

AdvNLP/E Lecture 1

Lexical & Distributional Semantics

Dr Julie Weeds, Spring 2021



Lecture 1 Overview

PART 1

- Lexical semantics
 - word senses
 - semantic relationships
 - WordNet
 - semantic similarity measures based on WordNet
 - evaluation

PART 2

- Distributional Semantics
 - bootstrapping semantics from context
 - cosine similarity
 - (positive) pointwise mutual information
 - evaluation
 - word ambiguity
 - semantic relationships
 - sparsity

Much of this is revision from NLE / Applied NLP!

Lexical Semantics

Lecture 1, part 1

Dr Julie Weeds, Spring 2021



Word senses

- Words are often **ambiguous**
- Words can have **multiple senses** (i.e., meanings)

*I placed the book on the **counter**.*

vs

*I placed my **counter** on the gameboard.*

- How many more senses of **counter** can you think of?

Dictionaries

- **Lexicographers** produce dictionaries which:
 - enumerate the senses of all of the words in a language
 - provide definitions of different sense
 - provide examples of usage of different senses

WordNet online search:

<http://wordnetweb.princeton.edu/perl/webwn>

Oxford English Dictionary online search:

https://en.oxforddictionaries.com/?utm_source=od-panel&utm_campaign=en

How many different senses do words have?

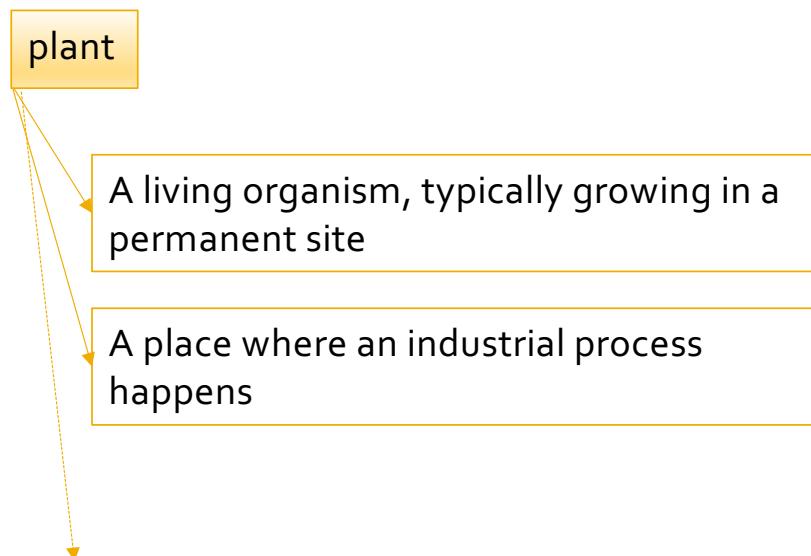
	WordNet	Oxford
plant	Noun:4, Verb:6	N:6, V:11
chicken	Noun:4, adj:1	N:4, V:1, J: 1
book	Noun: 11, Verb:4	N:14, V:9
twig	Noun:1, Verb: 2	N:2
counter	Noun: 9, Verb:2, adj:1, adverb: 1	N:13, V: 3, J: 1, R: 1

Dictionaries do not always agree on this! [Why is it so difficult?](#)

Sense distinctions

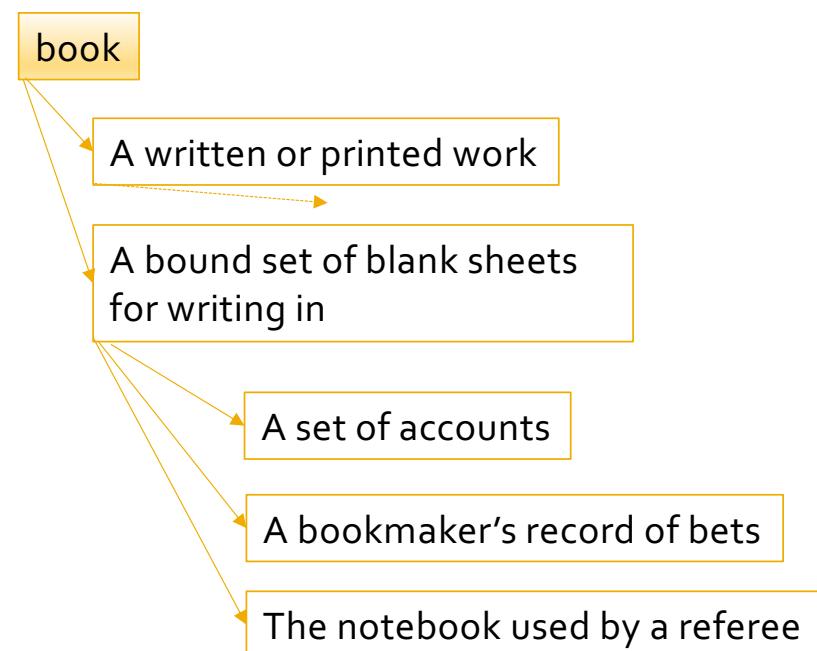
HOMONYMY

- Broad distinctions



POLYSEMY

- fine-grained distinctions



Lexical semantic relationships

- synonymy
- antonymy
- hyponymy / hypernymy
- meronymy / holonymy
- topical relatedness

Synonymy

fast

==

quickly

- Words which mean the same thing
- *Two words are **synonymous** if they can be substituted in all possible contexts without changing the meaning of the utterance.*
- True synonyms are very rare
- Choice of synonym usually gives us some extra information about the situation or speaker e.g., *car* vs *automobile*
- It is often defined as a relationship between word senses rather than between words. e.g., *plant* == *spy* ?

Antonymy

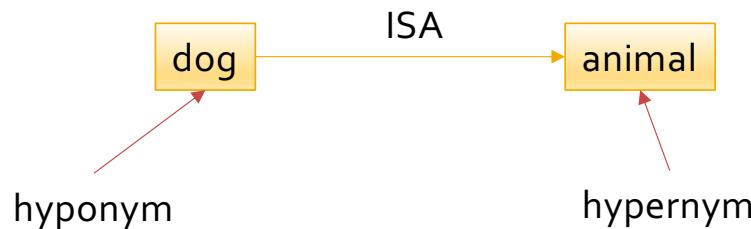
hot

≠

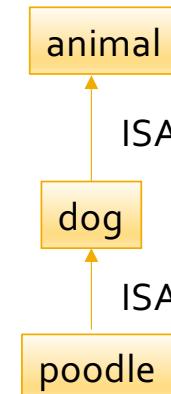
cold

- Words which are opposite in meaning
- Substituting one for the other would often cause a contradiction:
 - *The food is hot.*
 - *The food is cold.*
- Antonyms are actually very similar in meaning
 - *hot* and *cold* both describe the temperature of an object
 - *rise* and *fall* both describe an object which is moving in the vertical plane
- Most antonym pairs are adjectives, verbs or adverbs

Hyponymy and Hypernymy

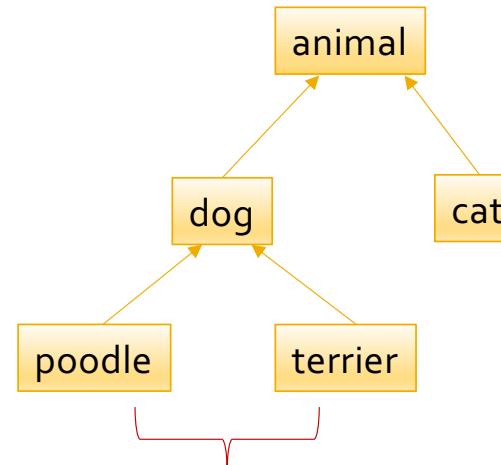


- Linguistic terms which capture the idea of class inclusion
- A *dog* is a type of *animal* so:
 - *dog* is a **hyponym** of *animal*
 - *animal* is a **hypernym** of *dog*
- It's a transitive relationship so
 - If *dog* is a hyponym of *animal*
 - And *poodle* is a hyponym of *dog*
 - *Poodle* is also a hyponym of *animal*



Hyponym Hierarchies

- The hyponymy relationship links together large numbers of concepts in a tree or hierarchy
- Most general superclass at the top
- Most specific types at the leaves



Words which share a common hypernym are called **co-hyponyms**

WordNet

- More than an electronic dictionary!
- See <http://wordnet.princeton.edu> for more general information
- Or see: Christiane Fellbaum (1998, ed.) *WordNet: An Electronic Lexical Database*. MIT Press.

WordNet

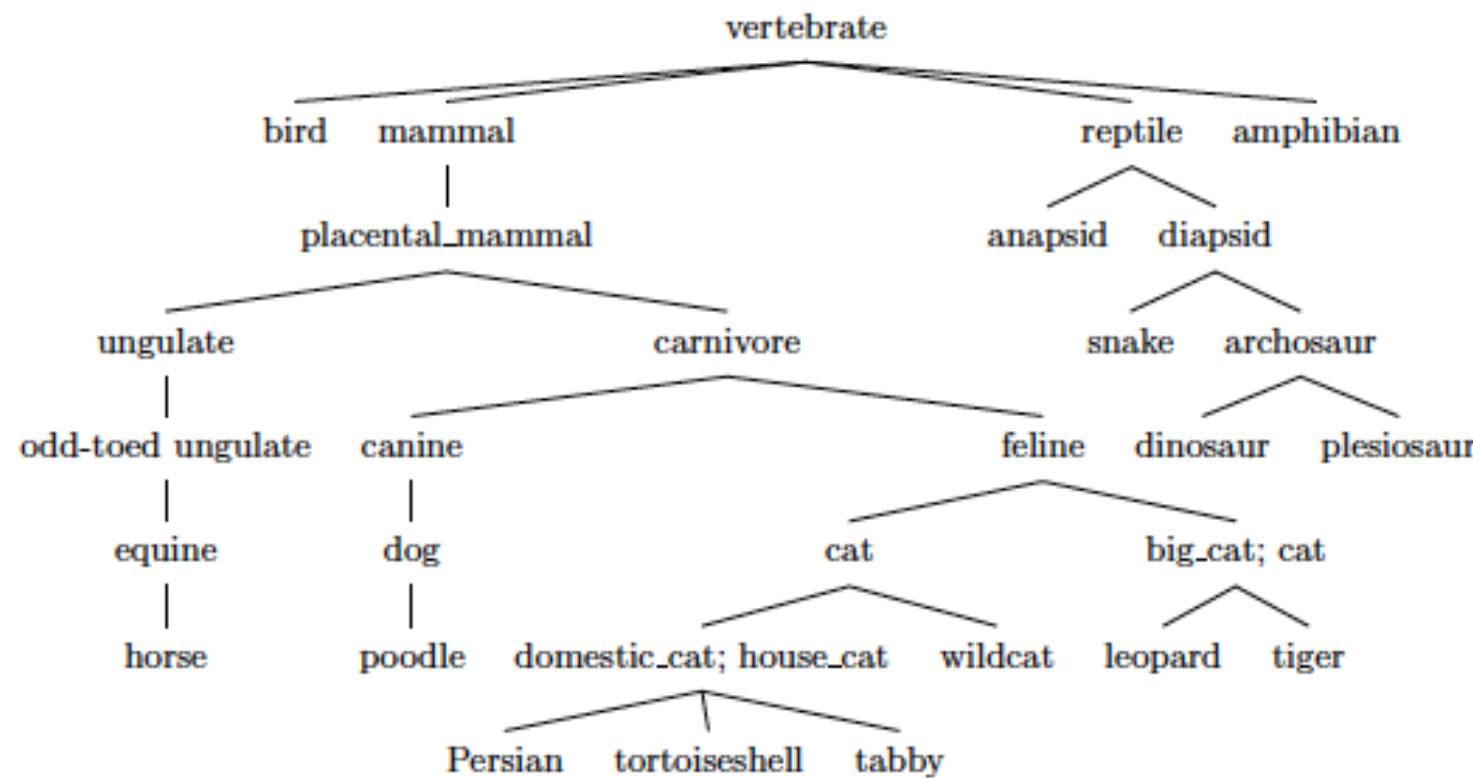
- A linguistic network organized around **synonymy** and **hyponymy**
- Core unit is the **synset**
 - a set of synonymous word senses
 - a set may contain a single word
 - synset items may be bigrams (e.g., “plant life”) as well as unigrams
 - each synset is also associated with a single definition
- Polysemous words appear in multiple synsets
 - One for each sense
- Synsets are then connected via hyponymy.....

{**plant, flora, plant life**} = a living organism lacking the power of locomotion

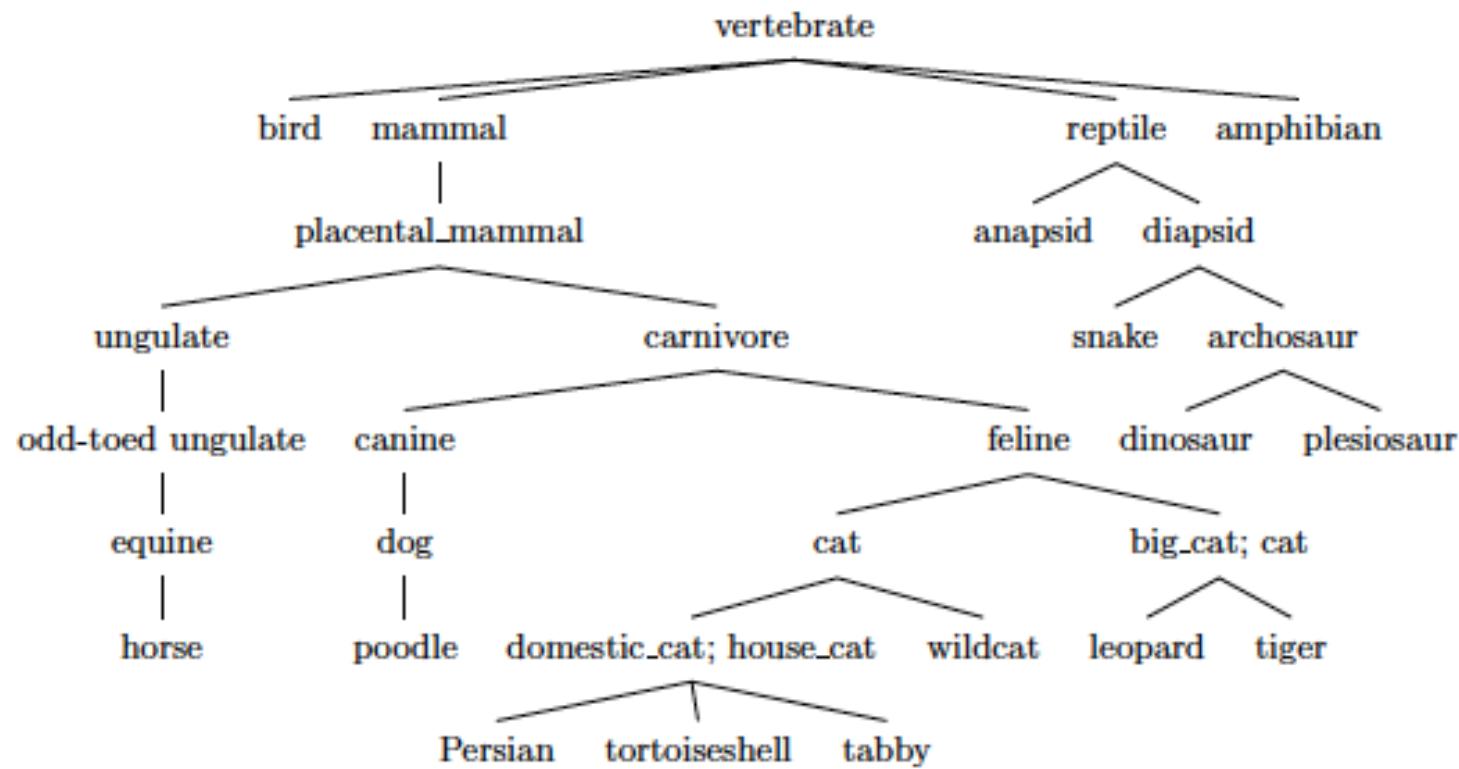
{**plant**} = something planted secretly for discovery by another

{**plant, works, industrial plant**} = buildings for carrying on industrial labour

Extract from the WordNet noun hierarchy

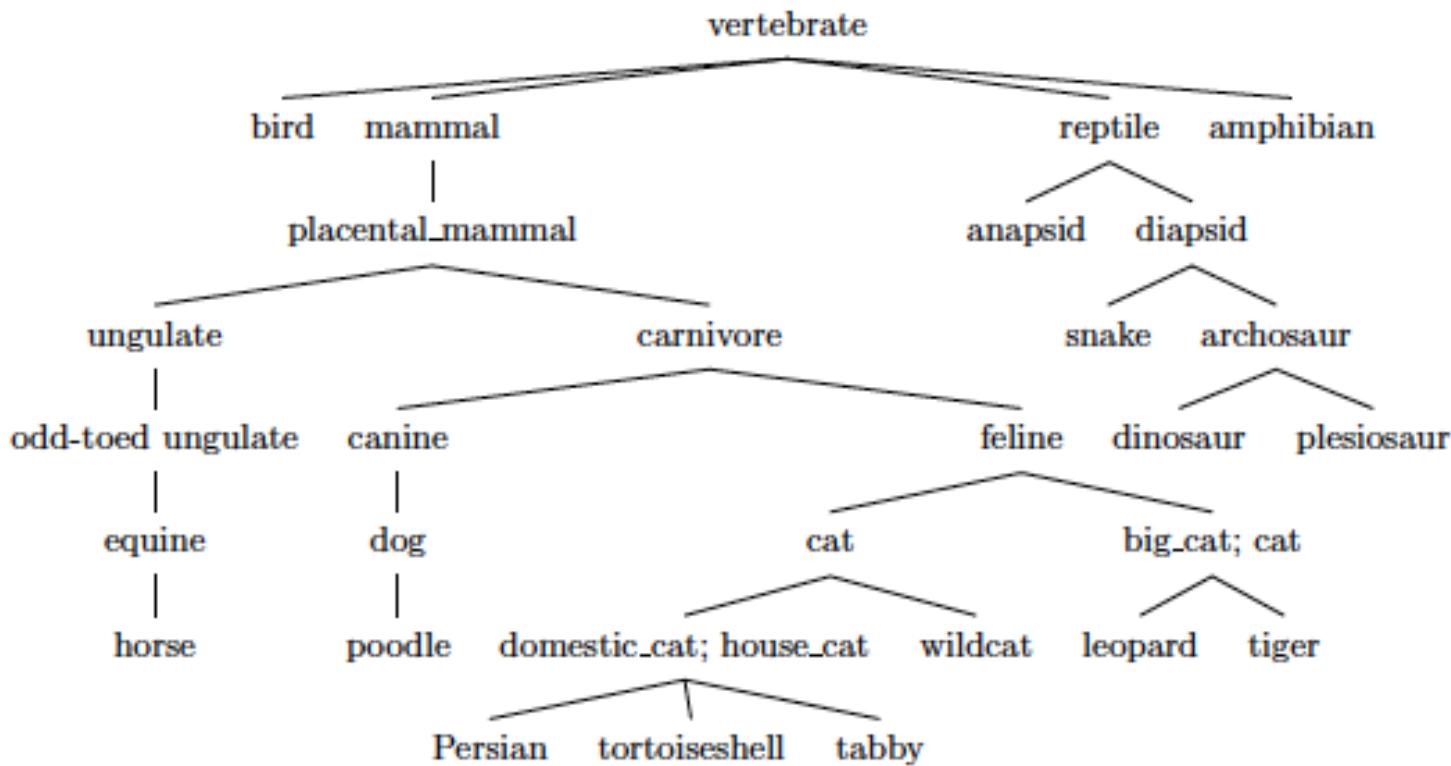


Semantic similarity based on WordNet



Intuition: More similar concepts are closer together in the hierarchy.

Path length: shorter path -> greater similarity

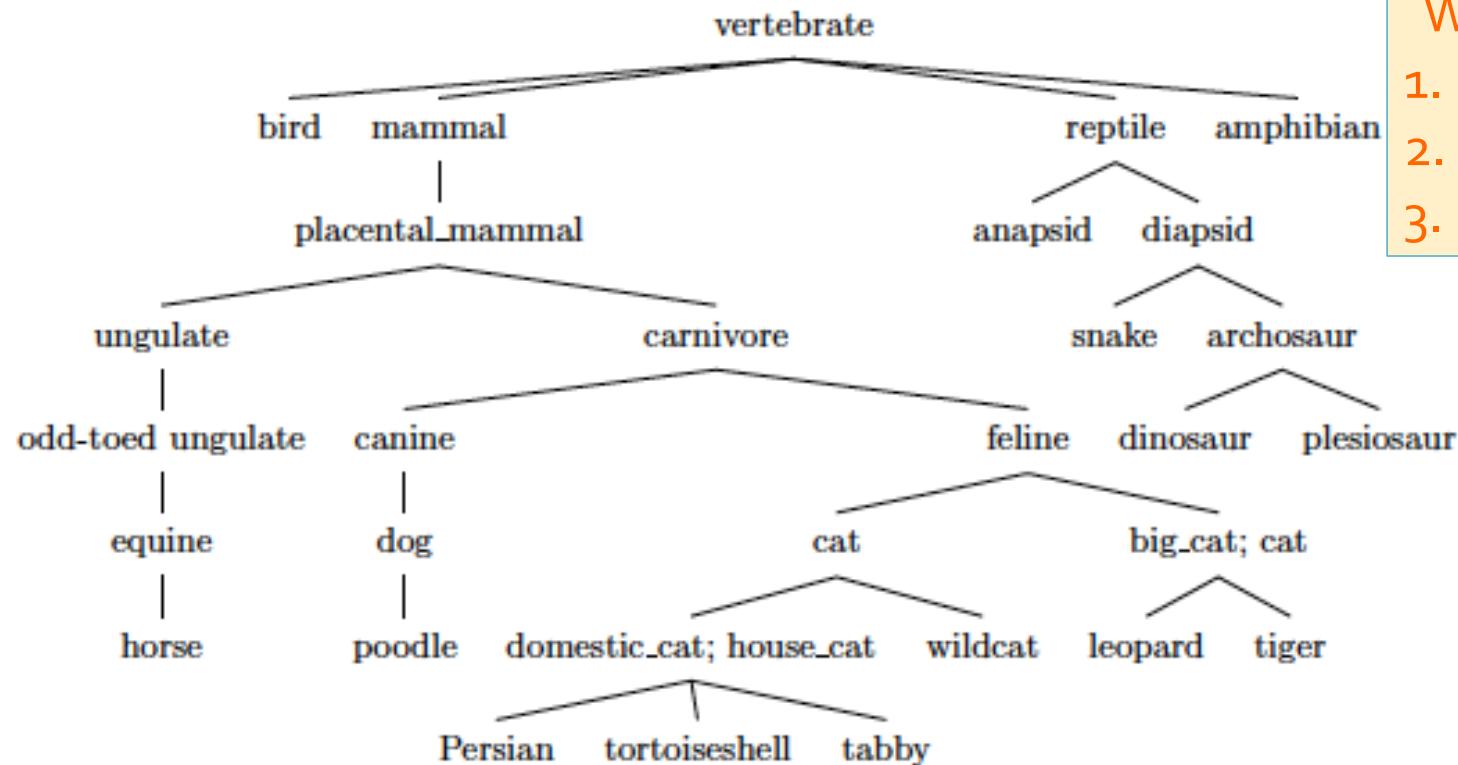


$$\text{sim}_{\text{path}}(c_1, c_2) = \frac{1}{1 + \text{pathlen}(c_1, c_2)}$$

Potential problems with pathlength

- Pathlength does not differentiate between different types of path e.g., *canine* → ... → *vertebrate* vs *dog* → ... ← *cat*
- Intuitively, concepts (separated by same path length) are more dissimilar higher up the tree; but this is not captured by path length similarity measure
- Some parts of tree may be densely populated with rare terminology

Lowest common subsumer: similarity based on what two concepts share



What is the LCS of:

1. tabby and tiger?
2. poodle and carnivore?
3. poodle and tiger?

Information content

- Intuition: concepts which have the LCS *carnivore* are more similar than concepts which have the LCS *vertebrate*
- We gain more **information** when we are told two objects are both *carnivores* than when we are told they are both *vertebrates*.
- We capture this probabilistically via the information content (IC) of a concept
 - Annotate the hierarchy with the frequency of occurrence of each concept in some corpus
 - Remember that the occurrence of a concept implies the occurrence of all of its hypernyms (if something is a *dog*, it is also a *canine* and so on)

$$P(c) = \frac{\text{freq}(c)}{\sum_c \text{freq}(c)}$$

$$\text{IC}(c) = -\log P(c)$$

Question

How do we count the number of times a concept has occurred in a corpus?

WordNet similarity measures based on information content (IC)

$$IC(c) = -\log P(c)$$

Information content in a concept

$$\text{sim}_{\text{res}}(c_1, c_2) = IC(\text{LCS}(c_1, c_2))$$

Information content in what the concepts share (their lowest common subsumer)

See Resnik, 1995

$$\text{sim}_{\text{lin}}(c_1, c_2) = \frac{2 \times \text{sim}_{\text{res}}(c_1, c_2)}{IC(c_1) + IC(c_2)}$$

Ratio of shared information content to total information content

See Lin 1998b

Word similarity

$$\text{wordsim}(w_1, w_2) = \max_{\substack{c1 \in \text{senses}(w_1) \\ c2 \in \text{senses}(w_2)}} \text{sim}(c_1, c_2)$$

- Can you write python code to implement this function?

Evaluation

- How do we evaluate semantic similarity measures?
- What is the right answer?

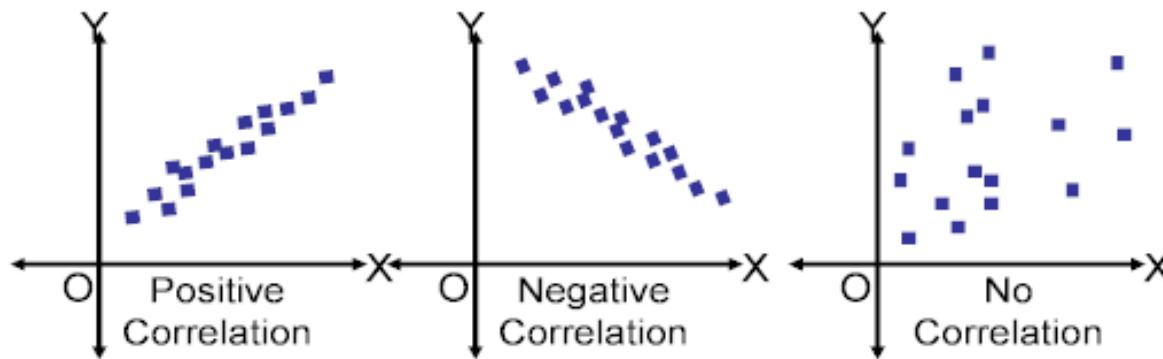
Human synonymy judgements

- Rubenstein & Goodenough 1965 (65 pairs)
- Miller and Charles 1991 (30 pairs)
- WordSim-353 2002 (353 pairs)
- MEN dataset 2012 (3000 pairs)

	M&C	WN
car-automobile	3.92	1.0
magician-wizard	3.5	1.00
journey-car	1.16	0.0
coast-forest	0.42	0.15
noon-string	0.08	0.0

Correlation

SCATTER PLOT EXAMPLES



- Pearson's product-moment correlation coefficient X
- Spearman's rank correlation coefficient ✓

Distributional Semantics

Lecture 1, Part 2

Dr Julie Weeds, Spring 2021



Distributional Semantics

"You shall know a word by the company it keeps."

Firth (1957)

The Distributional Hypothesis: "Words that occur in the same contexts tend to have similar meanings."

Harris (1954)

What does *tezguino* mean?

1. A bottle of *tezguino* is on the table.
2. Everyone likes *tezguino*.
3. *Tezguino* makes you drunk.
4. We make *tezguino* out of corn.

(Lin, 1998)

Bootstrapping the semantics of unknown words

- The **contexts** in which *tezguino* is used suggest that *tezguino* may be:
 - *A kind of alcoholic beverage made from corn mash*
- Similarity plays an important role in word acquisition (Gentner, 1982)
- Can we use corpora to infer similarity between words i.e., infer that *tezguino* is similar to *beer, wine, vodka* etc?

Applications of distributional semantics

- Automatic thesaurus construction
 - For any language, genre, domain ... where we have a corpus
- Overcoming data sparseness in models which require labelled training data

Distributional semantics in document classification

- Imagine we have built a Naïve Bayes document relevancy classifier using a relatively small training sample (e.g., 500 documents)
- A test document contains the word *tezguino* which has not been seen in the training sample
 - so it cannot contribute to the relevancy classification
- But by applying distributional semantics to a very large unlabeled corpus (e.g., the web), we know that *tezguino* is very similar to *beer*
 - *beer* has been seen in the training sample
 - Assume $P(\text{tezguino}|\text{class}) \approx P(\text{beer}|\text{class})$

Facets of meaning

- Tigers **eat** meat.
- The monkey **ate** a banana.
- X17 likes to **eat** falafel.
- My son does not **eat** courgettes.
- The machine **ate** my credit card.

From these examples we can learn:

- What can be **eaten**?
- What **eats** things?
- *Meat, banana, falafel, courgettes* and *credit card* all share 1 facet of meaning – that they can be eaten
- *Tigers, monkey, X17, son* and *machine* all share 1 facet of meaning – that they eat things

Features to capture facets of meaning

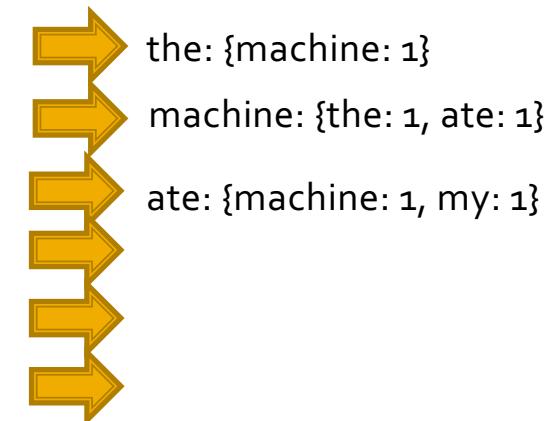
- Dependency relationships between words:
 - “is subject of *eat*”
 - “is object of *eat*”
- Proximity between words
 - “occurs within a **window of +- m words** either side of the word *eat*”
- Feature values can be Boolean but are usually real-valued
 - strength of association
- Dependency parsing is difficult
- Windows are easy to construct
- Window size can be varied to capture different types of semantic relationships

Context windows

window size around target word = +-1

The	machine	ate	my	credit	card
The	machine	ate	my	credit	card
The	machine	ate	my	credit	card
The	machine	ate	my	credit	card
The	machine	ate	my	credit	card
The	machine	ate	my	credit	card

Features added per target word



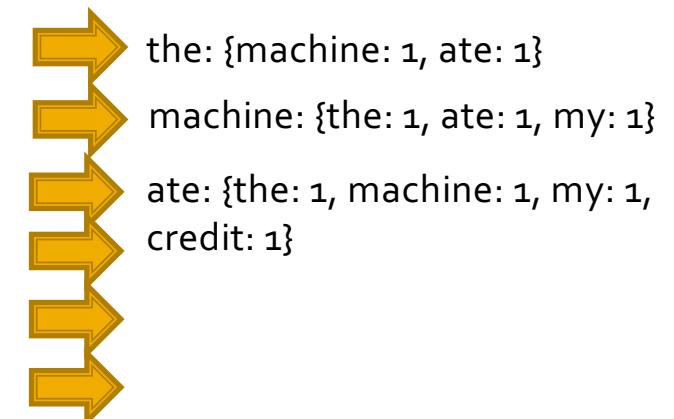
What features will be added for
my, credit and card?

Context windows

window size around target word = +-2

The	machine	ate	my	credit	card
The	machine	ate	my	credit	card
The	machine	ate	my	credit	card
The	machine	ate	my	credit	card
The	machine	ate	my	credit	card
The	machine	ate	my	credit	card

Features added per target word



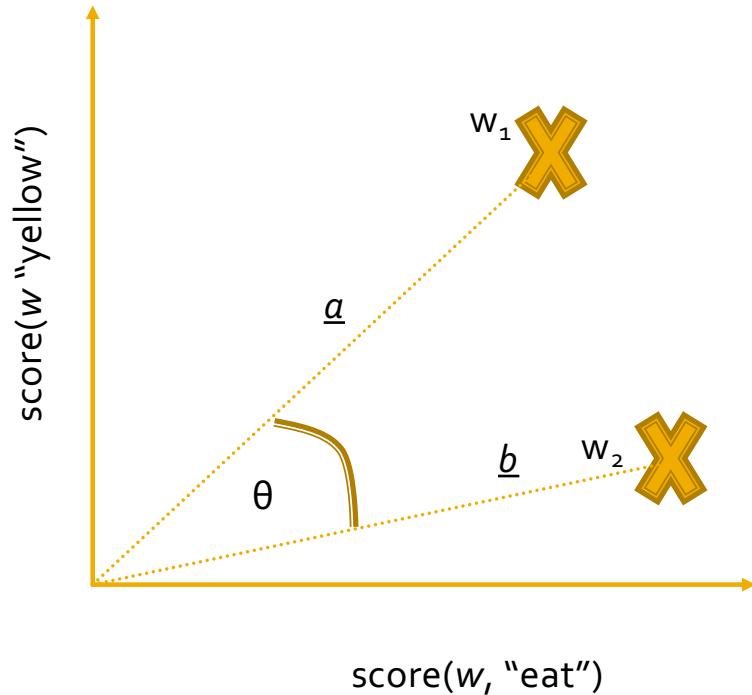
What features will be added for
my, *credit* and *card*?

Distributional Representations

Use windowing to extract and count features for all words in a large corpus i.e., distributional representations or vectors

<i>feature</i>	banana	meat	credit	Total
yellow	10	2	3	15
red	2	14	19	35
eat	20	9	1	30
spend	1	2	27	30
card	3	2	50	55
the	25	25	50	100
is	20	20	40	80
tiger	3	17	0	20
man	6	9	10	25
monkey	10	0	0	10
Total	100	100	200	400

Cosine similarity



- The more similar two words are, the smaller the angle θ between their vectors will be.
- So:

$$\text{sim}(w_1, w_2) = \cos(\theta)$$

$$= \frac{\underline{a} \cdot \underline{b}}{\sqrt{\underline{a} \cdot \underline{a} \times \underline{b} \cdot \underline{b}}}$$

Where:

$$\underline{a} \cdot \underline{b} = \sum_i^m a_i b_i$$

$m = \text{number of dimensions}$

Calculating cosine

<i>feature</i>	banana	meat	<i>a.b</i>	<i>a.a</i>	<i>b.b</i>
yellow	10	2	20	100	4
red	2	14	28	4	196
eat	20	9	180	400	81
spend	1	2	2	1	4
card	3	2	6	9	4
the	25	25	625	625	625
is	20	20	400	400	400
tiger	3	17	51	9	289
man	6	9	54	36	81
monkey	10	0	0	100	0
Total	100	100	1366	1684	1684

$$\cos(\text{banana}, \text{meat}) = \frac{1366}{1684} = 0.81$$

Pointwise Mutual information (PMI)

- Frequency and/or simple conditional probability do not capture the intuition that some features are more informative than others
- *the* and *is* appear relatively frequently with all of the words
 - so their contribution to similarity should be smaller
- PMI measures the amount of information gained by seeing a word and a feature together
- A feature which co-occurs with a target word more than we would expect (if words and features occurred independently) has more weight in the similarity calculation

calculating PMI

$$I(w, f) = \log \frac{P(f | w)}{P(f)} = \log \frac{P(f \cap w)}{P(f) \times P(w)}$$

$$I(w, f) = \log \frac{\text{freq}(f, w) \times \text{freq}(*, *)}{\text{freq}(*, w) \times \text{freq}(f, *)}$$

grand total

row total

column total

Representations based on PMI

<i>feature</i>	banana	meat	credit	Total
yellow	10	2	3	15
red	2	14	19	35
eat	20	9	1	30
spend	1	2	27	30
card	3	2	50	55
the	25	25	50	100
is	20	20	40	80
tiger	3	17	0	20
man	6	9	10	25
monkey	10	0	0	10
Total	100	100	200	400

$$\log \frac{10 \times 400}{100 \times 15}$$



<i>feature</i>	banana	meat	credit
yellow	1.42		
red			
eat			
spend			
card			
the			
is			
tiger			
man			
monkey			

Positive PMI (PPMI)

- What happens when frequency of co-occurrence is 0?
- PMI = negative infinity!!!
- positive PMI avoids this problem
 - similarity is then also based on shared features rather than the sharing of absent features

$$\text{PPMI}(w, f) = \begin{cases} I(w, f) & I(w, f) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Representations based on PPMI

<i>feature</i>	banana	meat	credit	Total
yellow	10	2	3	15
red	2	14	19	35
eat	20	9	1	30
spend	1	2	27	30
card	3	2	50	55
the	25	25	50	100
is	20	20	40	80
tiger	3	17	0	20
man	6	9	10	25
monkey	10	0	0	10
Total	100	100	200	400

$$\log \frac{10 \times 400}{100 \times 15}$$



<i>feature</i>	banana	meat	credit
yellow	1.42	0	0
red	0	0.68	0.12
eat	1.42	0.26	0
spend			
card			
the			
is			
tiger			0
man			
monkey	0	0	

Automatic thesaurus generation

- Extract feature representations based on corpus co-occurrence frequencies
- Convert representations to PPMI
- Calculate cosine similarities for all pairs of words
 - computationally very expensive
 - may want to reduce the number of words considered in vocab
 - e.g., top 10,000 words
- Find nearest neighbours of each word

Evaluation

- Difficult – why?
- Intrinsic evaluation
 - human synonymy judgements
 - manually compiled thesauruses
- Extrinsic evaluation
 - performance gain in an application

Word ambiguity

Here is the distributional thesaurus entry for the noun *bow* (derived using `nltk.lin_thesaurus`)

bow	
ribbon	0.09
machete	0.07
spear	0.07
hull	0.07
sword	0.07
knife	0.07
arrow	0.06
scarf	0.06
rope	0.06
streamer	0.06

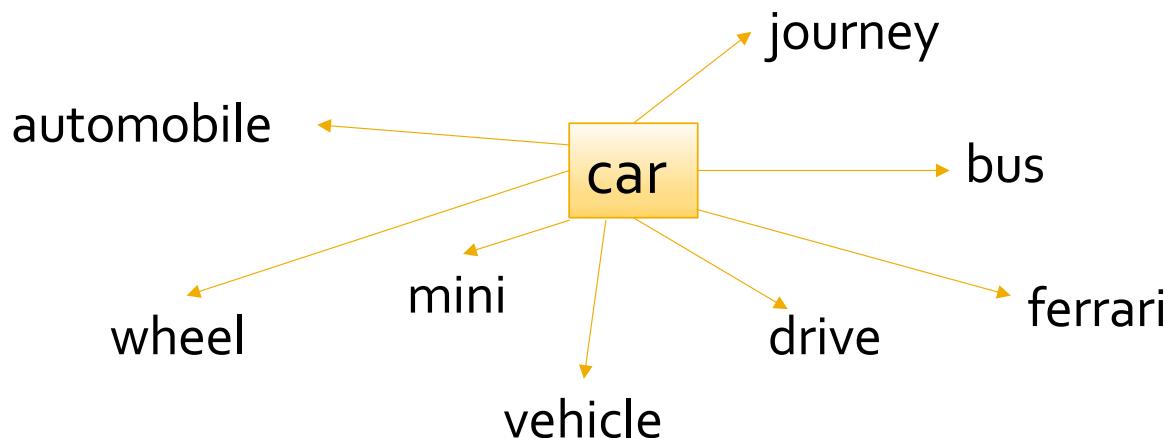
- What different senses of the word *bow* do you think are captured by the thesaurus entry?
- Are the neighbours distributed evenly between the senses or do some senses have more neighbours than others?
- Why do you think this is?

Senses in Distributional Semantics

- Distributional representations are of words not senses
 - mixture of senses in distributional neighbourhoods
 - this can be a problem in some applications.?
 - possible solutions: carry out WSD
 - before finding distributional neighbours
 - after finding distributional neighbours
- Distributional neighbours tend to reflect predominant sense of word
 - **how could this be useful?**

Semantic relationships

- Similar words are not necessarily synonyms
- Neighbourhoods typically contain:
 - synonyms, antonyms, hypernyms, hyponyms, co-hyponyms, meronyms, topically related words

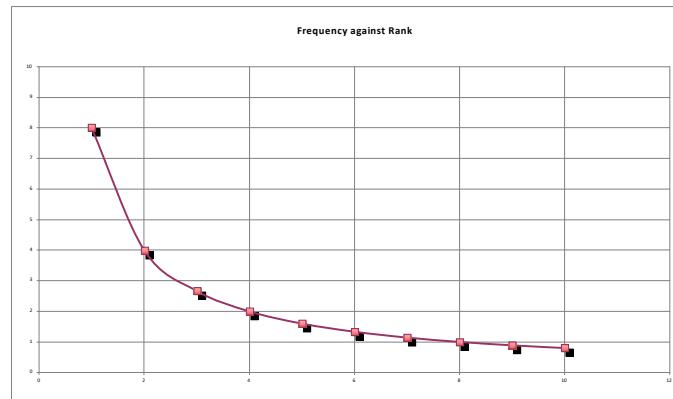


The nearest neighbour of a word is often an antonym (or co-hyponym). Why might this be a problem?

Sparsity

- Zipf's Law: “**The product of the frequency of a word and its rank is approximately constant.**”

Rank	1/Rank	Freq
1	8	8
2	4	4
3	2.667	3
4	2	2
5	1.6	2
6	1.333	1
7	1.143	1
8	1	1
9	0.889	1
10	0.8	1
	23.43	24



Hapax Legomena : words which only occur once. However large the corpus, these make up approximate half the vocabulary.

Consequences of Zipf's Law

- 100k dimensional co-occurrence vectors will be very sparse (lots of zeros)
- difficult to compare vectors because of all of this unseen stuff
- What can we do?

Possible solutions

- Smoothing
- Dimensionality reduction
- Language models with fixed dimensionality e.g., recurrent neural network language models (RNNLMs)
- More on this in week 3!

Next time: Language modelling

- Probabilistic language models
 - n-gram modelling
 - evaluation and perplexity
 - generalization and smoothing

Reading

- Week 1 seminar:
 - Pedersen (2010): Information Content Measures of Semantic Similarity Perform Better without Sense Tagged Text
- Week 2 seminar:
 - Zweig et al. (2011): Zweig, G. and Burges, A. 2011. The Microsoft Research Sentence Completion Challenge.

References

1. Christiane Fellbaum (1998, ed.) WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press.
2. Lin, D. (1998a) Automatic retrieval and clustering of similar words. In Proceedings of COLING/ACL.
3. Lin, D (1998b) An information-theoretic definition of similarity. In ICML 1998, San Francisco, pp 296-304
4. Mikolov, Yih and Zweig 2013 – Linguistic Regularities in Continuous Space Word Representations, (NAACL-HCT 2013)
5. Pedersen, T. (2010). Information Content Measures of Semantic Similarity Perform Better without Sense Tagged Text
6. Resnik, P (1995) Using information content to evaluate semantic similarity in a taxonomy. In IJCAI-95, pp. 448-424
7. Zweig, G. and Burges, A. 2011. The Microsoft Research Sentence Completion Challenge. Microsoft Technical Report

AdvNLP/E Lecture 2

Lexical and Distributional Semantics 2

Dr Julie Weeds, Spring 2024



Previously (week 1)

- Lexical semantics
 - word senses
 - semantic relationships
 - semantic similarity
- Distributional semantics
 - bootstrapping semantics from context
 - cosine similarity
 - (positive) pointwise mutual information
 - challenges

Challenges for measures of distributional similarity

- Mixture of senses (lecture 1)
- Mixture of relationships (lecture 1)
- Sparsity (this time!)

Lecture 2 overview

- Part 1
 - Sparsity and Zipf's Law
 - Smoothing
 - Dimensionality Reduction
- Part 2
 - Word Embeddings
 - Word2Vec
 - GloVe
 - Composition

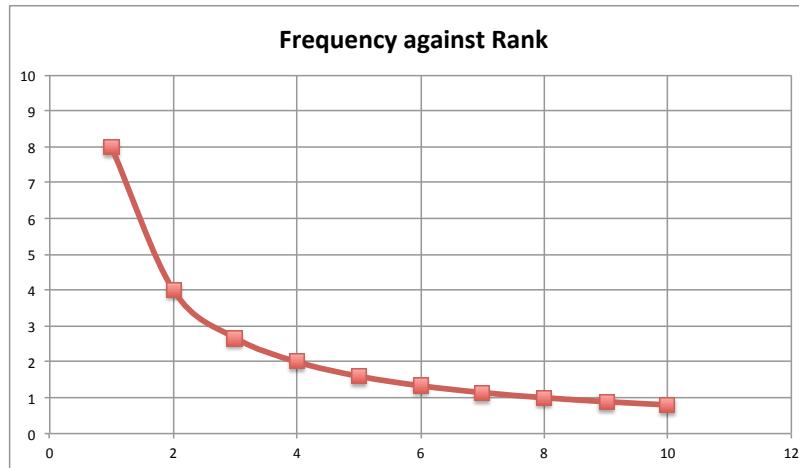
Sparsity

- Google News corpus contains **320M words (tokens)** with a vocabulary of **82K (types)**.
- What does that suggest about the distribution of types?
- Mean frequency = 3900
- Is that enough?
- Maybe if it was a Normal distribution with a small standard deviation

Zipf's Law

- “The product of the frequency of a word and its rank is approximately constant.”

Rank	8/Rank	Freq
1	8	8
2	4	4
3	2.667	3
4	2	2
5	1.6	2
6	1.333	1
7	1.143	1
8	1	1
9	0.889	1
10	0.8	1
	23.43	24



Hapax Legomena : words which only occur once. However large the corpus, these make up approximate half the vocabulary.

Consequences of Zipf's Law

- 82k dimensional co-occurrence vectors will be very sparse (lots of zeros)
- difficult to compare vectors because of all of this unseen stuff
- What can we do?

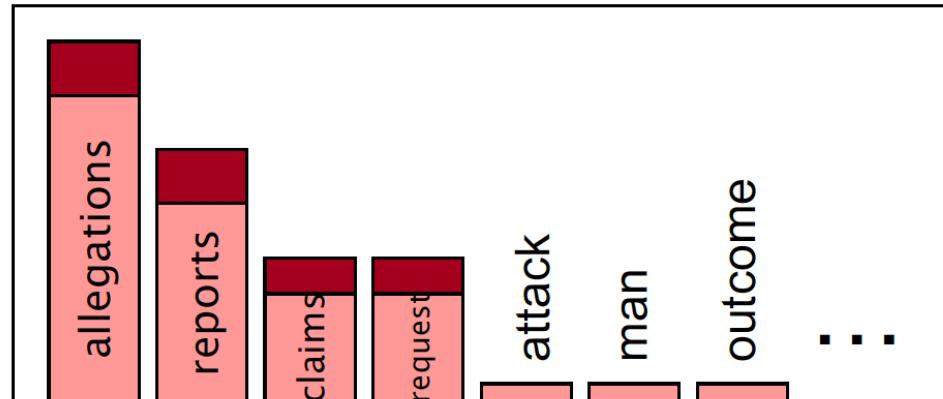
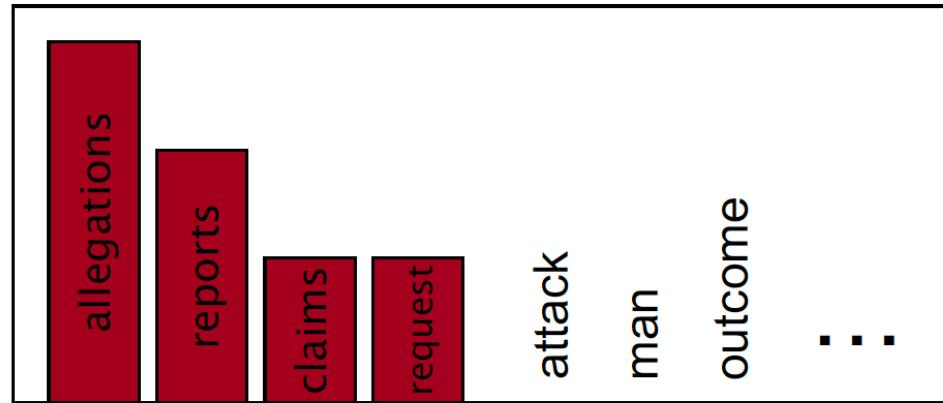
Possible solutions

1. Smoothing
2. Dimensionality reduction
3. Word embeddings / fixed dimensionality representations from language models

Smoothing intuition

- Anything is possible – even previously unobserved events
- When we have sparse statistics, steal probability mass from observed events generalise to unobserved events

count (w "denied")	smoothed count
allegations 3	allegations 2.5
reports 2	reports 1.5
claims 1	claims 0.5
request 1	request 0.5
	OTHER 2
total 7	total 7



Add-one smoothing

- Also called **Laplace smoothing**
- Pretend we saw each word one more time than we did (in each possible scenario)
- Just add on one to each count in the frequency matrix

feature	banana	meat	credit	Total
yellow	10	2	3	15
red	2	14	19	35
eat	20	9	1	30
spend	1	2	27	30
card	3	2	50	55
the	25	25	50	100
is	20	20	40	80
tiger	3	17	0	20
man	6	9	10	25
monkey	10	0	0	10
Total	100	100	200	400

feature	banana	meat	credit	Total
yellow	11	3	4	18
red	3	15	20	38
eat	21	10	2	33
spend	2	3	28	33
card	4	3	51	58
the	26	26	51	103
is	21	21	41	83
tiger	4	18	1	23
man	7	10	11	28
monkey	11	1	1	13
Total	110	110	210	430

Applying Add-1 Smoothing

- Can be very effective for some problems
 - If you did **AppliedNLP**, you might remember Add-1 Smoothing being used in the *Naïve Bayes text classifier*
 - Prevents a zero probability being assigned to a word occurring in a particular class just because it has never been seen with that class before
 - But here, the number of zeros isn't so huge
 - number of classes << size of English vocabulary
- Rarely used for co-occurrence data / language models
 - assigns too much mass to unseen co-occurrences (even add-k)
 - leads to massive, unwieldy models
 - and it wouldn't help us with the sparsity problem for distributional semantics anyway.....

