [Lord of the Rings]$_i$, I read $\_\_ _i$.
- [Lord of the Rings]$_i$, I managed to read $\_\_ _i$.
- [Lord of the Rings]$_i$, I believe John managed to read $\_\_ _i$.
- I believe [Lord of the Rings]$_i$, John managed to read $\_\_ _i$.

- Doing deep compositional semantics is still a challenge for current machine learning methods

Constraints on gaps

Gaps must be strictly subcategorised

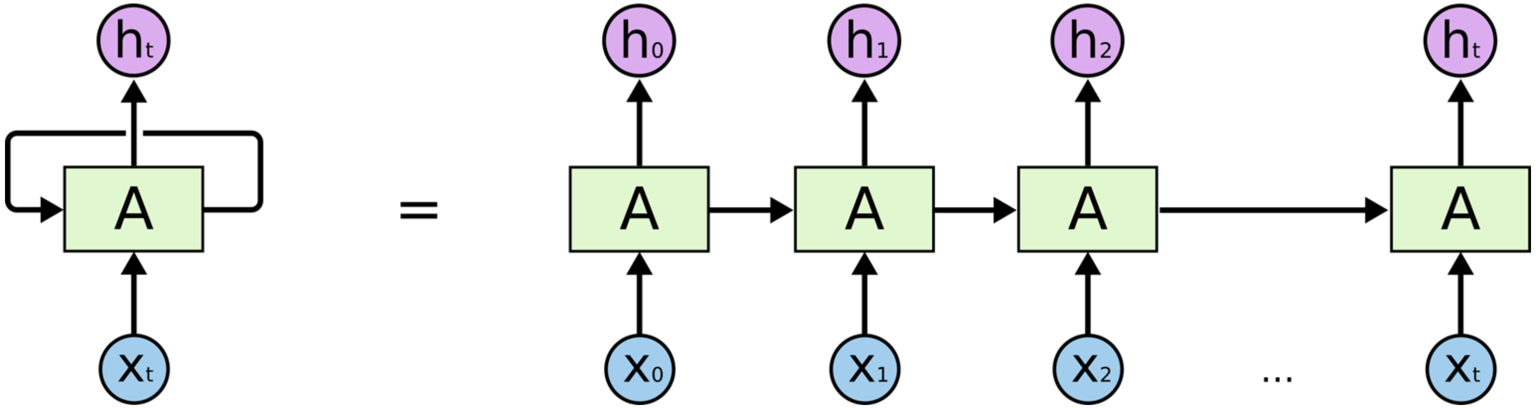
- *Kim$_i$, Sandy loves Bill __$_i$.
- Sandy loves the man in the grey suit.
- *[in the grey suit]$_i$, Sandy loves the man __$_i$.

Syntactic barriers on gaps

- Mary ordered cake and soda.
- *What$_i$ did Mary order cake and __$_i$. (co-ordinate structure)
- John saw Mary’s brother.
- *Whose$_i$ did John see __$_i$ brother ? (possesive NP)
- That John liked Mary surprised everyone.
- *Who$_i$ did that John like __$_i$ surprise everyone? (sentential subject NP)

Sequence models

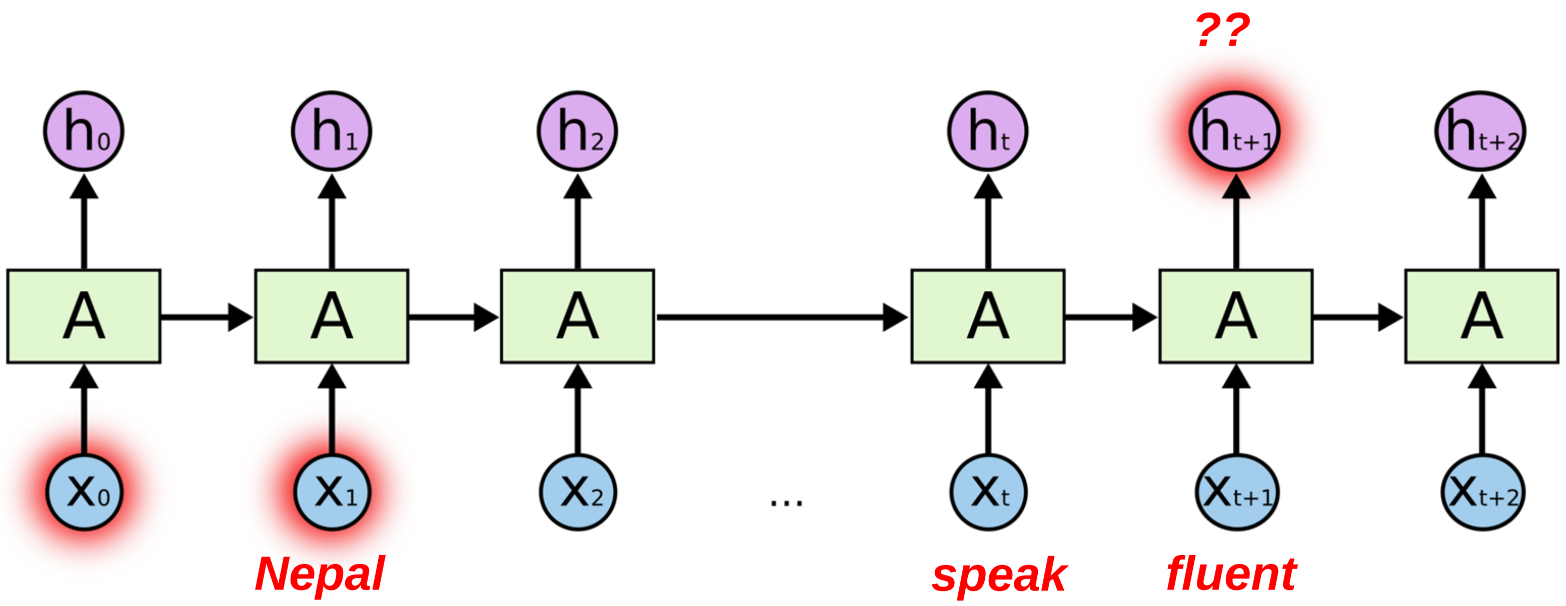
Recurrent nets (RNNs)