Think about it

- Why is Add-1 smoothing unlikely to be very effective in helping us with the sparsity problem for distributional semantics?

Distributional smoothing

Dagan, Pereira and Lee (1994), define $S(w)$ as the k -most similar words to w according to some distributional similarity measure. The co-occurrence probability of two words is then:

$$P_{SIM}(w_2|w_1) = \sum_{w' \in S(w_1)} P(w_2|w') \frac{sim(w', w_1)}{\sum_{w' \in S(w_1)} sim(w', w_1)}$$

Smoothing Example



- *snort* and *aardvark* are both rare words
- observed co-occurrence frequency is zero
- estimate probability of co-occurrence based on
 - co-occurrence of snort with neighbours of aardvark

$$P_{SIM}(\text{snort}|\text{aardvark}) = \sum_{w' \in \{\text{pig, animal, ...}\}} P(\text{snort}|w') \frac{\text{sim}(w', \text{aardvark})}{\sum \text{sim}(w', \text{aardvark})}$$

- and/or co-occurrence of aardvark with neighbours of snort

$$P_{SIM}(\text{aardvark}|\text{snort}) = \sum_{w' \in \{\text{grunt, sniff, ...}\}} P(\text{aardvark}|w') \frac{\text{sim}(w', \text{snort})}{\sum \text{sim}(w', \text{snort})}$$

Distributional smoothing in practice

- Build sparse distributional representations of all words
- Automatically generate thesaurus (find k nearest neighbours of every word)
- Re-estimate co-occurrence probabilities of every pair of words using distributional smoothing formula
- Regenerate thesaurus

Think about it

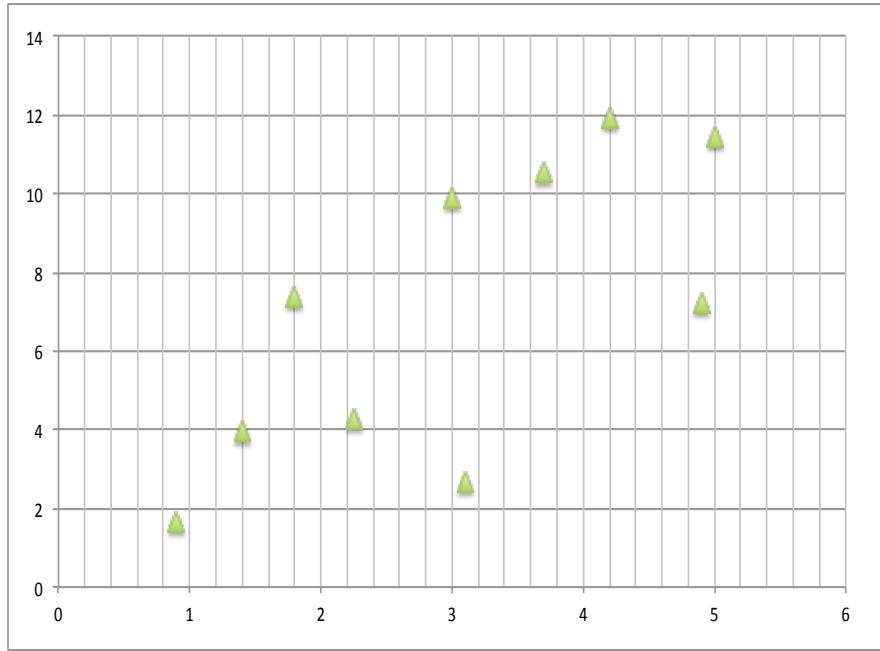
- What disadvantages can you see to this potential solution to the sparsity problem?

Dimensionality Reduction

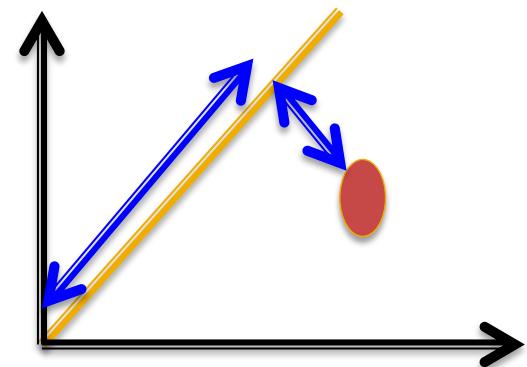
- Principal Component Analysis (PCA) – statistical technique
- Singular Value Decomposition (SVD) – matrix factorisation technique
- Non-negative Matrix Factorisation (NNMF)

Intuition: Find the k axes/bases on which there is most variation, transform the data so these are the bases, ignore other variations.

Principal Component Analysis (PCA) (2d->1d)



- Find the regression line (least squares linear regression)
- Each point can be represented by how far along the line it is and how far away from the line.



For a good line of best fit, most of the variation is contained within the principal component (distance along the line). Ignoring non-principal components leads to good approximation and good compression

PCA (2)

- In general, we want to reduce n dimensions (n very large) to k dimensions (e.g., $k = 300$)
- PCA can be applied iteratively to find all of the k dimensions (which are further required to be orthogonal)
- Variance in the data (not captured by principal components) can be computed as a measure of the loss of information

Latent Semantic Analysis (LSA)

- Deerwester et al. 1990, Landauer et al. 1998
- Represent text as a matrix X where each row is a unique word and each column is a document (or other context).
- Values are transformed frequencies (typically tf-idf or PPMI)
- Apply SVD where X is decomposed into a product of 3 matrices

SVD (Singular Value Decomposition)

$$X = W.S.P$$

$$\begin{pmatrix} 2 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 1 & 1 \\ 4 & 2 & 2 \\ 0 & 0 & 5 \end{pmatrix} = \begin{pmatrix} 2 & ? \\ ? & ? \\ ? & ? \\ ? & ? \\ ? & ? \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1.5 \end{pmatrix} \begin{pmatrix} 1 & ? & ? \\ 0 & ? & ? \end{pmatrix}$$

$$[5 \times 3] = [5 \times 2][2 \times 2][2 \times 3]$$

- S is a diagonal matrix, the dimensions of which define the reduced space
- W is the reduced dimensionality set of word vectors
- P is the reduced dimensionality set of document/co-occurrence vectors
- W, S and P found using standard matrix techniques e.g., by finding eigenvalues and eigenvectors
- Also known as eigendecomposition

NNMF

- Non-negative matrix factorization
- Many possible solutions to the factorization formulae
- NNMF has further constraint that all of the values in factorized space are non-negative
- Potentially leads to more interpretable spaces
 - why?

Using PCA, SVD, LSA or NNMF

ADVANTAGES

- can massively reduce dimensionality
- theoretically should capture similarities between words in the dimensions (co-occurrences with words describing dogs *could* all be on some “dogginess” dimension)

DISADVANTAGES

- computationally expensive
- generally impossible to interpret/interrogate dimensions

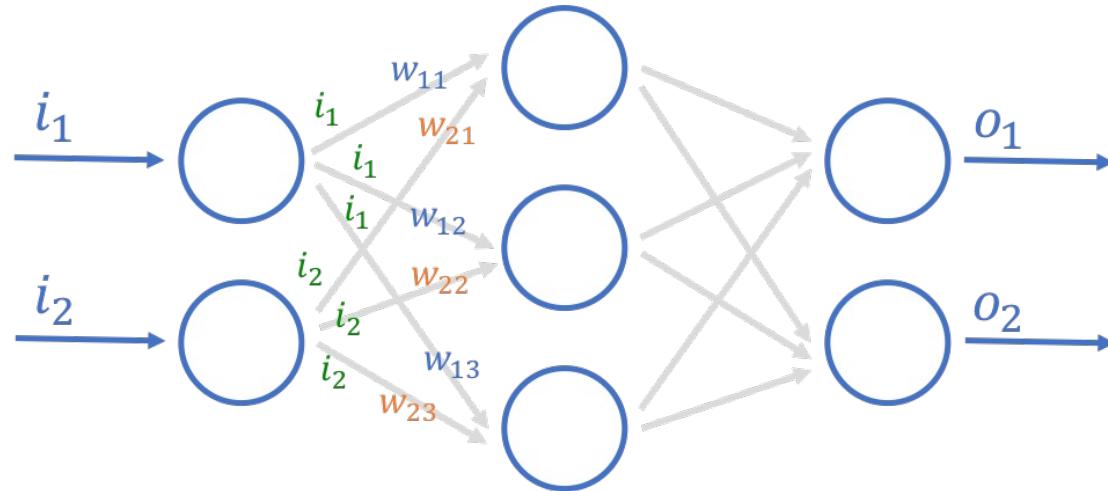
Week 2 Lecture: part 2

Word embeddings

Word Embeddings

- Start with the assumption that each word can be represented by a fixed set of dimensions
- Learn the best values to optimise performance on some task e.g., the log-likelihood of some training corpus
 - What's the probability of seeing this sequence of words?
 - What values would make it more likely that we would see this sequence of words (which has been observed and therefore we know it should be highly likely)?

A Simple NN



$$\begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \\ w_{13} & w_{23} \end{bmatrix} \cdot \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} = \begin{bmatrix} (w_{11} \times i_1) + (w_{21} \times i_2) \\ (w_{12} \times i_1) + (w_{22} \times i_2) \\ (w_{13} \times i_1) + (w_{23} \times i_2) \end{bmatrix}$$

One hot encoding

- Input space with V dimensions where V is size of vocabulary
- Exactly one of the input features is one, the others are zero
- This selects a column from the weight matrix

$$\begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \\ w_{13} & w_{23} \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} w_{11} \\ w_{12} \\ w_{13} \end{bmatrix}$$



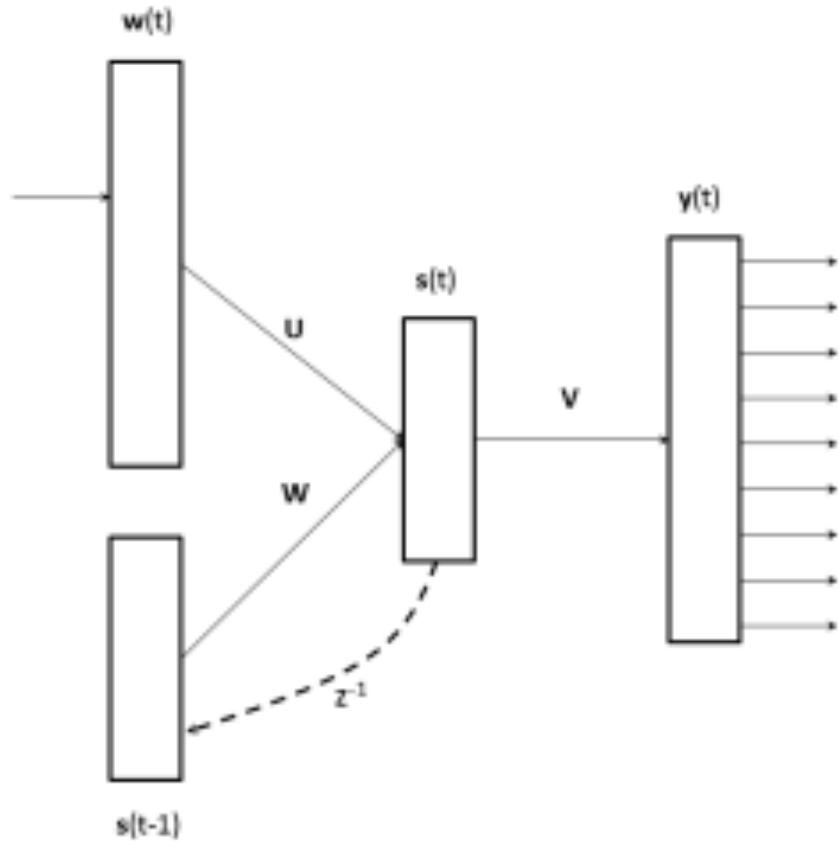
Embedding for word 1 (1,0)

$$\begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \\ w_{13} & w_{23} \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} w_{21} \\ w_{22} \\ w_{23} \end{bmatrix}$$



Embedding for word 2 (0,1)

Recurrent Neural Network Language Model (Mikolov et al. NAACL 2013)



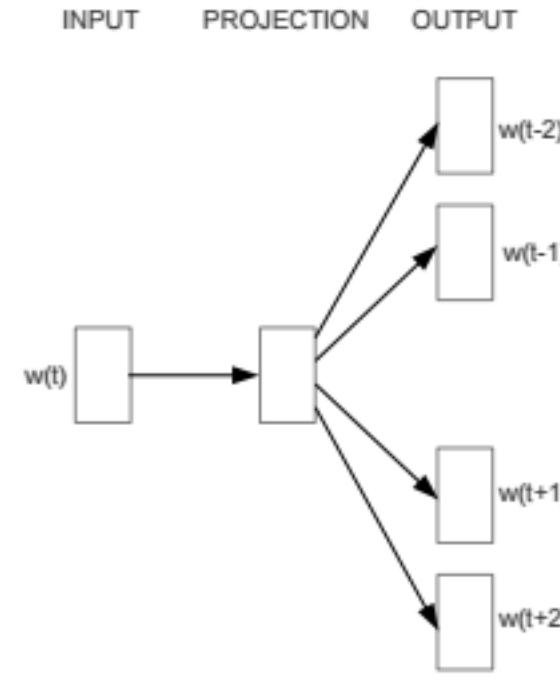
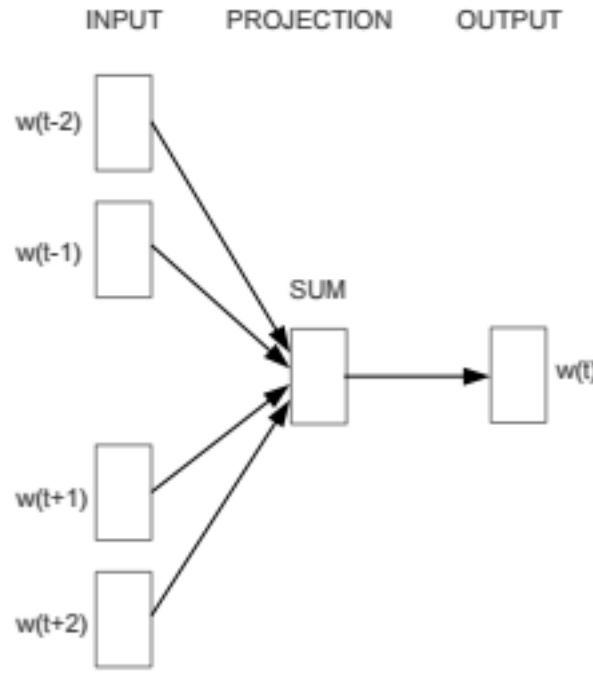
This model architecture comes up in the paper this week. However, in the same year, Mikolov et al. proposed 2 much simpler models (i.e., easier to understand) which achieved comparable results and which came to be known as **word2vec**.

So we are going to ignore it for now (we will have a look at it in week 4) and read the paper assuming that the architecture used is one of the standard word2vec ones.

Word2Vec Intuition

- The current word in the corpus is referred to as the **target word**
- Use a **fixed length context window** (e.g., +/- 2 words either side of the target word)
- Each word has 2 different embeddings
 - **Remember** – an embedding is a representation / vector / set of weights
 - One embedding used when it is a target word
 - The other embedding when it is a context word
- In CBOW, the context embeddings are used to predict the target embedding
- In Skip-gram, the target embedding is used to predict the context embeddings

Word2Vec (Mikolov et al. arXiv 2013)



Continuous Bag of Words (CBOW) : predict the current word given the context
Skip-gram : predict the context given the current word

Word2Vec training

- Embeddings are randomly initialised
- Iterate over text:
 - compute objective function for target word using positive and negative samples
- Stochastic gradient descent / backpropagation is used to 'improve' the weights
- Training continues

Gradient descent

- Randomly initialise parameter settings (W)
- Compute predictions
- Compute cost function (J)
- Compute (partial) derivatives of cost function with respect to parameters
- Back-propagate i.e. update parameters:

$$W = W - \alpha \frac{dJ}{dW}$$

learning rate

- Repeat until convergence / cost is acceptably small

Skip-gram Intuition

- Is word w_2 likely to show up near word w_1 ?
- Can we train a classifier on running text?
- Treat the target word and a neighbouring context word as positive examples
- Randomly sample other words in lexicon to get negative examples
- Train a classifier to distinguish those two cases

Skip-gram training

- Randomly initialise two sets of embeddings – each word in the vocabulary has a target embedding, t_w and a context embedding, c_w
- Iteratively shift target and context embeddings so that:
 - target embedding of word w is more like the context embeddings of words that occur nearby and less like the embeddings of words that don't occur nearby

Skip-gram Objective

$$J = \sum_{(t,c) \in +} \log P(+) | t, c) + \sum_{(t,c) \in -} \log P(- | t, c)$$

$$J = \sum_{(t,c) \in +} \sigma(c \cdot t) + \sum_{(t,c) \in -} \sigma(-c \cdot t)$$

sigmoid function

The diagram illustrates the sigmoid function's role in the skip-gram objective. It features two parallel arrows originating from the same point at the bottom left. The left arrow points upwards towards the term $\sigma(c \cdot t)$, and the right arrow points upwards towards the term $\sigma(-c \cdot t)$. Both arrows are colored yellow. Below these arrows, the text "sigmoid function" is centered.

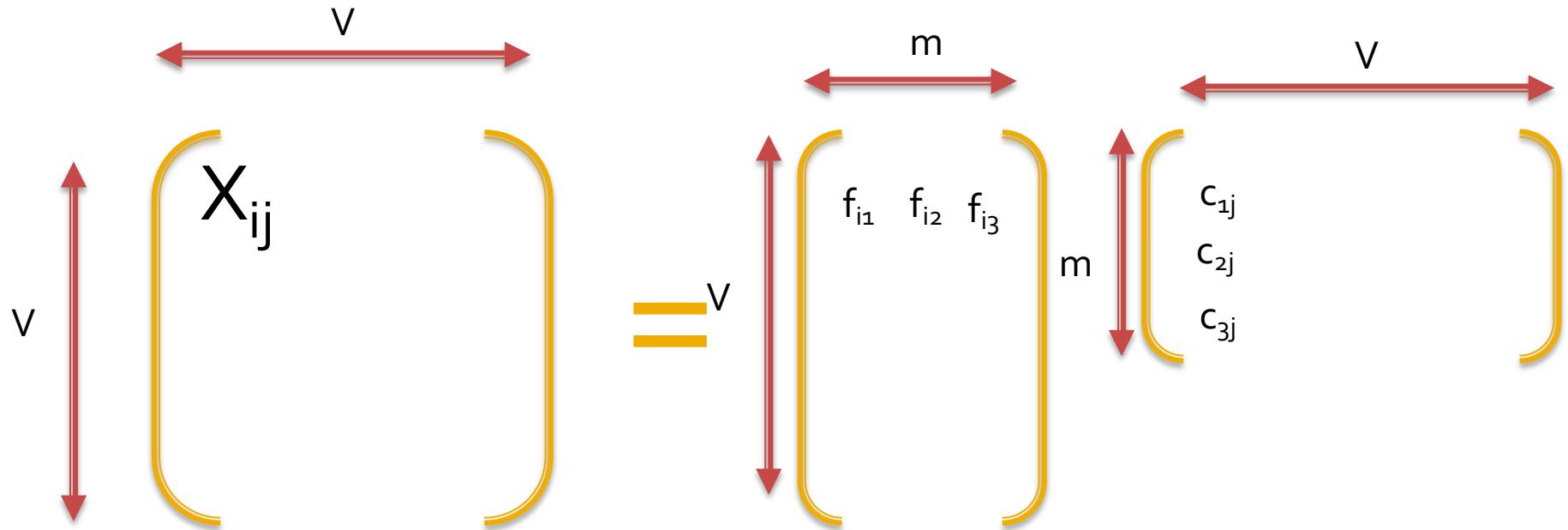
Word2Vec parameters

- Number of epochs
 - how many times is the corpus iterated over?
- Learning rate
 - how much are the weights updated each time?
- Batch size
 - how many predictions are considered before updates made?
- Window size
 - how many words of context are considered?

GloVe (Pennington et al. 2014)

- alternative approach to building low-dimensional dense word representations
- combines
 - mathematical simplicity of matrix factorization; with
 - speed and efficiency of neural techniques

GloVe Factorization



X_{ij} = co-occurrence frequency of word i and word j

Aim: find a set of m dimensional focal embeddings (F) and m -dimensional context embeddings (C) such that $X=FC^T$

GloVe Training

- Build standard co-occurrence matrix (based on context windows)
- Randomly initialise focal and context embeddings
- Batch sample co-occurrence matrix and minimise the loss function:

least squares problem

$$J = \sum_{i,j=1}^V f(X_{ij}) (F_i \cdot C_j^T + b_i + b_j - \log X_{ij})^2$$

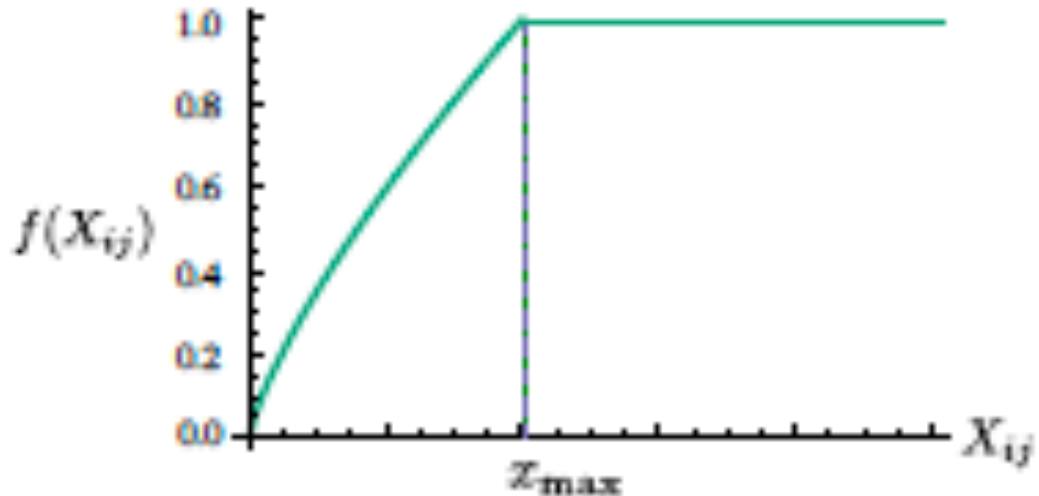
weighting function

dot product of focal and context embedding for each word

bias terms for each word

Weighting Function

$$f(x) = \begin{cases} (x/x_{max})^\alpha & \text{if } x < x_{max} \\ 1 & \text{otherwise} \end{cases}$$



- Aim: reduce the impact of less reliable, lower frequency events
- x_{\max} is typically around 100

Figure 1: Weighting function f with $\alpha = 3/4$.

Correlation with human judgements

Baroni et al. (2014)'s comparison of word2vec embeddings against best-performing count-based distributional representations on word similarity tasks

Dataset	Count-based (PPMI + NNMF)	Word2Vec
Rubinstein & Goodenough	0.74	0.84
WordSim353 (similarity)	0.70	0.80
WordSim353 (relatedness)	0.59	0.70
MEN dataset	0.72	0.80

“Don’t Count, predict!”

What is so special about word2vec or GloVe?

- Omer & Levy 2014 showed that word2vec embeddings are an implicit factorisation of standard co-occurrence vectors with weights as PPMI
- BUT there are a lot more hyper-parameters which are tuneable (Omer & Levy 2015)

Hyper-parameters exploited by Word2Vec et al. (1)

1. dynamic context window: weight words by their distance from the focus word
2. subsampling: dilute/remove very frequent words
3. deleting rare words: ignore these when computing context windows

Hyper-parameters exploited by Word2Vec et al. (2)

4. context distribution smoothing: when computing PMI, smooth the unigram distribution
5. negative sampling/shifted PMI: shift all of the PMI values by k so only really important contexts are seen as features
6. Combining word and context vectors
7. Eigenvalue weighting in SVD

Disadvantages of low-density representations

- Dimensions are un-interpretable
 - there may be a 'dogginess' dimension
 - but no way of knowing which one
- Non-determinism
 - random initialisation
 - different solution found each time
 - even if globally optimal solution exists, space could be rotated on different runs
 - cannot compare embeddings produced on different runs / for different corpora
 - but it does provide a way of testing for statistical significance of results

Where next with distributional semantics ... what about phrases?

- What is the similarity of “*nurse*” to “*man*”?
- What is the similarity of “*male nurse*” to “*man*”?
- How do we know the meaning of the word “*male nurse*”?
- Is this true for all (noun) phrases?

Composition: intersective

$$F(X \cdot Y) = F(X) \cap F(Y)$$

- Captures the intuition that features of the phrase must have occurred with ALL of the constituents of the phrase
- E.g., a feature is associated with “male nurse” if it is associated with “male” AND it is associated with “nurse”
- Commonly implemented with pointwise multiplication of vectors (or minimum)
- As more words are composed, will tend to zero vector (particularly if the vectors are sparse)

Composition: additive

$$F(X \cdot Y) = F(X) \cup F(Y)$$

- Captures the intuition that features of the phrase must have occurred with ONE of the constituents of the phrase
- E.g., a feature is associated with “male nurse” if it is associated with “male” OR it is associated with “nurse”
- Commonly implemented with pointwise addition of vectors (or maximum)
- Re-normalising makes this the centroid (or average) of the constituents
- As more words are composed, will tend to the centre of the semantic space

Evaluation of compositional models

- Very difficult
- Correlation with human similarity judgements for phrases
- Semantic similarity in context tasks
- Paraphrase recognition / textual entailment tasks
- Application-based evaluation

Challenges for compositional models

- Not all phrases are compositional
- Word order is important (surely?) but often totally ignored!
- Negation!!!!
- Other functional words (e.g., determiners)
- What are the distributional contexts of a sentence anyway?

Reading

- Mikolov, Yih and Zweig (NAACL, 2013:
Linguistic regularities in continuous space
word representations

Coming up

- Probabilistic language models (week 3)
 - n-gram modelling
 - evaluation and perplexity
 - generalization and smoothing
- Neural language models (week 4)
 - feed-forward
 - RNNs and LSTMs
 - character-based

References

- Baroni, Dinu and Kruszewski (ACL, 2014): Don't count, predict! A systematic comparison of context-counting vs context-predicting semantic vectors
- Dagan, Pereira and Lee (ACL, 1994): Similarity-based estimation of word co-occurrence probabilities
- Levy and Goldberg (NIPS, 2014): Neural Word Embeddings as Implicit Matrix Factorization
- Levy, Goldberg and Dagan (TACL, 2015): Improving Distributional Similarity with Lessons Learned from Word Embeddings
- Landauer, Foltz and Laham (Discourse Processes, 1998): An Introduction to Latent Semantic Analysis
- Mikolov, Chen, Corrado and Dean (arXiv pre-print, 2013): Efficient Estimation of Word Representations in Vector Space
- Mikolov, Yih and Zweig (NAACL, 2013): Linguistic regularities in continuous space word representations
- Zweig and Burges (Microsoft Research Technical Report, 2011): The Microsoft Research sentence completion challenge

AdvNLP/E Lecture 3

Language Modelling 1

Dr Julie Weeds, Spring 2024



Language models

PREVIOUSLY

- Lexical and distributional semantics
 - semantic relationships
 - WordNet
 - distributional hypothesis
 - vector spaces and word representations
 - sparsity and Zipf's Law
 - dimensionality reduction
 - word embeddings

THIS TIME

- Probabilistic language models
 - n-gram modelling
 - evaluation and perplexity
 - generation
 - generalization and smoothing

N-gram models

Lecture 2, Part 1



Why do we want to be able to assign a probability to a sentence?

- Machine translation

$$P(\text{high winds tonight}) > P(\text{large winds tonight})$$

- Spelling correction

$$\begin{aligned} P(\text{The office is about 15 minutes from my house}) \\ &> P(\text{The office is about 15 minuets from my house}) \end{aligned}$$

- Speech recognition

$$P(\text{I saw a van}) > P(\text{eyes awe of an})$$

Probabilistic language modelling

- Goal: compute the probability of a sentence or sequence of words

$$P(W) = P(w_1, w_2, \dots, w_n)$$

- Related task: probability of an upcoming word

$$P(w_5 | w_1, w_2, w_3, w_4)$$

- A model that computes either of these is called a **language model (LM)**

The Chain Rule for Probabilities

- The definition of conditional probabilities give us:

$$P(B|A) = \frac{P(A, B)}{P(A)} \quad \longrightarrow \quad P(A, B) = P(A)P(B|A)$$

- More variables

$$P(A, B, C, D) = P(A)P(B|A)P(C|A, B)P(D|A, B, C)$$

- The general case (chain rule):

$$P(x_1, x_2, x_3, \dots, x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1, x_2) \dots P(x_n|x_1, \dots, x_{n-1})$$

Applying the chain rule to words

$$P(w_1, w_2, w_3, \dots, w_k) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2) \dots P(w_k|w_1, \dots, w_{k-1})$$



$P(\text{"Then he hovered over the hill"})$
 $= P(\text{"Then"}) \times P(\text{"he"} | \text{"Then"}) \times P(\text{"hovered"} | \text{"Then he"})$
 $\times P(\text{"over"} | \text{"Then he hovered"})$
 $\times P(\text{"the"} | \text{"Then he hovered over"})$
 $\times P(\text{"hill"} | \text{"Then he hovered over the"})$

Estimating probabilities

- Can we just count and divide?

$$P(\text{"hill"} \mid \text{"Then he hovered over the"}) \\ = \frac{\text{freq}(\text{"Then he hovered over the hill"})}{\text{freq}(\text{"Then he hovered over the"})}$$

Markov Assumptions

- First order

$$P(\text{"hill"} \mid \text{"Then he hovered over the"}) \approx P(\text{"hill"} \mid \text{"the"})$$


- Second order

$$P(\text{"hill"} \mid \text{"Then he hovered over the"}) \approx P(\text{"hill"} \mid \text{"over the"})$$

N-gram language model

$$P(w_1, w_2, w_3, \dots, w_k) = \prod_{i=1}^k P(w_i | w_{i-(n-1)}, \dots, w_{i-1})$$

Considers only n words at a time, the current word and the previous $n-1$ words

- approximates each component in the product
- these approximations can be estimated using maximum likelihood estimation (MLE) on a training corpus

$$P(w_i | w_{i-(n-1)}, \dots, w_{i-1}) = \frac{freq(w_{i-(n-1)}, \dots, w_{i-1}, w_i)}{freq(w_{i-(n-1)}, \dots, w_{i-1})}$$

Unigram model

- $n = 1$

$$P(w_1, w_2, w_3, \dots, w_k) = \prod_{i=1}^k P(w_i)$$



$P(\text{"Then he hovered over the hill"})$
 $= P(\text{"Then"}) \times P(\text{"he"}) \times P(\text{"hovered"}) \times P(\text{"over"}) \times P(\text{"the"}) \times P(\text{"hill"})$

Bigram model

- $n=2$

$$P(w_1, w_2, w_3, \dots, w_k) = \prod_{i=1}^k P(w_i | w_{i-1})$$



$$\begin{aligned} & P(\text{"Then he hovered over the hill"}) \\ & = P(\text{"Then"}) \times P(\text{"he"} | \text{"Then"}) \times P(\text{"hovered"} | \text{"he"}) \\ & \quad \times P(\text{"over"} | \text{"hovered"}) \times P(\text{"the"} | \text{"over"}) \times P(\text{"hill"} | \text{"the"}) \end{aligned}$$

Trigrams and beyond

- We can extend to:
 - trigrams ($n=3$)
 - quadrigrams ($n=4$)
 - 5-grams($n=5$)
- The higher n is, the more long range dependencies can be captured ...
but the models will also become more sparse and unreliable

6-gram



Presently he emerged, looking even more _____ than before.

Products of probabilities

- Use logs
- Avoid underflow
- Computationally more efficient (adding is easier than multiplying)
- Convert back into probability at the end (if necessary!)

$$\log(p_1 \times p_2 \times \cdots \times p_n) = \log(p_1) + \log(p_2) + \cdots + \log(p_n)$$

Evaluation

- How good is a language model?
- Does it prefer “good” sentences to “bad” ones?
- Does it assign higher probabilities to “real” sentences rather than “ungrammatical” or “implausible” sentences?

Extrinsic evaluation

- Put each model in a task which requires a language model
 - spelling correction
 - machine translation
 - speech recognition
- Run the task and get an accuracy for each model
 - how many misspelt words corrected properly?
 - how many words translated correctly?
- Problems:
 - time-consuming
 - other factors affecting performance

Intrinsic evaluation

- Does the model assign higher probabilities to seen sentences than to unseen sentences?
- We trained the model's parameters on a training set
- We must test it on data that was **not used to train the model**
 - a test set
 - if we test on the training set, sentences will have artificially high probabilities
 - and it would be cheating!

Perplexity

- The best language model is one that best predicts an unseen test set
 - returns the highest $P(\text{sentences})$
- Perplexity is the inverse probability of the test set, normalised by the number of words

$$PP(W) = P(w_1, w_2, w_3, \dots, w_N)^{-1/N}$$



$$PP(W) = e^{-1/N \log P(w_1, w_2, w_3, \dots, w_N)}$$

- this assumes that we have calculated probability as a sum of logs
- multiplying by $-1/N$ first and then raising e to this power, makes the computation possible with floating point numbers

Minimising perplexity

- Example:
 - training 38 million words, testing 1.5 million words (WSJ text)

	unigram	bigram	trigram
Perplexity	962	170	109

Maximising probability is the same as minimising perplexity

- Perplexity should only really be compared for the same training and testing corpora

Generalisation in N-gram Language Models

Lecture 2, Part 2



A Toy Bigram Model

- I like to cook Chinese food.
 - I want to eat dinner.
 - They want to eat Indian food.

A Toy Bigram Model

- I like to cook Chinese food.
 - I want to eat dinner.
 - They want to eat Indian food.
 - They eat Chinese dinner.

Generation

- The Shannon-Visualisation Method
 - Choose a random bigram (_ST.,w) according to its probability
 - Now choose another random bigram (w,x) according to its probability
 - And so on until we choose _END
 - Then string the words together

_ST.	I						
	I	want					
		want	to				
			to	eat			
				eat	Chinese		
					Chinese	food	

I want to eat Chinese food

Approximating Shakespeare

1 gram	-To him swallowed confess hear both. Which. Of save on train for are ay device and rote life have
2 gram	- What means, sir. I confess she? then all sorts, he is trim, captain.
3 gram	- This shall forbid it should be branded, if renown made it empty.
4 gram	- King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in; - It cannot be but so.

- $N = 884,647$ tokens. $V = 29,066$
- Shakespeare produced 300,000 bigram types out of V^2 possible bigrams (840 million)
 - so 99.96% of the possible bigrams will have 0 probabilities
- Quadrigrams even worse
- What's coming out looks like Shakespeare because it is Shakespeare

Overfitting

- N-grams only work well for word prediction if the test corpus looks like the training corpus
- It often doesn't
- If Shakespeare had written one more play ... new possibilities for quadrigrams, trigrams, bigrams and even unigrams
- Models need to be robust – they need to generalise to unseen data

Zeros

TRAINING SET

- ... denied the allegations
- ... denied the reports
- ... denied the claims
- ... denied the request

TEST SET

- ... denied the loan
- ... denied the offer

$$P(\text{"offer"} \mid \text{"denied the"}) = 0$$

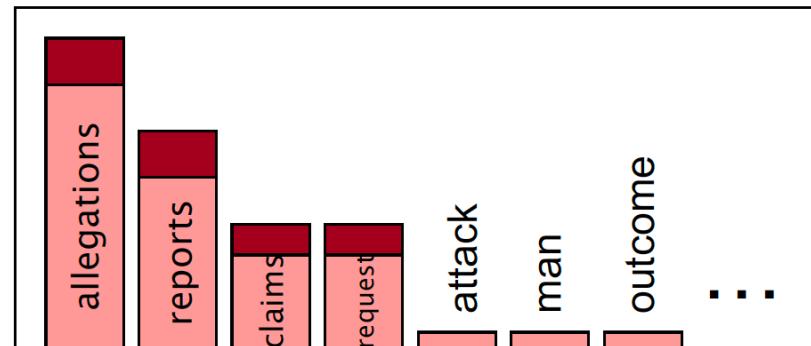
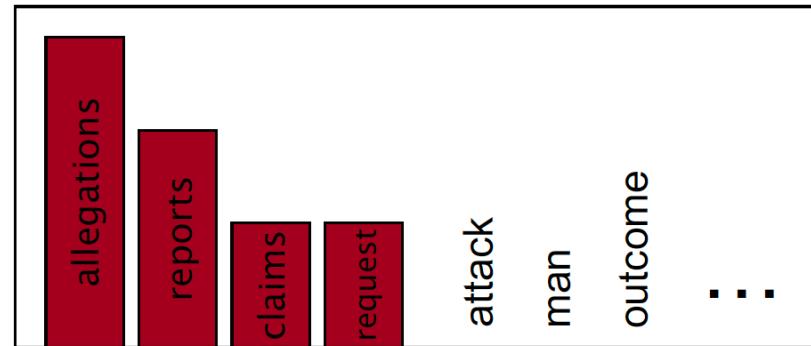
Trigrams (or even bigrams or unigrams) with zero probability in the training set mean that we

- assign zero probability to test set
- cannot calculate perplexity

Smoothing intuition

- When we have sparse statistics, steal probability mass from observed events to generalise to unobserved events

count (w "denied the")	smoothed count
allegations 3	allegations 2.5
reports 2	reports 1.5
claims 1	claims 0.5
request 1	request 0.5
	OTHER 2
total 7	total 7



Add-one estimation

- Also called Laplace smoothing
- Pretend we saw each word one more time than we did
- Just add one to each count!
- Can be very effective for some problems
 - where number of zeros isn't so huge
 - e.g., text classification
- But for n-grams, rarely used
 - assigns too much mass to unseen co-occurrences (even add-k)
 - and leads to massive, unwieldy models

Unknown words

- Test corpus contains words that the training corpus doesn't
- Training corpus also contains words that the test corpus doesn't
- Which training corpus words are least likely to be in the test corpus?
- Fix the vocabulary (top N words in training corpus or all words which occur *for more times*)
- Create a <UNK> token which captures probabilities for *Out-Of-Vocabulary (OOV)* words.

Unseen bigrams

- The <UNK> token allows us to estimate the probability of seeing an *OOV* word
- It even lets us estimate the probability of seeing two *OOV* words together or an *OOV* word with an in-vocabulary word
- But it does not allow us to estimate the probability of two in-vocabulary words which have not been seen together before
- What to do?

Absolute discounting

- Subtract a little from each bigram count in order to save probability mass for unseen events.
- How much?
- Church and Gale (1991)
 - Divided 22 million words of newswire text into training and testing sets
 - for each bigram count in the training set, what is its average bigram count in the test set?

Training count	Testing count
0	0.0000270
1	0.448
2	1.25
3	2.24
4	3.23
5	4.21
6	5.23
7	6.21
8	7.21
9	8.26

Absolute discounting interpolation

- If we subtract d from each bigram count, how much probability mass do we save for unobserved bigrams?
- We need to keep track of the discounts made for each word
 - each time we discount a bigram $c(w_2|w_1)$, we add that discount to a dummy token **lambda** for that word $c(\lambda|w_1)$
 - normalise counts as probability distributions as before
 - For a smoothed probability estimate of any bigram, **interpolate**
 - sum the observed (discounted) probability and a proportion of reserved probability mass (according to the unigram probability of w_2)

$$P_e(w_2|w_1) = P_d(w_2|w_1) + P_d(\lambda|w_1) \times P(w_2)$$

The San Francisco problem

- The *absolute discounting interpolation method* divides up the reserved probability mass according to the **unigram** probability of the target word
- **Assumption:** Higher probability words are more likely to be seen in novel word combinations
- **Not always true**
- “Francisco” is a high frequency word but only in the context of “San”
- In fact, high frequency means we have more evidence that a novel combination is unlikely

Kneser-Ney smoothing

- Don't assign the reserved probability mass according to the unigram probability
- Calculate a separate probability for each word which is its likelihood of being seen in novel word combinations

$$P_{KN}(w) = \frac{|\{w_j | c(w_j, w) > 0\}|}{\sum_i |\{w_j | c(w_j, w_i) > 0\}|}$$

$$P_e(w_2 | w_1) = P_d(w_2 | w_1) + P_d(\lambda | w_1) \times P_{KN}(w_2)$$

Web-scale language models

- Google n-gram corpus
 - 1 billion five-word sequences over 13 million unique word types

4-gram	Count
serve as the incoming	92
serve as the incubator	99
serve as the independent	794
serve as the index	223
serve as the indication	72
serve as the indicator	120
serve as the indicators	45

Efficiency considerations:

- words stored as 64-bit hash number
- probabilities quantized using 4-8 bits (rather than 8-byte floats)
- n-grams stored in reverse tries
- n-grams shrunk by pruning

Stupid backoff

- Can apply full Kneser-Ney smoothing to web-scale language models
- But Brants et al. (2007) showed that a much simpler algorithm might be sufficient at this scale
- Stupid backoff gives up on the idea of making it a true probability distribution
- No discounting of higher order probabilities
- If a higher-order n-gram has a zero count, simply “*backoff*” to a lower order n-gram, with a fixed weight ($\lambda = 0.4$)

$$S(w_i|w_{i-k+1}^{i-1}) = \begin{cases} \frac{\text{count}(w_{i-k+1}^i)}{\text{count}(w_{i-k+1}^{i-1})} & \text{if } \text{count}(w_{i-k+1}^i) > 0 \\ \lambda S(w_i|w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$$

Coming up

- Neural language models (week 4)
 - feed-forward
 - RNNs and LSTMs
 - character-based

References

- Brants, T. et al. 2007. Large language models in machine translation. *In EMNLP/CONLL 2007*
- Church, K.W. and Gale, W.A. 1991. A Comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of English bigrams. *Computer Speech and Language, 5, 19-54*
- Mikolov, Yih and Zweig 2013 – Linguistic Regularities in Continuous Space Word Representations, (NAACL-HCT 2013)
- Zweig, G. and Burges, A. 2011. The Microsoft Research Sentence Completion Challenge. Microsoft Technical Report

AdvNLP/E Lecture 4

Language Modelling 2

Dr Julie Weeds, Spring 2024



Language Models

PREVIOUSLY

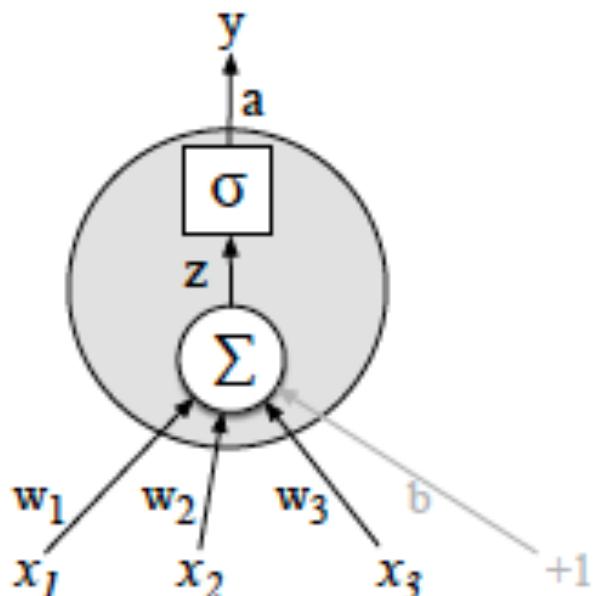
- N-gram language modelling
 - Markov assumptions
 - perplexity and evaluation
 - smoothing

THIS TIME

- Neural language models
 - feed forward
 - recurrent
 - long-short term memory
 - convolutional
 - word-based vs character-based

Neural Networks

A Neural Unit



y is the output and is equal to the activation of the unit

$$y = f(z)$$

z is a weighted sum of inputs

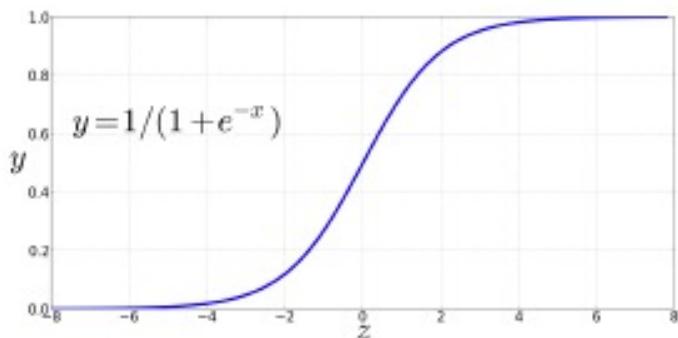
$$z = w \cdot x + b$$

w is the weight vector (same dimensions as input)

x is the input vector (3 dimensions here)

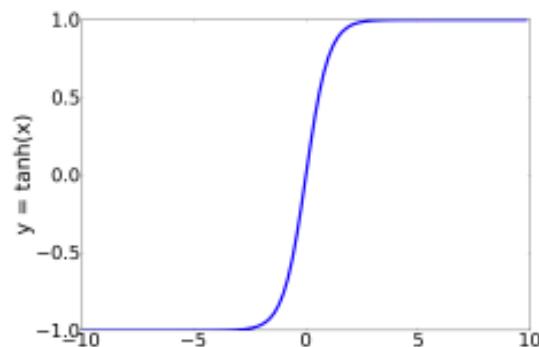
Activation functions

■ Introduce non-linearity



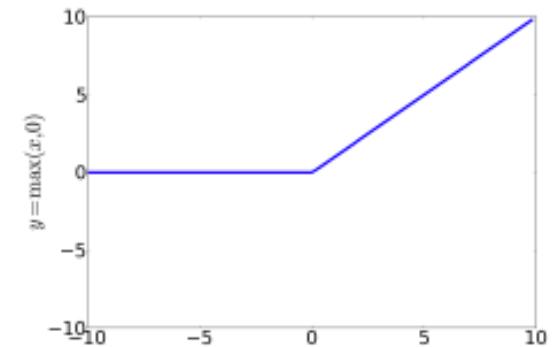
sigmoid

$$y = \sigma(z) = \frac{1}{1 + e^{-z}}$$



tanh

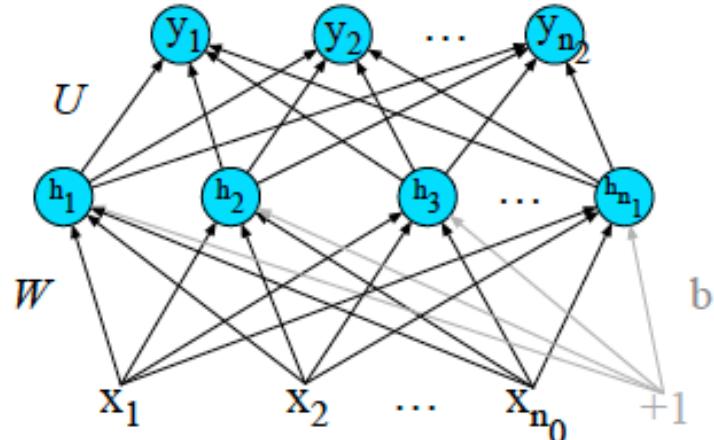
$$y = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$