- 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

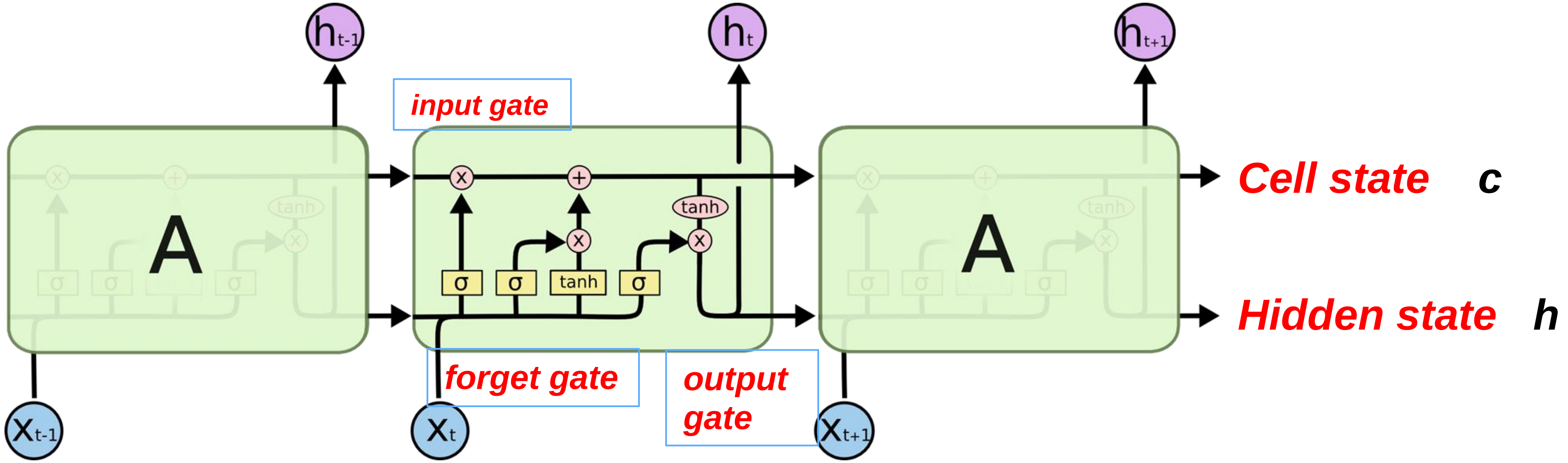
Issues with standard RNNs



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

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

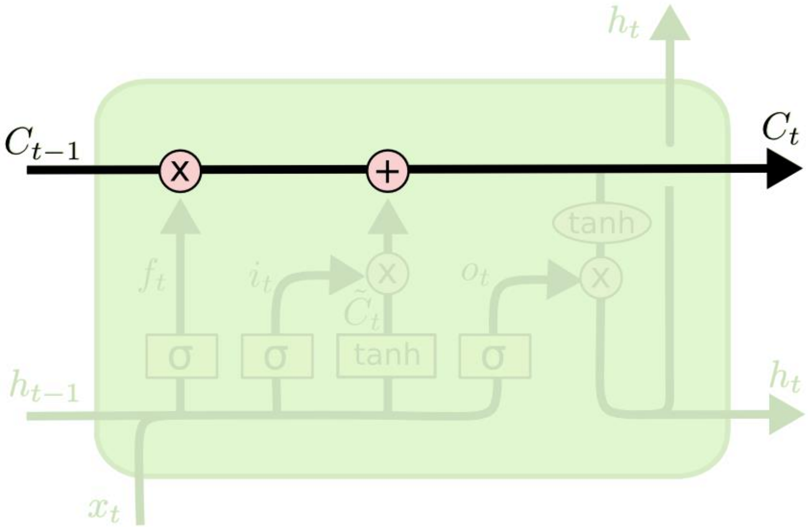
LSTM networks