ReLU

$$y = \max(z, 0)$$

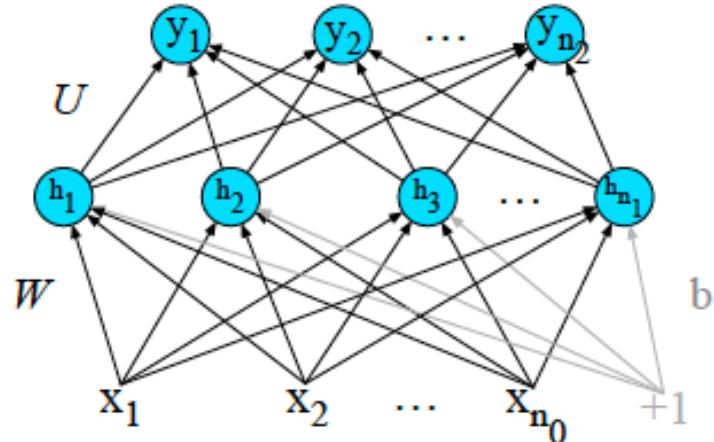
Feed-forward network



2-layer FF network with one hidden layer and one output layer

- Multi-layer network of connected neural units
- no cycles
- outputs from one layer are inputs to the next layer
- each unit takes a weighted sum of inputs and applies non-linearity
- fully-connected if each unit in each layer takes as input, outputs from all units in previous layer

Feed-forward network



2-layer FF network with one hidden layer and one output layer

$$y = \text{softmax}(Uh)$$

$$h = \sigma(Wx + b)$$

Softmax is used to turn the outputs into a probability distribution

$$y = \text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^d e^{z_j}}$$

Training neural networks

- For a **hard classification** task (i.e., one where there is only one correct answer)
 - cross-entropy loss** or **negative log-likelihood loss** is simply the log probability, computed by the network, of the correct class. Let \hat{y} be the output vector from the network and y be a k -dimensional one-hot vector encoding the correct answer (i)
- we want to find the parameters / weights which **minimize the loss function**
 - loss functions* are also sometimes referred to as *cost functions* and *objective functions*. Loss or cost function should always refer to penalties which we want to minimize. An objective function is anything we want to optimize on (and could include a reward function we want to maximise)

$$J = \text{Loss}(\hat{y}, y) = - \sum_{i=1}^k y_i \log \hat{y}_i = - \log \hat{y}_i$$

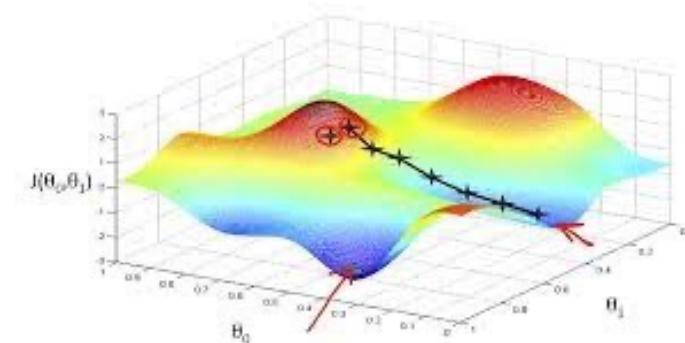
Gradient descent

- Randomly initialise parameter settings (W)
- Compute predictions
- Compute loss function (J)
- Compute (partial) derivatives of loss function with respect to parameters
- Back-propagate i.e. update parameters:

$$W = W - \alpha \frac{dJ}{dW}$$

learning rate

■ Repeat until convergence / loss is acceptably small



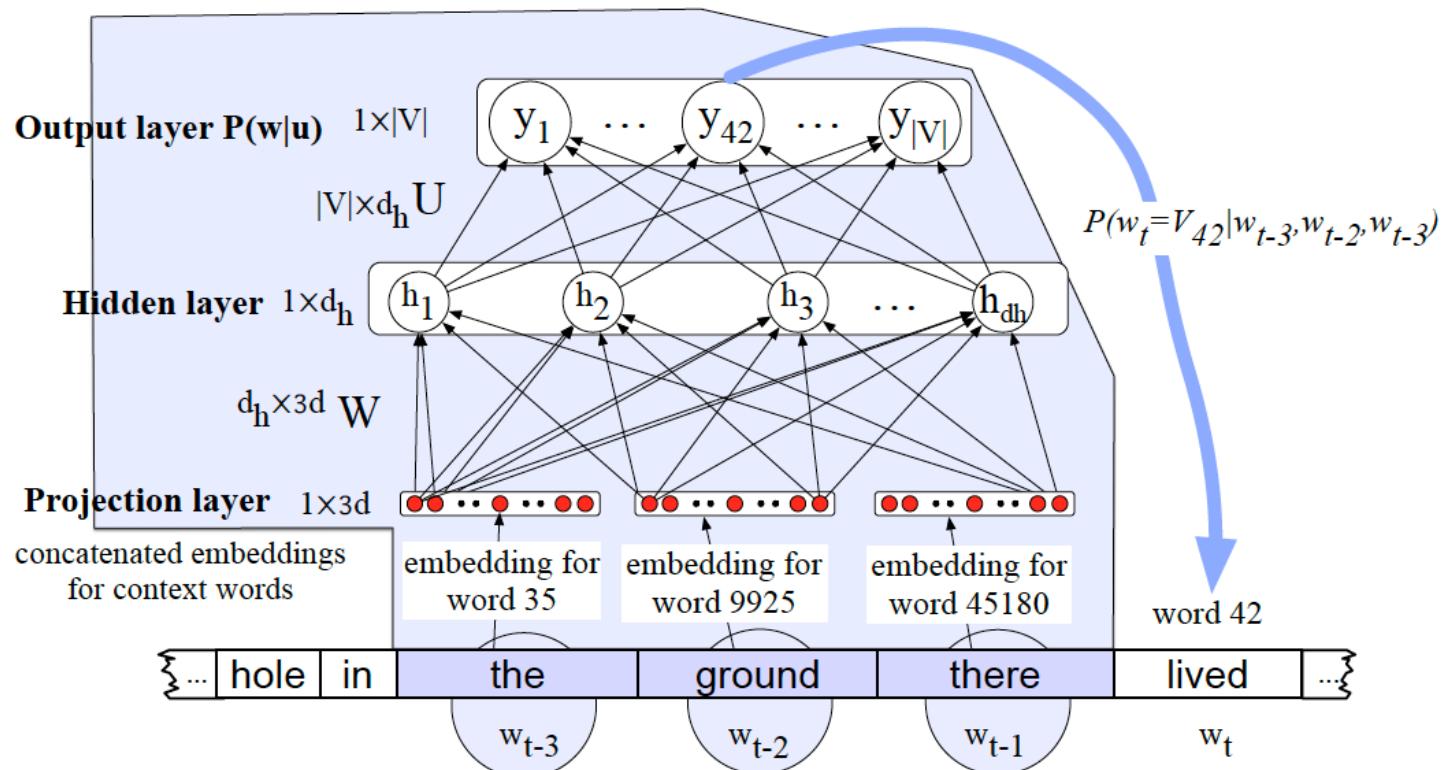
FF-NLM

Feed-forward neural language model

FF-NLMs (Bengio et al. 2003)

Output at time t: a probability distribution over possible next words

Input at time t: a representation of some number of previous words (w_{t-1}, w_{t-2} , etc)



FF-NLM

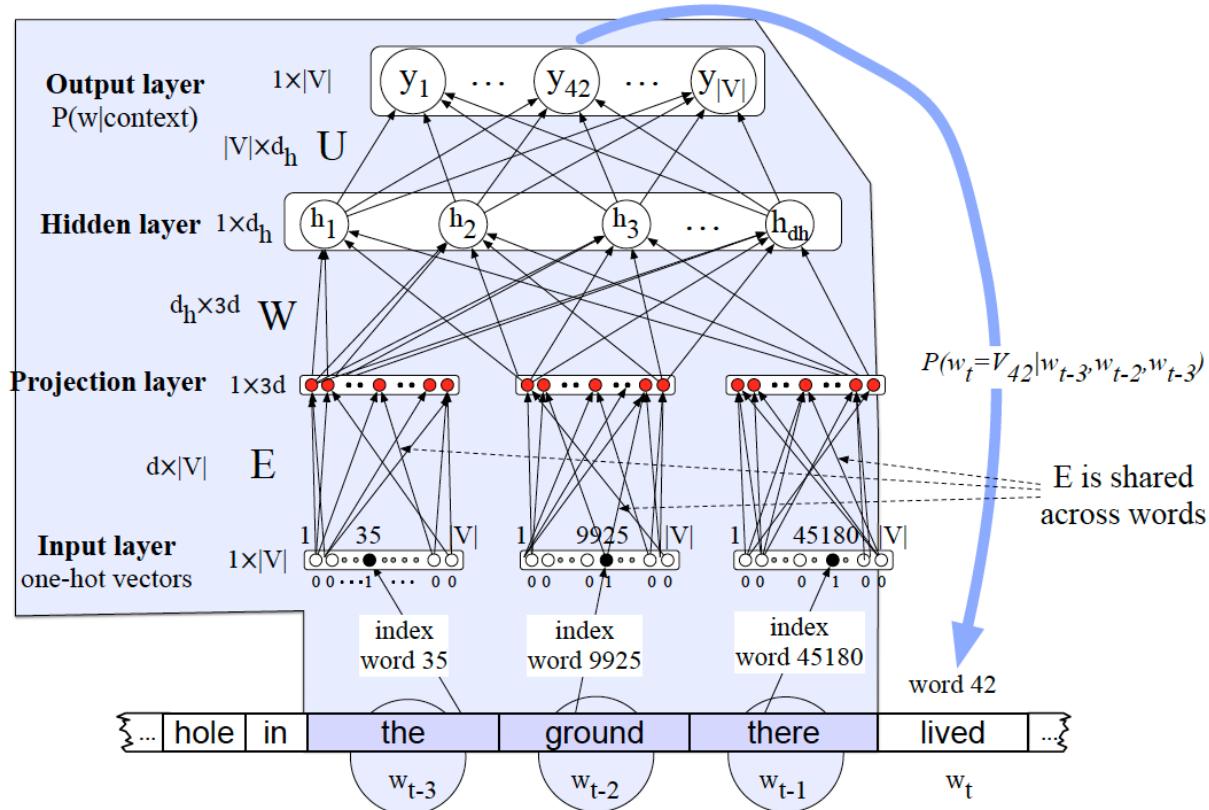
ADVANTAGES

- Generalizes over contexts of similar words (due to embeddings)
- Doesn't need smoothing
- Can handle longer histories

DISADVANTAGES

- Slower to train than N-gram model
- Still based on Markov assumptions
 - probability of word given the entire context is approximated based on the N previous words

Where do the embeddings come from?



- Embedding weight matrix E learnt at the same time as W and U
 - 3 layer network
 - random initialisation
- Pre-trained embeddings (e.g., LSA)
 - 2 layer network (or 3 layer network with **frozen** embedding matrix)
 - or **tuned** in a 3 layer network

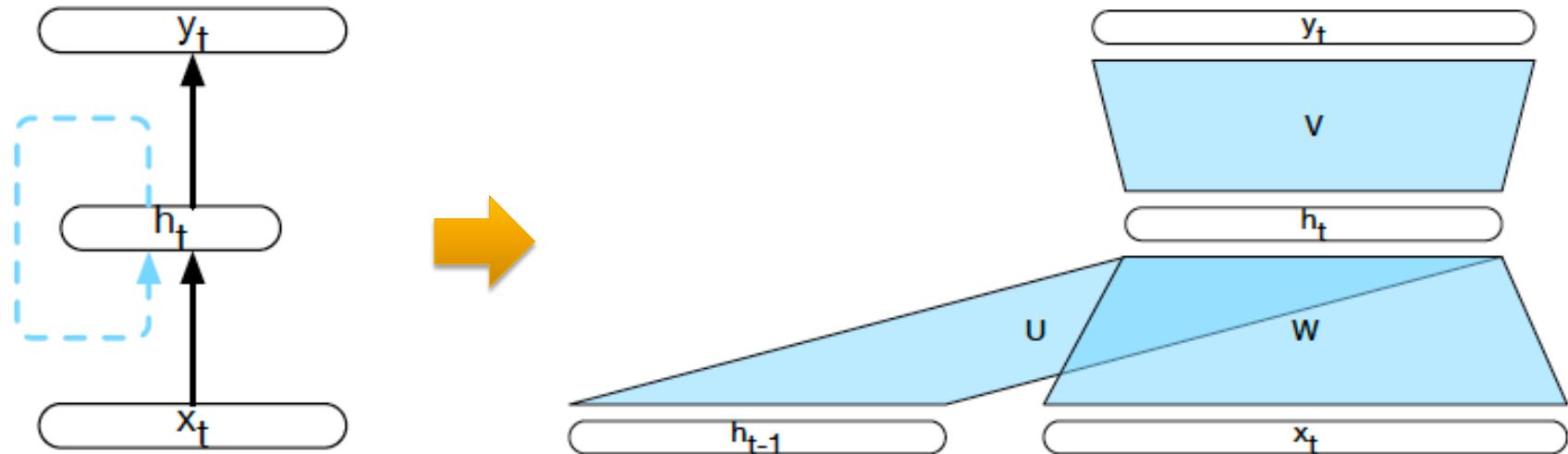
Log-bilinear language model

- Variant on FF-NLM
- No non-linearity in the hidden layer
- Faster to train
- See Mnih and Hinton (2008), Botha and Blunsom (2014)

RNNs

Recurrent neural networks

Simple Recurrent Networks



- activation value of the hidden layer depends on
 - the current input; and
 - the activation value of the hidden layer from the previous step

RNN

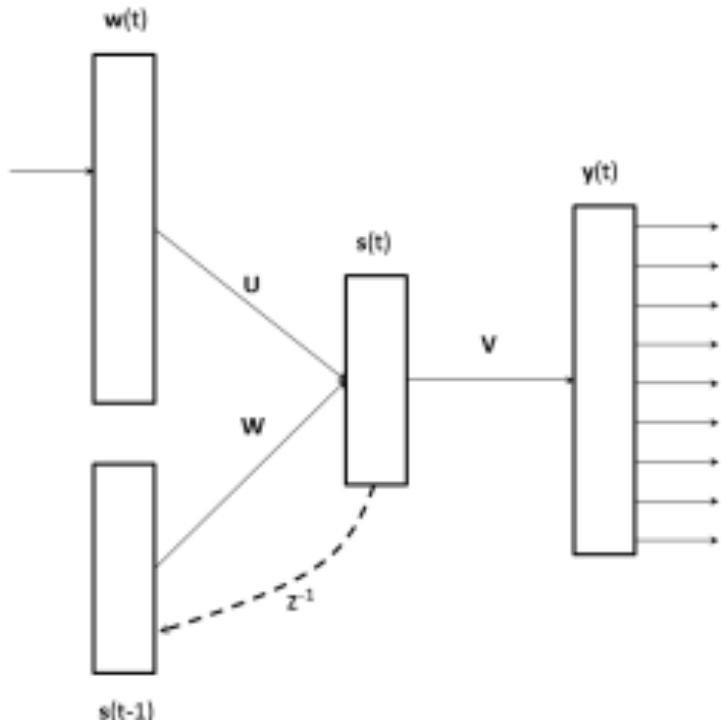
ADVANTAGES

- Hidden layer from previous timestep provides memory
- No fixed length limit on amount of prior context
 - The weights U determine how the network should use past context in calculating output for current input

DISADVANTAGES

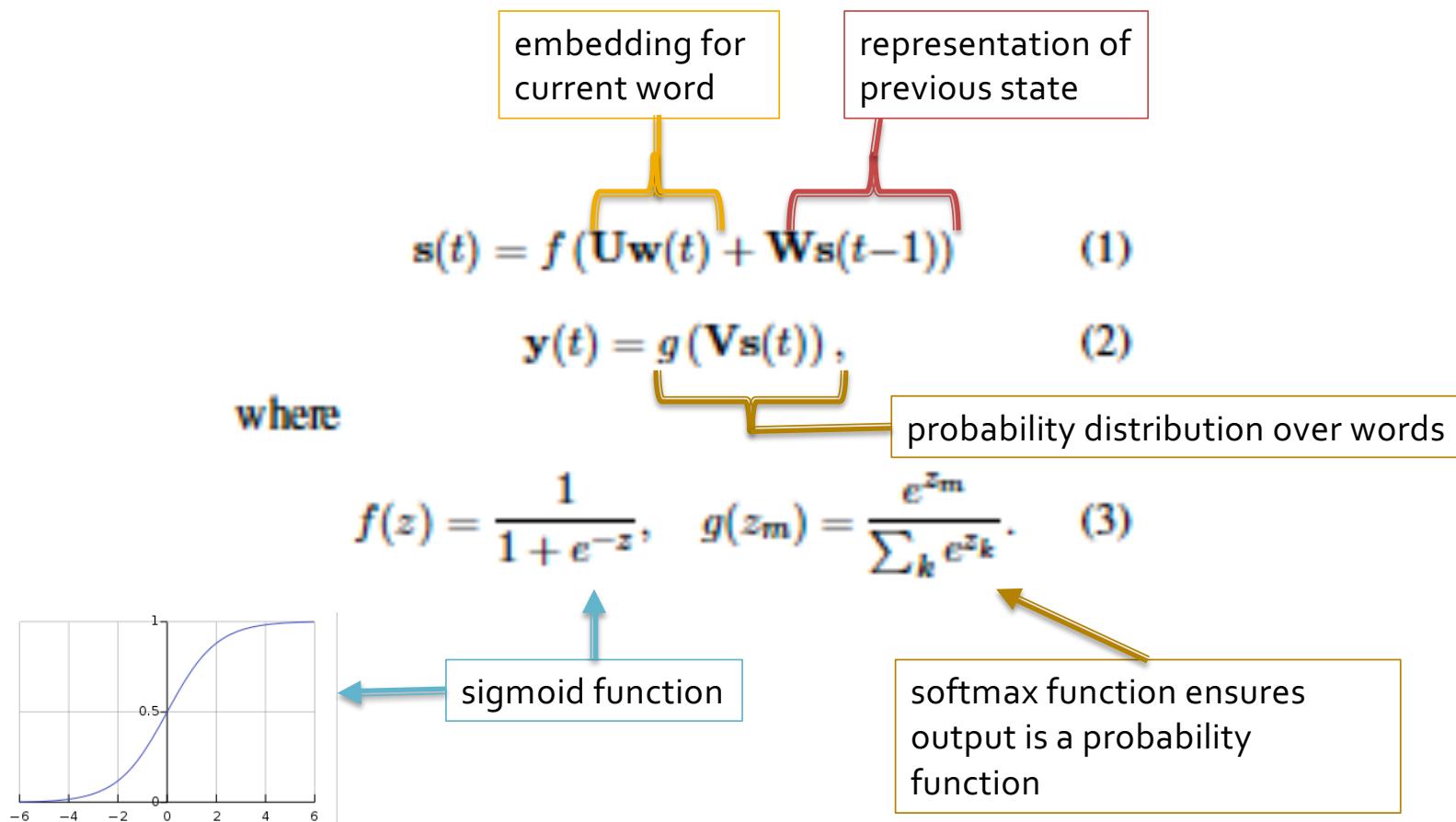
- Whilst RNN theoretically has access to the entire preceding sequence, in practice, information tends to be fairly local
- Difficult to capture long-range semantic dependencies
 - Hidden layer weights need to capture information for short range dependencies (e.g., pluralisation agreement) and long range dependencies

Recurrent Neural Network Language Model (Mikolov et al. NAACL 2013)



- Input and output vectors (w and y) have dimensionality of the vocabulary
- Output vector y is a probability distribution over words
- Word representations are found in the columns of a matrix U (selected by the z of N encoding of the current word $w(t)$)
- The hidden layer ($s(t)$) maintains a representation of sentence history
- Trained with back-propagation to maximise data log-likelihood

Computation of hidden and output layers

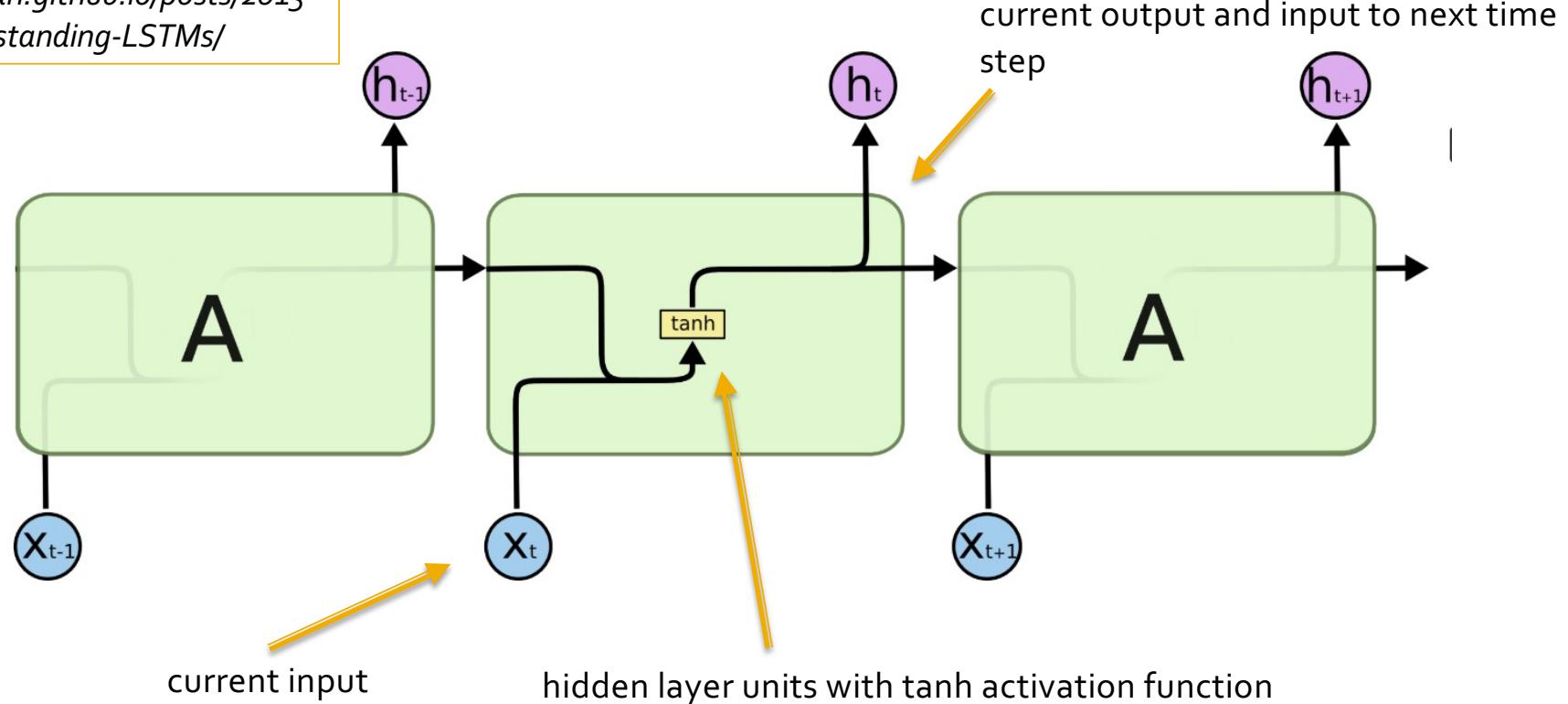


LSTMs

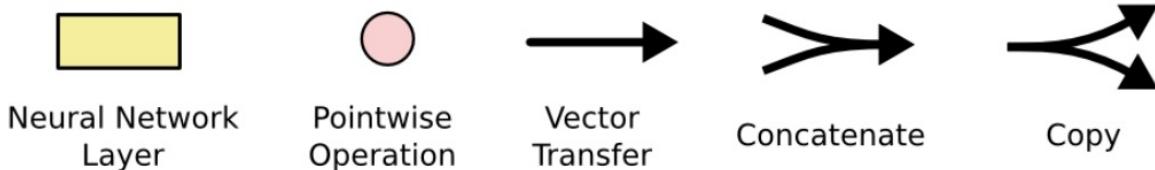
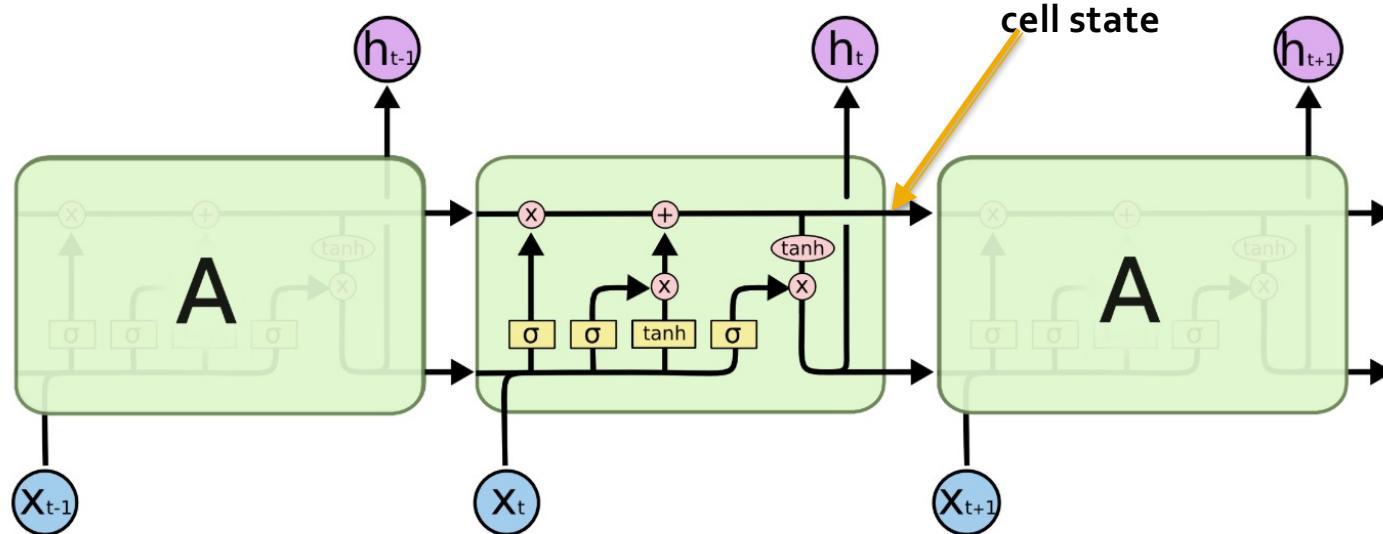
Long short-term memory networks

Unrolling a simple RNN

Diagrams from Colah's blog:
<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>



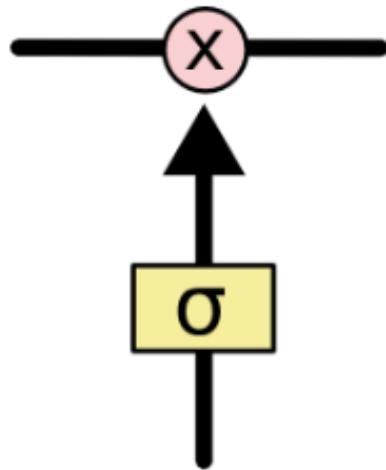
Unrolling an LSTM



Core idea is that at each time step, hidden layer activation values depend on

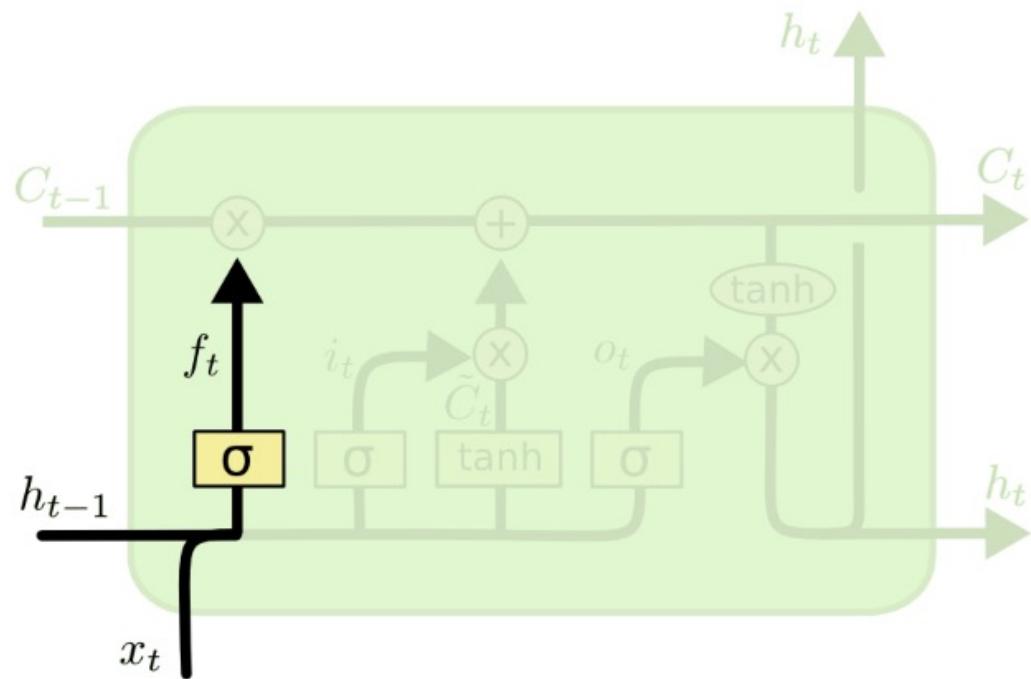
- current input
- activation values from the hidden layer at previous time step (*short term memory*)
- activation values from the **cell state** (*long term memory*)

Gates inside neural units



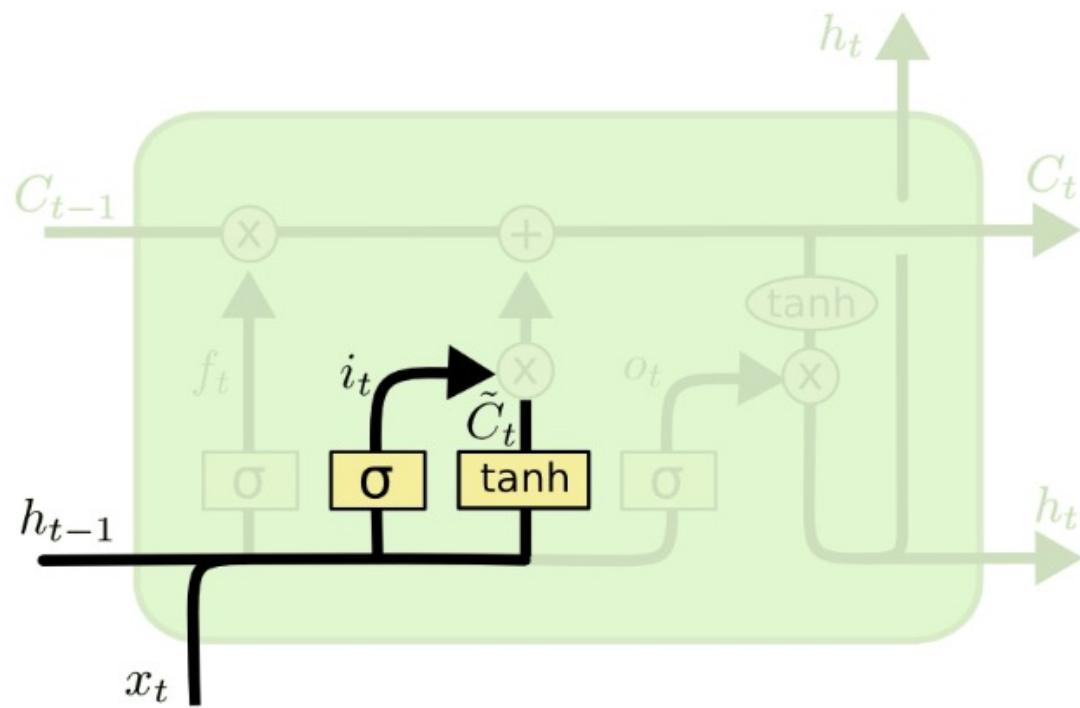
- a way to control flow of information
 - optionally allow information through
- composed of
 - a sigmoid neural net layer
 - a pointwise multiplication
- Sigmoid outputs numbers between 0 and 1, describing how much of each component should be let through
 - it controls the gate

Forget



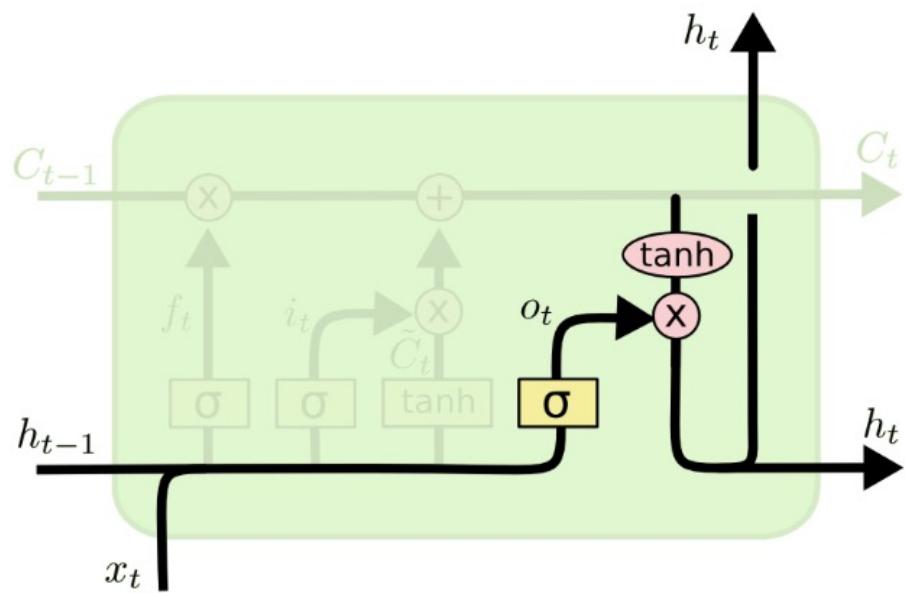
- What in the current cell state should be forgotten?
- Controlled by the current input and the short term memory

Input



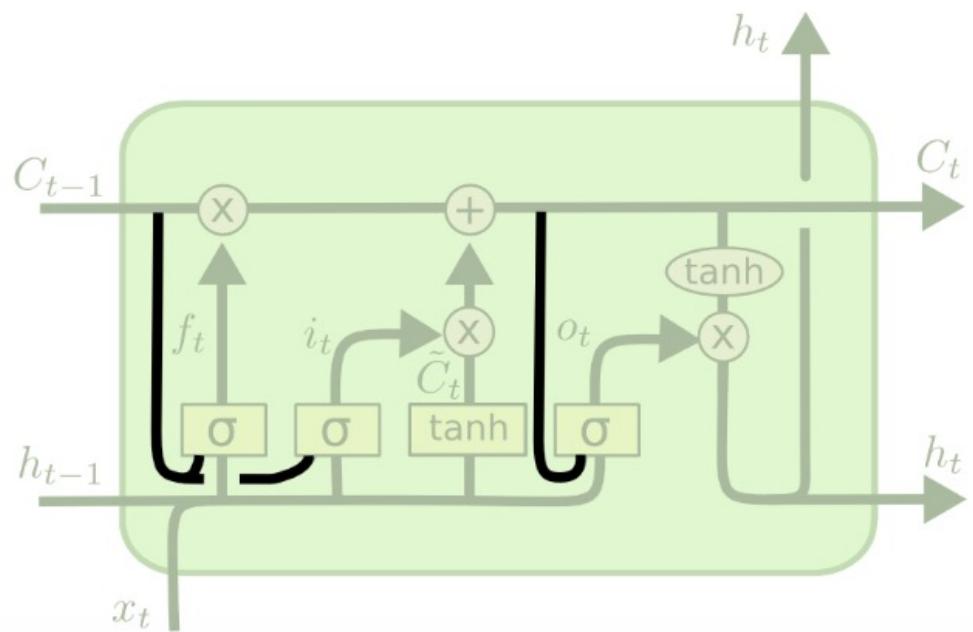
- What of the current input and short-term memory should be put into the long-term memory?
 - sigmoid decides which values to update
 - tanh decides what those values should be

Output



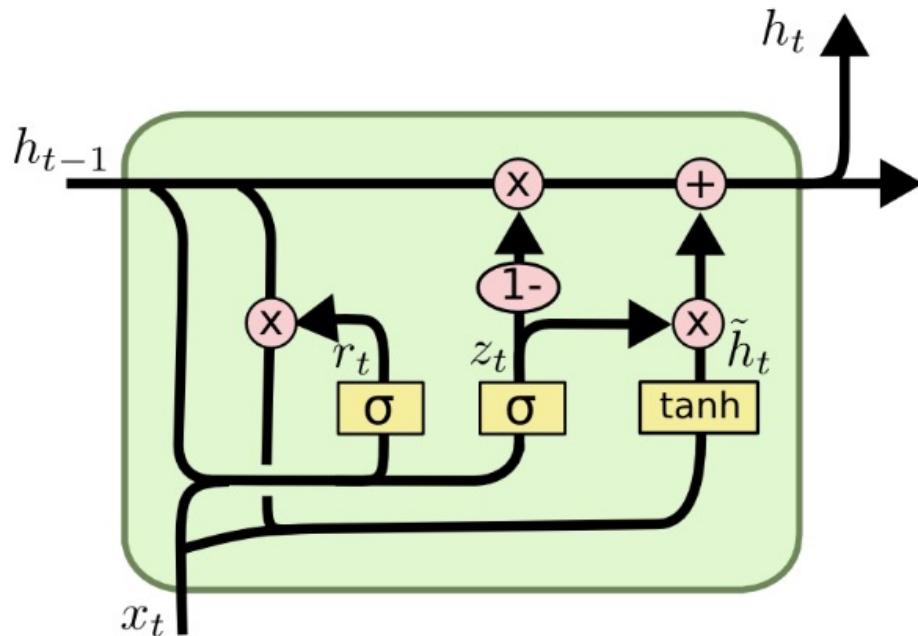
What should be **output** at this timestep?

Peephole variant



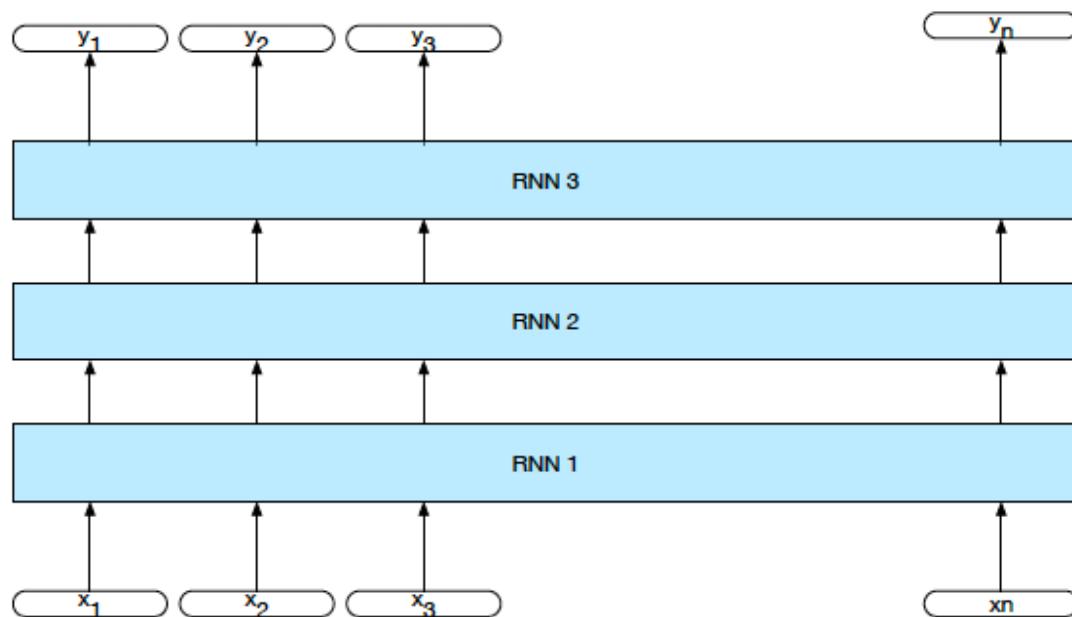
Gates also have access to the cell state when deciding what to let through

Gated recurrent unit (GRU) variant



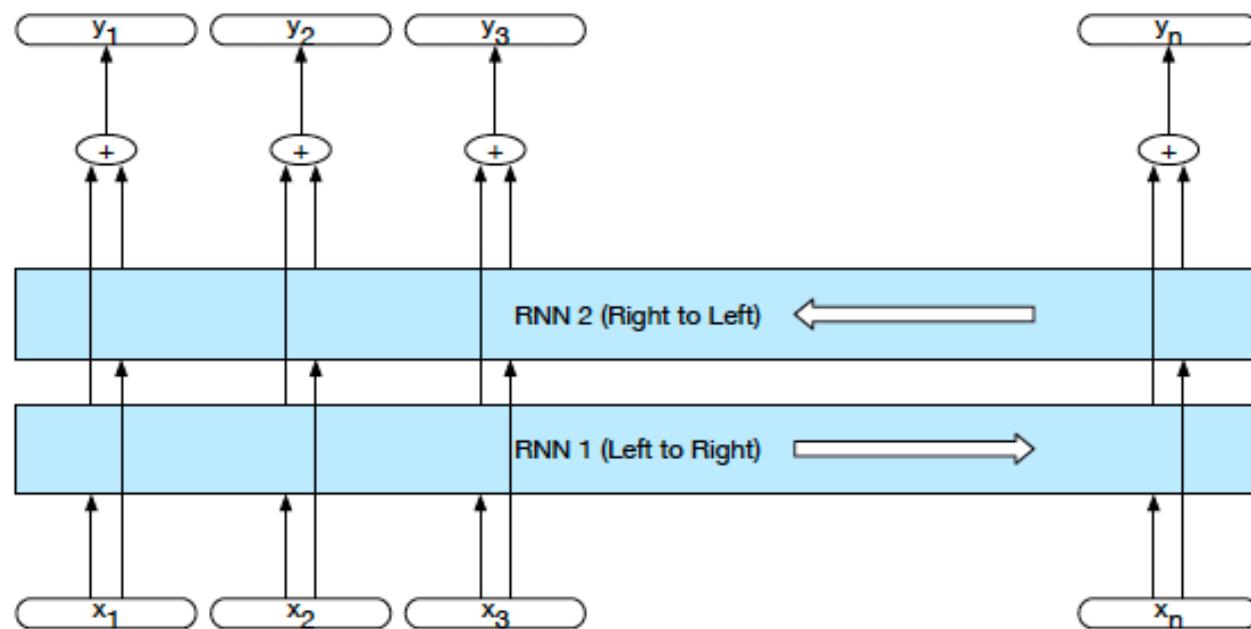
- Simpler alternative
 - Forget and input gates merged into a single update gate
 - hidden and cell state merged

Stacked RNNs



- Output (at each time step) from RNN is fed as input into another RNN
- Stacking leads to **deep** networks and **deep** learning
- Different layers induce representations at different levels of abstraction
 - short range syntactic dependencies in early layers
 - long range semantic dependencies in deeper layers

Bidirectional RNNs



- Take advantage of right context as well as left context
- Pair of RNNs
 - one is trained from left-to-right
 - the other is trained from right-to-left
- Add or concatenate hidden layers from both RNNs to produce output at each timestep

Beyond stacked bidirectional LSTMs

- Attention
 - at each step allow the RNN to pick information from larger collection of information
- Encoder-decoder networks
- Transformers
- Grid LSTMs

Characters and Convolutions

Part 2 of lecture

Problems for word-based NLMs

- Sparsity!
- Can generalise using similar words but
- How reliable are embeddings for low frequency / unknown words?
- Should we replace all low-frequency words with “UNK” token?
- Can we do better?

Character-based NLMs

- So far core input unit has been the word
 - sparsity problems
- What if we make the core input unit a character?
 - no sparsity problems
 - can we learn to make language predictions based on character embeddings?