- 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

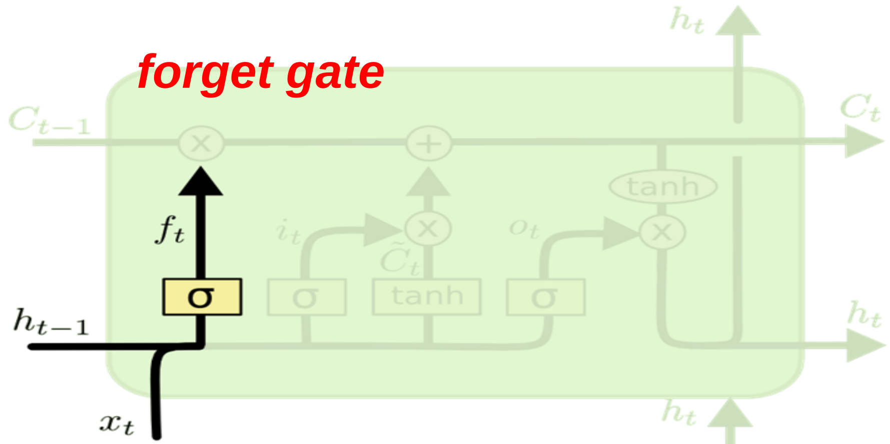
LSTM networks



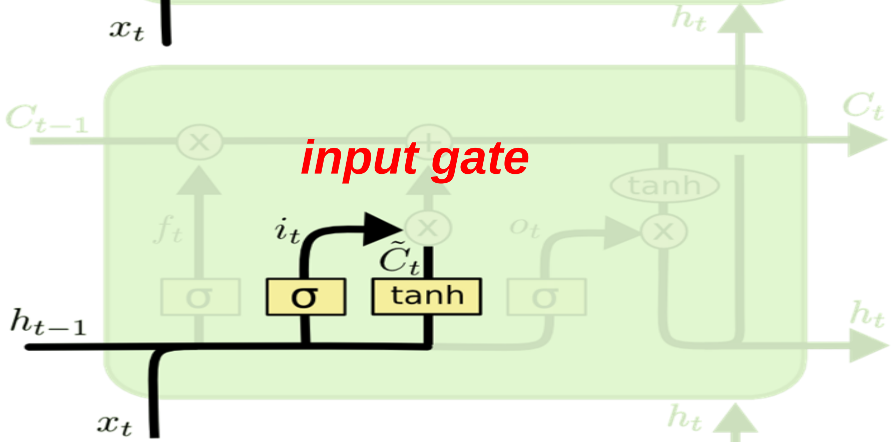
Cell state

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

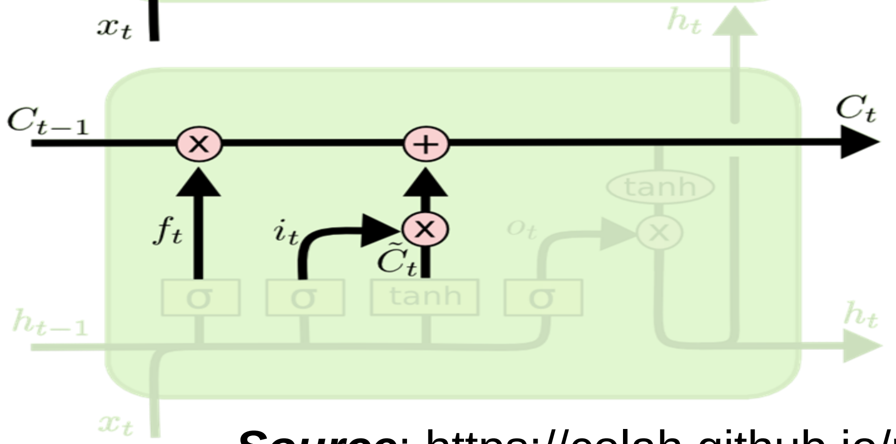
LSTM networks



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



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

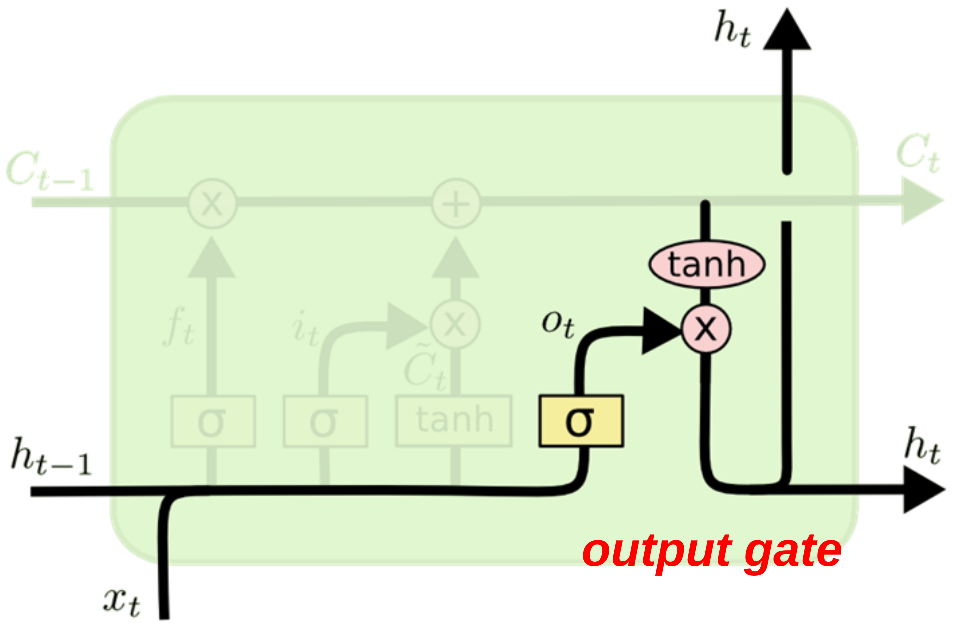


Cell state ***c***

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

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

LSTM networks