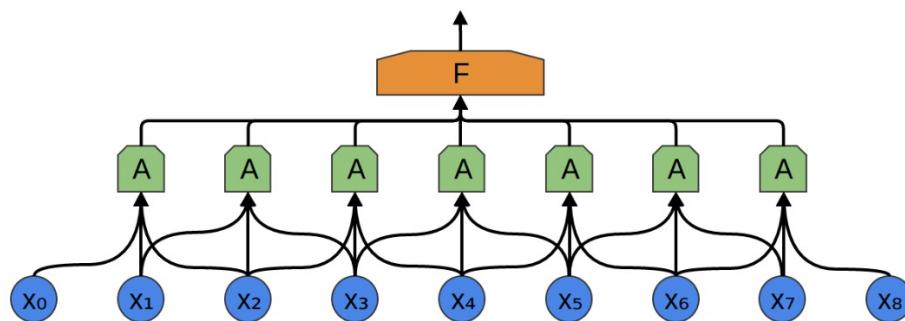
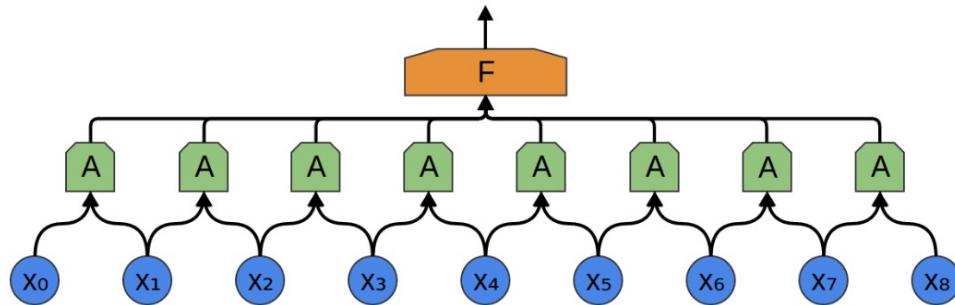
CNNs

Convolutional neural networks

CNNs

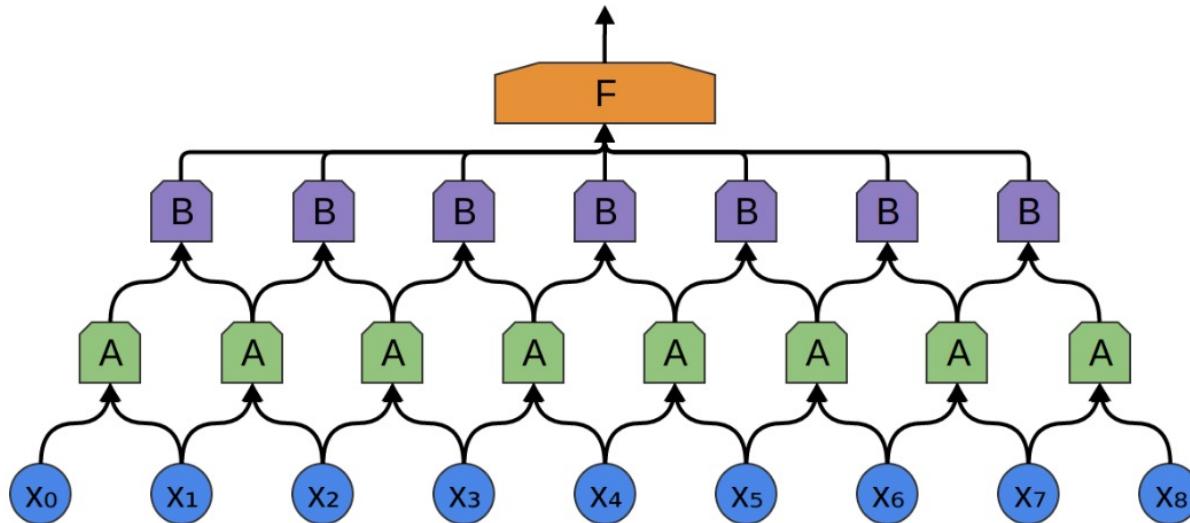
- Very common in vision
 - convolutional layers used to detect features in an image
 - e.g. an edge
 - features can be anywhere in the image
 - but presence determined by looking at neighbouring pixels / inputs
- Can also be used on text

Convolutions



- Convolutional layers look at groups of inputs or neurons, e.g.,
 - convolution with window size 2
 - convolution with window size 3
- **kernel function A** is learnt which looks for or filters for a particular pattern / feature in the input e.g.,
 - 2 character sequence "un"
 - 3 character sequence "ing"

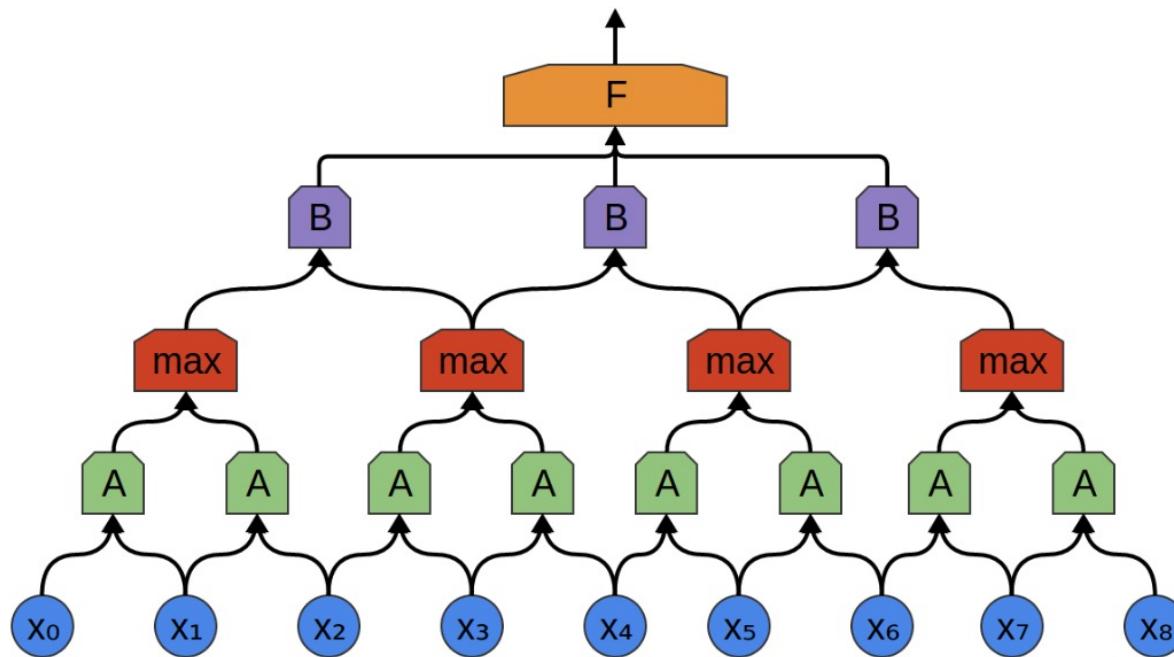
Stacking convolutional layers



Stacking allows detection of complex features composed from simpler features e.g.,

- face detection requires eye detection

Max pooling



Max-pooling layer takes the maximum of its inputs

- is a feature present anywhere in this chunk of input?
- does not care where it is in the input

Kernel Functions

- Individual kernel function detects an individual feature in the input
- In practice hundreds or thousands of kernel functions needed for different character combinations
- Kernel functions are just neural network weights
 - learnt by training on a corpus
 - kernels learnt depend on the corpus / task

Advantages of character-aware NLMs

- allows morphological information to be utilised
- particular useful for low-frequency words

Using NLMs

- What can we do with a NLM beyond predicting the next word?
- Calculate the probability of a word sequence
 - speech recognition
 - machine translation
 - spelling correction
- Use as input to text classification task
 - relevancy or sentiment classification
 - paraphrase identification
- Use in sequence labelling tasks
 - part of speech tagging
 - named entity recognition

Next time:

- Sequence labelling (Named Entity Recognition)

References

- Bengio, Y. et al. 2003. A neural probabilistic language model. In *Journal of Machine Learning Research*.
- Botha, J., and Blunsom, P. 2014. Compositional morphology for word representations and language modelling. In *Proceedings of ICML*
- Kim, Y. et al. 2016. Character-aware neural language models. In *Proceedings of the AAAI*.
- Mikolov, T. et al. 2010. Recurrent neural network based language model. In *Proceedings of Interspeech*
- Mnih and Hinton, G. 2008. A scalable hierarchical distributed language model. In *Proceedings of NIPS*.

AdvNLP Week 5

Named Entity Recognition

Julie Weeds, Spring 2024



Previously

1. Distributional Semantics
 - Bootstrapping semantics from context
2. Word embeddings
 - Dimensionality reduction, neural language models, word2vec, GloVe
3. Probabilistic language models
 - n-gram modelling, perplexity and generalization
4. Neural language models
 - Word-based and character-based

Week 5 overview

1. Named Entity Recognition

- What and why
- Challenges
- NE types
- Sequence labelling
- evaluation
- features

2. Approaches to NER

- Rule-based
- Generative (e.g. HMM)
- Discriminative (e.g. CRF)
- Neural Models (e.g. CNN-BiLSTM-CRF, ACE)
- Transformer LMs

3. Getting Data

- Supervised
- Semi-supervised
- Unsupervised

Named Entity Recognition

- The detection and classification of named entities in a text
- A named entity is anything which can be referred to with a **proper name** e.g., “*Boris Johnson*”, “*Pizza Hut*”, “*Brighton*”, but may also include dates, times, prices and more
- Often multi-word phrases
- Semantic structured prediction

Why Named Entity Recognition?

NER is the basis for downstream NLP tasks:

- Sentiment or intent regarding entities
- The relation between entities in fact/relation extraction
- Entity-linking (co-reference resolution) and Knowledge Base Population (KBP)
- Entity-informed search and question answering regarding an entity (Google/Bing Infobox, Google knowledge graph (GKG))
- Competitive/Product Intelligence, brand insights

Sample Text

1. Manchester United striker Wayne Rooney has agreed a new five-and-a-half year contract worth up to £300,000 a week.
2. The 28-year-old England international will extend his current contract by four years, tying him to the club until June 2019.
3. Rooney joined United from Everton in August 2004 and is only 42 goals shy of passing Bobby Charlton's record of 249 for United.
4. The former Everton forward has taken 430 games in all competitions to score his 208 goals for the Red Devils.
5. The club have yet to confirm the contract but BBC sports editor David Bond said the deal had been agreed.

Types of Named Entity

Type	Tag	Examples
People	PER	Wayne Rooney
Organization	ORG	Manchester United
Location	LOC	Manchester
Miscellaneous	MISC	£300,000

- The most popular NER benchmark, CoNLL-2003, includes only these 4 entity types
- Other benchmarks include more types, e.g. OntoNotes v5.0 includes 18 types
- Temporal expressions and numerical expressions are often included
- Some approaches aim to discover any entity regardless of type
- Which entity type set might be most useful?

Sample Text

1. Manchester United striker Wayne Rooney has agreed a new five-and-a-half year contract worth up to £300,000 a week.
2. The 28-year-old England international will extend his current contract by four years, tying him to the club until June 2019.
3. Rooney joined United from Everton in August 2004 and is only 42 goals shy of passing Bobby Charlton's record of 249 for United.
4. The former Everton forward has taken 430 games in all competitions to score his 208 goals for the Red Devils.
5. The club have yet to confirm the contract but BBC sports editor David Bond said the deal had been agreed.

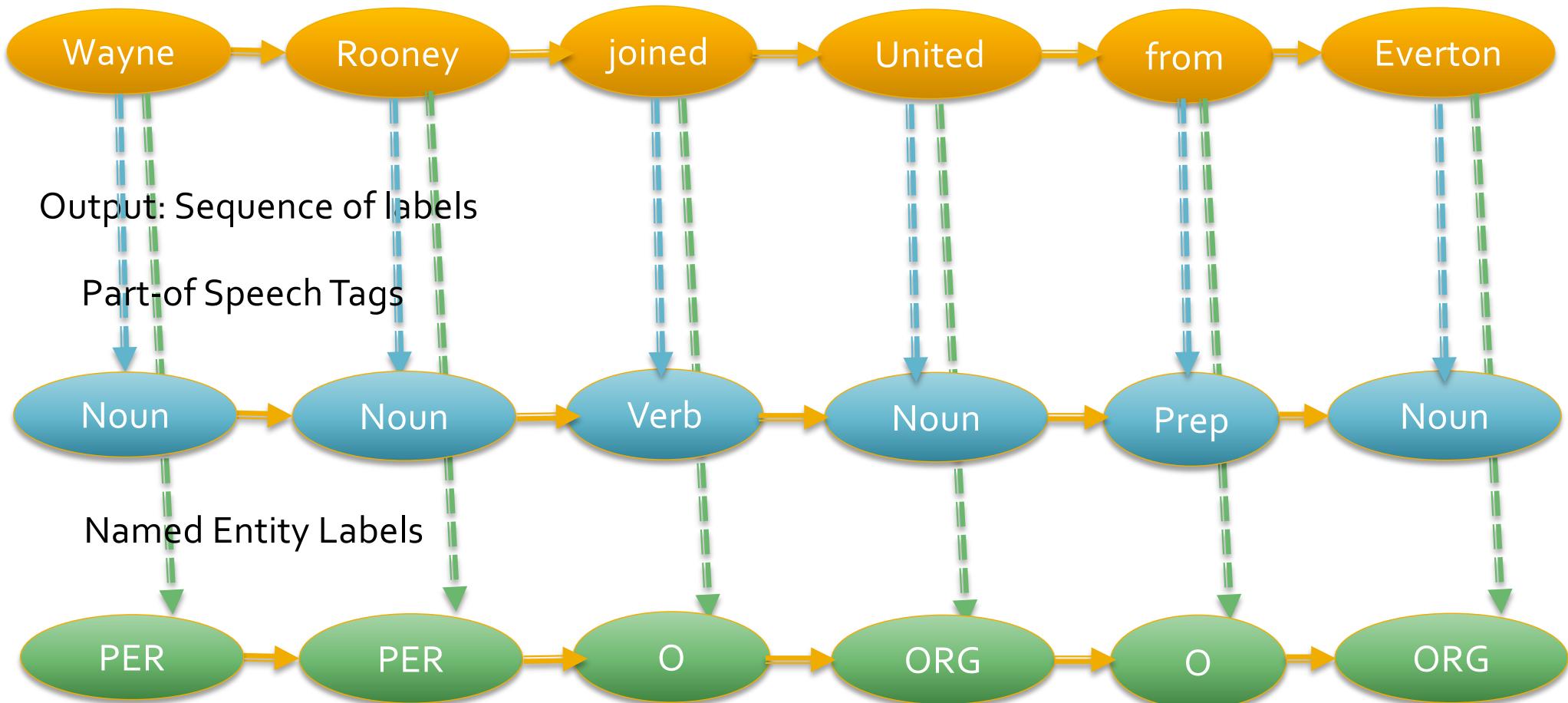
Types of Named Entity

Type	Tag	Examples
People	PER	Wayne Rooney
Organization	ORG	Manchester United
Location	LOC	Manchester
Miscellaneous	MISC	£300,000

- Difficulties:
 - Two types of ambiguity: boundary and type
 - Boundary detection (e.g. the New York Times)
 - Two types of type ambiguity: entity-noun, entity-entity
 - Variation, even with reliable text sources (e.g. United, Utd, Utd.)

Sequence Labelling

Input: Sequence of Words



Sequence Labelling – IOB encoding

Words	Label
Manchester	B-ORG
United	I-ORG
striker	O
Wayne	B-PER
Rooney	I-PER
has	O
agreed	O
a	O
new	O
£300,000	B-MISC
contract	O
.	O

I → inside a chunk
O → outside a chunk
B → beginning a chunk

Sequence Labelling – IOB encoding

Words	Label
Manchester	B-ORG
United	I-ORG
striker	O
Wayne	B-PER
Rooney	I-PER
has	O

I → inside a chunk
O → outside a chunk
B → beginning a chunk

- Turn span identification into a sequence labelling problem using IOB encoding – 1 tag per token
- Find the correct spans of text that constitute an entity
- Typically entities should be identified with the correct type and exact position
- Sometimes multiple scores are included including partial overlaps
- Also known as BIO encoding

Evaluation

- $P = \frac{TP}{TP+FP}$
 - i.e., the proportion of named entities identified that are actually named entities
- $R = \frac{TP}{TP+FN}$
 - i.e., the proportion of named entities in the test data that were correctly identified
- $F_1 = \frac{2PR}{P+R}$
 - i.e., a harmonic mean of both

Features commonly used in NER

Entity Type	Tag	Examples
People	PER	Wayne Rooney
Organization	ORG	Manchester United
Location	LOC	Manchester
Miscellaneous	MISC	£300,000

Which rules or features might be useful in identifying these entity types?

Features commonly used in NER

Feature	example
lexical item	united
stemmed lexical item	unite
shape	initial capitalization
part-of-speech	N
syntactic chunk labels	part of NP
presence in gazetteer or name list	e.g., list of first names or place names
predictive words in context	Mr.
Bag of words/n-grams	preceding word(s) e.g., "striker"

What might the choice of features depend on?

Typical Supervised Approach

- Humans annotate training documents
- IOB encoding
- Feature extraction performed if necessary
- Classifier (e.g., HMM, SVM, MEMM, CRF, LSTM or a combination) trained
- Evaluation carried out on held out test data typically using precision, recall and the F-measure (F1)
- Paper is published (possibly)

Classification

- Sequence labelling task –
 - We want to assign the most likely sequence of labels given the observed tokens
 - NOT the most likely label for each token given all of the other tokens
 - So, this might rule out simple classifiers such as Naïve Bayes and Logistic Regression / MaxEnt

AdvNLP Week 5

Named Entity Recognition

Part 2

Julie Weeds, Spring 2024



NER Approaches

- Approximate chronology of NER approaches
 - Rule-based
 - Generative (e.g. HMM)
 - Discriminative (e.g. MEMM, CRF)
 - RNN-based, usually LSTM
 - (Usually) Transformer/attention-based large language models, fine-tuned (e.g. BERT)
 - Hybrid

Rule-based

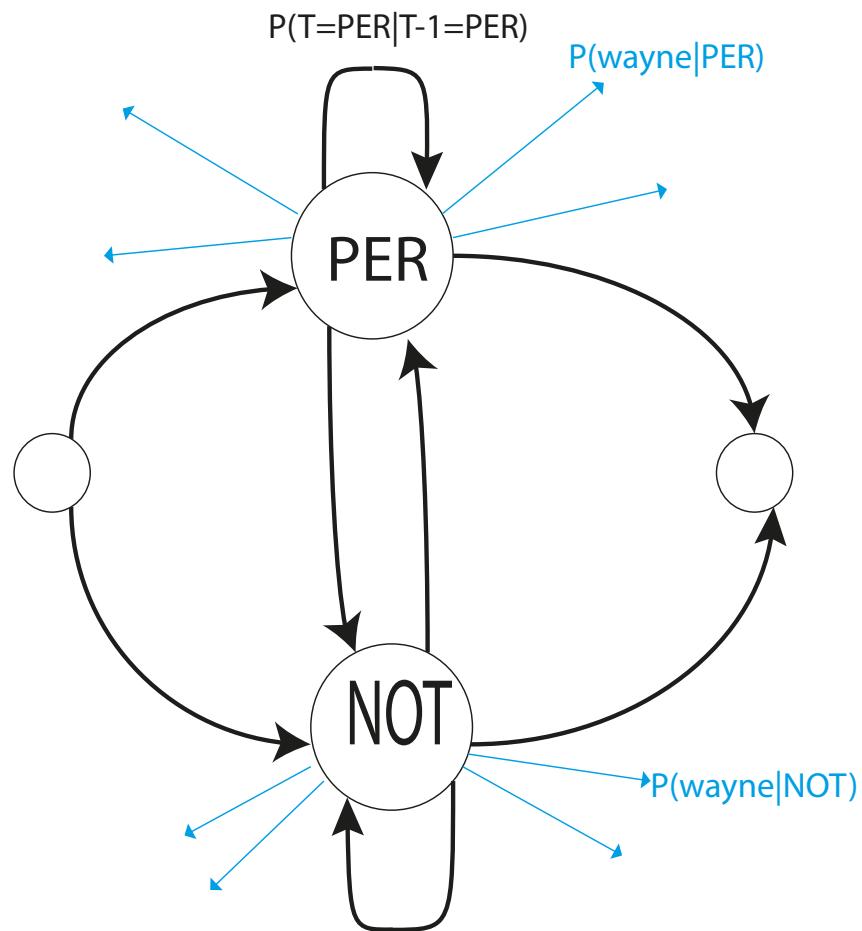
- Heuristics, entity lookup lists (gazeteers)
- Varying degrees of inclusion of supervised ML
- Bootstrapping using initial set of seeds
- Still used in non-academic scenarios

	Pros	Cons
Rule-based	<ul style="list-style-type: none">• Declarative• Easy to comprehend• Easy to maintain• Easy to incorporate domain knowledge• Easy to trace and fix the cause of errors	<ul style="list-style-type: none">• Heuristic• Requires tedious manual labor
ML-based	<ul style="list-style-type: none">• Trainable• Adaptable• Reduces manual effort	<ul style="list-style-type: none">• Requires labeled data• Requires retraining for domain adaptation• Requires ML expertise to use or maintain• Opaque

Generative models

- Model joint probability distributions from training data
- Can be used to generate new data from these distributions
- Use Bayes theorem to obtain a conditional probability to label unseen data
- Examples of generative models are Naïve Bayes and the Hidden Markov Model

Simple HMM



Defined by

- **state transition probabilities**
(independence assumption that current state depend only on previous state) often given as a **state transition matrix**; and
- **emission probabilities**
 $P(\text{observed word} \mid \text{state})$

These probabilities can be derived from labelled corpora using maximum likelihood estimation (MLE).

More on HMMs

- To maximise probability of tag sequence given word sequence, Bayes Rule is applied

$$\Pr(tags \mid words) = \frac{\Pr(words \mid tags) \times \Pr(tags)}{P(words)}$$

- The denominator is a constant so it can be ignored.
- The probability of the tag sequence comes from the transition probabilities
- The probability of the word sequence given the tag sequence comes from the emission probabilities
- Typically use the Viterbi Algorithm to find the tag sequence which optimises this probability

Hidden Markov Model (HMM)

■ Drawbacks

- Arise from the two simplifying assumptions
- Bigram limits model history
- Feature independence limits model richness
- Difficult to add additional features

$$P(\text{words}|\text{tags}) = \prod_{j=1}^n P(\text{word}_j|\text{tag}_j)$$

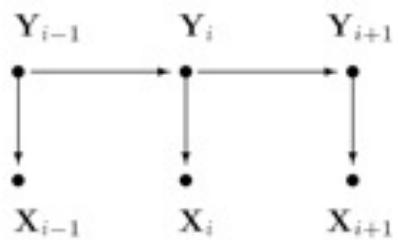
ONLY VALID THROUGH INDEPENDENCE ASSUMPTION!

Discriminative models

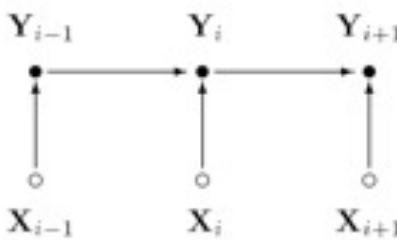
- Simplify the problem by discriminating between classes (NE tag types) rather than modelling every data point
- Leverage multiple (potentially interdependent) features (e.g. word shape, presence in gazeteer, PoS, word embeddings) which can improve prediction and help with sparsity
- Examples of discriminative models are the Maximum Entropy Markov Model (MEMM) and the Conditional Random Field (CRF)

Model for MEMM

HMM



MEMM



If the observations are given by \mathbf{X} and the tag sequence is given by \mathbf{Y} ,

- In a HMM, we model $P(\mathbf{X}|\mathbf{Y})$
- In a MEMM, we model $P(\mathbf{Y}|\mathbf{X})$

$$P(y_1 \dots y_n | x_1 \dots x_n) = \prod_{i=1}^n P(y_i | x_i, y_{i-1})$$

$$P(y_i | x_i, y_{i-1}) = \frac{1}{Z} \exp \left(\sum_{j=1}^k \lambda_j f_j(x_i, y_i, y_{i-1}) \right)$$

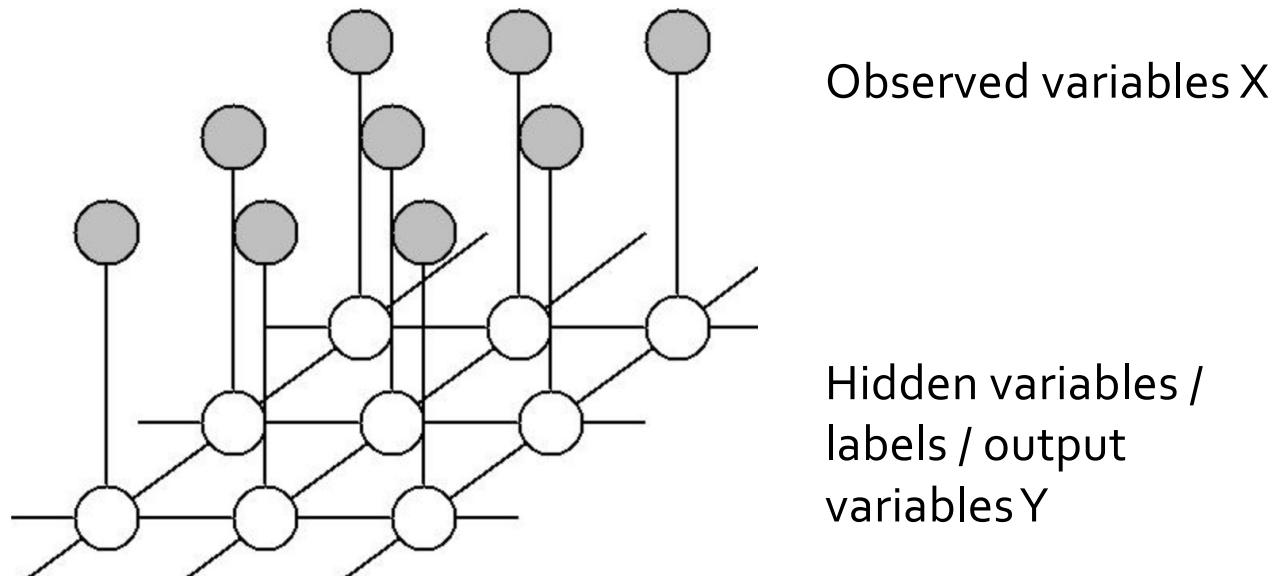
Label Bias Problem

- MEMMs use a per-state exponential model
 - Leads to label bias problem
 - Role of second (or subsequent) tokens in a chunk (multi-word NE) in distinguishing class can be lost
- Solution: Conditional Random Fields (CRFs):
 - CRFs have a single exponential model for the joint probability of the entire label sequence

So what is a CRF?

- A CRF is an undirected graphical model where the vertices can be divided exactly into two disjoint sets, X and Y , which are the observed and output variables respectively. We then model the conditional distribution $p(Y|X)$.

A general
CRF could
look like this



Linear Chain CRFs

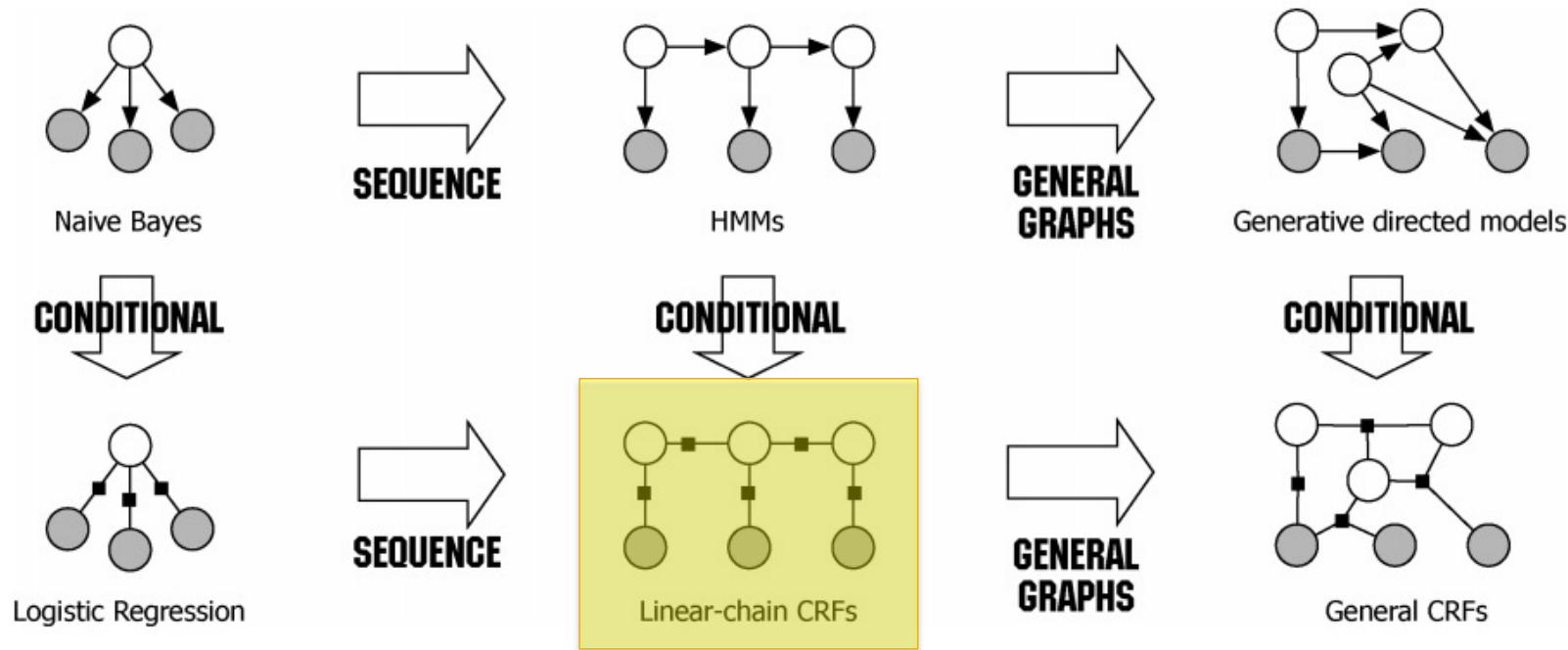


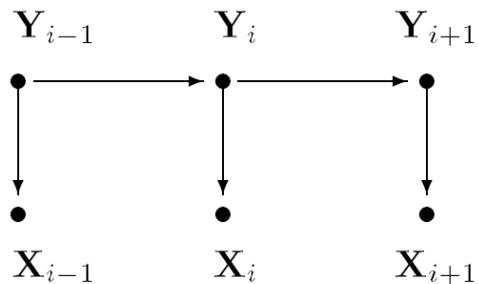
Fig. 2.4 Diagram of the relationship between naive Bayes, logistic regression, HMMs, linear-chain CRFs, generative models, and general CRFs.

[Sutton and McCallum, 2010]

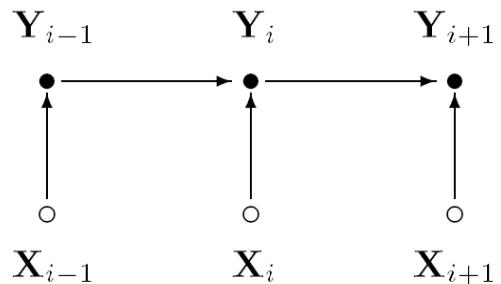
Graphical comparison among HMMs, MEMMs and CRFs

(from Lafferty et al.)

HMM



MEMM



CRF

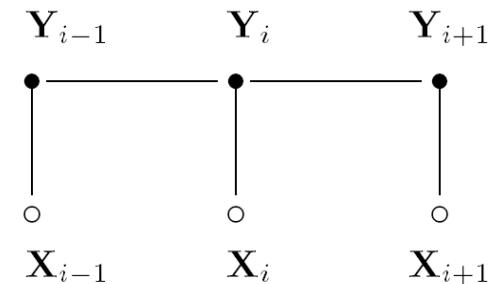


Figure 2. Graphical structures of simple HMMs (left), MEMMs (center), and the chain-structured case of CRFs (right) for sequences. An open circle indicates that the variable is not generated by the model.

Linear Chain CRF

- If the $G=(V,E)$ of Y is a chain, the joint distribution over the label sequence Y given X has the form

$$p(Y|X) \propto \exp \left(\sum_{e \in E, k} \lambda_k f_k \left(e, Y \Big|_e, X \right) + \sum_{v \in V, k} \mu_k g_k \left(v, Y \Big|_v, X \right) \right)$$

where x is a data sequence, y a label sequence and $y|_S$ is the set of components of y associated with the vertices in subgraph S

Parameter Estimation

- Need to determine the parameters $\theta = (\lambda_1 \dots \lambda_n, \mu_1 \dots \mu_n)$ from the training data $D \{(x^i, y^i)\}_{i=1}^N$ with empirical distribution $p(x, y)$
- Want to maximise the log-likelihood objective function

$$O(\theta) \propto \sum_{x,y} p(x, y) \log p_\theta(y | x)$$

- This can be done with a gradient descent algorithm or an iterative scaling algorithm, e.g., improved iterative scaling (IIS) of Della Pietra et al. 1997

Drawbacks of using CRFs

- CRF performs well but is subject to drawbacks
- Very slow to train
- Lafferty et al. 2001 report (on the same dataset):
 - MEMM+ trained to convergence from uniform distribution in 100 iterations
 - CRF did not converge after 2000 iterations from the uniform distribution
 - using the MEMM+ parameters to initialise the CRF, it converged in 1000 iterations
- Features need to be hand-crafted

Neural Models

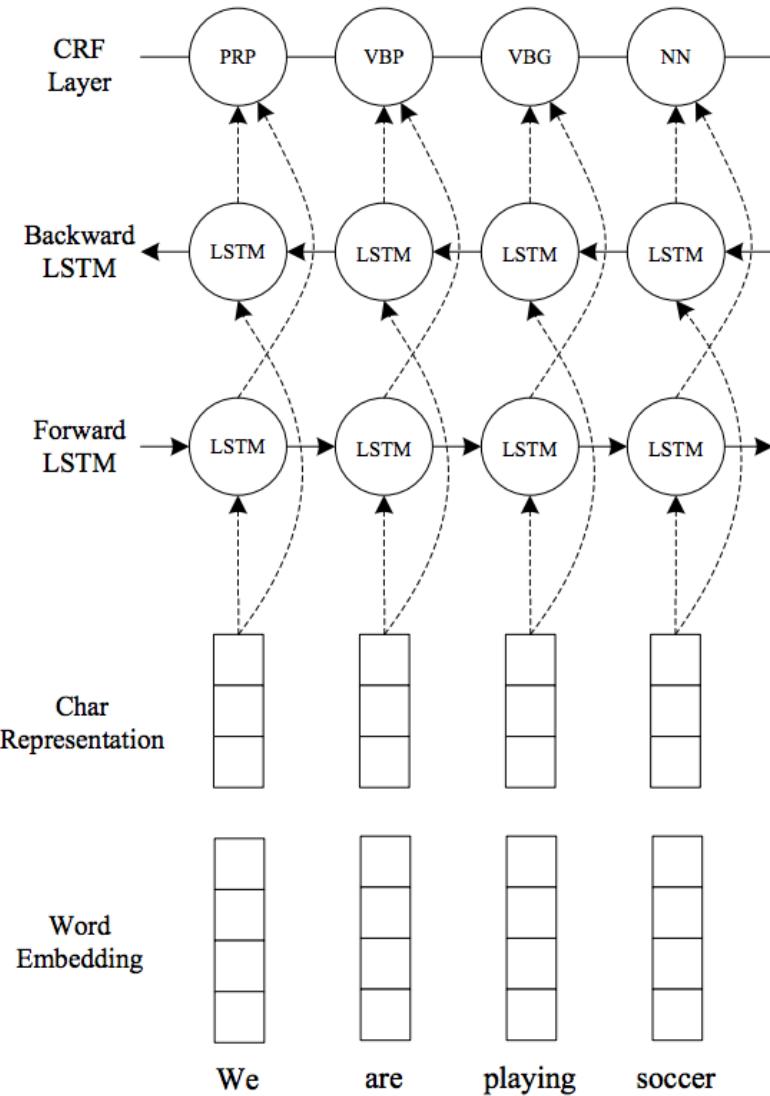
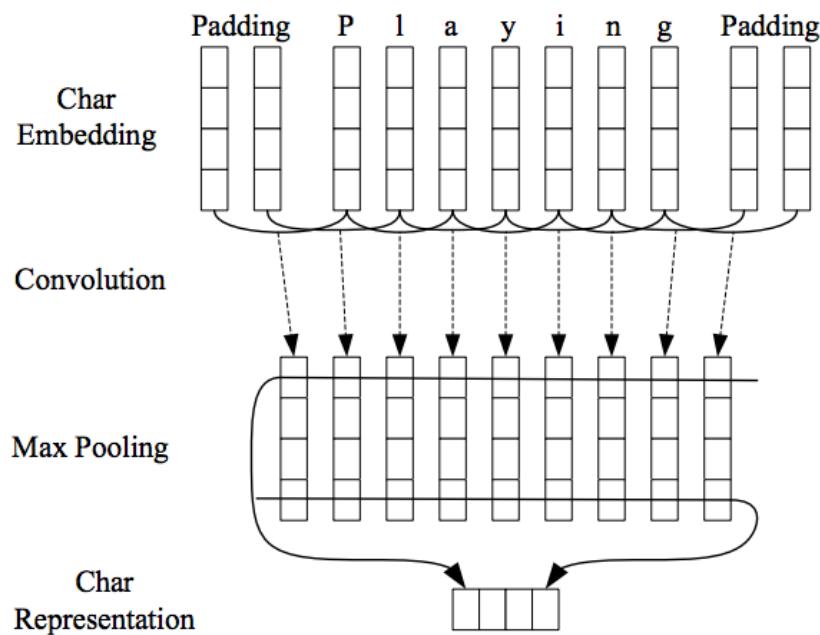
- Bidirectional-LSTM is most popular (non-transformer-based)
- Vanilla RNN and GRU have also been used
- NNs allow word, char or other embeddings to be used (or generated)
- Ma and Hovy (2016) is a good base Bi-LSTM architecture
(CoNLL English SotA in 2016 and basis for Wang et al. SotA
in 2021)

Ma and Hovy (2016)

- Also our seminar paper for this week
- A char representation for each word is generated from a CNN based on char embeddings
- Char rep is concatenated with GloVe word embedding
- Bi-directional LSTM is achieved by concatenating left-to-right and right-to-left LSTM hidden states
- CRF is used as output layer (rather than Softmax) to decode label sequences

Ma and Hovy (2016)

- Architecture diagrams from paper
- CRF shows tags from PoS task, NER task uses same config (except for initial LR)



Advantages of Neural Approach

- End-to-end sequence labelling
 - No feature-engineering
- Achieved F1 91.21 which was SotA in 2016

AdvNLP Week 5

Named Entity Recognition

Part 3

Julie Weeds, Spring 2024



Supervised Learning

- For tasks like NER, supervised learning is often the best performing option
- Data requirements - as much as possible?
- Have some expectations of cost/time/acceptable performance
- Reduce data requirements using transfer learning e.g. pre-trained embeddings or deep LMs with fine-tuning

Sequence Labelling – IOB encoding

Words	Label
Manchester	B-ORG
United	I-ORG
striker	O
Wayne	B-PER
Rooney	I-PER
has	O
agreed	O
a	O
new	O
£300,000	B-MISC
contract	O
.	O

I → inside a chunk
O → outside a chunk
B → beginning a chunk

Existing Datasets

- Unless your task is completely novel, a dataset likely already exists
- Most papers eval on multiple datasets
- NER examples: CoNLL, OntoNotes, ACE, WNUT, various versions and languages
- There are many more
- See
<https://paperswithcode.com/task/named-entity-recognition-ner>

Hand-labelling

- No existing dataset
- Novel task (e.g. NER with HTML or a low-resource language)
- Unlabelled data is abundant
- Labelling it by hand is costly and tedious
- Not best use of a researcher's time!
- Improve efficiency with:
 - UI support
 - Active learning
 - Crowdsourcing

Annotation User Interfaces

- As with most tasks, a good UI = efficiency
- A large unlabelled dataset can be labelled efficiently
- A trained model is required to suggest annotations which are then accepted/rejected/amended by user
- NER labelling applications:
 - prodigy (<https://prodi.gy/>)
Paid, tightly integrated with spaCy for Active Learning
 - doccano (<https://dокументation.github.io/doceanno/>)
Open source, labelling model integration through API
 - Label Studio (<https://labelstud.io/>)
Open source/paid, backend model integration for AL

prodigy

PERSON 1 ORG 2

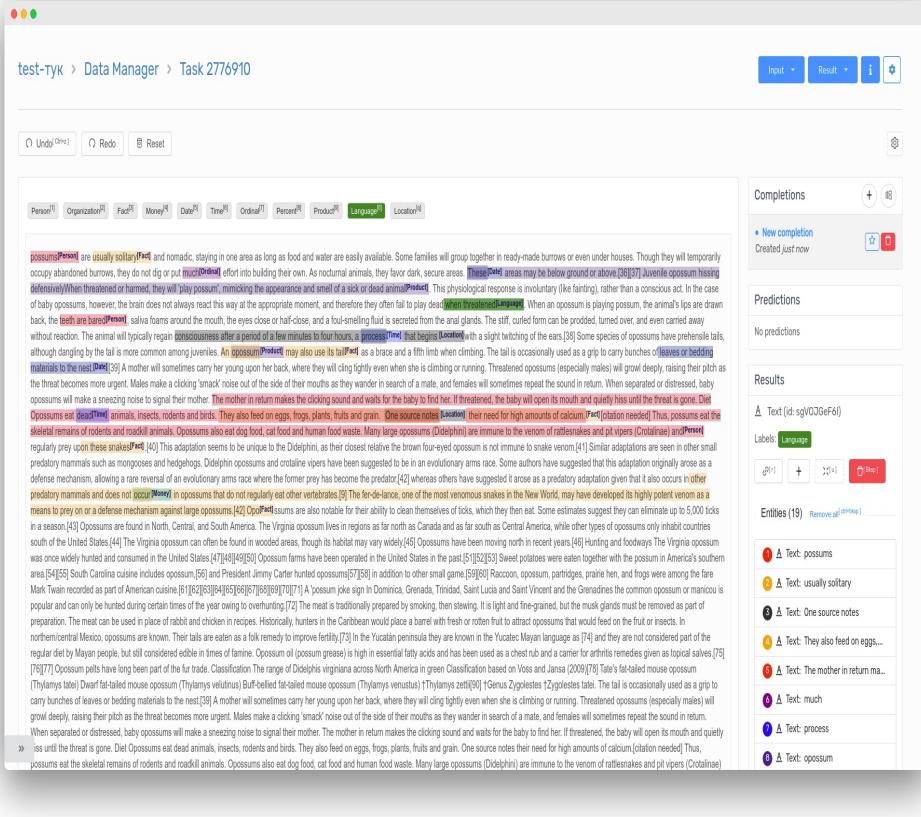
The film begins with the birth of Karishma PERSON and Karan ORG on the same day in the same hospital. 23 years later, unknown of their family backgrounds, they meet each other one summer at Krakow University Poland ORG and fall in love. When Karan ORG finds out her family background, he starts avoiding Karishma PERSON . What happens thereafter is a succession of interesting events that you would get to see in this musical extravaganza from the house of Shakti Samanta PERSON which made some memorable romantic films like Aradhana PERSON , Kati Patang PERSON , Amar Prem PERSON , Kashmir Ki Kali and others.

SOURCE: CMU Movie Summary Corpus



- Accept, reject or ignore all annotations
- Amend or delete (or undo) individual annotations
- Simple clear UI
- Binary training mode

Label Studio



- Similar UI to prodigy
- Integrates with your model (e.g. Torch)
- Larger entity set and longer text harder to label
- Takes time and concentration
- Design labelling task accordingly

Active Learning

- Don't label your whole dataset
- Actively label the best sentences while training
- Process:
 1. Select a model and unlabelled dataset
 2. Hand-label a small sample of data
 3. Train the model
 4. Select a batch of the most informative sentences
 5. Repeat from 2 until convergence (or some other constraint)
- What do we mean by “most informative”?

Semi-Supervised Learning

- Generate training data with minimal human interaction
- Unlabelled data is abundant
- Focus on labels i.e. providing rules or examples for identifying each entity type
- The main two semi-supervised techniques are bootstrapping and distant supervision, which share roughly equal popularity

Bootstrapping

- Using a classifier's own predictions to label more data
- General process:
 1. Train classifier (minimally trained initially) to convergence
 2. Label an unlabelled corpus with this classifier
 3. Add the most confident predictions to training data
 4. Iterate (25 iterations – Kozareva (2006))
- The most confident predictions? The reverse of Active Learning!

Disadvantages of Bootstrapping

- Problem 1: classifier errors are compounded with each iteration (semantic drift)
- Problem 2: the most confident classifications are not the most informative
- Solution: co-training (or even tri-training)
 - Use two (or more) classifiers
 - Use different classifier algorithms, biases, feature sets or differently sampled data
 - Use agreement between classifiers as the criteria for adding instances to training data
 - New instances can be swapped between classifiers for training
 - Disagreement (of one classifier) can be used in tri-training to prevent only easy data being supplied
- Can be expensive computationally (we have lots of unlabelled data to iterate over)

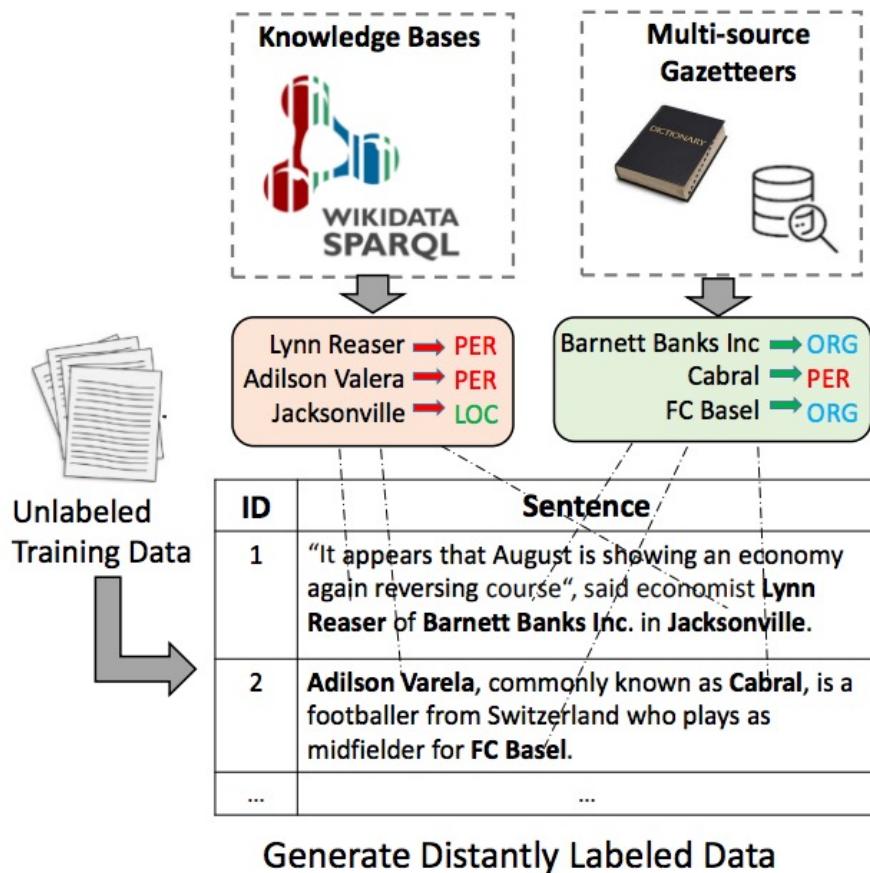
Ruder and Plank (2018)

- Strong baselines for neural semi-supervised learning under domain shift
- Bootstrapping brought up to date
- Char + word embedding (GloVe) + Bi-LSTM
- They evaluate self-training, tri-training and tri-training with disagreement
- PoS (not NER) and sentiment tasks
- Standard tri-training performed well on PoS SANCL 2021

Distant Supervision

- Use a knowledge base as a large set of examples of a given label (e.g. entity) type
- Commonly used KBs: DBpedia, Wikidata, YAGO, Cyc
- KBs typically contain tuples/relations, so Relation Extraction task fits well
- RE process can also be used to perform NER
- Basic process:
 1. Select relation type (e.g. places of birth)
 2. Get tuples from KB that express that relation (e.g. person->birthPlace->location)
 3. Find sentences in an unlabelled corpus (e.g. the Web) that contain both entities
 4. Label those entities as person and location and add sentence to training data OR train a model on the context surrounding those entities
- Matching techniques (between KB and unlabelled corpus):
 - String or RegEx
 - Heuristics (e.g. PoS rules, titles, word shape)

Distant Supervision



- Distant labelling example with direct matching
- From Liang et al. (2020)

Disadvantages of Distant Supervision

- Opposing problems
 - Incomplete annotation
 - KBs have limited coverage
 - Entities in unlabelled corpus unmatched
 - False negatives / limited *recall*
 - Noisy annotation
 - Ambiguity causes mis-labelling
 - E.g. KB contains “Manchester” (a LOC), corpus contains “Manchester Utd.” (an ORG)
 - False positives / limited *precision*

Distant Supervision Solutions

- Incomplete annotation solutions
 - Use more sources!
 - Or knowledge of language (i.e. pre-trained LMs)
- Noisy annotation solutions
 - Induce labels based on occurrence counts
 - Not good for open-domain with high-ambiguity labels
 - Filter out sentences with low matching quality (e.g. indicators of unlabelled entities, e.g. Mr/Ms or Inc. next to unlabelled tokens)
 - Can be effective, but hand-written for each entity type
- Liang et al. (2020) tackle both problems

Distant Supervision

- Other methods that may qualify as “distant”
 - Wikipedia hyperlinks and page object_type
 - KB used directly as a gazeteer input to model
- Applications exist to support Distant Supervision:
 - Stanford Snorkel
 - Clean and integrate weak/distant data sources
 - Create labelling functions
 - Evaluate accuracy, coverage, overlaps, conflicts of functions
 - Combine functions and sources into a labelling model

Unsupervised Learning

- For NER – the techniques used are generally heuristics, patterns and clustering
 - Hearst patterns (1992) is still a good baseline to beat: Lexical patterns used to identify semantically similar nouns e.g. the city of <city>
 - Clustering embeddings of words to find words of similar types (what is close to “London” in the vector space?)

Week 5 summary

1. Named Entity Recognition

- What and why
- Challenges
- NE types
- Sequence labelling
- evaluation
- features

2. Approaches to NER

- Rule-based
- Generative (e.g. HMM)
- Discriminative (e.g. CRF)
- Neural Models (e.g. CNN-BiLSTM-CRF, ACE)
- Transformer LMs

3. Getting Data

- Supervised
- Semi-supervised
- Unsupervised

Before the seminar

- Please read Ma and Hovy (2016)

AdvNLE Seminar 5

Sequence Classification / Propaganda Detection

Dr Julie Weeds, Spring 2023



Warm-up

- What is the difference between sequence labelling and sequence classification? How many examples can you come up with of applications of each.
- Can you think of ways that either could be applied in the following scenarios:
 - Machine translation?
 - Summarization?