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

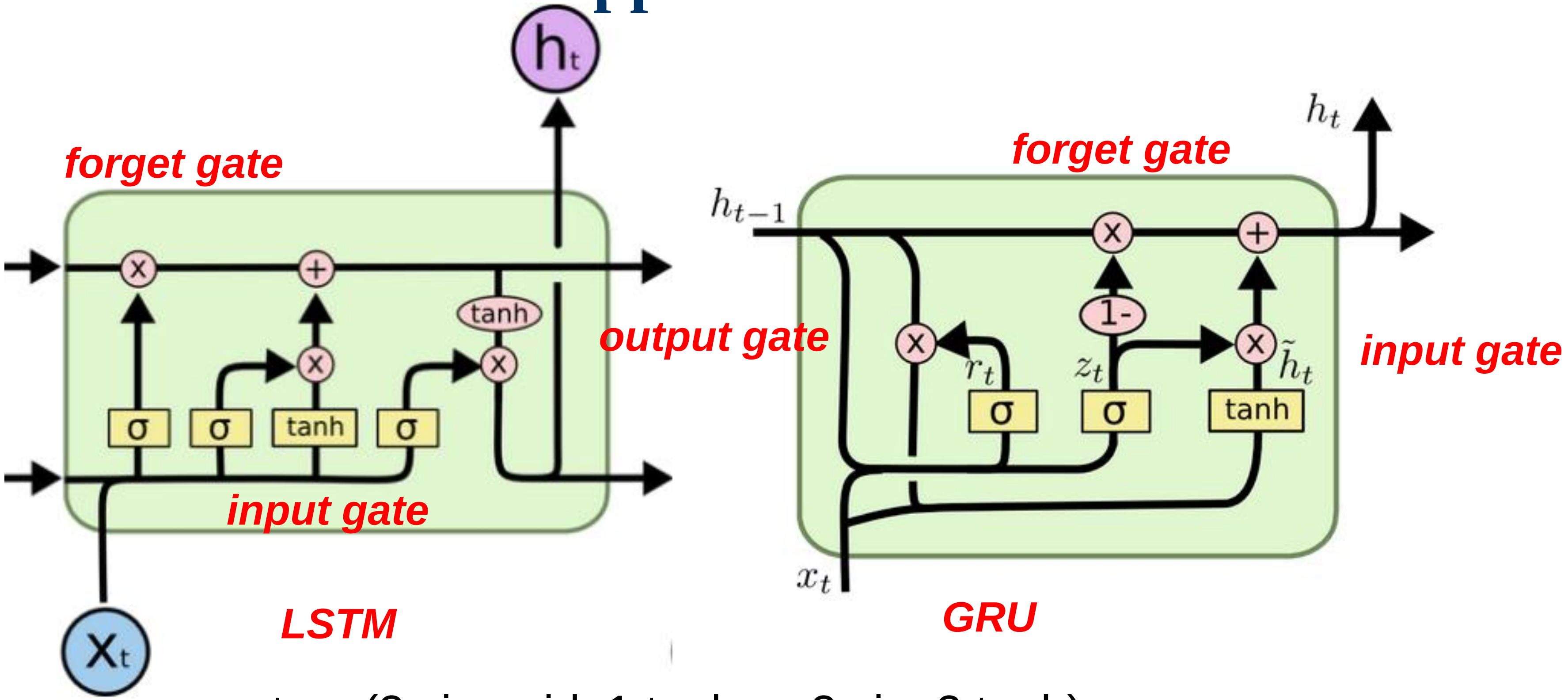
$$h_t = o_t * \tanh\left(C_t\right)$$

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

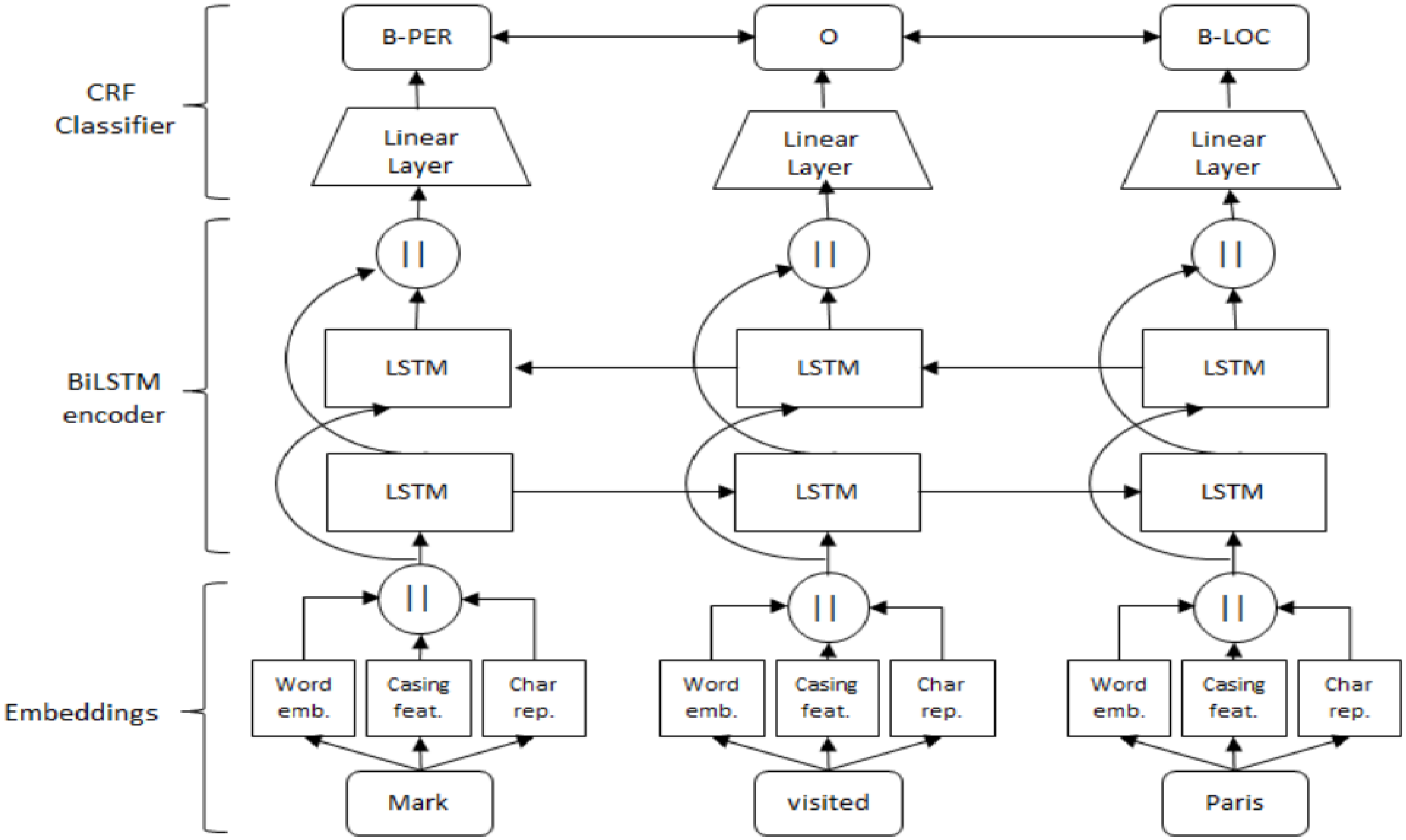
Using LSTM networks in NLP applications



- 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

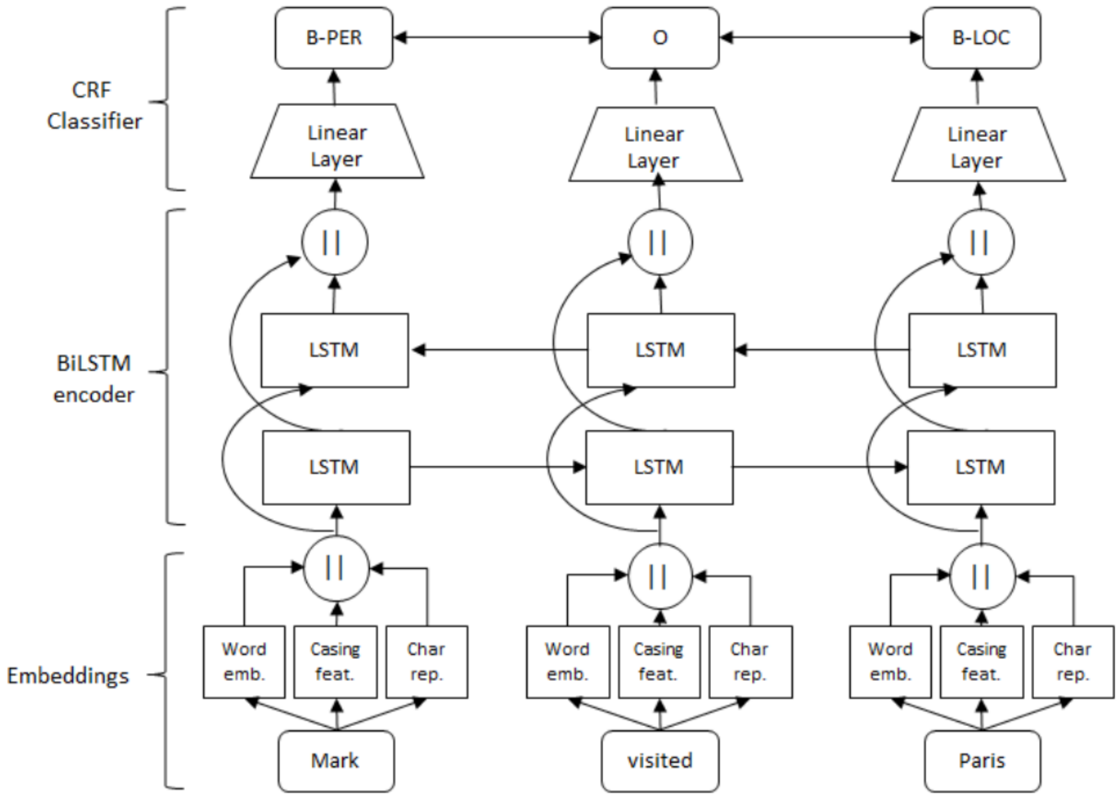
Source: https://colah.github.io/posts/2015-08-Understanding-LSTMs

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%

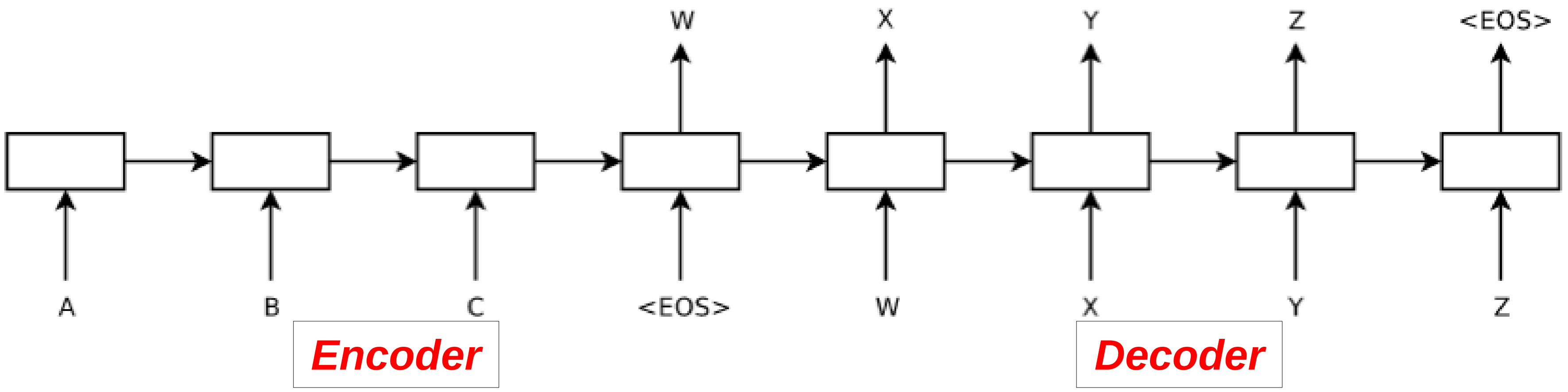
Training data sizes: **GloVe3 840B** **Komn. 2B**