Previously

- Distributional models of word meaning
 - how similar are two words based on how they are used in text?
- Language models
 - how likely is a sequence of words in a language?
- Neural language models
- Sequence Labelling (Named Entity Recognition)

This week

- Sequence classification
- Document / sequence representations
- Evaluation
- Distributional representations of meaning
- Composition
- Propaganda detection

Sequence classification

- Aka document classification, text classification
- Make a single sequence-level classification decision per sentence / per document
- Classification could be
 - *sentiment, topic, relevance ... spam, hate speech, machine-generated*
...
- Classification could be
 - *Binary or multi-class*
- Classification could be
 - *Hard or soft*

Evaluation

- Is class distribution balanced?
- Accuracy
- Precision, Recall, F₁

		Actual Class			
		1	2	3	4
Predicted Class	1	TP	FP	FP	FP
	2	FN	TN	TN	TN
	3	FN	TN	TN	TN
	4	FN	TN	TN	TN

		Actual Class	
		1	0
Predicted Class	1	TP	FP
	0	FN	TN

- In multi-class scenario, need to compute precision, recall and F₁ for each class
- Macro-average => unweighted average
- Micro-average => average weighted by the size of each class

Representing sequences for classification

- Classical document-level representation is **bag-of-words**
- What are the benefits and limitations of this?

	the	plot	is	great	not	boring	dull
1. The plot is great	1	1	1	1	0	0	0
2. The plot is not great	1	1	1	1	1	0	0
3. The great plot is not boring	1	1	1	1	1	1	0
4. The boring plot is not great	1	1	1	1	1	1	0
5. The plot is dull	1	1	1	0	0	0	1

Using Word Embeddings

- Find the **sum/centroid/max** of all of the word embeddings for words in the sequence
- Pass this as input to a classifier (e.g., logistic regression / SVM / NN)

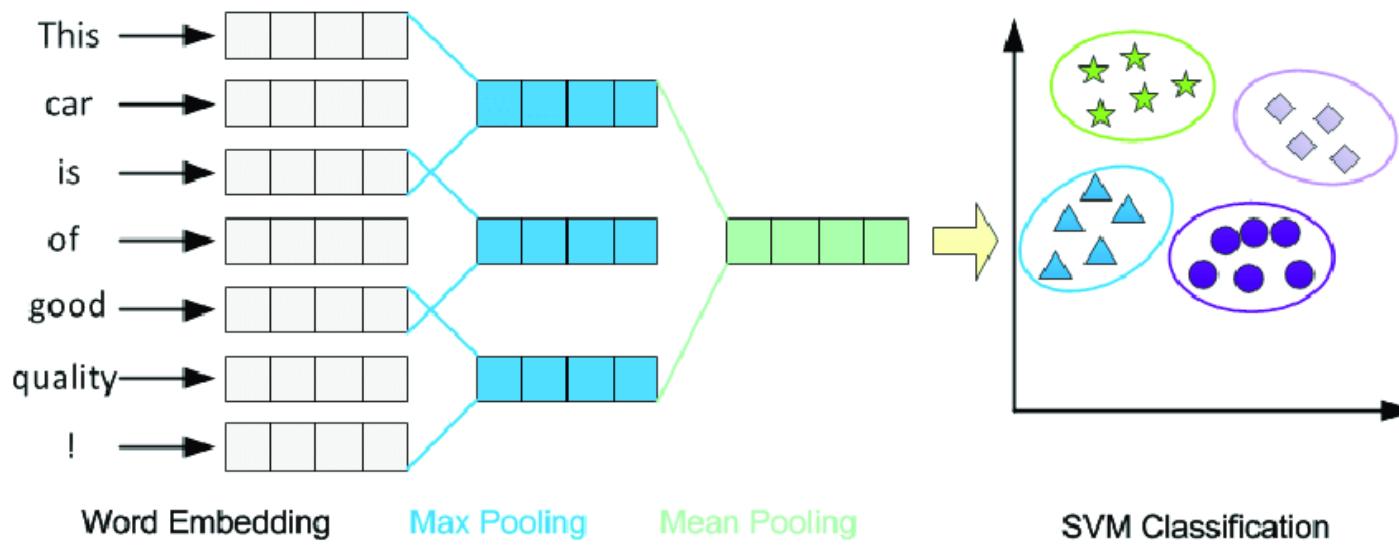


Figure from Pan et al. 2019

Question

- What are the benefits / disadvantages of using word embeddings in this way?

Distributed Representations of Word Meaning

- Distributional Hypothesis
- “*words which mean similar things tend to behave in similar ways*”
- i.e., they co-occur with similar words

	miaows	elected	...	decides	...	comfy	...
leader	0		3		5		0	
president	0		5		5		0	
ruler	0		3		5		0	
cat	5		0		0		1	
chair	0		0		0		5	

Distributed Representations of Sentential Meaning

- Can we do the same as for words?
 - Hypothesize that sentences which mean similar things tend to behave in similar ways?
 - Collect all of the contexts of sentences
 - Represent using vectors and compare
- Why / why not?

The Principle of Compositionality

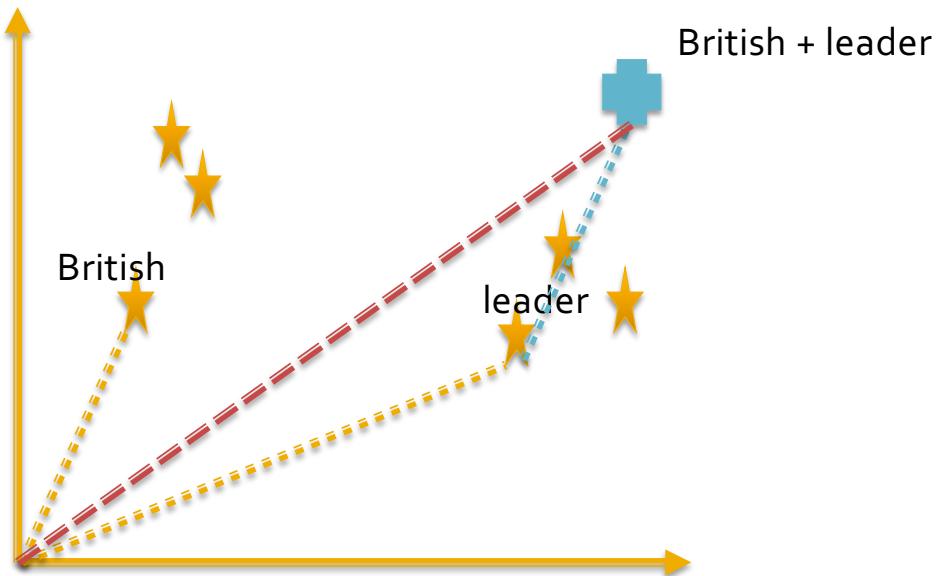
- widely attributed to Gottlieb Frege, but assumed by others
e.g., George Boole
- *"The meaning of a complex expression is determined by the meanings of its constituent expressions and the rules used to combine them"*

Composition for Meaning Representations

- **Constituent** expressions are **words**
- Words are represented by distributional representations / **embeddings** (e.g., Word2Vec or GloVe)
- So to get a representation of a sentence we need to ...
 - ... compose the embeddings of the constituent words
- How? What are the rules for composition?

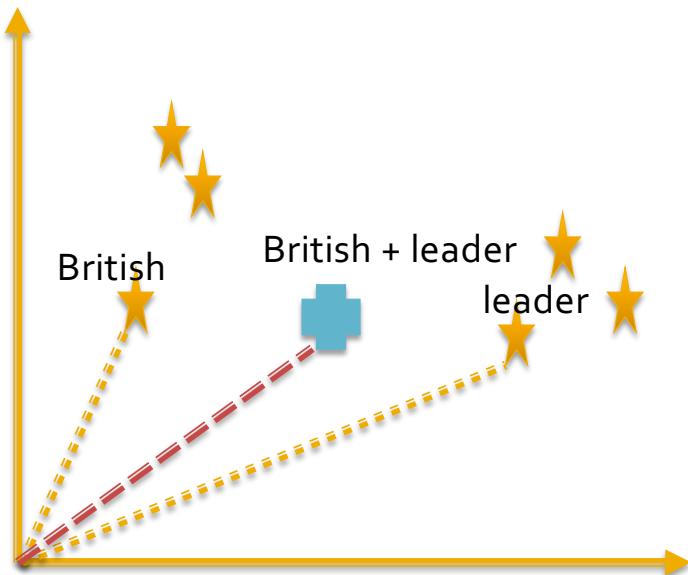
Additive composition

- Simply add the vectors



Additive composition

- Or average the vectors (find the centroid)



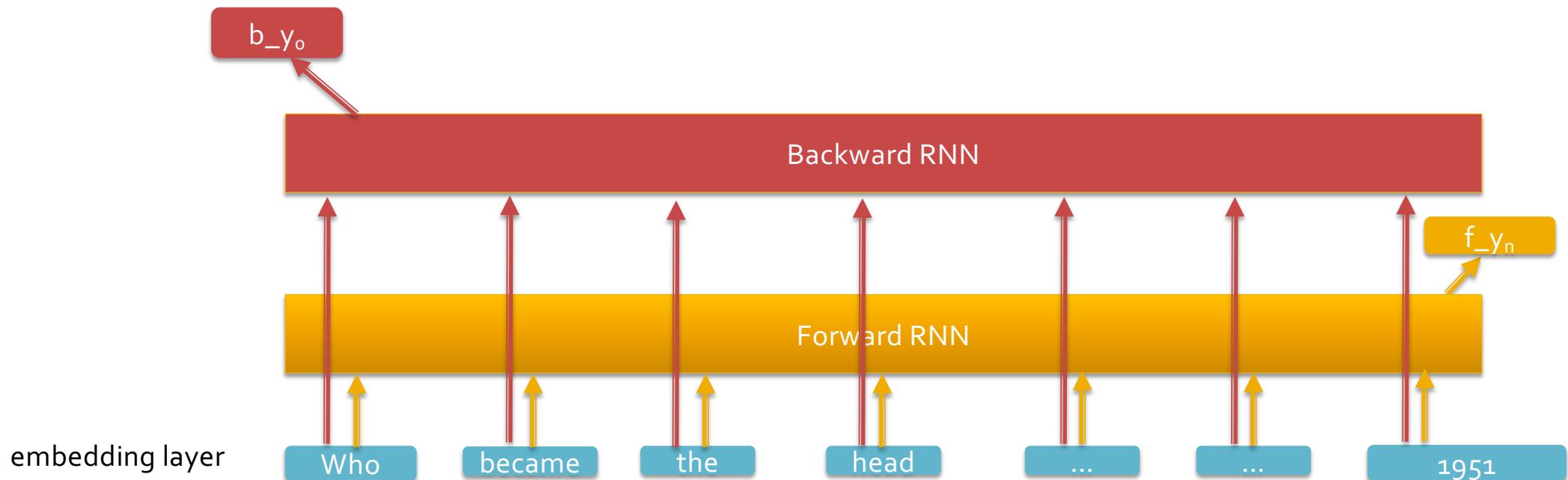
- remember, we are interested in cosine similarity between vectors
- → direction
- so very little difference between adding and averaging
- especially if vectors are normalised to unit length

Disadvantages of Adding Word Embeddings

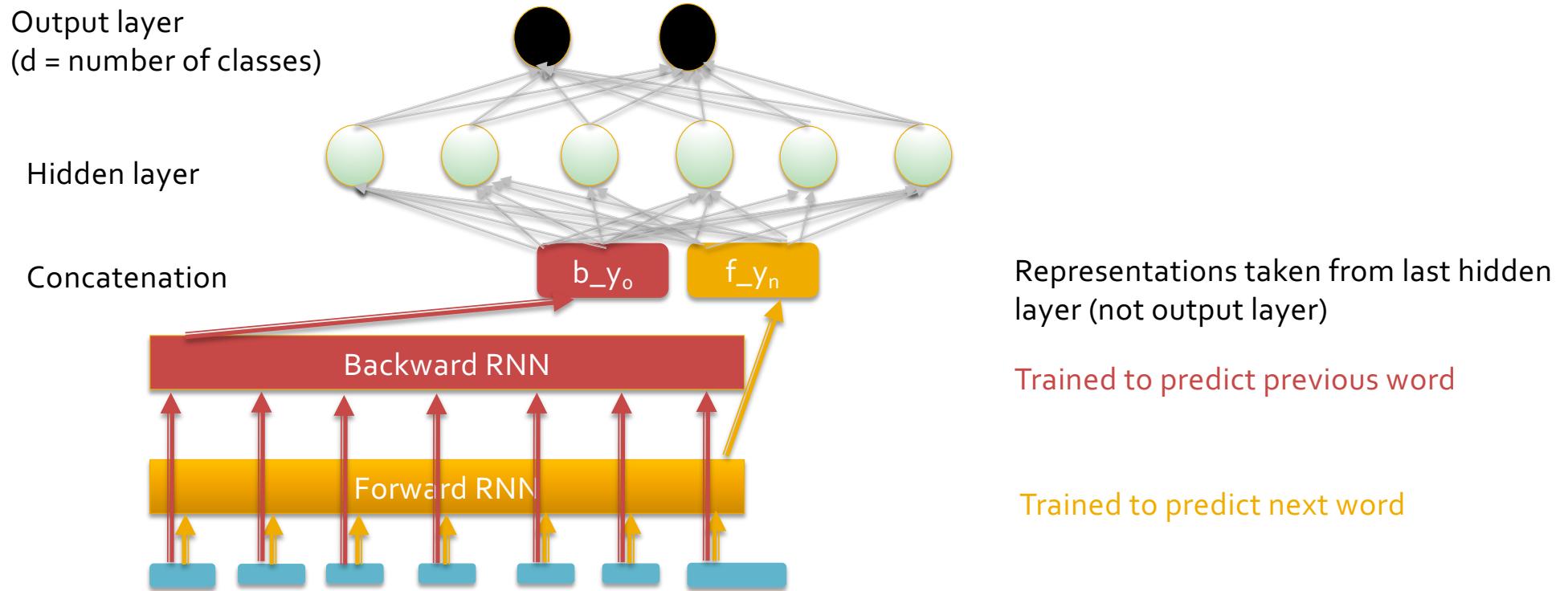
- Word embeddings are **uncontextualised**
 - contain a mixture of usages of all senses of the word
 - e.g., *head* as part of the body and *head* as leader
 - but only one sense is intended in a given sentence
- Pays no attention to word order or syntax
- What about function words such as *in*, *on*, *every* and *not*

Language modelling for Sequence Representations

- Represent a sequence by what is predicted to the left and right of it
- e.g., concatenation of $[b_y_o, f_y_n]$



Classification using representations from LMs



Propaganda Detection (Part 2)

[Da San Martino et al. \(2020\)](#) introduce a competition to detect and classify propaganda techniques in text. When reading this paper, do not be overly concerned with the different systems which took part in the competition. We will focus on the overall idea of propaganda detection, the two tasks introduced in this paper (span identification and technique classification), the dataset and the evaluation metrics. Once you have read the paper, consider the following questions.

1. What do you understand by the term propaganda and why might it be important to develop systems which can automatically detect propaganda in text?
2. Why is automatic propaganda detection difficult?
3. Give examples of 3 different propaganda techniques being used in text. Explain why this is propaganda.
4. What textual features might be useful to help a system detect propaganda?
5. Describe the pipeline proposed by the paper for propaganda identification. Can you think of any alternatives? What advantages / disadvantages are there of each?
6. How was the PTC-SemEval20 corpus collected and annotated? What do you understand by “the γ agreement on the annotated articles is on average 0.6”?
7. How do the authors evaluate systems on the span identification task?
8. Micro-average F_1 is used to evaluate systems on the technique classification task. The authors state that for a single-label task, this is equivalent to accuracy. Explain
9. Outline one method which could be used to carry out span identification.
10. Outline one method which could be used to carry out techniques classification.
11. Systems were evaluated for span identification on both the development set and the test set. Why do you think the results are not the same on both?
12. What is the predominant propaganda technique found in the corpus? If a system labelled every propaganda snippet with this label, how would it do? What do you think of the system results for techniques classification (Table 6)?

Question 1

- What do you understand by the term propaganda and why might it be important to develop systems which automatically detect propaganda in text?

Question 1

- What do you understand by the term propaganda and why might it be important to develop systems which automatically detect propaganda in text?

Propaganda comes in many forms, but it can be recognized by its persuasive function, sizable target audience, the representation of a specific group's agenda, and the use of faulty reasoning and/or emotional appeals (Miller, 1939). The term *propaganda* was coined in the 17th century, and initially referred to the propagation of the Catholic faith in the New World (Jowett and O'Donnell, 2012a, p. 2). It soon took a pejorative connotation, as its meaning was extended to also mean opposition to Protestantism. In more recent times, the Institute for Propaganda Analysis (Ins, 1938) proposed the following definition:

Propaganda. *Expression of opinion or action by individuals or groups deliberately designed to influence opinions or actions of other individuals or groups with reference to predetermined ends.*

Recently, Bolsover and Howard (2017) dug deeper into this definition identifying its two key elements: (i) trying to influence opinion, and (ii) doing so on purpose.

Question 2

- Why is automatic propaganda detection difficult?

Question 2

- Why is automatic propaganda detection difficult?
- At least 3 reasons:
 1.
 2.
 3.

Question 3

- Give examples of 3 different propaganda techniques being used in text. Explain why this is propaganda

# Technique	Snippet
1 Loaded language	Outrage as Donald Trump suggests injecting disinfectant to kill virus.
2 Name calling, labeling	WHO: Coronavirus emergency is ' Public Enemy Number 1 '
3 Repetition	I still have a dream . It is a dream deeply rooted in the American dream . I have a dream that one day ...
4 Exaggeration, minimization	Coronavirus ' risk to the American people remains very low ', Trump said.
5 Doubt	Can the same be said for the Obama Administration?
6 Appeal to fear/prejudice	A dark, impenetrable and “irreversible” winter of persecution of the faithful by their own shepherds will fall.
7 Flag-waving	Mueller attempts to stop the will of We the People!!! It's time to jail Mueller.
8 Causal oversimplification	If France had not have declared war on Germany then World War II would have never happened.
9 Slogans	“BUILD THE WALL!” Trump tweeted.
10 Appeal to authority	Monsignor Jean-Franois Lantheaume, who served as first Counsellor of the Nunciature in Washington, confirmed that “Vigan said the truth. That’s all.”
11 Black-and-white fallacy	Francis said these words: “Everyone is guilty for the good he could have done and did not do ... If we do not oppose evil, we tacitly feed it.”
12 Thought-terminating cliché	I do not really see any problems there. Marx is the President.
13 Whataboutism	President Trump — who himself avoided national military service in the 1960's— keeps beating the war drums over North Korea.
Straw man	“Take it seriously, but with a large grain of salt.” Which is just Allen's more nuanced way of saying: “Don't believe it.”
Red herring	“You may claim that the death penalty is an ineffective deterrent against crime – but what about the victims of crime? How do you think surviving family members feel when they see the man who murdered their son kept in prison at their expense? Is it right that they should pay for their son's murderer to be fed and housed?”
14 Bandwagon	He tweeted, “EU no longer considers #Hamas a terrorist group. Time for US to do same.”
Reductio ad hitlerum	“Vichy journalism,” a term which now fits so much of the mainstream media. It collaborates in the same way that the Vichy government in France collaborated with the Nazis.

Table 1: The 14 propaganda techniques with examples, where the propaganda span is shown in bold.

Question 4

- What textual features might be useful to help a system detect propaganda?

# Technique	Snippet
1 Loaded language	Outrage as Donald Trump suggests injecting disinfectant to kill virus.
2 Name calling, labeling	WHO: Coronavirus emergency is ' Public Enemy Number 1 '
3 Repetition	I still have a dream . It is a dream deeply rooted in the American dream . I have a dream that one day ...
4 Exaggeration, minimization	Coronavirus ' risk to the American people remains very low ', Trump said.
5 Doubt	Can the same be said for the Obama Administration?
6 Appeal to fear/prejudice	A dark, impenetrable and “irreversible” winter of persecution of the faithful by their own shepherds will fall.
7 Flag-waving	Mueller attempts to stop the will of We the People!!! It's time to jail Mueller.
8 Causal oversimplification	If France had not have declared war on Germany then World War II would have never happened.
9 Slogans	“BUILD THE WALL!” Trump tweeted.
10 Appeal to authority	Monsignor Jean-Franois Lantheaume, who served as first Counsellor of the Nunciature in Washington, confirmed that “Vigan said the truth. That’s all.”
11 Black-and-white fallacy	Francis said these words: “Everyone is guilty for the good he could have done and did not do ... If we do not oppose evil, we tacitly feed it.”
12 Thought-terminating cliché	I do not really see any problems there. Marx is the President.
13 Whataboutism	President Trump — who himself avoided national military service in the 1960's— keeps beating the war drums over North Korea.
Straw man	“Take it seriously, but with a large grain of salt.” Which is just Allen's more nuanced way of saying: “Don't believe it.”
Red herring	“You may claim that the death penalty is an ineffective deterrent against crime – but what about the victims of crime? How do you think surviving family members feel when they see the man who murdered their son kept in prison at their expense? Is it right that they should pay for their son's murderer to be fed and housed?”
14 Bandwagon	He tweeted, “EU no longer considers #Hamas a terrorist group. Time for US to do same.”
Reductio ad hitlerum	“Vichy journalism,” a term which now fits so much of the mainstream media. It collaborates in the same way that the Vichy government in France collaborated with the Nazis.

Table 1: The 14 propaganda techniques with examples, where the propaganda span is shown in bold.

Question 5

- Describe the pipeline proposed by the paper for propaganda identification. Can you think of any alternatives? What advantages / disadvantages are there of each?

Question 5

- Describe the pipeline proposed by the paper for propaganda identification. Can you think of any alternatives? What advantages / disadvantages are there of each?

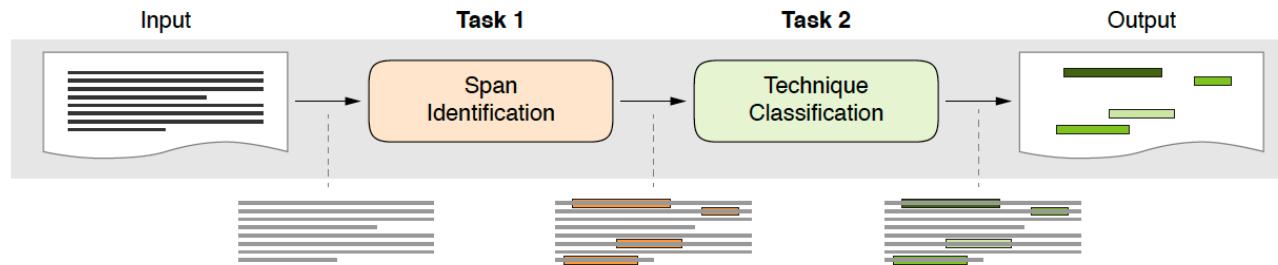


Figure 1: The full propaganda identification pipeline, including the two subtasks: Span Identification and Technique Classification.

Question 6

- How was the PTC-SemEval20 corpus collected and annotated? What do you understand by “the gamma agreement on the annotated articles is on average 0.6”?

Question 6

- How was the PTC-SemEval20 corpus collected and **annotated**?

Input article		Annotation file			
Article ID	Technique	Start	End		
	Name_Calling	34	40		
123456	Loaded_Language	83	89		
123456	Loaded_Language	94	99		
123456	Loaded_Language	350	368		
...	...				

Figure 2: Example of a plain-text article (left) and its annotation (right). The *Start* and the *End* columns are the indices representing the character span of the spotted technique.

Question 6

- What do you understand by “the gamma agreement on the annotated articles is on average 0.6”?

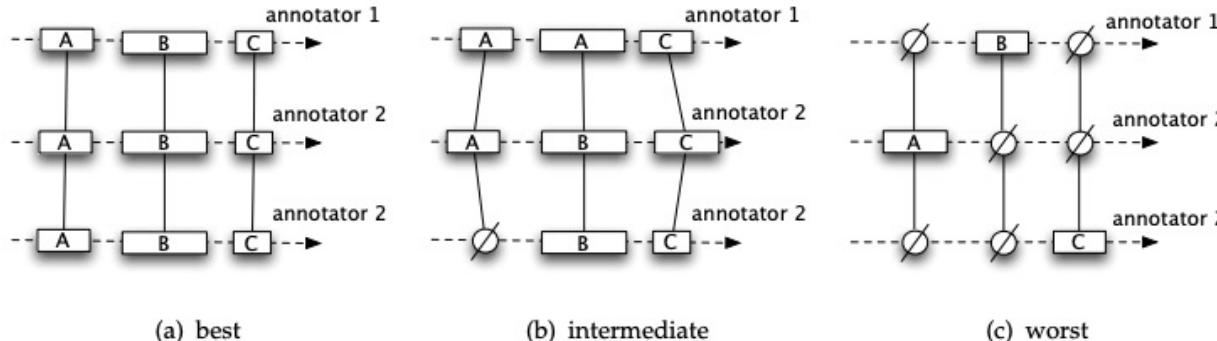


Figure 11

Examples of best, intermediate, and worst possible disorders.

$$\forall s \in c, \gamma = 1 - \frac{\delta(s)}{\delta_e(c)} \quad (8)$$

If all annotators perfectly agree (Figure 11a), $\gamma = 1$. Figure 11c corresponds to the worst case, where the annotators are worse than annotating at random, with $\gamma < 0$. Figure 11b shows an intermediate situation.

Question 7

- How do the authors evaluate systems on the span identification task?

Question 7

- How do the authors evaluate systems on the span identification task?

Let d be a news article in a set D . A gold span t is a sequence of contiguous indices of the characters composing a text fragment $t \subseteq d$. For example, in Figure 4 (top-left) the gold fragment “*stupid and petty*” is represented by the set of indices $t_1 = [4, 19]$. We denote with $T_d = \{t_1, \dots, t_n\}$ the set of all gold spans for an article d and with $T = \{T_d\}_d$ the set of all gold annotated spans in D . Similarly, we define $S_d = \{s_1, \dots, s_m\}$ and S to be the set of predicted spans for an article d and a dataset D , respectively. We compute precision P and recall R by adapting the formulas in (Potthast et al., 2010):

$$P(S, T) = \frac{1}{|S|} \cdot \sum_{d \in D} \sum_{s \in S_d, t \in T_d} \frac{|(s \cap t)|}{|t|}, \quad (1)$$

$$R(S, T) = \frac{1}{|T|} \cdot \sum_{d \in D} \sum_{s \in S_d, t \in T_d} \frac{|(s \cap t)|}{|s|}. \quad (2)$$

Question 8

- Micro-average F₁ is used to evaluate systems on the techniques classification task. The authors state that for a single-label task, this is equivalent to accuracy. Explain.

Evaluation

- Is class distribution balanced?
- Accuracy
- Precision, Recall, F₁

		Actual Class			
		1	2	3	4
Predicted Class	1	TP	FP	FP	FP
	2	FN	TN	TN	TN
	3	FN	TN	TN	TN
	4	FN	TN	TN	TN

		Actual Class	
		1	0
Predicted Class	1	TP	FP
	0	FN	TN

- In multi-class scenario, need to compute precision, recall and F₁ for each class
- Macro-average => unweighted average
- Micro-average => average weighted by the size of each class

Question 9

- Outline one method which could be used to carry out span identification.

Question 10

- Outline one method which could be used to carry out technique classification.

Question 11

- Systems were evaluated for span identification on both the development set and the test set. Why do you think the results are not the same on both?

Question 11

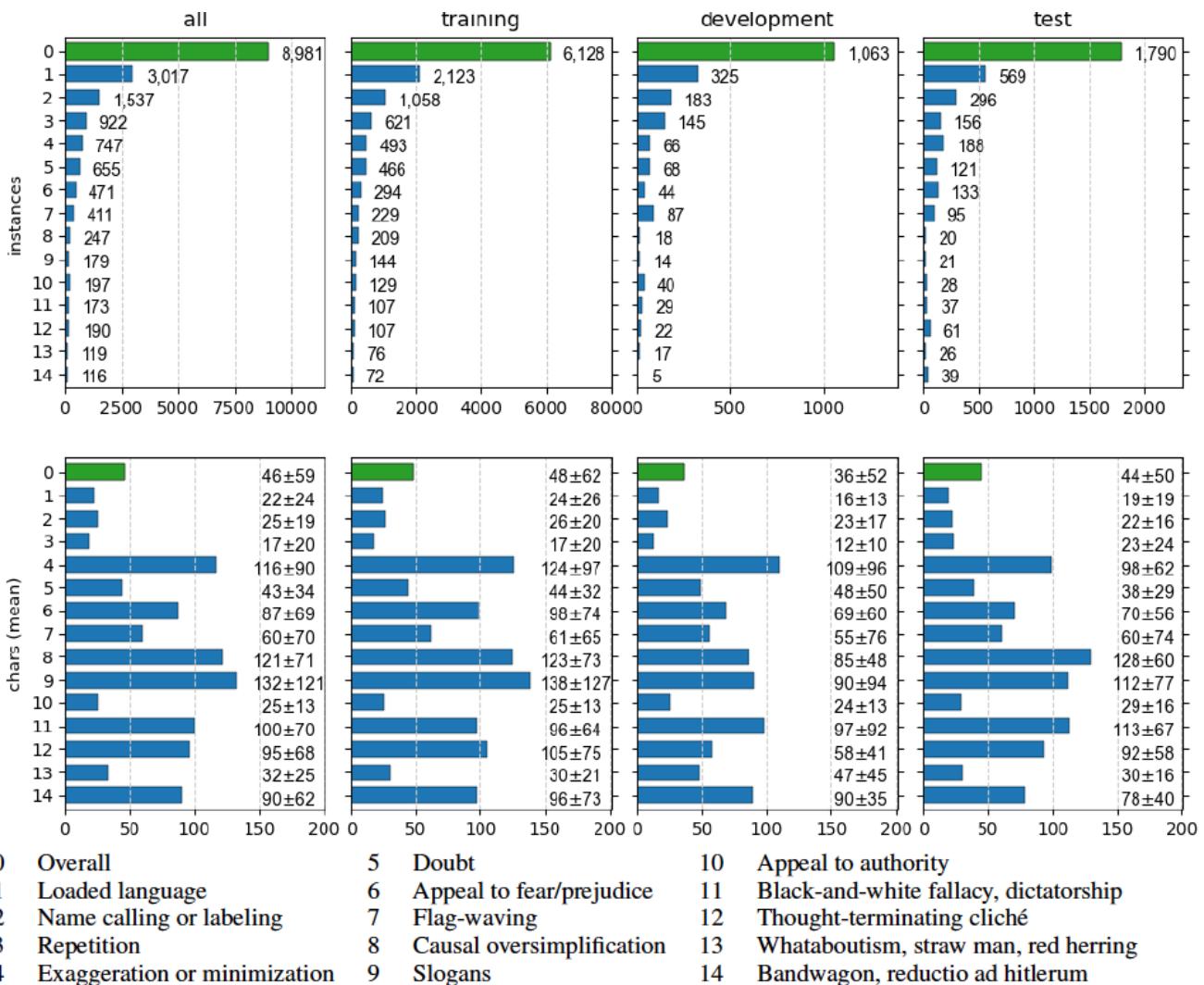
Team	Test			Development				
	Rnk	F ₁	P	R	Rnk	F ₁	P	R
Hitachi	1	51.55	56.54	47.37	4	50.12	42.26	61.56
ApplicaAI	2	49.15	59.95	41.65	3	52.19	47.15	58.44
aschern	3	49.10	53.23	45.56	5	49.99	44.53	56.98
LTIatCMU	4	47.66	50.97	44.76	7	49.06	43.38	56.47
UPB	5	46.06	58.61	37.94	8	46.79	42.44	52.13
Fragarach	6	45.96	54.26	39.86	12	44.27	41.68	47.21
NoPropaganda	7	44.68	55.62	37.34	9	46.13	40.65	53.31
CyberWallE	8	43.86	42.16	45.70	17	42.39	33.45	57.86
Transformers	9	43.60	49.86	38.74	14	43.06	40.85	45.52
SWEAT	10	43.22	52.77	36.59	16	42.51	42.97	42.06
YNUTaoxin	11	43.21	55.62	35.33	11	44.35	40.74	48.67
DREAM	12	43.10	54.54	35.63	19	42.15	42.66	41.65
newsSweeper	13	42.21	46.52	38.63	10	44.45	38.76	52.10
PsuedoProp	14	41.20	41.54	40.87	22	39.32	34.27	46.11
Solomon	15	40.68	53.95	32.66	15	42.86	43.24	42.49
YNUHPCC	16	40.63	36.55	45.74	18	42.27	32.08	61.95
NLFIIIT	17	40.58	50.91	33.73	21	39.67	35.04	45.72
PALI	18	40.57	53.20	32.79	2	52.35	49.64	55.37
UESTCICSA	19	39.85	56.09	30.90	13	44.17	43.21	45.18
TTUI	20	39.84	66.88	28.37	6	49.59	48.76	50.44
BPGC	21	38.74	49.39	31.88	25	36.79	34.72	39.12
DoNotDistribute	22	37.86	42.36	34.23	24	37.73	32.41	45.12
UTMNandOCAS	23	37.49	37.97	37.03	31	34.35	23.65	62.69
Entropy	24	37.23	41.68	33.63	32	32.89	30.82	35.25
syrapropa	25	36.20	49.53	28.52	1	53.40	39.88	80.80

- Systems were evaluated for span identification on both the development set and the test set. Why do you think the results are not the same on both?

Question 12

- What is the predominant propaganda technique found in the corpus? If a system labelled every propaganda snippet with this label, how would it do? What do you think of the system results for technique classification (Table 6)?

- What is the predominant propaganda technique found in the corpus?
- If a system labelled every propaganda snippet with this label, how would it do?



What do you think of the system results for technique classification (Table 6)?

Rnk	Team	Overall	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	ApplicaAI	62.07	77.12	74.38	54.55	33.59	56.23	45.49	69.43	22.73	51.28	48.15	49.02	39.22	25.00	8.33
2	aschern	62.01	77.02	75.65	53.38	32.65	59.44	41.78	66.35	25.97	54.24	35.29	53.57	42.55	18.87	14.93
3	Hitachi	61.73	75.64	74.20	37.88	34.58	63.43	38.94	68.02	36.62	45.61	40.00	47.92	29.41	26.92	4.88
4	Solomon	58.94	74.66	70.75	42.53	28.44	61.82	39.39	61.84	19.61	50.75	26.67	42.00	38.10	0.00	4.88
5	newsSweeper	58.44	75.32	74.23	20.69	37.10	56.55	42.80	60.53	19.72	50.75	41.67	25.00	21.62	8.00	13.04
6	NoPropaganda	58.27	77.17	73.90	42.71	37.99	56.27	38.02	59.30	12.12	42.42	23.26	8.70	23.26	0.00	0.00
7	Inno	57.99	73.31	74.30	24.89	35.39	58.65	45.09	59.41	24.32	43.75	43.14	40.40	29.63	19.36	10.71
8	CyberWallE	57.37	74.68	70.92	47.68	28.34	58.65	39.84	54.38	15.39	39.39	14.63	23.68	23.81	0.00	12.25
9	PALI	57.32	74.29	69.09	24.56	28.57	58.97	36.59	61.62	30.59	39.22	27.59	39.62	40.82	20.90	28.57
10	DUTH	57.21	73.71	71.41	20.10	28.24	59.16	33.33	58.95	26.23	34.78	44.44	33.33	27.03	17.78	9.30
11	DiSaster	56.65	74.49	68.10	20.44	30.64	59.12	35.25	58.25	14.63	42.55	51.16	26.67	19.05	4.35	20.41
12	djichen	56.54	73.21	68.38	29.75	31.42	60.00	33.65	56.19	22.79	30.77	37.50	43.81	27.91	18.87	20.83
13	SocCogCom	55.81	72.18	67.34	18.88	34.86	60.40	31.62	54.26	6.35	40.91	28.57	26.51	23.53	10.00	9.76
14	TTUI	55.64	73.22	68.49	21.18	32.20	57.40	41.48	61.68	23.08	37.50	28.24	35.29	25.00	20.29	24.56
15	JUST	55.31	71.96	64.73	21.94	29.57	58.26	37.10	62.56	27.27	33.33	48.89	28.89	31.82	28.57	24.49
16	NLFIIIT	55.25	72.55	69.30	21.55	30.30	55.66	24.89	63.32	0.00	41.67	29.63	32.10	13.64	0.00	9.30
17	UMSIForeseer	55.14	73.02	70.79	21.49	28.57	57.21	31.97	56.14	0.00	39.22	29.41	0.00	14.29	0.00	9.76
18	BPGC	54.81	71.58	67.51	23.74	33.47	53.78	33.65	58.93	24.18	40.00	30.77	40.00	20.69	20.90	12.50
19	UPB	54.30	70.09	68.86	20.00	30.62	52.55	30.00	55.87	16.95	34.62	20.00	19.72	22.86	4.88	0.00
20	syrapropa	54.25	71.47	68.44	30.77	28.10	56.14	29.77	57.02	21.51	29.03	31.58	30.61	28.57	9.09	19.61
21	WMD	52.01	69.33	64.67	13.89	25.46	53.94	29.20	52.08	5.71	6.90	7.14	0.00	7.41	0.00	5.00
22	YNUHPCC	50.50	68.08	62.33	17.72	21.54	51.04	26.40	55.56	3.45	27.59	29.79	38.38	17.78	15.00	13.79
23	UESTCICSA	49.94	68.23	66.88	27.96	25.44	44.99	22.75	53.14	3.74	41.38	12.77	11.27	28.57	3.70	0.00
24	DoNotDistribute	49.72	68.44	60.65	19.44	27.23	46.25	29.75	53.76	14.89	28.07	22.64	24.49	12.25	9.68	4.55
25	NTUAAILS	46.37	65.79	54.55	18.43	29.66	48.75	28.31	46.47	0.00	13.79	36.36	0.00	11.43	4.08	9.76
26	UAIC1860	41.17	62.33	42.97	11.16	21.01	36.41	22.12	38.78	7.60	11.43	17.39	2.90	5.56	4.26	9.76
27	UNTLing	39.11	62.57	36.74	7.78	11.82	32.65	5.29	40.48	2.86	17.65	4.35	0.00	0.00	0.00	0.00
28	HunAlize	37.10	58.59	15.82	2.09	23.81	31.76	11.83	29.95	7.84	4.55	6.45	8.00	0.00	0.00	0.00
29	Transformers	26.54	47.55	24.06	2.86	0.00	0.98	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30	Baseline	25.20	46.48	0.00	19.26	14.42	29.14	3.68	6.20	11.56	0.00	0.00	0.00	0.00	0.00	0.00
31	Entropy	20.39	37.74	15.49	5.83	6.39	12.81	6.32	4.95	7.41	0.00	3.92	2.27	0.00	6.78	0.00
32	IUSE8	19.72	38.07	14.70	4.92	8.23	15.47	7.07	8.57	2.27	0.00	0.00	0.00	0.00	0.00	0.00

Table 6: **Technique classification F1 performance on the test set.** The systems are ordered on the basis of the final ranking. Columns 1 to 14 show the performance for each of the propaganda techniques (cf. Section 2). The best score for each technique appears highlighted. (Note: We found a bug in the evaluation script after the end of the competition. The correct ranking, shown in Appendix B, does not differ substantially from above.)

AdvNLP Week 7

Machine Translation

Dr Julie Weeds, Spring 2024



Previously

- Distributional semantics
- Language models
- Neural language models
- Sequence labelling
- Sequence classification

Overview

- What makes machine translation (MT) hard?
- Evaluation of MT
- Classical MT (Pre 1990s)
- Statistical MT (1990-2015)
 - Word-based models
 - Phrase-based models
- Neural MT (2015 -)
 - Encoder-decoder models

... was hard!

French

Pour une journée au ski réussie à Samoëns, n'hésitez pas, consultez les prévisions météo sur la station, les conditions d'enneigement et l'ouverture des pistes. Profitez d'une bonne journée au ski à Samoens, si vous venez en week end ou pour une semaine, bénéficiez de prévision météo à 2 jours sur notre site et à plus long terme sur le site de Météo Chamonix !

English (Google 2014)

For a successful day at Ski Samoens , please, check the forecast weather station on the snow conditions and opening tracks. Enjoy a good day skiing Samoens , if you come on a weekend or for a week, get forecasting weather two days on our website and in the longer term on the site Weather Chamonix !

... was hard!

French

Pour une journée au ski réussie à Samoëns, n'hésitez pas, consultez les prévisions météo sur la station, les conditions d'enneigement et l'ouverture des pistes. Profitez d'une bonne journée au ski à Samoens, si vous venez en week end ou pour une semaine, bénéficiez de prévision météo à 2 jours sur notre site et à plus long terme sur le site de Météo Chamonix !

English (website/human)

To ensure a great ski day in Samoëns, don't hesitate to check the weather report of this ski station, the snow report and the slopes opening. Enjoy a great ski day in Samoëns, if you come for the week-end or for a week, check the Samoëns weather report for the next 2 days on this website and for a longer period of time on the Météo Chamonix website.

... is solved?

French

Pour une journée au ski réussie à Samoëns, n'hésitez pas, consultez les prévisions météo sur la station, les conditions d'enneigement et l'ouverture des pistes. Profitez d'une bonne journée au ski à Samoens, si vous venez en week end ou pour une semaine, bénéficiez de prévision météo à 2 jours sur notre site et à plus long terme sur le site de Météo Chamonix !

English (Google 2019)

For a successful ski day in Samoens, do not hesitate, check out the weather forecast for the resort, the snow conditions and the opening of the slopes. Enjoy a good day skiing in Samoens, if you come for a weekend or for a week, enjoy 2-day weather forecast on our site and longer term on the site of Weather Chamonix!

Why is/was MT hard?

- Lexical differences
- Structural differences (morphological differences and syntactic differences)
- Study of systematic cross-linguistic similarities and differences is called **linguistic typology**
 - See World Atlas of Language Structures (Dryer and Haspelmath, 2013)

Lexical Divergences

- Homonymy and polysemy (in both languages)
- e.g., “I know that machine translation is hard.” → French *savoir*
- whereas “I know David Weir.” → French *connaître*
- different distinctions in different languages e.g., Chinese distinguishes *older brother* and *younger brother*

哥哥
Gēgē

弟弟
Dìdì

I met my brother → 我遇到了我的哥哥
Wǒ yù dàole wǒ dí gēgē

Lexical Divergences

- can be grammatical e.g., the English verb *to like* corresponds to the German adverb *gern*
- gender marking
 - *they* -> *elles* or *ils* in French?
 - In Spanish, different forms for male and female animals)
- lexical gaps
 - *la baguette*
 - *le weekend*

Morphological differences

unhappiness

=un + happy + ness

- *agglutinative vs fusion*

agglutinative => different morphemes within a word are clearly differentiable, e.g., Finnish

fusion => morphemes not clearly distinguishable, may contain multiple bits of information e.g., Indo-European languages

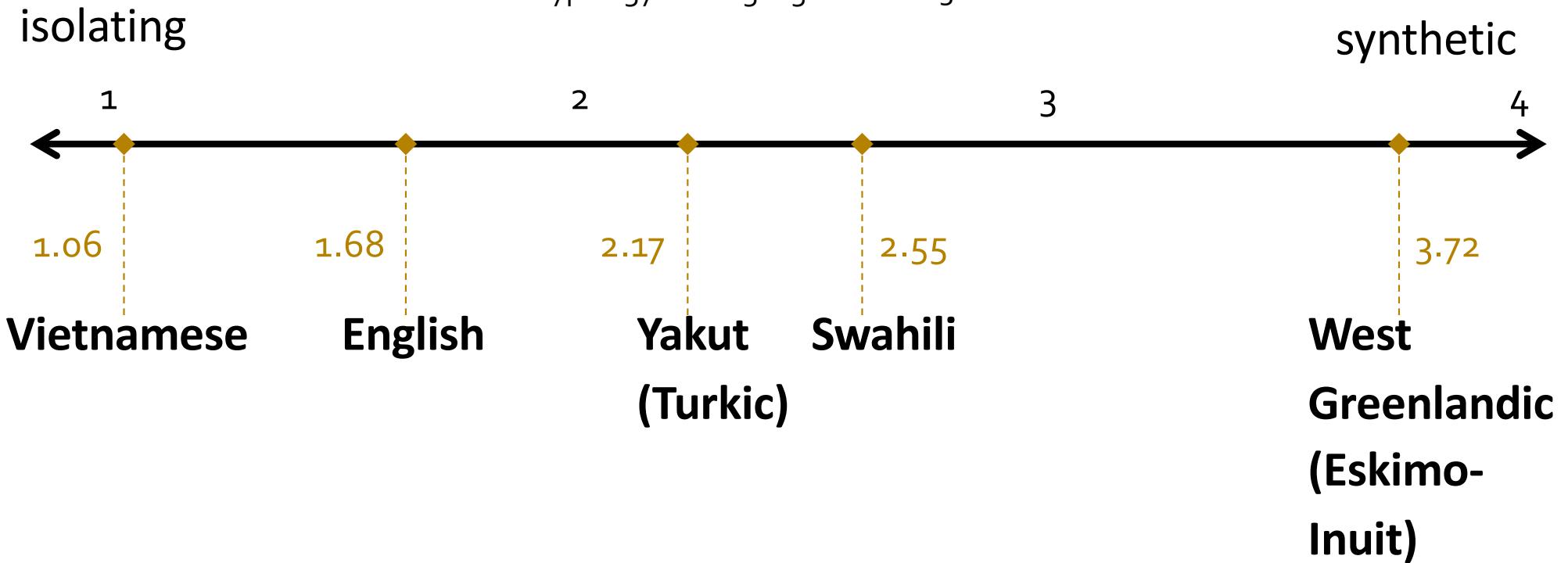
- *isolating vs polysynthetic*

isolating => little or no morphological change e.g., Chinese

polysynthetic => high morpheme to word ratio, the ability to form words which are equivalent to sentences in other languages, e.g., many Amerindian languages

Morphemes per Word

Joseph Greenberg. 1954. A Quantitative Approach to the Morphological Typology of Language. IJAL 26:3.



Many Morphemes per word: Turkish

uygarlaştıramadıklarımızdanmışsınızcasına

uygar+laş+tır+ama+dık+lar+ımız+dan+mış+sınız+casına

Behaving as if you are among those whom we could not cause to become civilized

Syntactic Differences

- See Jurafsky and Martin Chapter 13
 - SVO vs SOV vs VSO
 - adjective noun order
 - prepositions vs postpositions
 - head marking vs dependent marking
 - verb-framed vs satellite framed
 - pro-drop languages, referential density, hot and cold languages
 - idiosyncratic constructions e.g., “There burst into the room three men with guns.”

Evaluation

- Human Raters
 - Fluency :
 - ratings on scale of 1 -5
 - Cloze task
 - delete every nth word – can human readers guess the missing words?
 - Fidelity
 - adequacy/informativeness
 - bilingual vs monolingual raters
 - post-edit cost
- Automatic Evaluation e.g., BLEU, chrF

BLEU

Machine: check the forecast weather station

Human 1: consult the resort weather forecast

Human 2:check the weather forecast at the resort

- Average Unigram Precision= $\frac{1}{2} (3/5 + 4/5)$
- What are the flaws?
- Why not recall?

Modified precision

Machine: the the the the the

Human 1: consult the resort weather forecast

Human 2: check the weather forecast at the resort

- average unigram precision = $\frac{1}{2} (5/5 + 5/5) = 1$
- is this a good translation?

Modified precision

- For each word in the machine translation, take the maximum number of times it occurs in any human reference
- For example, $m_{\max}(\text{the}) = 2$
- Restrict the number of times a word can appear in machine translation to its m_{\max}
- In above example, modified unigram precision = $2/5$

BLEU (continued)

- computes modified precision for unigrams, bigrams, trigrams and often quadrigrams
- combines using geometric mean
- incorporates a penalty for translations which are too short
- good for evaluation of incremental changes to same general architecture
- see Papineni 2002

chrF

- Character-based F score(Popovic 2015)
- Overcomes problems with different tokenization standards
- In practice, if there are different possible human translations, machine translation cannot capture all of the variation, so recall may be lower than precision
 - But this is the same for all of the hypotheses (different machine translations)
 - So, recall still provides a robust way of balancing precision

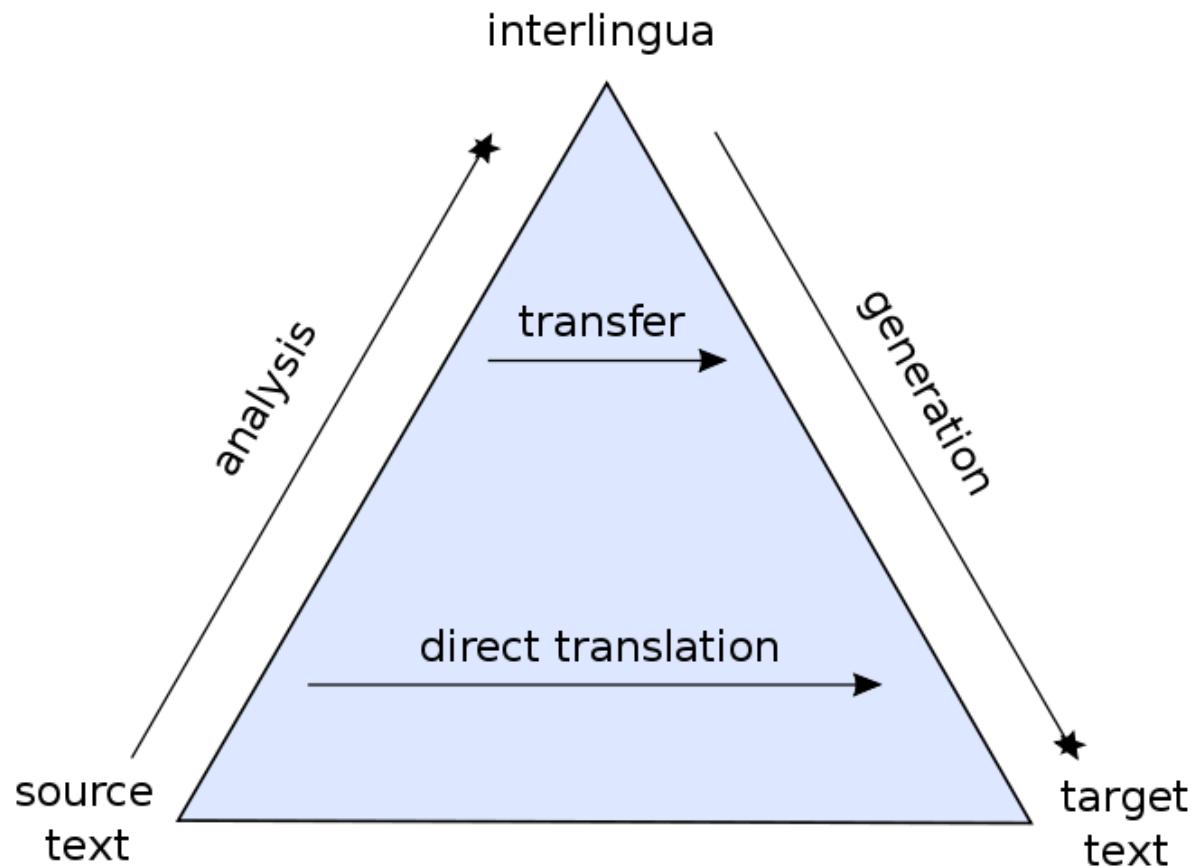
chrF (cont)

- chrP = percentage of character 1-grams, 2-grams, ..., k-grams in the hypothesis that occur in the reference, averaged
- chrR = percentage of character 1-grams, 2-grams, ..., k-grams in the reference that occur in the hypothesis, averaged

$$chrF\beta = (1 + \beta)^2 \frac{chrP \cdot chrR}{\beta^2 \cdot chrP + chrR}$$

$\beta=2$ gives twice as much weight to chrR as to chrP

Classical MT: Vauquois Triangle



Classical MT systems developed before 1990s were generally **rule-based** systems used 3 basic approaches:

- Direct translation
- Transfer-based
- Interlingua based

Statistical MT

- focus on the result NOT the process
- What does it mean for a sentence to be a translation of some other sentence?
- faithfulness and fluency
- based on probabilities derived from *parallel corpora*
 - *Sentences in a source language matched with sentences in the target language*

Bayesian Noisy Channel Model

Channel source = target language E

I saw the black dog

$P(E)$

Channel output = source language F

J'ai vu le chien noir

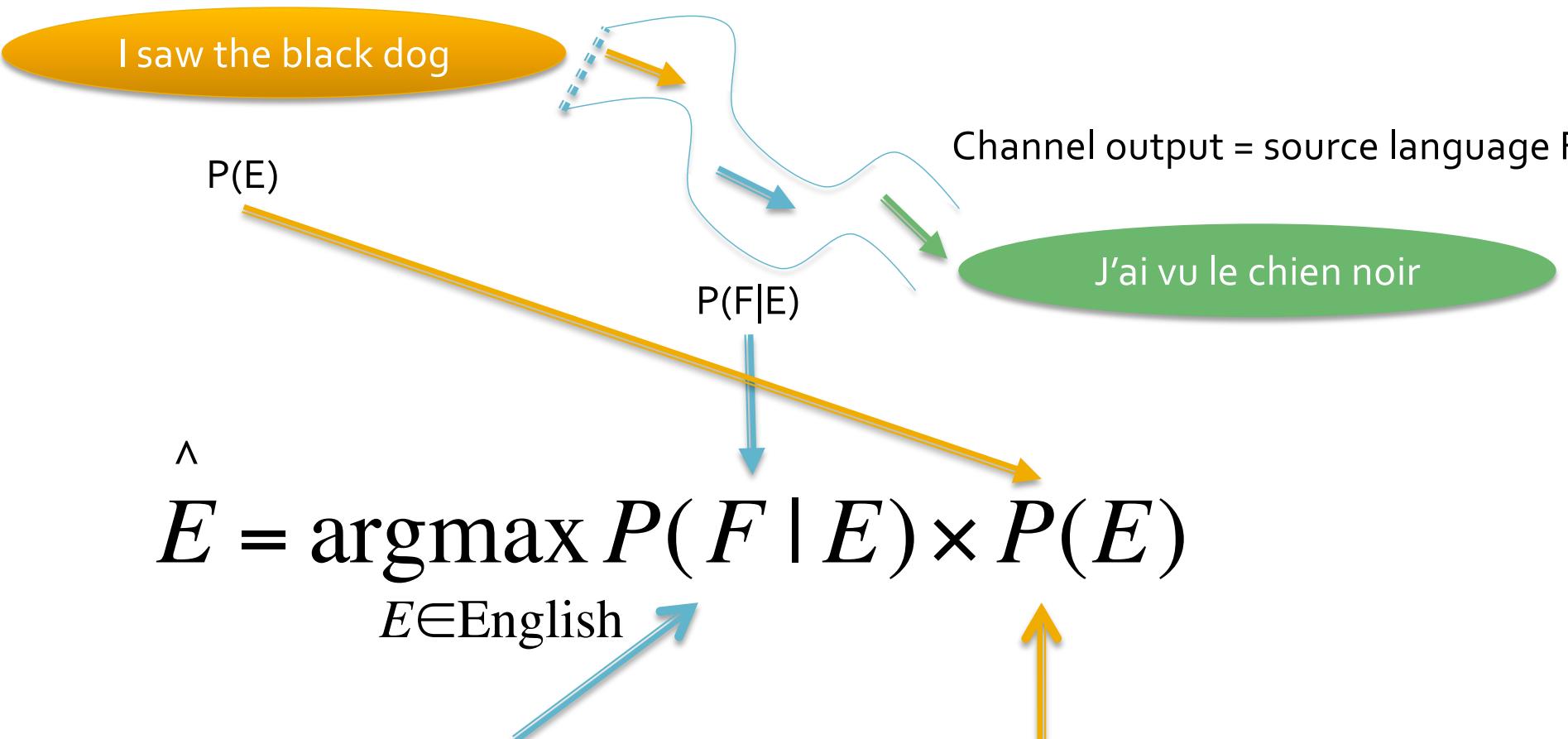
$P(F|E)$

$$E = \operatorname{argmax} P(F | E) \times P(E)$$

$E \in \text{English}$

translation model
captures faithfulness

language model
captures fluency



P(E): fluency

- The language model captures the fact that
I the dog black saw.

is less fluent / less probable in English than

I saw the black dog.

P(E) can be obtained from any monolingual corpus

- Most systems used an n-gram language model e.g., a bigram model

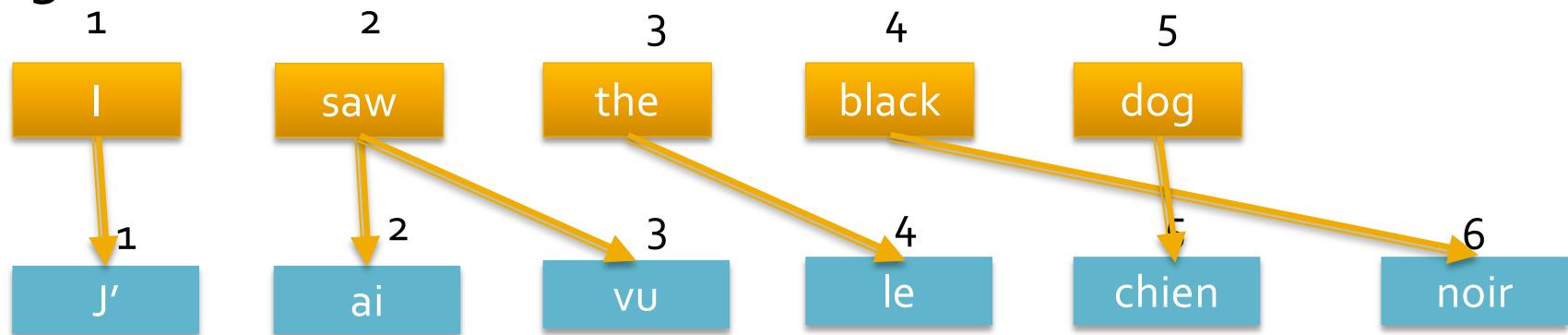
$$P(E_1) = P(I | \text{START}) \times P(\text{the}|I) \times P(\text{dog}|\text{the}) \times P(\text{black}|\text{dog}) \times P(\text{saw}|\text{black}) \times P(\text{END}|\text{saw})$$

$$P(E_2) = P(I | \text{START}) \times P(\text{saw}|I) \times P(\text{the}|\text{saw}) \times P(\text{black}|\text{the}) \times P(\text{dog}|\text{black}) \times P(\text{END}|\text{dog})$$

- More sophisticated models (e.g., based on grammar) could be used

$P(F|E)$: faithfulness

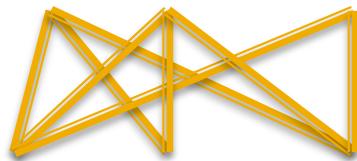
- Simplest translation models are based on *word alignment*
- A word alignment is a mapping between words in F and words in E
- Common simplifying assumption is that it is a one-to-many alignment – each French word comes from exactly one English word



Alignment can be represented by a vector of indices.
Here, the vector would be [1, 2, 2, 3, 5, 4]

Estimating Translation Probabilities

- Parallel corpora → sentence-aligned data
...le chien noir ...



...the black dog ..

- English and French words but no connections between them
- How can we use this to estimate the probability of a French word being generated by an English word?

Example

The three goats all loved to eat grass.

Les trois chèvres ont tous adoré manger de l'herbe.

They ate grass all day long on the hill.

Ils mangeaient de l'herbe toute la journée sur la colline.

But they never crossed the bridge to eat the grass on the other side.

Mais ils n'ont jamais traversé le pont de manger l'herbe de l'autre côté.

They never crossed the bridge because the Troll lived under the bridge.

Ils n'ont jamais traversé le pont parce que le Troll vivait sous le pont.

The Troll was very bad.

Le Troll était très mauvaise.

He ate anyone who dared to cross his bridge.

Il a mangé quiconque osait traverser son pont.

Expectation Maximization (EM)

- initialise model parameters (all connections equally likely)
- assign probabilities (E-step)
- estimate parameters (M-step)
- iterate

From word-based models to phrase-based models

- Word-based models assume one-to-many alignment of words
- Word-based models cannot (easily) handle non-compositional phrases
- Phrase-based models treat phrases as atomic units
- Many-to-many translation can handle non-compositional phrases
- Was (until 2016) the state-of-the-art used by Google Translate.

Phrase-alignment

- Generated by running word alignment algorithms in both directions to give
 - a one-to-many alignment
 - a many-to-one alignment
- Classifiers developed to decide how to symmetrize the alignments somewhere between intersection and union

Phrase Translation Table

- Given a phrase alignment, we can store each pair of aligned phrases in a phrase translation table together with its MLE translation probability:

$$\phi(f, e) = \frac{\text{count}(f, e)}{\sum_f \text{count}(f, e)}$$

- This gives us the “translation options” for each phrase at decode time.

Standard Model for PBMT

- For translating French (source language) to English (target language), use a log-linear model:

$$\hat{E} = \operatorname{argmax}_{E \in English} P(E|F) = \exp \left(\sum_i \lambda_i h_i(e, f) \right)$$

- The feature functions h_i are typically
 - a language model;
 - a reordering model;
 - a word penalty; and
 - various translation models (phrase translation and word translation)
- Discriminative or generative? Why?

Decoding for Phrase-Based Statistical MT

- Finding the sentence which maximises translation probability is a search problem
- Exhaustive search impossible!
- Decoders in MT tend to use best-first search
- Maintain a priority queue (or stack) with all partial translation hypotheses and their scores
- Use the phrase-translation table to limit the search space to target language sentences which are possible translations of the source language sentence
- Beam search: At every iteration, only keep the k most promising search states and prune high-cost states

Shortcomings of PBMT

- Brittle design choices (Wu et al. 2016)
- Large phrase translation tables
- Inability to generalise (Kalchbrenner and Blunsom 2013)
 - i.e., similar phrase pairs do not share statistical weight in the models' estimation of their translation probabilities
 - Leading to general sparsity issues
 - Imperfect translations

PBMT sample output

French

Pour une journée au ski réussie à Samoëns, n'hésitez pas, consultez les prévisions météo sur la station, les conditions d'enneigement et l'ouverture des pistes. Profitez d'une bonne journée au ski à Samoens, si vous venez en week end ou pour une semaine, bénéficiez de prévision météo à 2 jours sur notre site et à plus long terme sur le site de Météo Chamonix !

English (Google 2014)

For a successful day at Ski Samoens , please, check the forecast weather station on the snow conditions and opening tracks. Enjoy a good day skiing Samoens , if you come on a weekend or for a week, get forecasting weather two days on our website and in the longer term on the site Weather Chamonix !

Part 3

NMT

Neural Machine Translation (NMT)

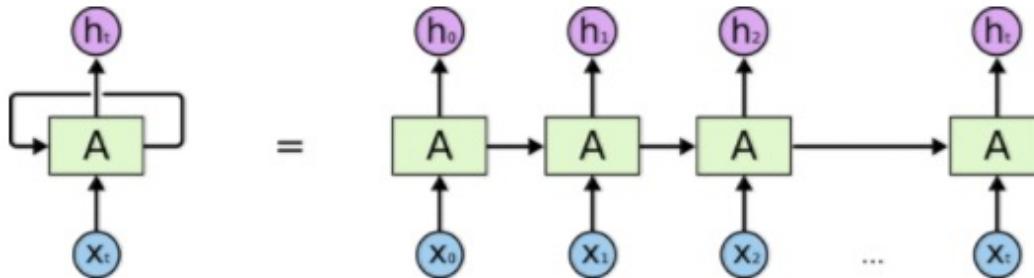
- Continuous representations (e.g., word2vec embeddings) for words and phrases are able to capture their morphological, syntactic and semantic similarity
- As in SMT, train on parallel corpora where sentences are aligned
- Maximise the probability of the sequence of tokens in the target language $y_1 \dots y_m$ given the sequence of tokens in the source language $x_1 \dots x_n$

$$P(y_1, \dots, y_m | x_1, \dots, x_n)$$

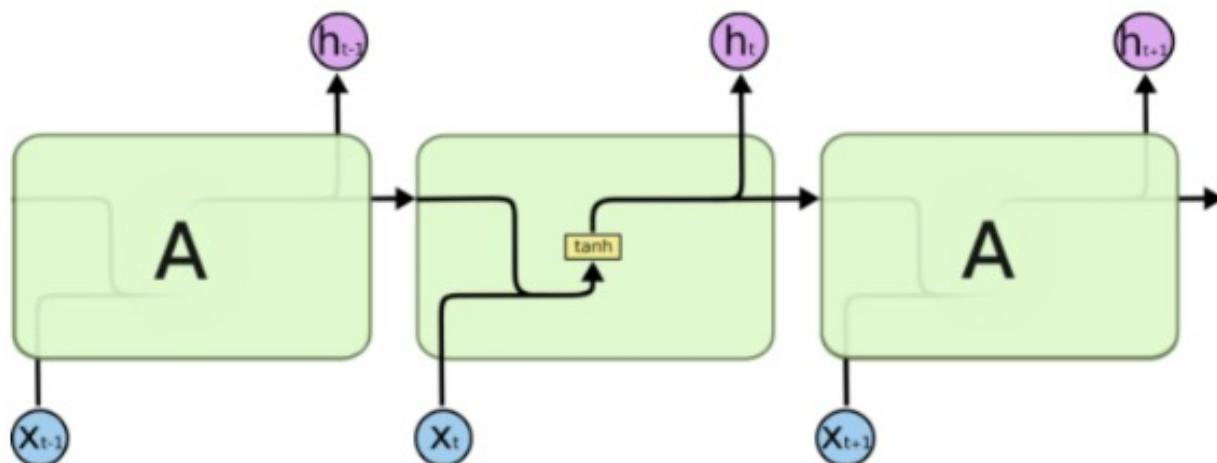
Basic architecture for NMT

- Encoder – decoder architecture
 - Aka sequence-to-sequence or seq2seq architecture
- 2 recurrent neural networks (RNNs) – one to consume the input text sequence and one to generate translated output text.

RNNs



An unrolled recurrent neural network.

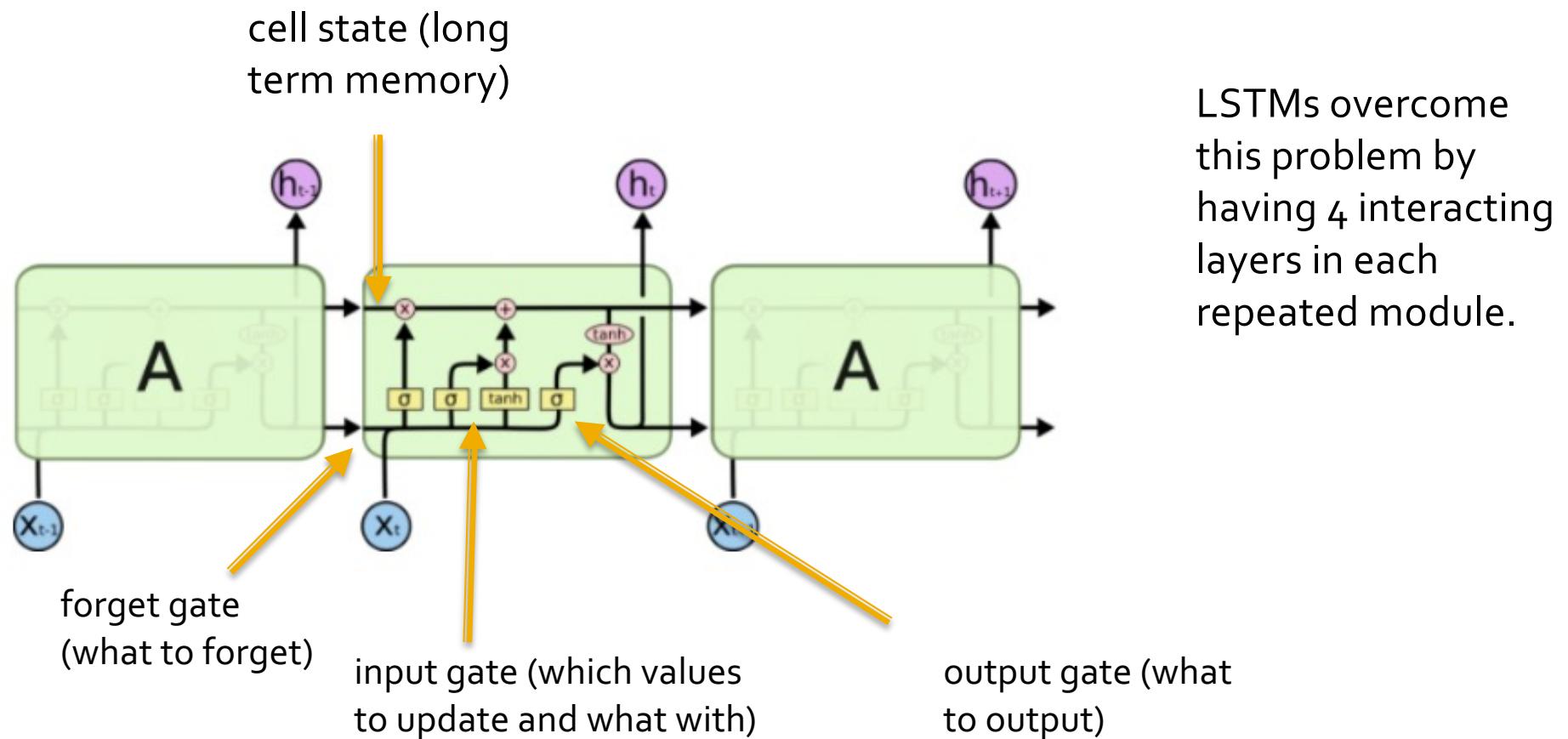


The repeating module in a standard RNN contains a single layer.

RNNs are very effective at learning language models i.e., $P(E)$ the probability of a sentence in a given language. During training, the error (i.e., difference between output and next word) is back-propagated to update the weights used to combine X_t and h_{t-1} AND the representations of the words (X_t)

Long short term memory networks (LSTMs)

- Simple RNNs struggle with long term dependencies e.g., “He grew up in Spain. He speaks fluent ...”

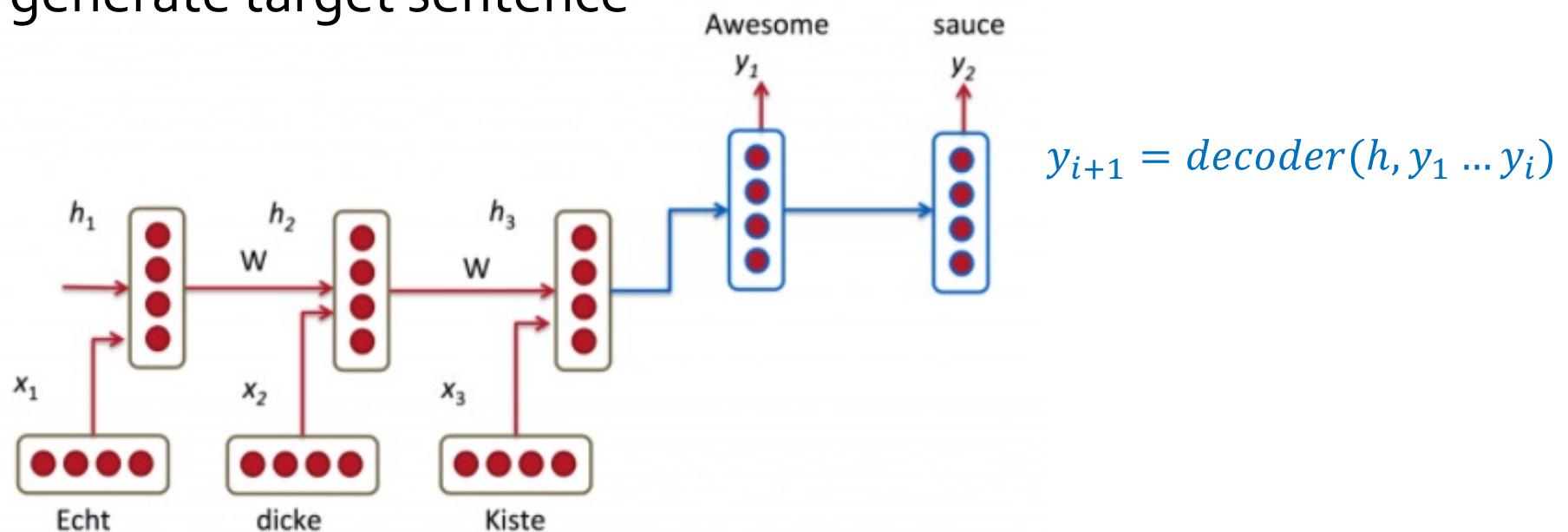


Basic architecture for NMT

- RNN₁, the encoder, builds a representation of the source sentence $x = x_1 \dots x_n$

$$h = \text{encoder}(x)$$

- The output from RNN₁ (after the complete source sentence has been read) is input to RNN₂, the decoder to generate target sentence



Encoder-decoder details

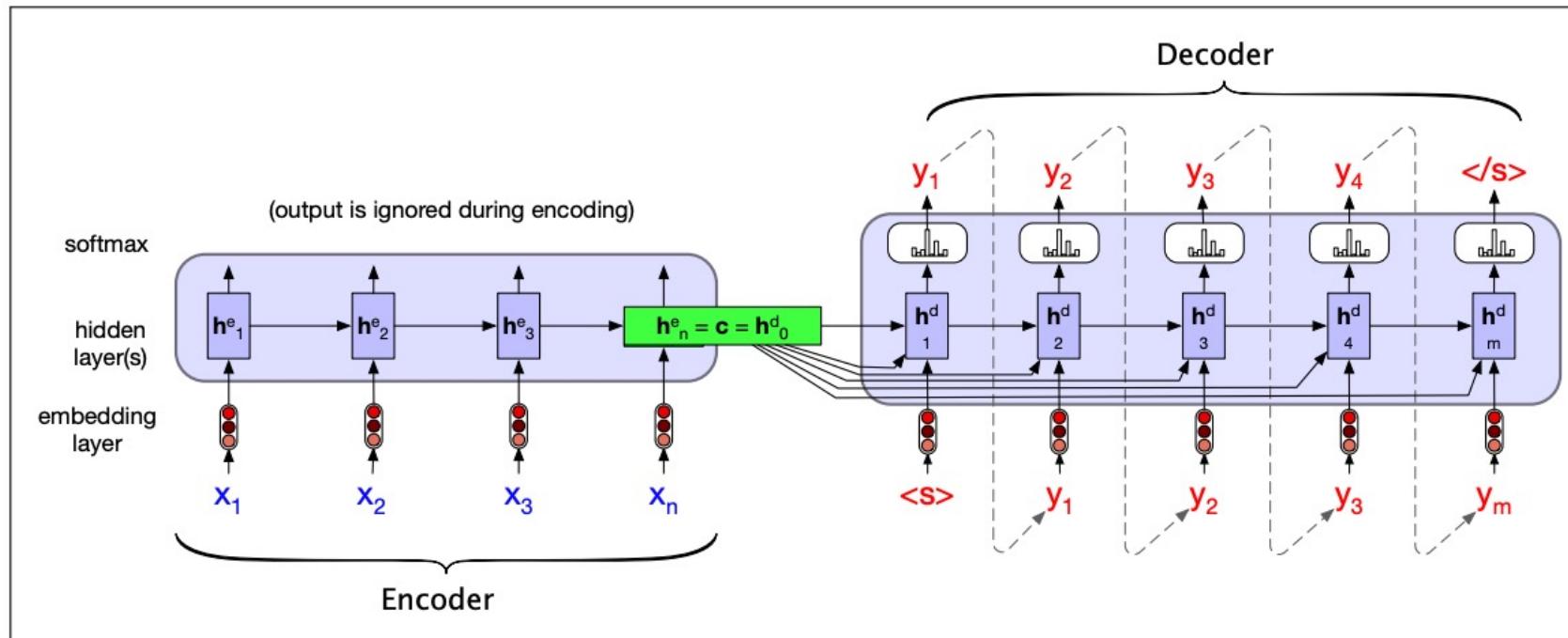


Figure 9.18 A more formal version of translating a sentence at inference time in the basic RNN-based encoder-decoder architecture. The final hidden state of the encoder RNN, h_e^n , serves as the context for the decoder in its role as h_d^0 in the decoder RNN, and is also made available to each decoder hidden state.

Possible weaknesses

- Slow training and inference speed
- Ineffectiveness at dealing with rare words
- Output sentences that do not translate all words of the input sentence
- Difficulty in translating long sentences since the encoder output (or context) needs to encode the whole sentence
 - Information from start of sentence may be lost

Rare words (Luong et al. 2015)

- Due to computational constraints, NMT systems usually limited to top 30K-80K of most frequent words in each language
- Unknown/rare words can be translated using a dictionary or exact copy provided it is known which source word generated UNK token in target.
- Problem when sentence contains multiple rare words
- Luong et al. first use a word alignment of parallel corpora and annotate unknown words with positional information (e.g., UNK₁)
- Output from NMT can then be post-processed

Subword tokenization

- Word vocabulary is huge and sparse
- Character vocabulary is small and dense, but lacking in semantic meaning
- Subword tokenization provides a compromise
- Frequent words tend to be a token whereas rare words will be broken down into subwords based on character n-grams
- Shared vocabulary for source and target languages – makes it easy to copy tokens like names from source to target
- Common algorithms include
 - BytePiece Encoding (BPE)
 - Wordpiece algorithm
 - Unigram / SentencePiece algorithm

Long sentences

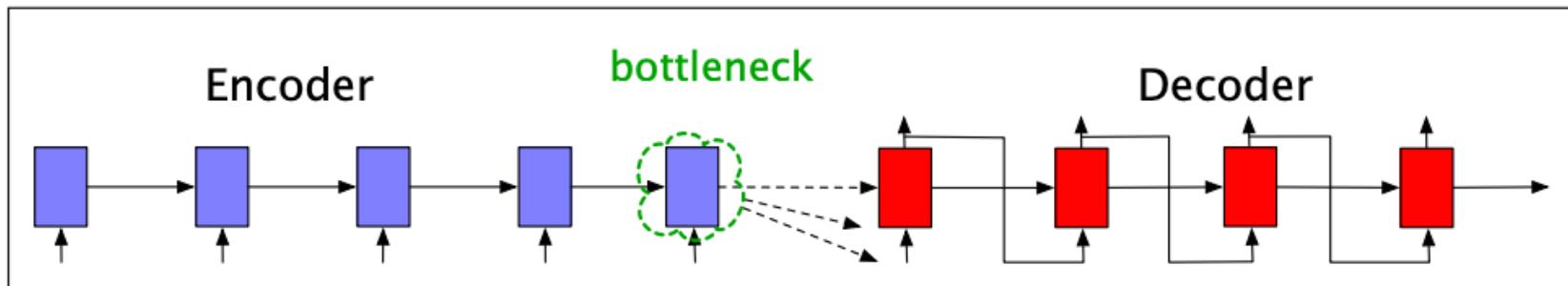


Figure 9.20 Requiring the context c to be only the encoder's final hidden state forces all the information from the entire source sentence to pass through this representational bottleneck.

- Attention (more on this next week) provides a way for the decoder to get information from all of the hidden states of the encoder rather than just the last hidden state

Attention

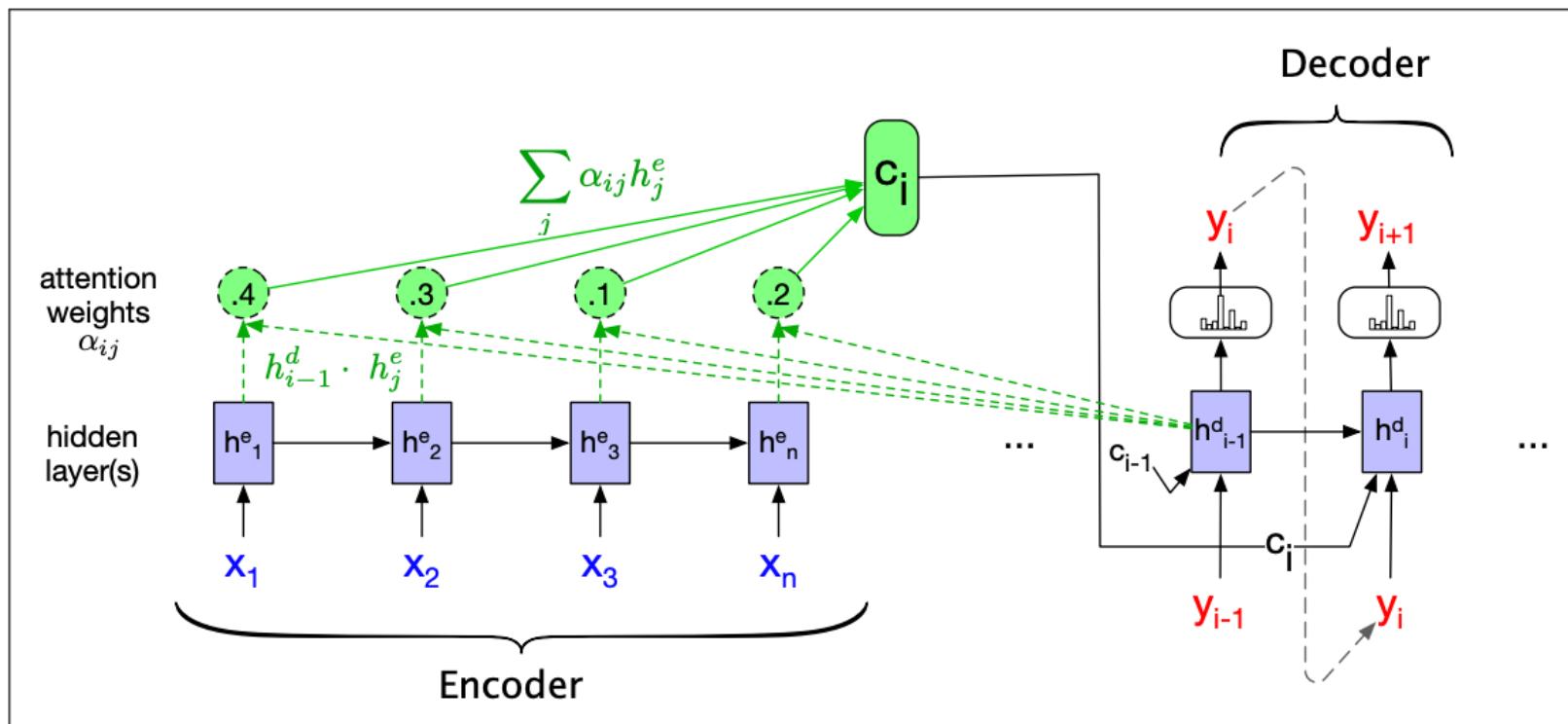
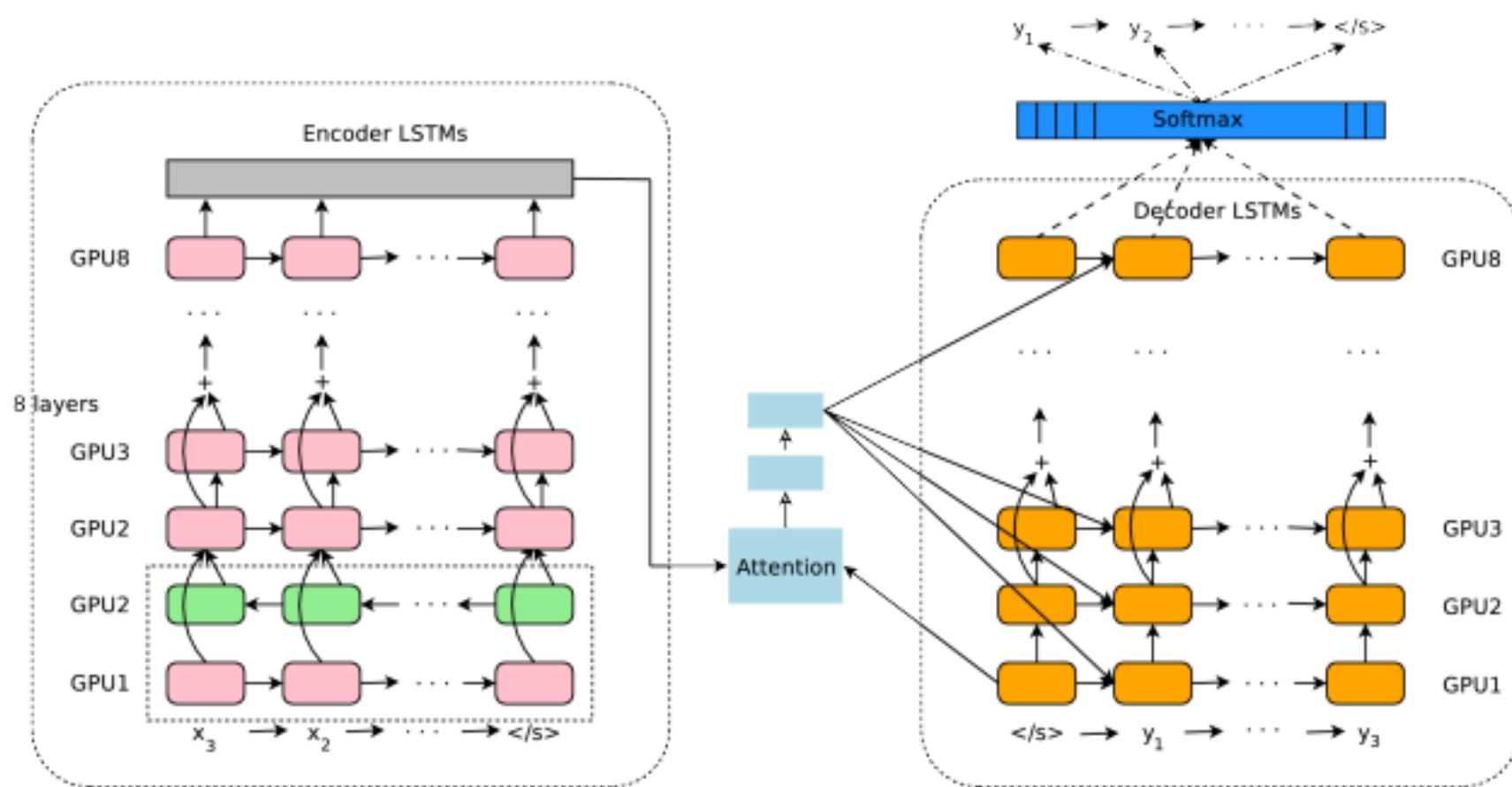


Figure 9.22 A sketch of the encoder-decoder network with attention, focusing on the computation of c_i . The context value c_i is one of the inputs to the computation of h_i^d . It is computed by taking the weighted sum of all the encoder hidden states, each weighted by their dot product with the prior decoder hidden state h_{i-1}^d .

Google NMT (GNMT)

- Recurrent networks are LSTMs with attention (8 layers - residual connections between layers to encourage gradient flow)
- For parallelism, the attention from the decoder network connect to top layer of encoder network
- Low-precision arithmetic for inference, accelerated using special hardware (Google's TPU)
- Rare words dealt with using sub-word units (balancing flexibility of single characters with efficiency of full words)
- Beam search techniques includes a length normalization procedure and a coverage penalty to encourage model to translate all of the input

GNMT Architecture



Transformers and LLMs in MT?

- Transformers generally have higher performance than LSTMS and GRUs
 - Generally replacing seq2seq architectures
 - More on this in weeks 8-10
-
- <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9355969>
 - <https://arxiv.org/abs/2209.07417>

Open questions

- High resource vs low resource languages

References

- Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models, In *EMNLP*
- Philip Koehn. 2004. Pharoah: A Beam Search Decoder for Phrase-Based Statistical Machine Translation Models. In *AMTA*
- Minh-Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals and Wojciech Zaremba. 2015. Addressing the Rare Word Problem in Neural Machine Translation. In *ACL*
- Ilya Sutskever, Oriol Vinyals and Quoc Le. Sequence to Sequence Learning with Neural Networks. In *NIPS*
- Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc Le and Mohammad Norouzi. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. ArXiv Oct 2016

AdvNLE Lecture 8

Pre-training Large Language Models

Dr Julie Weeds, Spring 2024



Previously

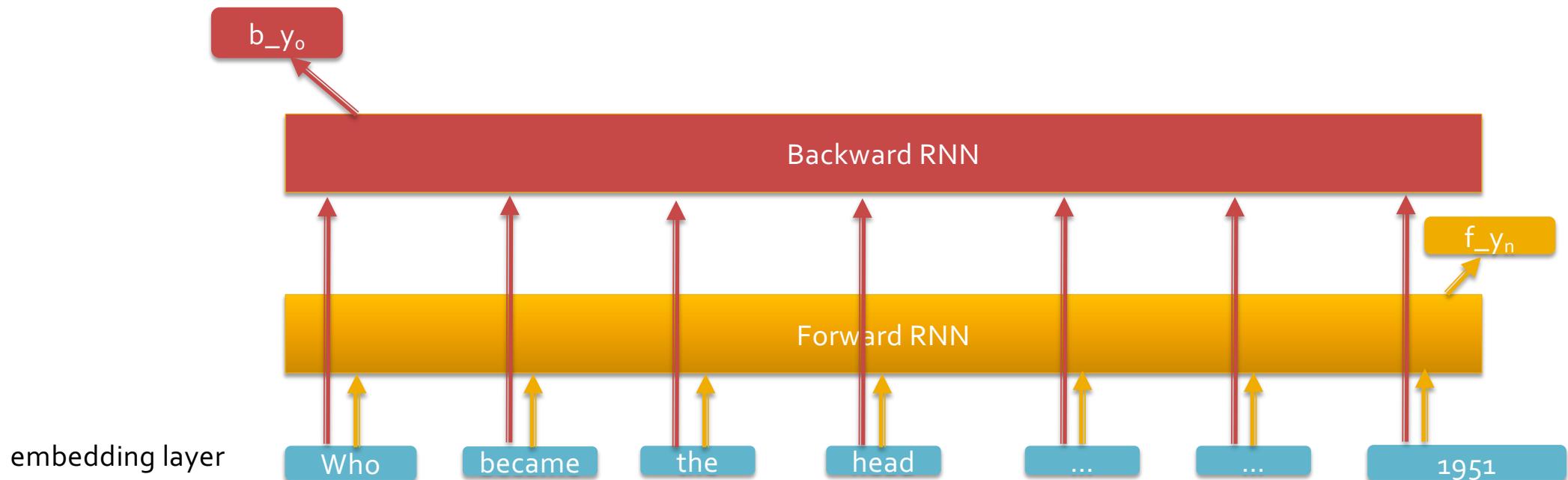
- Distributional models of word meaning
 - how similar are two words based on how they are used in text?
- Language models
 - how likely is a sequence of words in a language?
- Neural language models
- Tasks
 - Sequence labelling
 - Sequence classification
 - Sequence generation

This week

- Large language models
 - Contextualised word embeddings (ELMo)
 - Transformer architecture and attention
 - BERT (bidirectional encoder representation from transformers)
 - Pre-training:
 - Masked language modelling
 - Autoregressive language modelling
 - Next sentence prediction
 - Sentence representations / paraphrasing (Seminar)

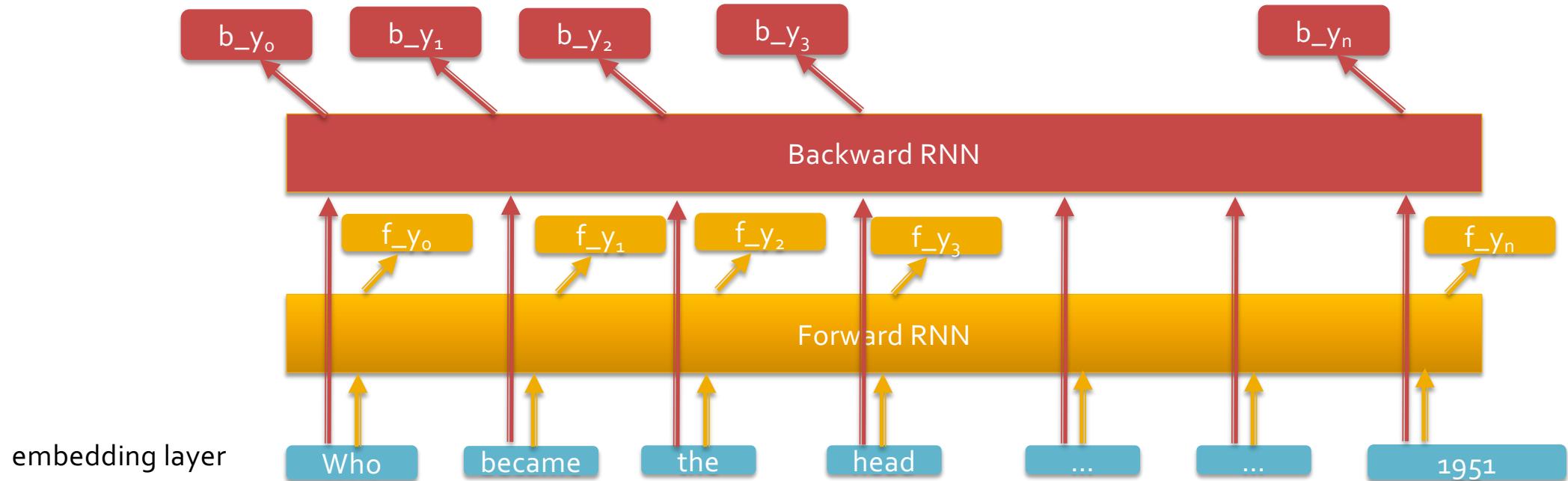
Language modelling for Sentence Representations

- Represent a sequence by what is predicted to the left and right of it
- e.g., concatenation of $[b_y_o, f_y_n]$



Contextualised word embeddings

- Represent each word based on its context
- e.g., concatenation of $[b_y_t, f_y_t]$
- compose contextualised word embeddings as before



ELMo (Peters et al. 2018)

- Embeddings from Language Models
- representations are functions of the entire word sequence
- computed
 - on top of two-layer biLMS with character convolutions
 - as a linear function of internal network states
- multiple representations of the same word:
 - higher-level layers capture context dependent aspects of word embeddings
 - lower-level layers capture aspects of syntax
- semi-supervised learning –
 - biLMs can be pre-trained at a large scale
 - final layers fine-tuned on smaller / task-specific data-set

ELMo Base Model

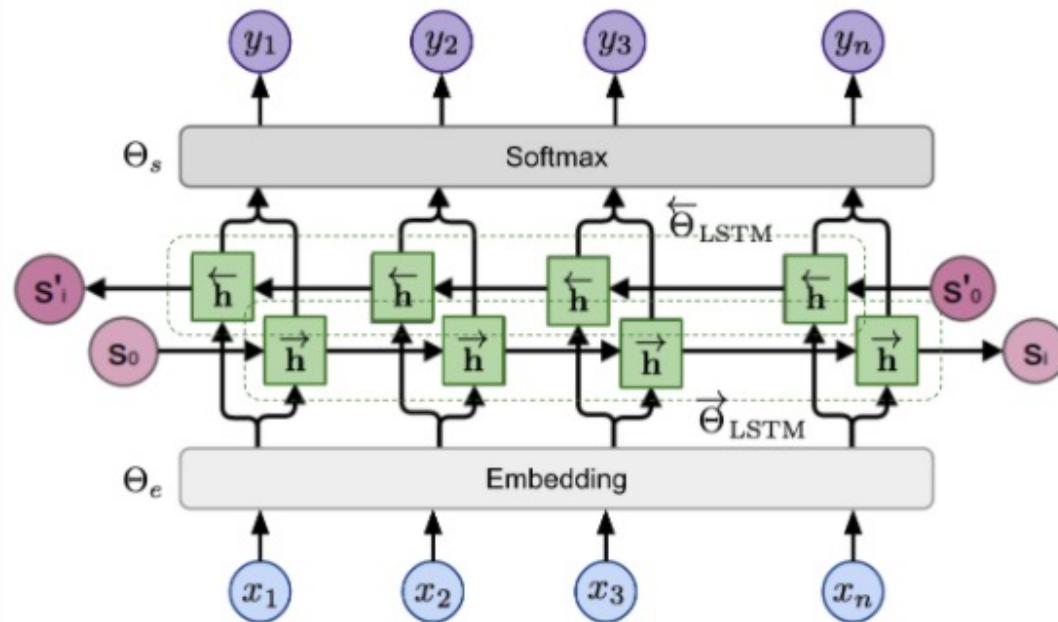
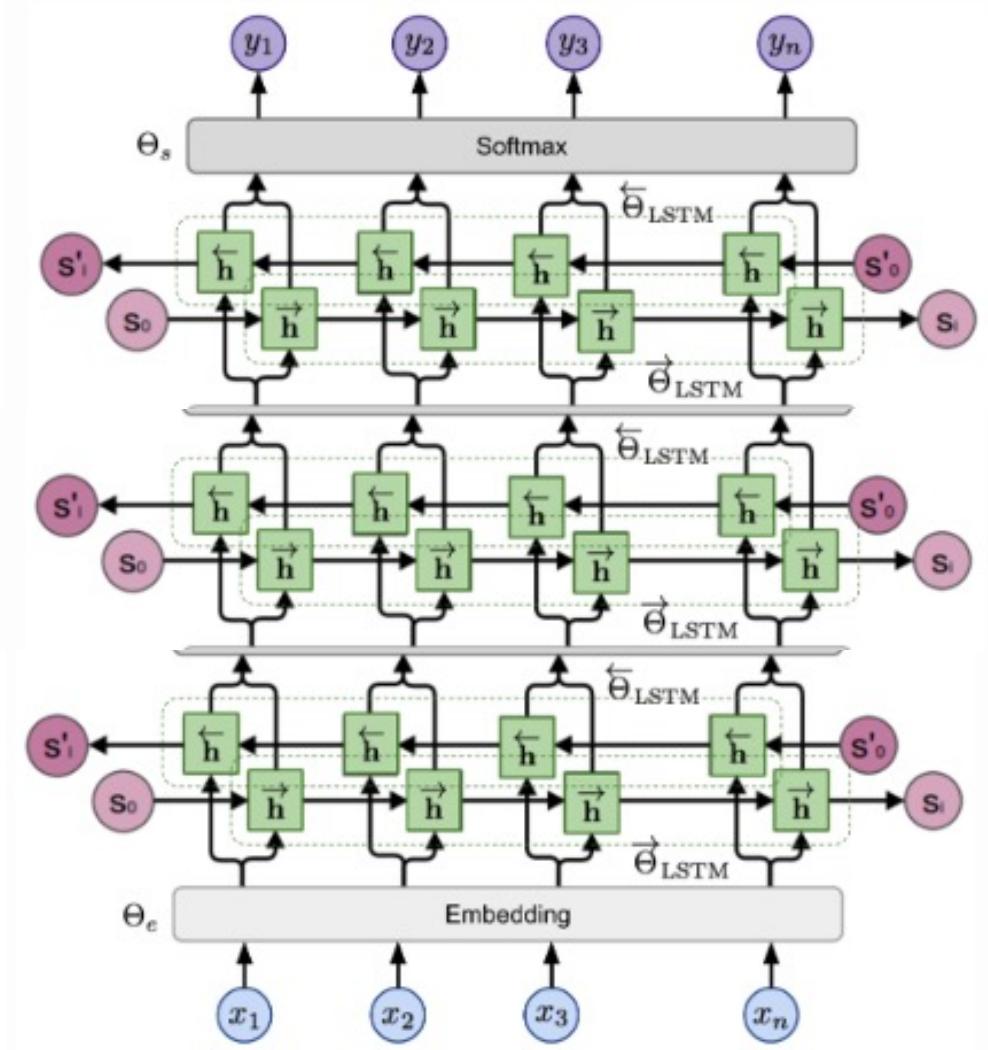


Fig. 3. The biLSTM base model of ELMo. (Image source: recreated based on the figure in “[Neural Networks, Types, and Functional Programming](#)” by Christopher Olah.)