- ***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

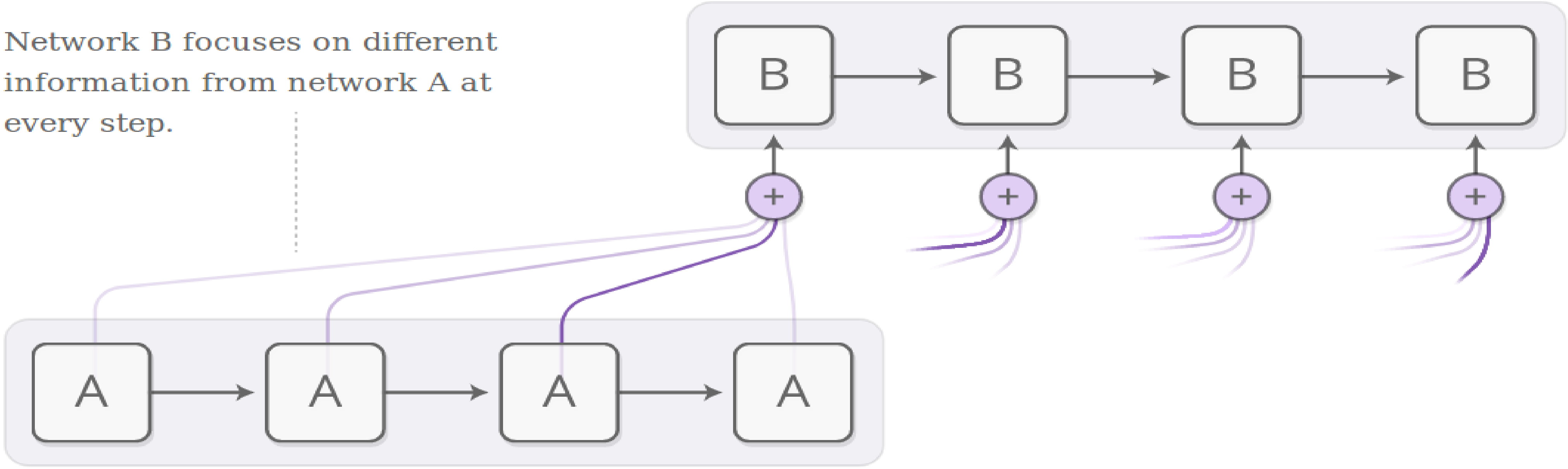
Encoder-Decoder Architectures



- 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***

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

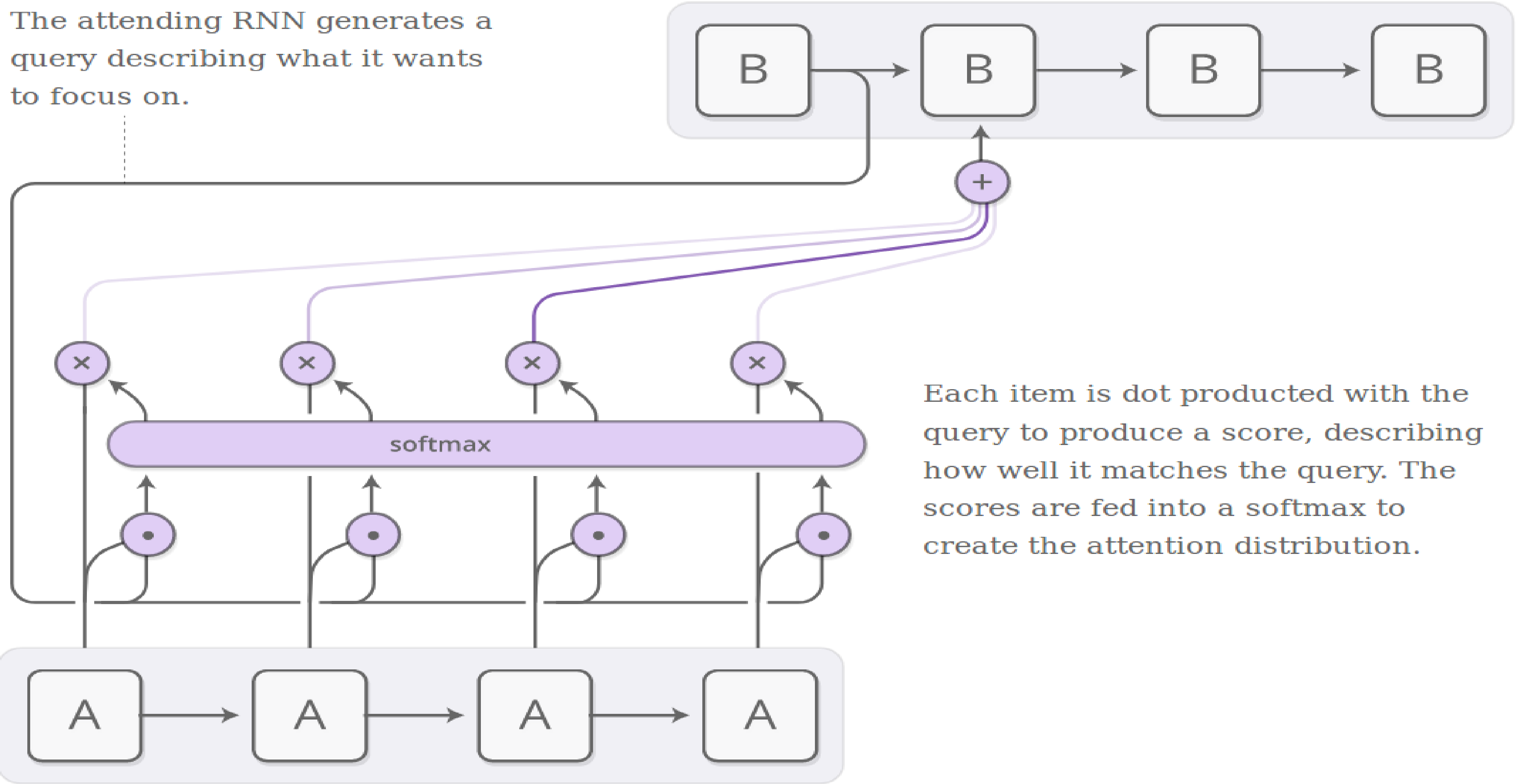
Attention based models



- **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 **<u>hidden output from B</u>** 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 **<u>most relevant outputs from A</u>**

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

Attention based models

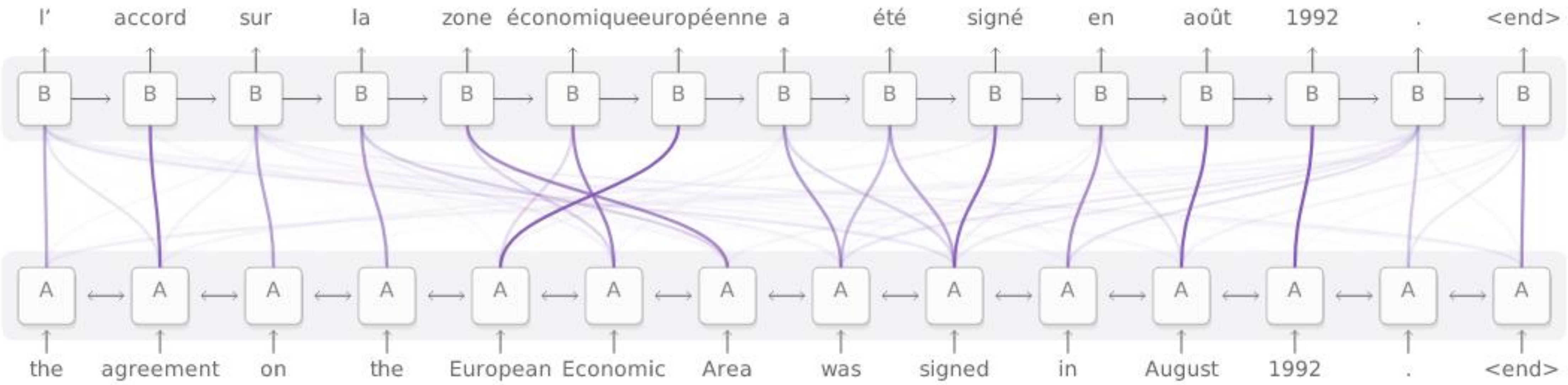


Diagram derived from Fig. 3 of Bahdanau, *et al.* 2014

- The attention mechanism generates a simpler architecture compared to the encoder-decoder setup
- In the 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/

Slides Credits [some slides are from]:

Kalchbrenner N., Grefenstette E., Blunsom P. A Convolutional Neural Network for Modelling Sentences. ACL 2014.

Alexandros Komninos and Suresh Manandhar. Dependency based embeddings for sentence classification tasks. NAACL 2016.

Omer Levy and Yoav Goldberg. Neural word embeddings as implicit matrix factorization. NIPS 2014.

Omer Levy

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

Olah and Carter https://distill.pub/2016