ELMo Deep Model

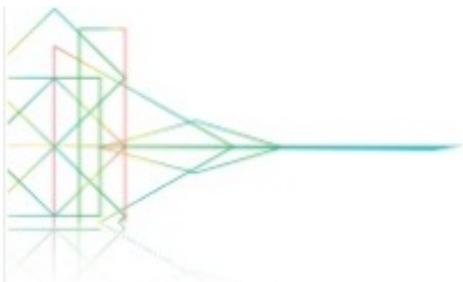
- Stack L biLSTMs on top of each other
- inputs to second layer are contextualised by the words either side of a target word
- inputs to third layer are contextualised by 2 words either side of target word
- higher layers better at capturing longer-range semantic dependencies



ELMo Embeddings

- Each layer learns its own vector representation for every token
- These are combined and a final softmax layer is trained to collapse these vectors into a single vector and a task-specific scalar
- Trained ELMo word embeddings often used as input to another neural model!

Method

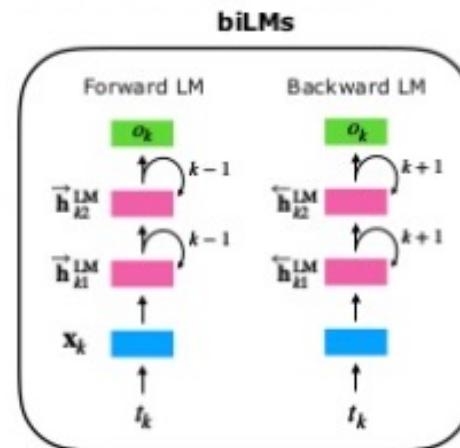


ELMo is a task specific representation. A down-stream task learns weighting parameters

$$\text{ELMo}_k^{\text{task}} = \gamma^{\text{task}} \times \sum \left\{ \begin{array}{l} s_2^{\text{task}} \times h_{k2}^{\text{LM}} \\ s_1^{\text{task}} \times h_{k1}^{\text{LM}} \\ s_0^{\text{task}} \times h_{k0}^{\text{LM}} \\ ([x_k; x_k]) \end{array} \right. \xrightarrow{\text{Concatenate hidden layers}} (\vec{h}_k^{\text{LM}}, \overleftarrow{h}_k^{\text{LM}})$$

Unlike usual word embeddings, ELMo is assigned to every token instead of a type

ELMo represents a word t_k as a linear combination of corresponding hidden layers (inc. its embedding)



ELMo Fine Tuning Tasks

- Prediction of word-level tags
 - NER, semantic role labelling (SRL)
- Prediction of sentence-level labels
 - sentiment analysis (SST-5)
- Prediction of sentence-pair labels
 - textual entailment / natural language inference (SNLI), question-answering (SQuAD)

ELMo Results

Task	Previous SOTA	Our Baseline	ELMo + Baseline	Increase (Absolute/Relative)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17
SRL	He et al. (2017)	81.7	81.4	84.6
Coref	Lee et al. (2017)	67.2	67.2	70.4
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5

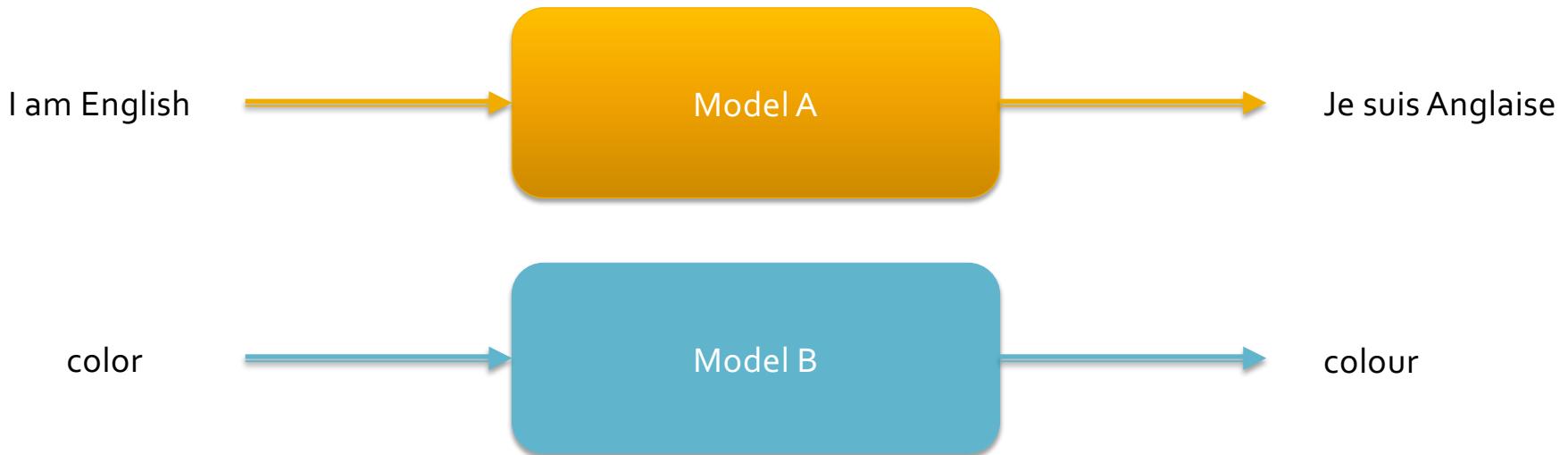
Table 1: Test set comparison of ELMo enhanced neural models with state-of-the-art single model baselines across six benchmark NLP tasks. The performance metric varies across tasks – accuracy for SNLI and SST-5; F_1 for SQuAD, SRL and NER; average F_1 for Coref. Due to the small test sizes for NER and SST-5, we report the mean and standard deviation across five runs with different random seeds. The “increase” column lists both the absolute and relative improvements over our baseline.

Transformers (Vashwani et al. 2017)

- Sequence transduction model using stacked self-attention
 - no convolutions or recurrence
 - easier to parallelize than RNNs
 - faster to train than RNNs
 - captures more long-range dependencies than CNNs with fewer parameters
- What's a sequence transduction model?
- What's stacked self-attention?

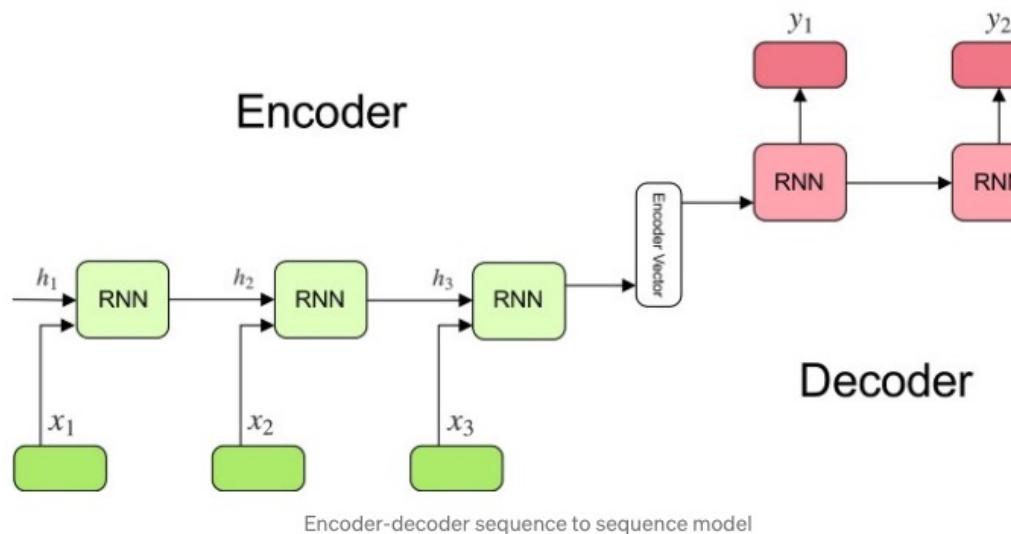
Sequence transduction

- Sequence to sequence models
 - Input: Sequence A
 - Output: Sequence B
- learning to convert one string into another

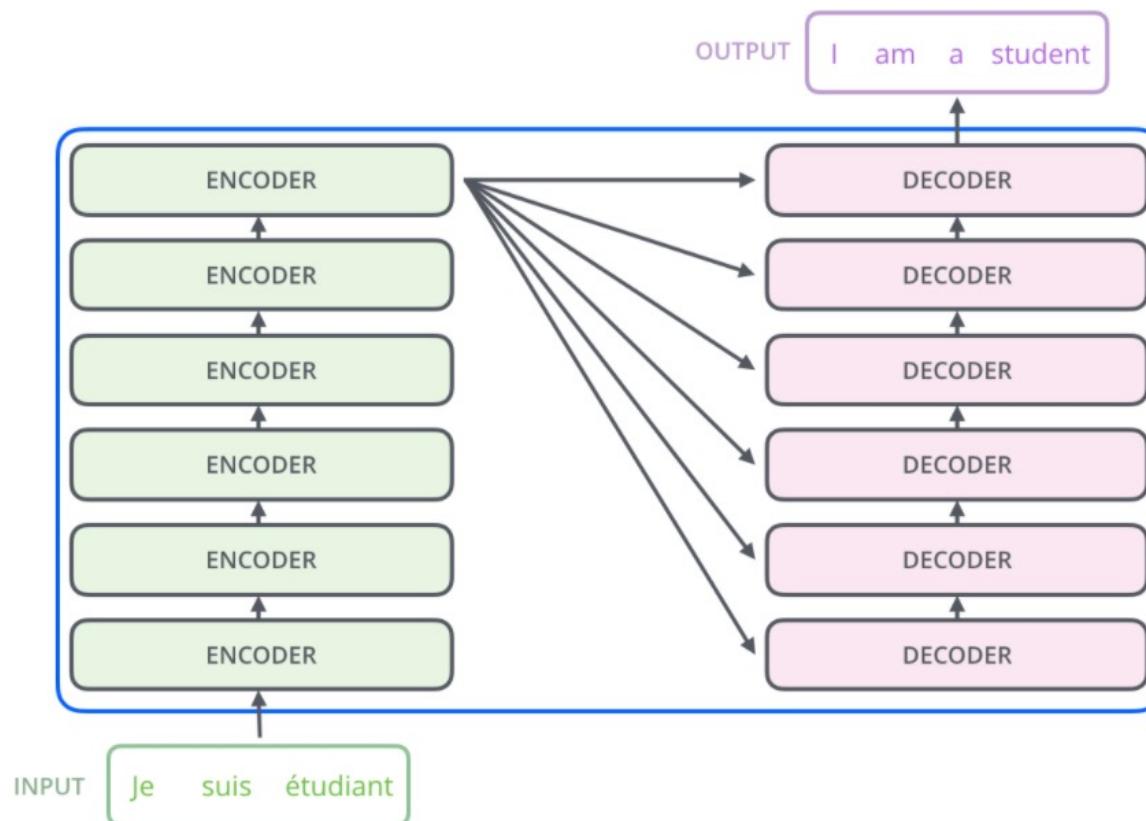


Encoder – Decoder Networks

- **Encode** the input sequence into a point in some latent space (*encoder vector*)
- **Decode** the point in latent space (*encoder vector*) into the output sequence
- The encoders and decoders can be any kind of neural networks, typically RNNs or attention-based



Stacked Encoder and Decoder Networks

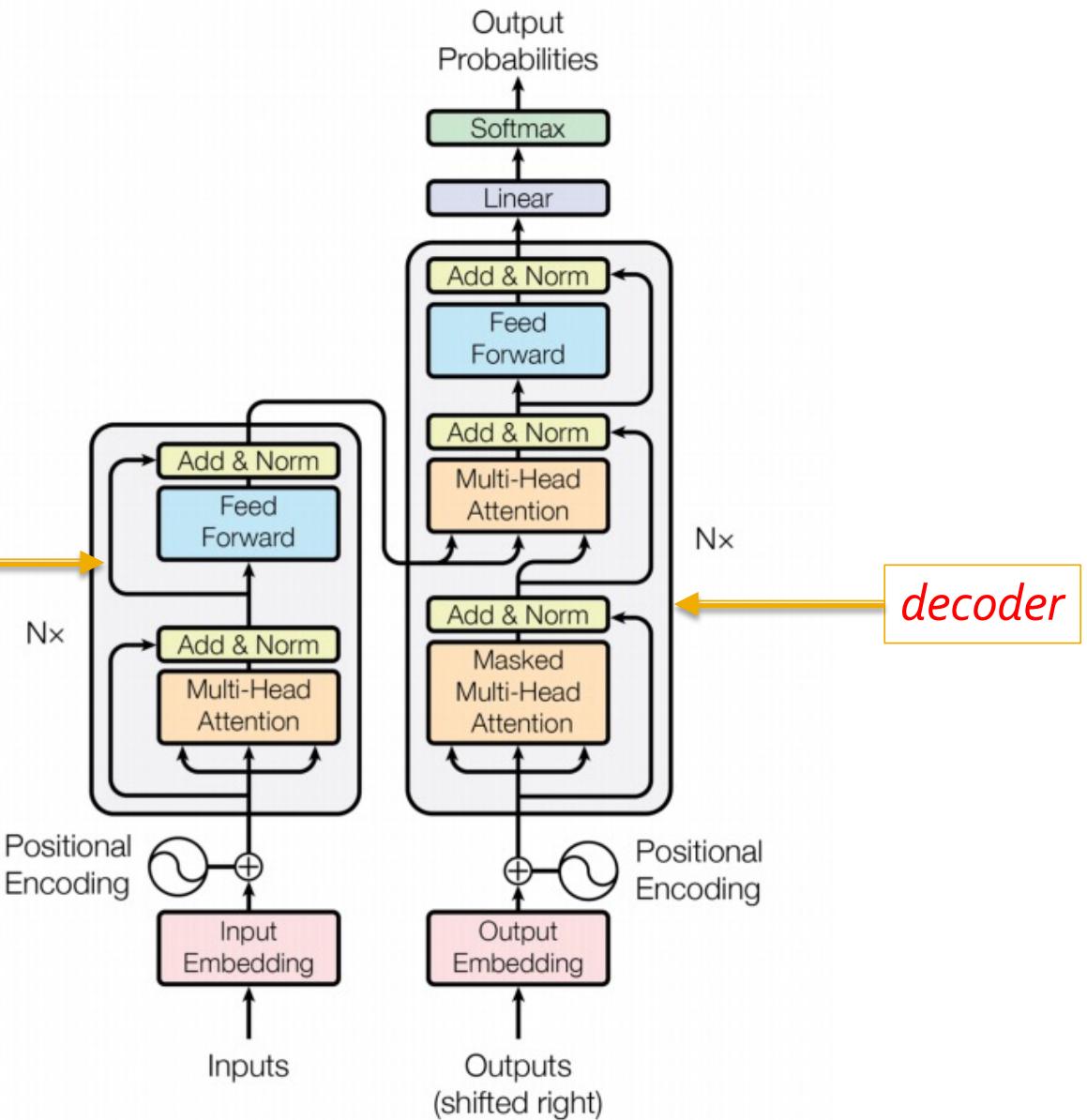


typically:

- same number of decoders as encoders
- typically 6 of each
- each decoder has access to the output of the last encoder (the *encoder vector*) as well as the previous decoder in the stack

Transformer Architecture

encoder

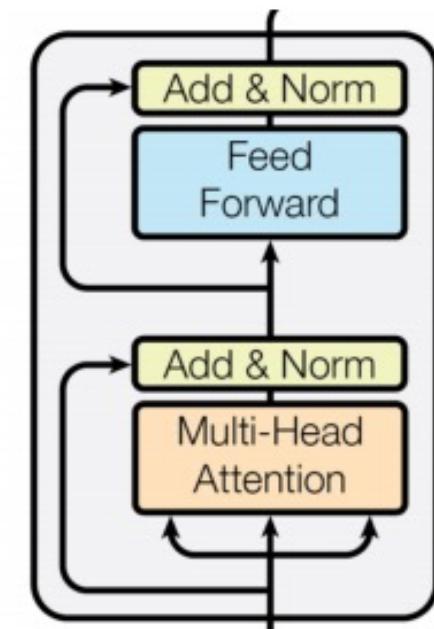


Diagrams borrowed from Julia Hockenmaier:

[https://courses.engr.illinois.edu/cs546/
sp2020/Slides/Lecture09.pdf](https://courses.engr.illinois.edu/cs546/sp2020/Slides/Lecture09.pdf)

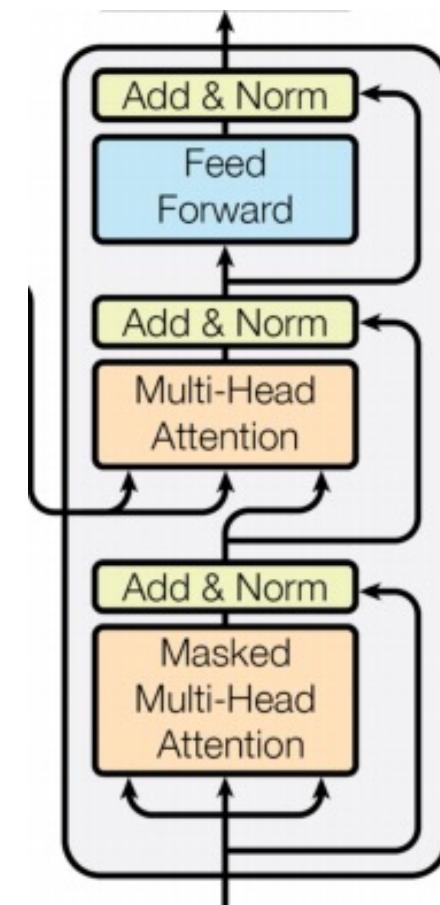
Encoder

- a stack of $N=6$ identical layers
- all layers and sublayers are 512-dimensional
- Each layer consists of **2** sublayers
 - one multi-headed self-attention layer
 - one position-wise fully connected layer
- Each sublayer has a residual connection and is normalised:
 - $\text{LayerNorm}(x + \text{Sublayer}(x))$



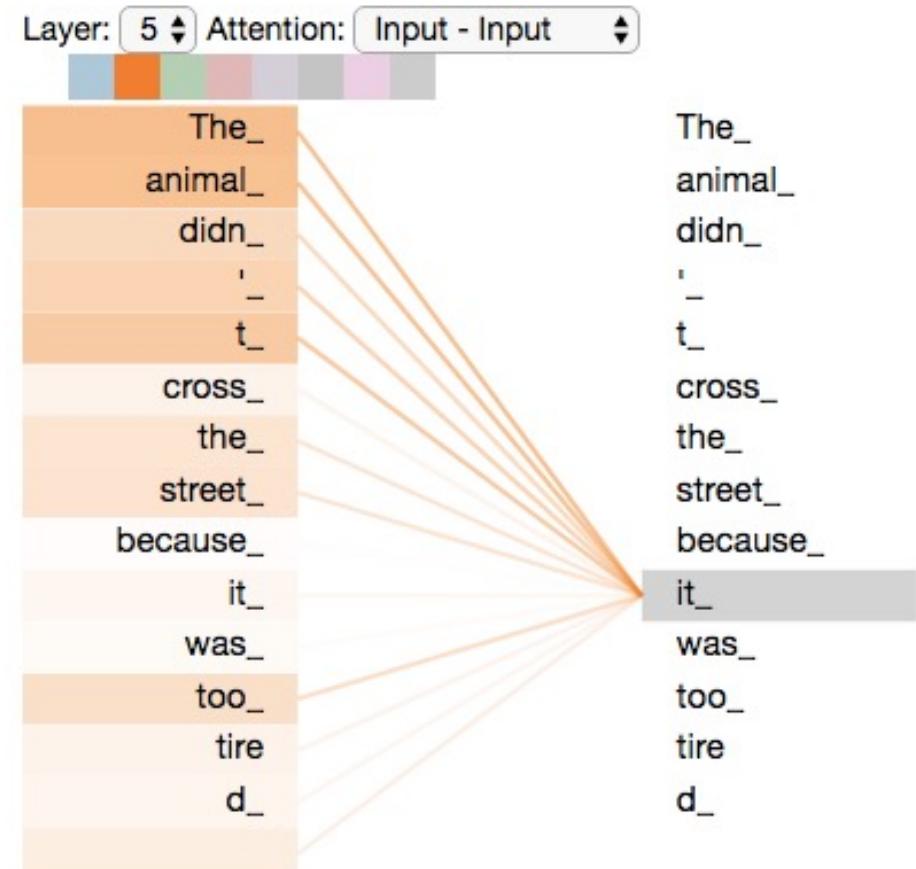
Decoder

- stack of $N=6$ identical layers
- all layers and sublayers are 512-dimensional
- Each layer consists of 3 sublayers:
 - one multi-headed self-attention layer over decoder output (ignore future tokens)
 - one multi-headed attention layer over encoder output
 - one position-wised fully connected layer
- each sublayer has a residual connection and is normalised
 - $\text{LayerNorm}(x + \text{Sublayer}(x))$



Self-Attention (High level)

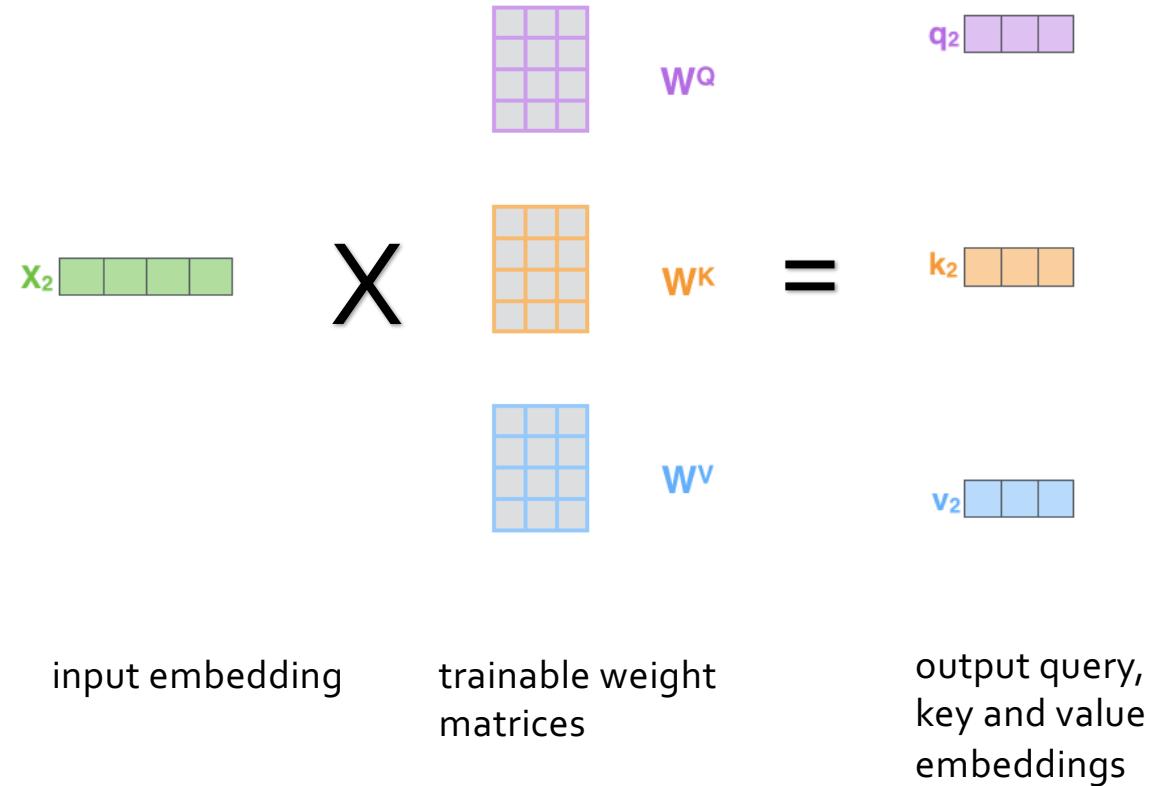
- What to pay attention to when encoding or decoding
 - **Attention:** in previous layer
 - **Self-attention:** in the same layer
- When encoding the word ***it*** in the sentence "*The animal didn't cross the street because it was too tired*", self-attention should allow the model to associate ***it*** with ***animal***



Example from <http://jalammar.github.io/illustrated-transformer/>

Self-attention - Step 1

- Step 1: from the encoder's input vector (e.g., word embedding), create 3 vectors: **Query**, **Key** and **Value**
- Q, K, and V embeddings usually smaller in number of dimensions e.g.,
 - input 512
 - output 64



Self-attention score

- Step 2: To work out how much the word in position_i (e.g., "it") should pay attention to a word in position_j (e.g., "animal") words in the sentence, we take the **dot product** of the Q_i (the **query** vector for *it*) with the K_j (the **key** vector for *animal*)

$$SelfAtt(w_i, w_j) = Q_i \cdot K_j$$

- Do this for every word in the sentence with every other word in the sentence

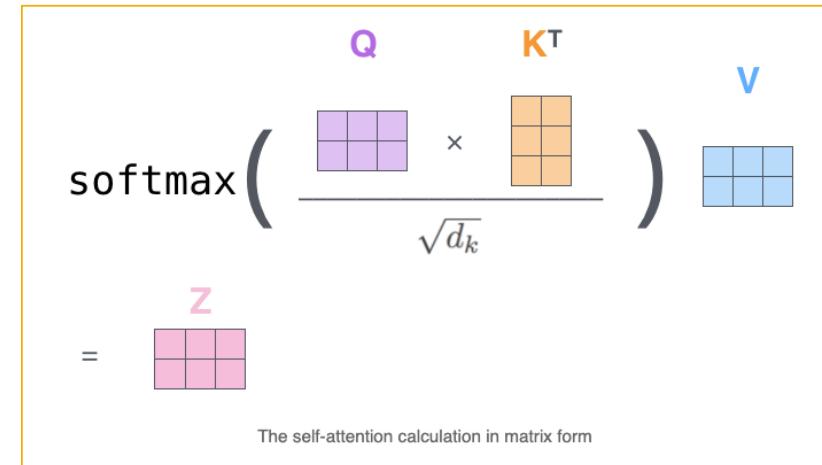
Normalise

- Step 3: Divide all of the scores by 8
 - the square root of the dimension of the key vectors which is 64 here
 - this is the default and seems to lead to more stable gradients
- Step 4: Apply a softmax to the scores
 - this results in them all being positive
 - and summing to 1
 - so can be thought of as a probability / proportion – “what proportion of my attention should I pay to word j?”

Weight sum of the value vectors

- Step 5: For each word w_i , multiply the value vector (V_j) of each other word w_j by its softmax score with w_i
- Step 6: Sum to produce output embedding (at this layer) for w_i

$$Z_i = \sum_j softmax_score(w_i, w_j) \times V_j$$



Self-attention summary

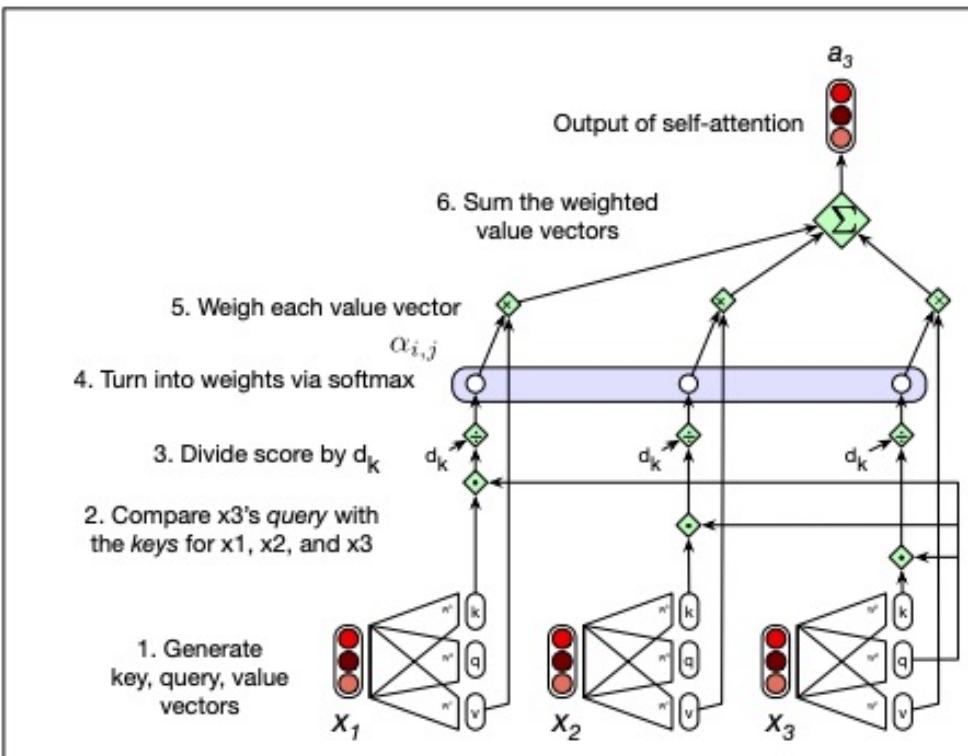


Figure 10.3 Calculating the value of a_3 , the third element of a sequence using causal (left-to-right) self-attention.

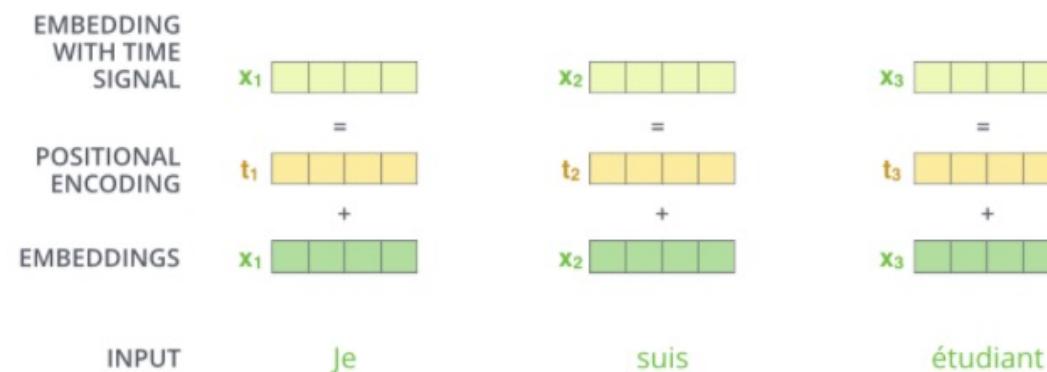
- Image taken from Chapter 10, Jurafsky and Martin
- This is actually for uni-directional self-attention
- However, it would be the same for the a_3 in the bidirectional case assuming an input sequence which is 3 tokens long!

Multi-head attention

- Paying attention to multiple things at the same time!
- Multiple “representation subspaces”
- Multiple sets of Query/Key/Value weight matrices
- Typical transformer uses 8 attention heads per self-attention layer
 - that’s 8 sets of Q,K,V matrices for each encoder and 16 for each decoder
- All randomly initialised
- Embeddings produced by multiple heads are concatenated and then multiplied by an additional weights matrix before passing to the feed-forward layer

Positional Encoding

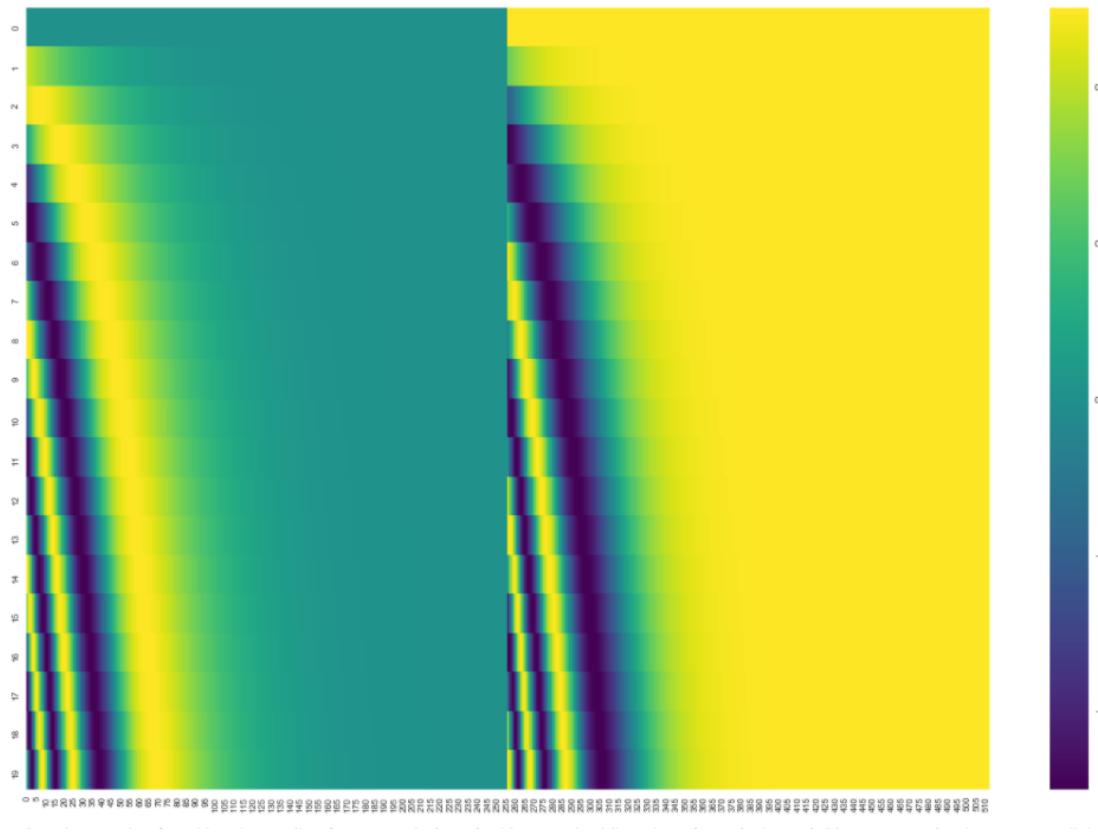
- How does the model account for word order?
- Adds a positional encoding vector to each input embedding
- Positional encoding follows a fixed pattern
 - enables the model to determine the position of each word and distance between them



Absolute position

- Start with randomly initialized embeddings corresponding to each possible input position up to some maximum length
- We have embeddings for
 - words e.g., student
 - Positions e.g., position 3
- Positional embeddings are learned / updated during training
- Alternatives to absolute position include
 - Static function maps integer inputs to real-valued vectors
Original transformers used combinations of sines and cosines
 - relative position

Positional encoding visualisation

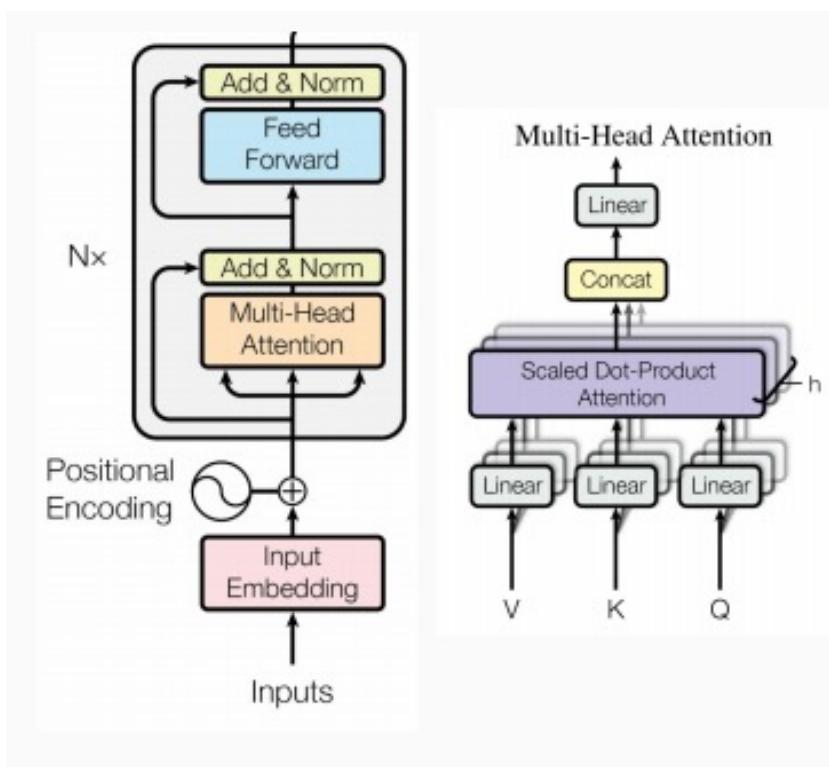


BERT (Devlin et al. 2019)

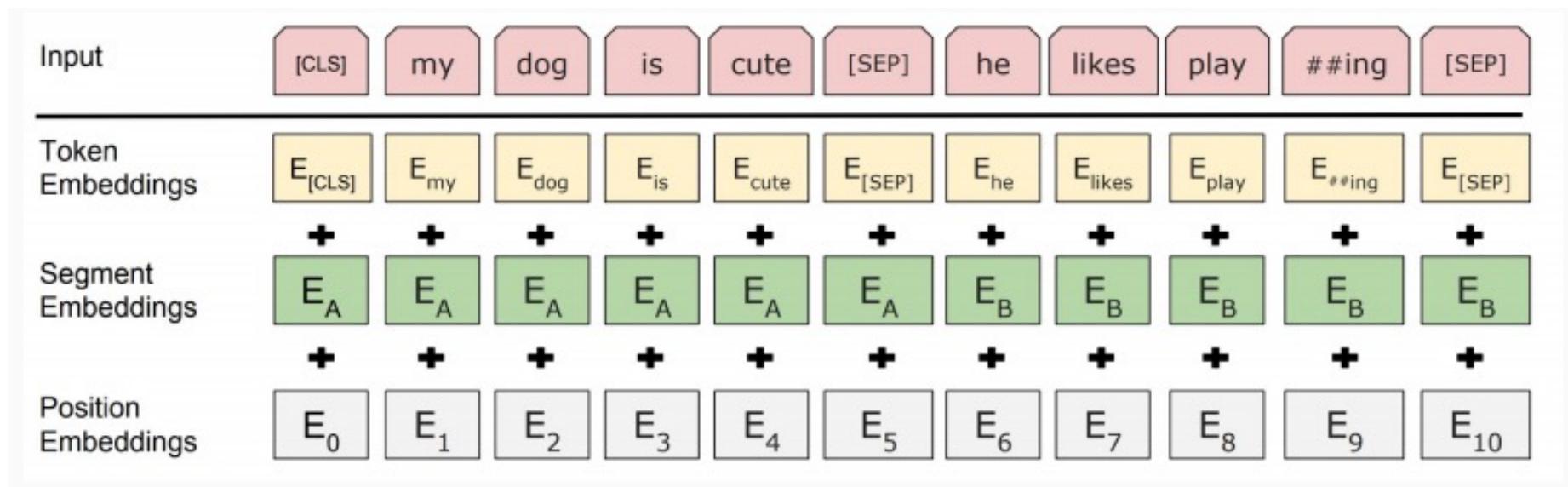
- Bidirectional Encoder Representations from Transformers
 - just uses the encoder portion of the transformer architecture
 - uses left and right context of a target word to build representation
- Pre-trained on general language modelling tasks
 - masked language modelling task
 - next sentence prediction
- fine-tuned on task-specific training data
- Pre-trained models available from Google:
 - <https://github.com/google-research/bert>
- And from huggingface!

Encoder Representation

- Multi-headed self-attention
 - models context
- Feed-forward layers
 - non-linear hierarchical features
- Layer norm and residuals
 - makes training deep networks healthy
- Positional embeddings
 - learn relative positioning



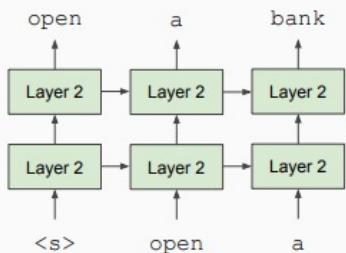
Input Representation



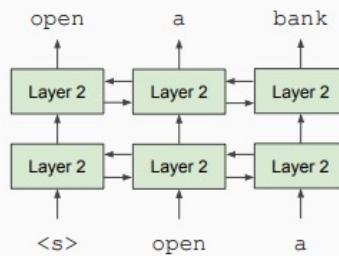
- Each token is sum of three embeddings
- 30,000 WordPiece vocabulary → morphology → better representations for rare words
- sentences are separated by a special “SEP” token
- a special “CLS” token is added at the beginning of the chunk
 - often used in classification

Unidirectional vs Bidirectional models

Unidirectional context
Build representation incrementally



Bidirectional context
Words can “see themselves”

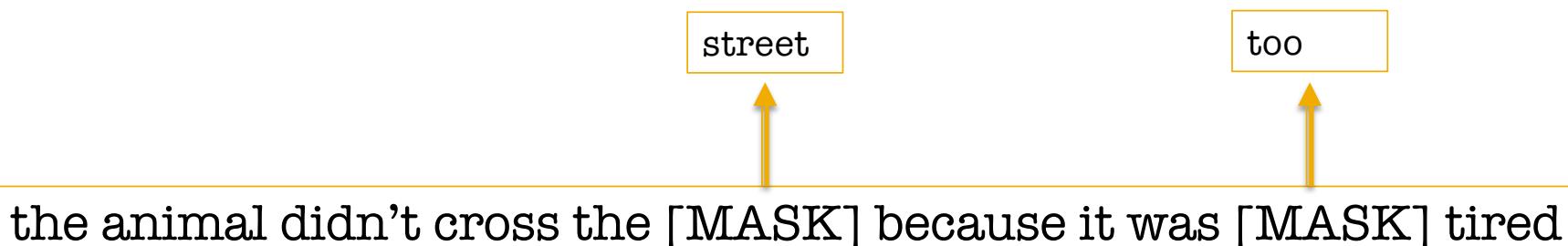


from <https://nlp.stanford.edu/seminar/details/jdevlin.pdf>

- Most language models trained to predict the next word in sequence
- Known as **auto-regressive** or **unidirectional**
- BERT uses the whole of the sentence to predict missing or masked word
- masked language model is **bidirectional**

Masked Language Model

- Mask out k% of the input words, and predict the masked words
 - if k is too small → expensive to train
 - if k is too large → not enough context
 - Google use k=15



Next Sentence Prediction

- To learn relationships between sentences, predict whether sentence B is actual sentence that follows sentence A or is a random sentence

sentence A = The man went to
the store

sentence B = He bought a
gallon of milk

Label = IsNextSentence

sentence A = The man went to
the store

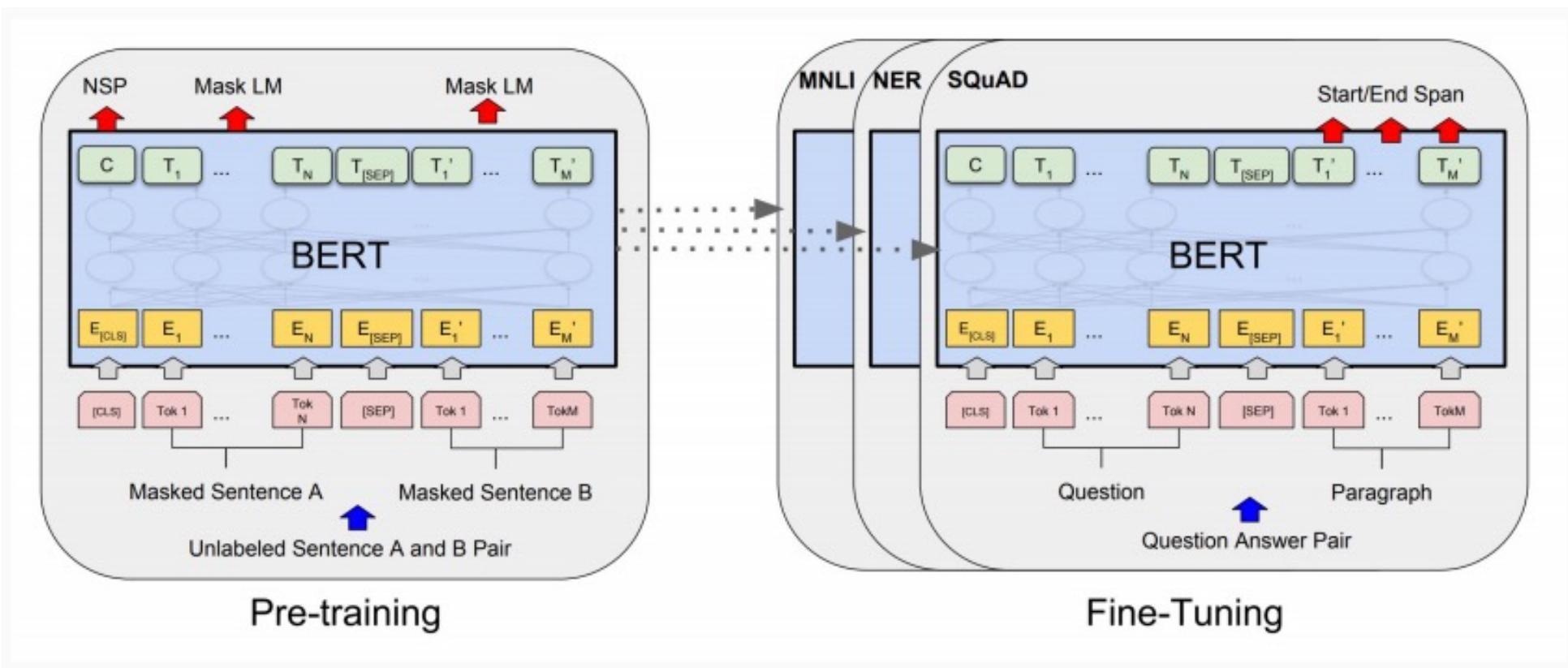
sentence B = Penguins are
flightless

Label = NotNextSentence

Pre-trained model details

- Data: Wikipedia (2.5B words) + BookCorpus (800M words)
- Batch Size: 131,072 words (1024 sequences * 128 length or 256 sequences * 512 length)
- Training Time: 1M steps (~40 epochs)
- Optimizer: AdamW, 1e-4 learning rate, linear decay
- BERT-Base: 12-layer, 768-hidden, 12-head
- BERT-Large: 24-layer, 1024-hidden, 16-head
- Trained on 4x4 or 8x8 TPU slice for 4 days

Fine-tuning Procedure



GLUE Results

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Average
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
BERT _{BASE}	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
BERT _{LARGE}	86.7/85.9	72.1	91.1	94.9	60.5	86.5	89.3	70.1	81.9

MultiNLI

Premise: Hills and mountains are especially sanctified in Jainism.

Hypothesis: Jainism hates nature.

Label: Contradiction

CoLa

Sentence: The wagon rumbled down the road.

Label: Acceptable

Sentence: The car honked down the road.

Label: Unacceptable

Further reading

- Devlin et al. (2019): Pre-training of Deep Bidirectional Transformers for Language Understanding in NAACL 2019, <https://www.aclweb.org/anthology/N19-1423/>
- Peters et al. (2018): Deep contextualised word representations in Proceedings of NAACL 2018 <https://arxiv.org/pdf/1802.05365.pdf>
- Peters et al. (2018): Dissecting Contextual Word Embeddings: Architecture and Representation in Proceedings of EMNLP 2018
<https://www.aclweb.org/anthology/D18-1179.pdf>
- Reimers et al. (2019):
- Vashwani et al. (2017): Attention is all you need in Proceedings of NIPS 2017
- Yang et al. (2020) :
(<https://arxiv.org/pdf/2004.12297.pdf>)

AdvNLE Lecture 9

Transfer Learning with Pretrained Large Language Models

Dr Julie Weeds, Spring 2024



Previously ...

- Paraphrase and semantic matching
 - applications
- Simple text matching
- Distributional representations of meaning
- Composition
- Contextualised word embeddings
 - ELMo
 - transformers
 - BERT

The Distributional Hypothesis: for words

"Words that occur in the same contexts tend to have similar meanings."

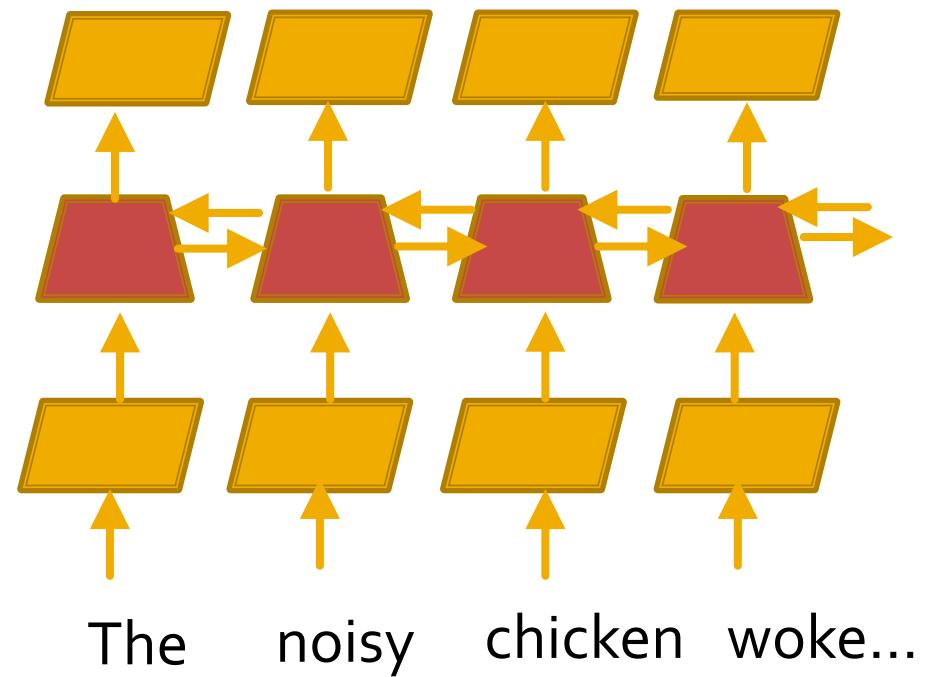
Harris, 1954

Words represented by real-valued vectors, where dimensions explicitly or implicitly represent contexts in which the word occurs:

chicken	0.3	0.1	0.8	0.5
beef	0.05	0.9	0.75	0.01

Contextualised Word Embeddings

- ELMo (Peters et al. 2018)
- BERT (Devlin et al. 2018/2019)
- combine word embeddings and bi-directional language models to provide (deep) contextualised word embeddings

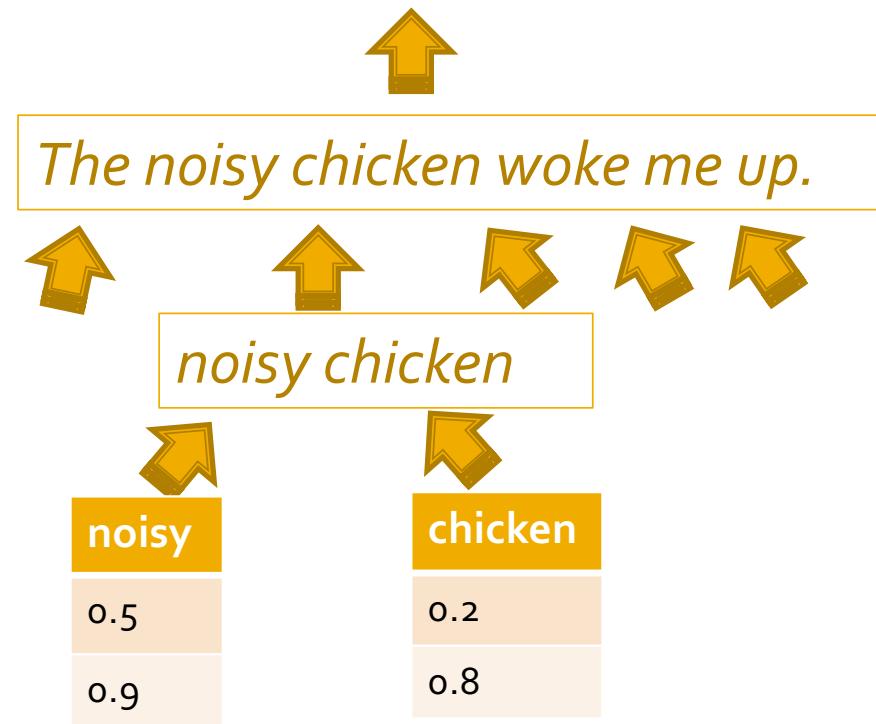


Beyond words

We have a model for **words**, how does this scale up to

- phrases
- sentences
- utterances
- documents
- discourse

➤ *Who woke me up?*
➤ *It was the noisy chicken.*



Principle of compositionality

“The meaning of a complex expression is determined by the meaning of its constituent parts and the rules used to combine them.”

[Boole; Frege and others]

Can we take distributional word representations and combine them to make distributional sentence representations?

The distributional hypothesis: for sentences?

"Words that occur in the same contexts tend to have similar meanings."



"Phrases that occur in the same contexts tend to have similar meanings."



"Sentences that occur in the same contexts tend to have similar meanings."

Sentential contexts

- Contexts of sentences are other sentences
- But most composition methods assume that the possible contexts of a sentence is some function of the possible contexts of all of the constituent words
- Word representations are often **simply added or averaged** to get *composed* sentential representations (Mitchell and Lapata, 2010)

SBert (Reimers and Gurevych 2019)

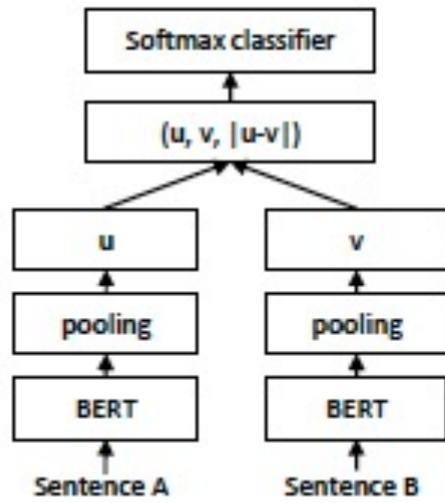


Figure 1: SBERT architecture with classification objective function, e.g., for fine-tuning on SNLI dataset. The two BERT networks have tied weights (siamese network structure).

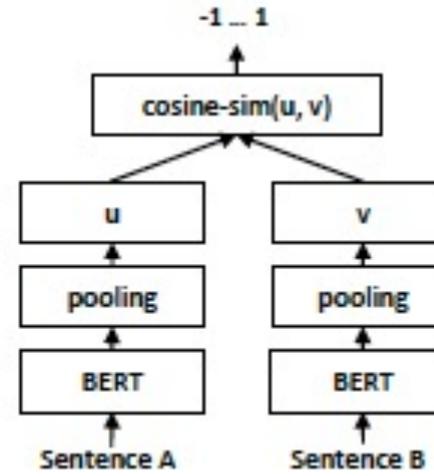


Figure 2: SBERT architecture at inference, for example, to compute similarity scores. This architecture is also used with the regression objective function.

Today

- Transfer learning
- Fine-tuning BERT-based models for
 - text classification
 - sequence labelling
- The BERT family
- More distant relatives
- Schick and Schütze (2021)

Transfer Learning through Fine-tuning

- **Transfer learning:** acquiring knowledge from one task or domain and then applying (transferring) it to solve a new task
- BERT-based models acquire knowledge about language through **pre-training** (masked language model prediction and next sentence prediction) on large unannotated corpora
- **Fine-tuning** is the process of transferring this knowledge to a specific task

Fine-tuning

- Fine-tuning **uses labelled data** from the application to train **additional application-specific parameters**
- Application-specific parameters could be a single layer neural network on top of the BERT architecture
- Fine-tuning might
 - freeze the pre-trained language model parameters
 - allow updates to be made to some or all of the pre-trained language model parameters

Text classification with BERT

- E.g., Is a review **positive** or **negative** in sentiment?
- [CLS] token is used to stand for the entire sequence
- [CLS] is part of the vocabulary and must be pre-pended to all input sequences during pre-training and fine-tuning
- [CLS] is the input to a classifier head, e.g., logistic regression, which makes relevant decision

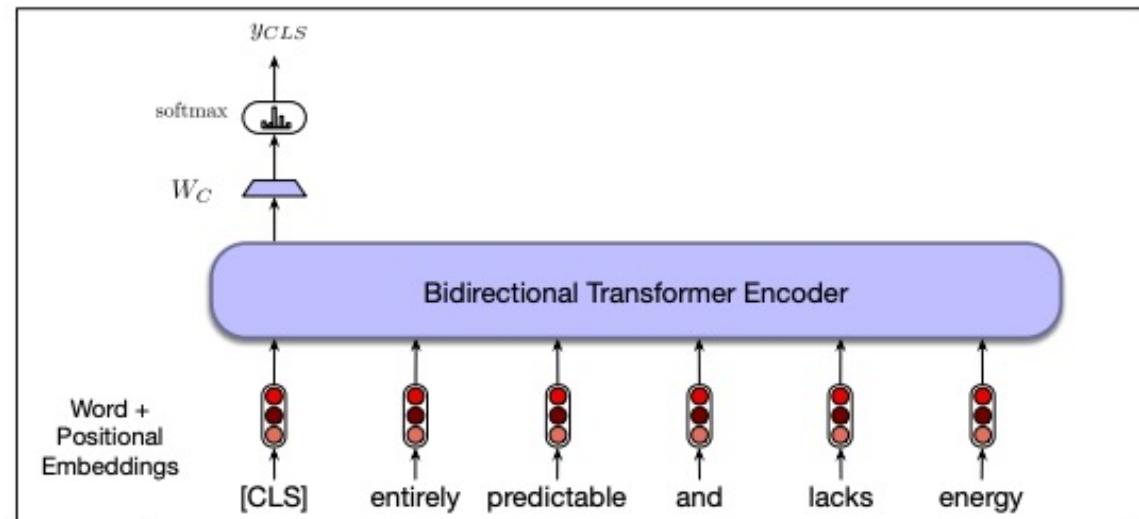


Figure 11.8 Sequence classification with a bidirectional transformer encoder. The output vector for the [CLS] token serves as input to a simple classifier.

Figure taken from Jurafsky and Martin (Section 11.3.1)

Text classification with BERT (2)

- learn a set of weights W_C which maps output vector for [CLS] to a set of scores for the possible sentiment classes
- Pass input text through pre-trained language model to generate y_{CLS} , multiply it by W_C and then pass this through softmax

$$y = \text{softmax}(W_C y_{CLS})$$

- Fine-tuning requires input sequences labelled with appropriate class
- Cross-entropy loss between softmax output and correct answer drives backpropagation
- Weights can be updated just for W_C (pre-trained language model is frozen) or in the pre-trained language model as well

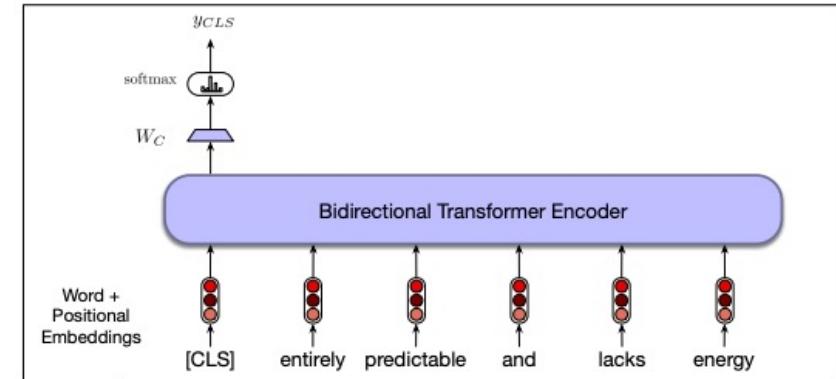


Figure 11.8 Sequence classification with a bidirectional transformer encoder. The output vector for the [CLS] token serves as input to a simple classifier.

In practice ... with Huggingface transformers library

- `transformers.BertForSequenceClassification`
 - BERT transformer with a sequence classification / regression head on top (linear layer on top of **pooled output**)
 - inherits from `PreTrainedModel`

```
: from transformers import BertTokenizerFast, BertForSequenceClassification
from transformers import Trainer, TrainingArguments

# check text classification models here: https://huggingface.co/models?filter=text-classification
model_name = "bert-base-uncased"

# load the tokenizer
tokenizer = BertTokenizerFast.from_pretrained(model_name, do_lower_case=True)

# load the model and pass to CUDA
model = BertForSequenceClassification.from_pretrained(model_name, num_labels=len(target_names)).to("cuda")
```

Downloading: 100%

466k/466k [00:00<00:00, 756kB/s]

Training the Sequence Classification Model

- See <https://www.thepythoncode.com/article/fine-tuning-bert-using-huggingface-transformers-python>
- Need to load, tokenize and encode the inputs
 - training and validation set
- Set up training_args e.g., number of epochs, batch size
- Set up metrics for evaluation e.g., accuracy

```
: trainer = Trainer(  
    model=model,  
    args=training_args,  
    train_dataset=train_dataset,  
    eval_dataset=valid_dataset,  
    compute_metrics=compute_metrics,  
)  
  
# train the model  
trainer.train()
```

Training arguments

```
args = TrainingArguments(  
    f"{model_name}-finetuned-{task}",  
    evaluation_strategy = "epoch",  
    save_strategy = "epoch",  
    learning_rate=2e-5,  
    per_device_train_batch_size=batch_size,  
    per_device_eval_batch_size=batch_size,  
    num_train_epochs=5,  
    weight_decay=0.01,  
    load_best_model_at_end=True,  
    metric_for_best_model=metric_name,  
    push_to_hub=True,  
)
```

e.g., "pearson", "accuracy", "f1" or "spearmanr"

The GLUE Benchmark tasks

The GLUE Benchmark is a group of nine classification tasks on sentences or pairs of sentences which are:

- [CoLA](#) (Corpus of Linguistic Acceptability) Determine if a sentence is grammatically correct or not. is a dataset containing sentences labeled grammatically correct or not.
- [MNLI](#) (Multi-Genre Natural Language Inference) Determine if a sentence entails, contradicts or is unrelated to a given hypothesis. (This dataset has two versions, one with the validation and test set coming from the same distribution, another called mismatched where the validation and test use out-of-domain data.)
- [MRPC](#) (Microsoft Research Paraphrase Corpus) Determine if two sentences are paraphrases from one another or not.
- [QNLI](#) (Question-answering Natural Language Inference) Determine if the answer to a question is in the second sentence or not. (This dataset is built from the SQuAD dataset.)
- [QQP](#) (Quora Question Pairs2) Determine if two questions are semantically equivalent or not.
- [RTE](#) (Recognizing Textual Entailment) Determine if a sentence entails a given hypothesis or not.
- [SST-2](#) (Stanford Sentiment Treebank) Determine if the sentence has a positive or negative sentiment.
- [STS-B](#) (Semantic Textual Similarity Benchmark) Determine the similarity of two sentences with a score from 1 to 5.
- [WNLI](#) (Winograd Natural Language Inference) Determine if a sentence with an anonymous pronoun and a sentence with this pronoun replaced are entailed or not. (This dataset is built from the Winograd Schema Challenge dataset.)

See https://colab.research.google.com/github/huggingface/notebooks/blob/main/examples/text_classification.ipynb

Compute metrics

```
In [ ]: import numpy as np
        from datasets import load_metric
        metric = load_metric("glue","sst-2")

        fake_preds=np.random.randint(0,2,size=(64,))
        fake_labels=np.random.randint(0,2,size=(64,))
        metric.compute(predictions=fake_preds,references=fake_labels)
```

```
In [ ]: def compute_metrics(eval_pred):
            predictions,labels=eval_pred
            # more code here if needed to pre-process predictions /labels into lists e.g., select appropriate column
            return metric.compute(predictions=predictions,references=labels)
```

Using the Sequence Classification Model

- Evaluate it using the trainer's evaluate() method
- Save the associated model and tokenizer for future use
- Make predictions on unseen data

```
: def get_prediction(text):
    # prepare our text into tokenized sequence
    inputs = tokenizer(text, padding=True, truncation=True, max_length=max_length, return_tensors="pt").to("cuda")
    # perform inference to our model
    outputs = model(**inputs)
    # get output probabilities by doing softmax
    probs = outputs[0].softmax(1)
    # executing argmax function to get the candidate label
    return target_names[probs.argmax()]

print(get_prediction(text))
```

Drawbacks

- BertForSequenceClassification is quite **opaque**
 - Dig through lots of code to find out the architecture of the classification head
 - What is the “pooled output”?
 - It is the CLS representation but this is not clear in the documentation and you might want to use a different pooling strategy
- You need to be signed up to huggingface-hub to be able to use all of the functionality (e.g., load_metric) which increases the burden on getting started
- **ALTERNATIVE:** build own classification head on top of pre-trained BERT model
 - See lab

Code for BERTclassifier (from lab)

```
]: #now we need to put a simple classification layer on top of BERT

from torch import nn
from transformers import BertModel

class BertClassifier(nn.Module):

    def __init__(self,dropout=0.5,num_classes=2):
        super(BertClassifier,self).__init__()

        self.bert=BertModel.from_pretrained('bert-base-uncased')
        self.dropout=nn.Dropout(dropout)
        self.linear=nn.Linear(768,num_classes)
        self.relu=nn.ReLU()

    def forward(self,input_id,mask):

        last_hidden_layer,pooled_output = self.bert(input_ids=input_id,attention_mask=mask,return_dict=False)
        dropout_output=self.dropout(pooled_output)
        linear_output=self.linear(dropout_output)
        final_layer=self.relu(linear_output)

    return final_layer
```

Freezing layers

```
In [ ]: model=BERTClassifier(num_classes=len(labels.keys()))

#this will freeze the pre-trained BERT model and just make the classification head trainable
#can speed things up and avoid "catastrophic forgetting" / overfitting on task-specific data
model.bert.requires_grad_(False)
```

Pairwise Sequence Classification

- Do two sentences entail or contradict each other?
 - “I’m confused”
 - “It is not completely clear to me”
- Concatenate the sentences, separated by the [SEP] token
- Proceed as for single sequence classification
- OR use something like SBERT. What’s the advantages of each approach?

Sequence Labelling with BERT

- E.g., POS tagging or NER
- Simplest approach is to pass each output from BERT to a simple classifier
- Or pass it to a CRF which considers tag-level transitions as well
- Complications can arise from subword tokenizations
 - simplest approach is to use the first subword token from each word

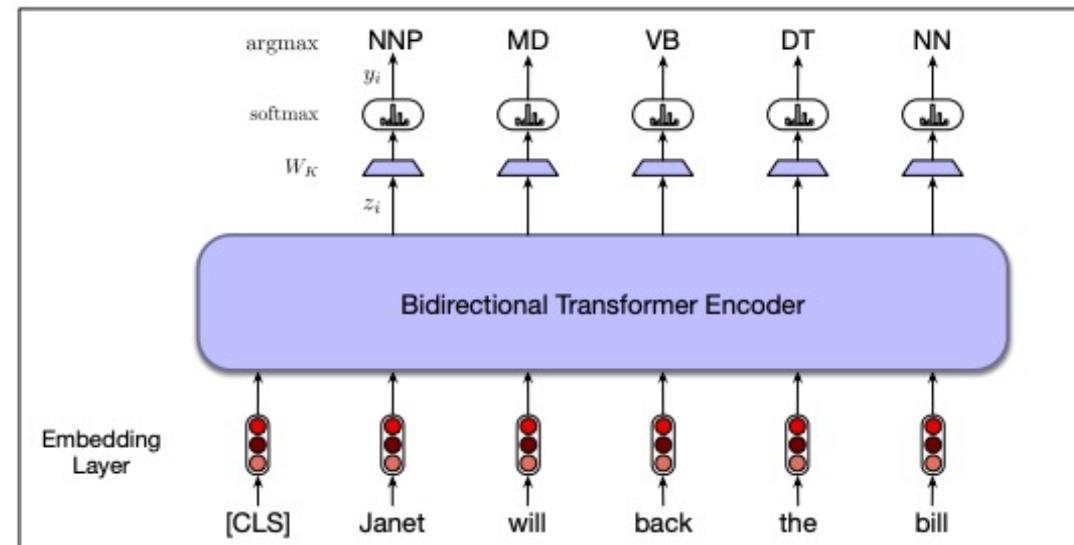


Figure 11.9 Sequence labeling for part-of-speech tagging with a bidirectional transformer encoder. The output vector for each input token is passed to a simple k -way classifier.

Sequence Labelling ... with Huggingface Transformers

- https://huggingface.co/docs/transformers/tasks/token_classification

```
>>> from transformers import AutoModelForTokenClassification, TrainingArguments, Trainer  
  
>>> model = AutoModelForTokenClassification.from_pretrained("distilbert-base-uncased", num_labels=2)
```

BERT Family

- RoBERTa
- ALBERT
- DistilBERT
- SciBERT
- MBert
- and more
 - StructBERT
 - AraBERT
 - DeBERTa

RoBERTa

- Alternative to BERT from Facebook (Liu et al (2019)): a robustly optimized **BERT** pretraining approach
- Pre-trained on even larger corpus
- Ditches next sentence prediction and just uses MLM for pre-training
 - dynamic masking (different masks each time sentence is used)
 - Full-sentences without NSP loss
 - large mini-batches (8K sequences)
 - larger byte-level BPE (50K subword units, 15M-20M additional parameters)

ALBERT

- A Lite BERT for Self-supervised learning of language representations (Lan et al. 2019)
- Incorporates 2 parameter reduction techniques with a view to making pre-trained models more scalable
 - factorization of the vocabulary embedding matrix
 - cross-layer parameter sharing
- Performance also improved with sentence order prediction task
- 18x fewer parameters than BERT and trained about 1.7x faster

DistilBERT

- a distilled version of BERT: smaller, faster, cheaper and lighter
(Sanh et al. 2019)
- knowledge distillation during pre-training
- a compact model (the student) is trained to reproduce the behaviour of a larger model (the teacher) or ensemble of models
 - reduces size by 50%
 - retains 97% of language understanding
 - 60% faster

SciBERT

- SciBERT: a pretrained language model for scientific text
(Beltagy et al. 2019)
- basically BERT pre-trained on large multi-domain corpus of scientific publications
- Improved in-domain results

MBert

- Multilingual BERT, model released by Devlin et al. at the same time as BERT
- See Pires et al (2019): How multilingual is Multilingual BERT
- trained on text from 104 languages
- does not contain any explicit translation information
- Intuition is that embeddings for words in different languages will be embedded in the same space due to commonalities in the vocabularies for the languages (e.g., names, numbers and other shared vocab)

More Distant Relatives of BERT

- Other Pretrained Large Language Models, generally still based on transformers e.g.,
 - GPT,
 - Turing-NLG,
 - T5,
 - XLNet,
 - Electra

Generative Pre-trained Transformer 3 (GPT-3)

- Brown et al. 2020: *Language Models are Few Shot Learners*
- autoregressive language model
 - this means it predicts the next token rather than masked tokens
- variable length inputs but uni-directional in nature
- largest non-sparse language model: 175 billion parameters,
10x bigger than competitors
- Trained on Common Crawl, WebText2, Books1, Books2 and Wikipedia
- No fine-tuning. Used to generate answers using a few-shot training / prompting paradigm

In-context learning / Prompting

- Zero-shot: model predicts the answer given only a natural language description of the task. No parameter updates.
 - “Translate English to French: cheese => _____”
- One-shot: model sees the description and one example of the task. No parameter updates.
 - “Translate English to French: sea otter => loutre de mer, cheese => _____”
- Few-shot: model sees the description and a few examples of the task. No parameter updates
 - “Translate English to French: sea otter => loutre de mer, peppermint => menth poivree, plush girafe => girafe peluche, cheese => _____”

Text-to-Text Transfer Transformer (T5)

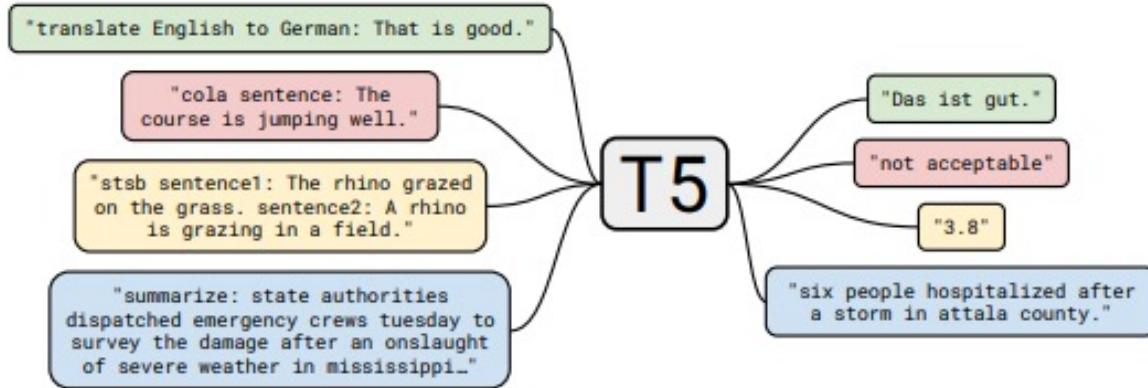
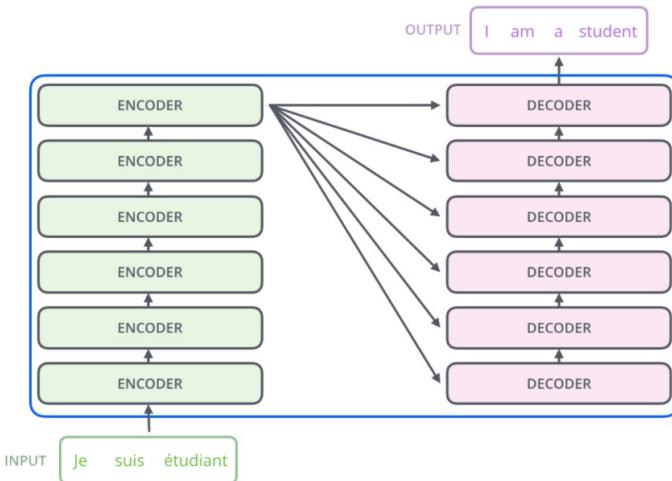


Figure 1: A diagram of our text-to-text framework. Every task we consider—including translation, question answering, and classification—is cast as feeding our model text as input and training it to generate some target text. This allows us to use the same model, loss function, hyperparameters, etc. across our diverse set of tasks. It also provides a standard testbed for the methods included in our empirical survey. “T5” refers to our model, which we dub the “Text-to-Text Transfer Transformer”.

- Raffel et al.
2020:
Exploring the
Limits of
Transfer
Learning with
a Unified Text-
to-Text
Transformer

T5 Architecture

- Encoder-decoder architecture closely following the original proposal by Vaswani et al. 2017
- Used for generation as well as encoding
- Every task is structured in such a way that the answer (be it a label or a translation) should be generated by the model i.e., prediction is done in an autoregressive way



T5 also uses MLM

- Pre-training involves unsupervised objectives which are similar to the MLM of BERT and “word dropout” regularization technique (Bowman et al. 2015)

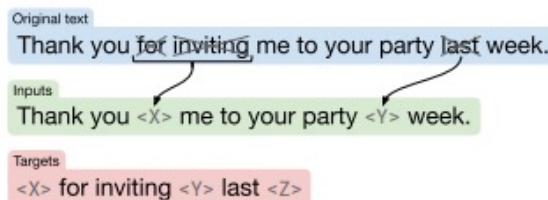


Figure 2: Schematic of the objective we use in our baseline model. In this example, we process the sentence “Thank you for inviting me to your party last week.” The words “for”, “inviting” and “last” (marked with an \times) are randomly chosen for corruption. Each consecutive span of corrupted tokens is replaced by a sentinel token (shown as $\langle X \rangle$ and $\langle Y \rangle$) that is unique over the example. Since “for” and “inviting” occur consecutively, they are replaced by a single sentinel $\langle X \rangle$. The output sequence then consists of the dropped-out spans, delimited by the sentinel tokens used to replace them in the input plus a final sentinel token $\langle Z \rangle$.

T5 Fine-Tuning

- Fine-tuned in a **supervised** fashion on a large number of tasks
- “unified” in the sense that all tasks can benefit from learning on other tasks
- To make best use of a pre-trained and fine-tuned T5 model, you need to know the prompt format which most closely matches your task e.g.,
 - input =“summarize: text.”
 - output = “summary”

Next time

- ChatGPT
- Generative applications and LLMs
 - Machine translation
 - Summarisation
- Trustworthy and responsible LLMs / AI
- Environmental impact of LLMs / AI

Seminar Reading

- Chick and Schütze (2021): Exploiting Cloze Questions for Few Shot Text Classification and Natural Language Inference

Further reading

- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. in *NAACL 2019* <https://www.aclweb.org/anthology/N19-1423/>
- Erk, K. and Padó, S. (2008) A structured vector space model for word meaning in context. In *EMNLP 2008*
- Levy, O., and Goldberg, Y. (2014) Dependency-based word embeddings. *In ACL 2014*
- Marcheggiani, D. and Titov, I. (2017). Encoding Sentences with Graph Convolutional Networks for Semantic Role Labelling. In *Proceedings of EMNLP 2017*. <https://www.aclweb.org/anthology/D17-1159/>
- Mitchell, J. and Lapata, M. (2010). Composition in distributional models of semantics. In *Cognitive Science*, 34(8):1388–1429.
- Padó, S. and Lapata, M., (2007). Dependency-based construction of semantic space models. In *Computational Linguistics*, 33(2):161–199,
- Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. In *NAACL 2018*
- Weir, D., Weeds, J., Reffin, J. and Kober, T. (2016). Aligning packed dependency trees: a theory of composition for distributional semantics. In *Computational Linguistics*.

AdvNLE Seminar 10

Using LLMs and the Future

Dr Julie Weeds, Spring 2024



Previously ...

- Distributional representations of meaning
- Neural Language modelling
- Contextualised word embeddings
- Large language models
 - BERT (Bi-directional Encoder Representations from Transformers)
 - Pre-training
 - Fine-tuning

Today

- More distant relatives of BERT
 - GPT and ChatGPT
- Generative applications with LLMs
 - MT
 - Summarization
 - Question-answering
- Trustworthy and responsible AI
- Environmental impact of LLMs
- Revision

More Distant Relatives of BERT

- Other Pretrained Large Language Models, generally still based on transformers e.g.,
 - GPT (GPT-2, GPT-3, ChatGPT GPT-4 ...)
 - Turing-NLG,
 - XLNet,
 - Electra
 - Dolly
 - NeMo
 - BLOOM
 - LLaMa
 - PaLM2

Generative Pre-trained Transformer 3 (GPT-3)

- Brown et al. 2020: *Language Models are Few Shot Learners*
- autoregressive language model
 - this means it predicts the next token rather than masked tokens
- variable length inputs but uni-directional in nature
- largest non-sparse language model: 175 billion parameters, 10x bigger than competitors
- Trained on Common Crawl, WebText2, Books1, Books2 and Wikipedia
- No fine-tuning. Used to generate answers using a few-shot training / prompting paradigm

Generating responses

- Take a ***prompt*** and use a language model to predict what comes next or what fills in the gaps.
- Working out what is the best prompt strategy (how to convert the user utterance into a prompt for the LLM) is known as **prompt engineering**

User: *When did Turner paint the picture The Lighthouse?*

Prompt: *When did Turner paint the picture The Lighthouse? _____*

Prompt: *Turner painted the picture The Lighthouse in _____.*

- Find the word(s) that fills in the blank(s) with the highest probability according to the large language model (pretrained on the very large corpus)

What could possibly go wrong?

What could possibly go wrong?

- Large Language Models are trained to give plausible / believable answers based on large text corpora
- If 2 (or more words) often occur together in the training data, they may lead to a higher probability response even if factually incorrect in the specific context –
 - This is referred to as “*hallucination*” where a large language model adds information which has no basis in say the text being summarized or translated
- Text corpora may contain factually incorrect documents (e.g., fiction) and / or text which exhibits *biases* so these can be replicated in generated texts

Common hallucinations

- Based on your knowledge of LLMs, what do you think the most common types of errors and hallucinations are likely to be?

Genuine example (Feb 24)

J You

Where in Rotherfield is the Six Bells pub?



ChatGPT

The Six Bells pub in Rotherfield is located at Church Street, Rotherfield, Crowborough, East Sussex, TN6 3LQ, United Kingdom.



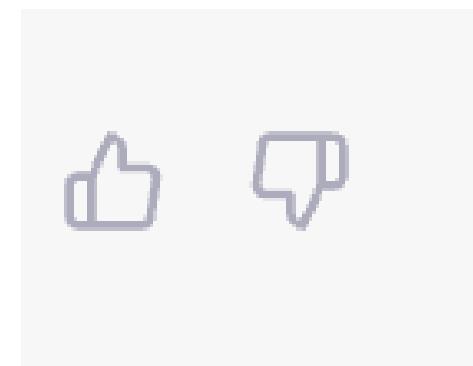
- There is no Six Bells pub in Rotherfield
- There is no Church Street in Rotherfield (only a Church Road)
- TN6 3LQ is Court Meadow, Rotherfield

ChatGPT (OpenAI, 2023)

- InstructGPT models
 - Trained with humans in the loop
 - Deployed as default language models on OpenAI's API
 - Better at following user intentions than GPT-3
 - More truthful and less toxic
 - Uses “alignment” technique
 - Reinforcement learning from human feedback (RLHF)

Reinforcement Learning from Human Feedback (RLHF)

- Customers submit prompts to the API
- Human labelers provide demonstrations of desired model behaviours
 - This data is used to fine-tune GPT-3
- Human labelers then rank different model outputs
 - This data is used to train a **reward model** to predict which output labellers would prefer
- GPT-3 has an additional input known as its “policy”
 - this is fine-tuned to maximise the reward using proximal policy optimization (PPO) (Schulman et al. 2017)



Training ChatGPT (Ouyang et al. 2022)

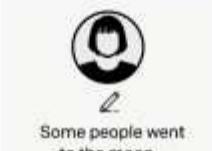
Step 1

Collect demonstration data, and train a supervised policy.

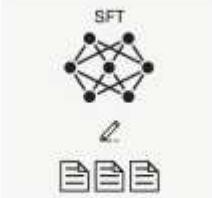
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



This data is used to fine-tune GPT-3 with supervised learning.



Step 2

Collect comparison data, and train a reward model.

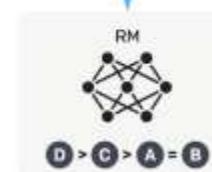
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



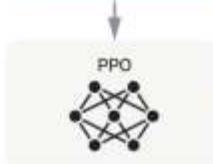
Step 3

Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.



The policy generates an output.



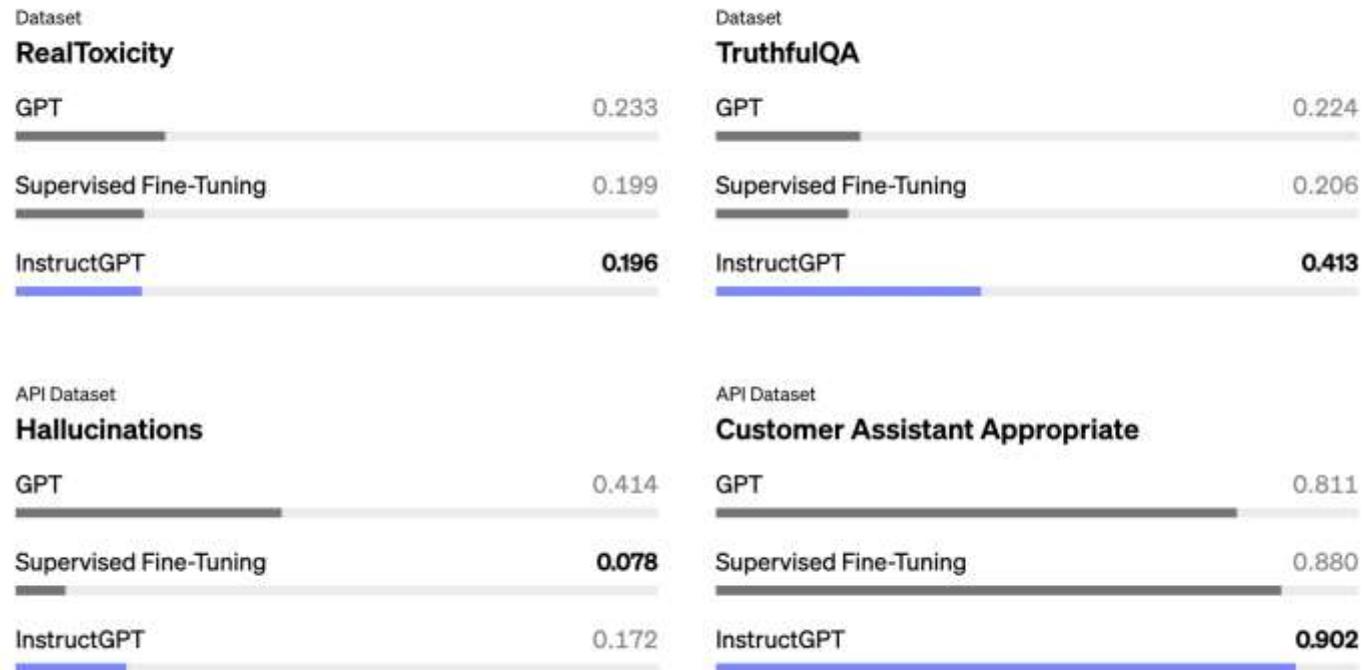
The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.

r_k

Does it work?



According to OpenAI

- Less toxic
- More truthful
- Less hallucinations
- More appropriate

Yes, but ...?

Weaknesses

ChatGPT is a pre-trained large language model

- It doesn't know or understand anything
- It doesn't look things up or carry out inferences
- Words which mean "similar" things are easy for it to mix up **especially numbers and dates**
- It has been trained on a very large corpus which is biased towards
 - a certain time period, particular geographic locations, culture and ways of thinking
- Fine-tuning process could be subverted

Using Generative Models

- Nearly all NLP applications can be posed as a prompt to a generative model:
 - Translate the following text from English to French: The cat sat on the mat
 - Summarise the following information in 3 sentences:
 - Who played Hans Solo in Star Wars?
 - What is the sentiment of the following review:
- **Encode** the prompt
- **Decode** and generate a response

Document – level MT (Wang et al. 2023)

Translate this document from English to Chinese:

The screenshot shows a document-level machine translation interface. On the left, there's a user instruction: "Translate this document from English to Chinese:". Below it is the original English text about Audi's history, with various discourse elements highlighted by red boxes and lines. These annotations include mentions of August Horch, Audi's early models, its transition to racing, and its acquisition by Volkswagen. On the right, the generated Chinese translation is shown, which appears to have correctly understood and translated these complex discourse references.

奥迪是一家生产豪华汽车和SUV的汽车制造商。该公司起源于德国。它由奥古斯特·霍希 (August Horch) 于1910年创立。霍希此前曾创立过另一家公司，其车型相当受欢迎。奥迪起初生产四缸车型。到1914年，霍希的新车开始参加比赛并获胜。奥古斯特·霍希于1920年离开奥迪公司，担任德国机动车行业联合会的行业代表。目前，奥迪是大众集团的子公司，生产质量卓越的汽车。

Figure 1: An example of translating a document-level text from English to Chinese using GPT-4 (Date: 2023.03.17). We highlight the discourse phenomena using figures and lines, which are invisible to GPT-4.

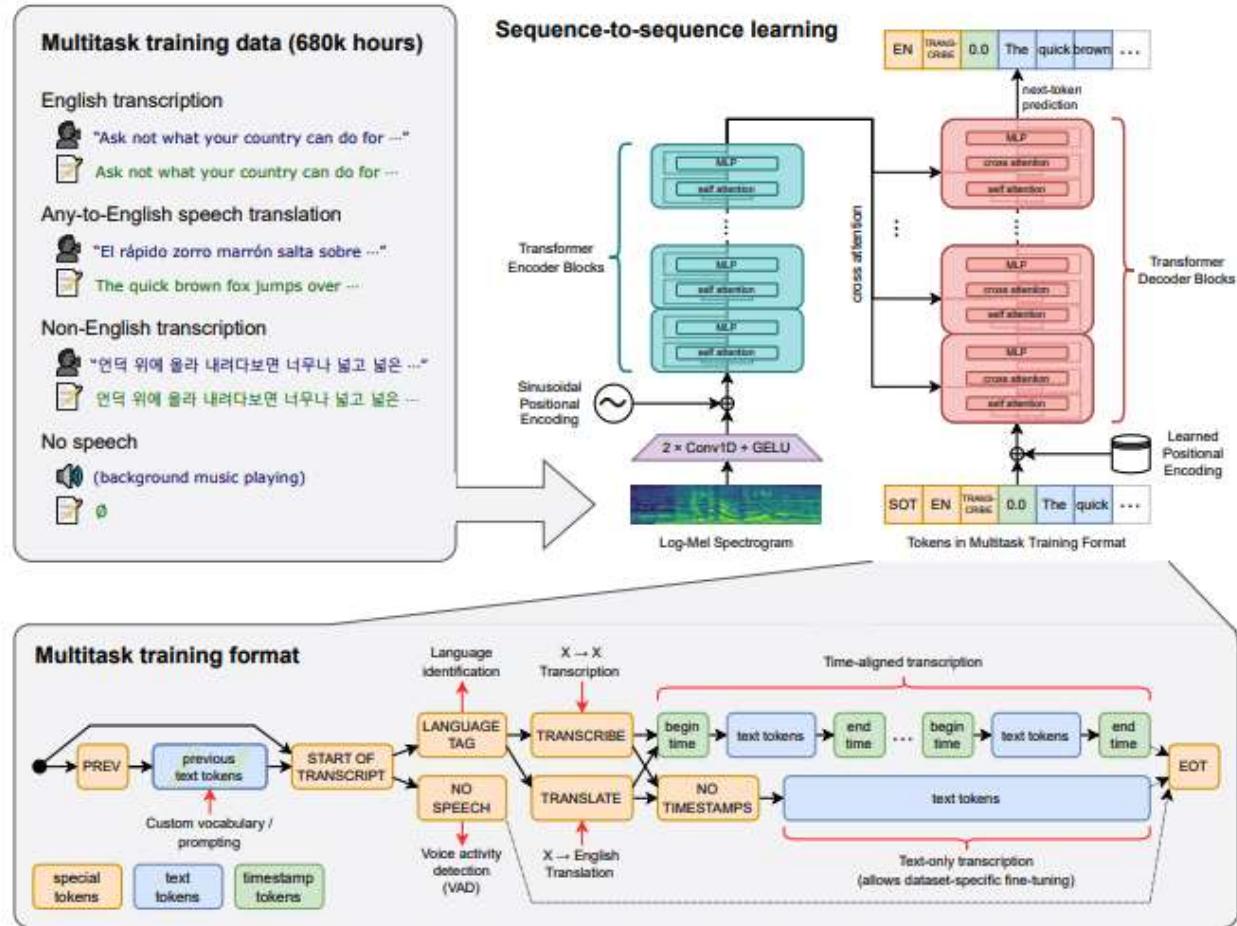
Model	Automatic (d-BLEU)					Human (General/Discourse)				
	News	Social	Fiction	Q&A	Ave.	News	Social	Fiction	Q&A	Ave.
Google	27.7	35.4	16.0	12.0	22.8	1.9/2.0	1.2/1.3	2.1/2.4	1.5/1.5	1.7/1.8
DeepL	30.3	33.4	16.1	11.9	22.9	2.2/2.2	1.3/1.1	2.4/2.6	1.6/1.5	1.9/1.9
Tencent	29.3	38.8	20.7	15.0	26.0	2.3/2.2	1.5/1.5	2.6/2.8	1.8/1.7	2.1/2.1
GPT-3.5	29.1	35.5	17.4	17.4	24.9	2.8/2.8	2.5/2.7	2.8/2.9	2.9/2.9	2.8/2.8
GPT-4	29.7	34.4	18.8	19.0	25.5	3.3/3.4	2.9/2.9	2.6/2.8	3.1/3.2	3.0/3.1

- Introduces evaluation metric which are “discourse aware”
- Shows the superiority of LLMs over other advanced MT systems
- Demonstrates the need for context-aware prompts
 - ChatGPT can translate sentence-by-sentence but remember the discourse-level context via the chat box
 - But best results generally by providing the whole document

Whisper (Radford et al. 2023)

- Trained to transcribe and translate audio
- Dataset constructed from audio paired with human-generated transcripts on the Internet
- Minimal pre-processing
- Audio language detection
- Multi-task format
- Zero-shot evaluation
- Available in Huggingface

Overview of Whisper



X → English	High	Mid	Low	All
XMEF-X	34.2	20.2	5.9	14.7
XLS-R (2B)	36.1	27.7	15.1	22.1
mSLAM-CTC (2B)	37.8	29.6	18.5	24.8
Maestro	38.2	31.3	18.4	25.2
Zero-Shot Whisper	36.2	32.6	25.2	29.1

Table 3. $X \rightarrow \text{en}$ Speech translation performance. Zero-shot Whisper outperforms existing models on CoVoST2 in the overall, medium, and low resource settings but still moderately underperforms on high-resource languages compared to prior directly supervised work.

Machine Translation: remaining challenges

- Long documents can still pose problems
- Stylised translation
- Interactivity
- Multi-modal
- Evaluation – quality and interpretability / explanations

Benchmarking LLMs for News Summarization

(Zhang et al. 2024)

Article: <https://www.dailymail.co.uk/femail/article-3052124>

Reference summary:

Rare tote is so exclusive that it can't even be brought in store. Experts say it's an extremely good investment as only increases in value. Kim Kardashian and Victoria Beckham are both fans of Hermes.

0-shot GPT-3 Davinci summary:

The most expensive handbag in the world is a pink Hermes Birkin bag, which has been put on sale for £140,000. The bag is so exclusive that it can't even be brought in store and experts say it's an extremely good investment. With the same amount of money that the handbag costs, you could splash out on an Aston Martin DB9 Volante.

What is the main idea of the article?

0-shot Instruct Davinci summary:

The article discusses a pink Hermes Birkin bag that is on sale for £140,000. The bag is made-to-measure and embellished with rubies. The article also discusses how the bag is a collector's item and an investment.

Freelance writer summary:

Hermes is a brand rich in history and value, so much so that experts are saying their new £140,000 handbag is an extremely good investment. The bag, studded in rubies, is only the latest in Hermes Birkin's long line of highly expensive handbags for highly stylish women, which include celebrities and royalty.

Figure 2: Examples summaries generated by GPT-3 models (Section 3) or written by freelance writers (Section 4) of an article from the CNN/DM dataset. We find that the instruction-tuned GPT-3 model can generate a much better summary compared to the non-instruction-tuned variant. The reference summary from CNN/DM is not coherent whereas the freelance writer summary is both coherent and relevant.

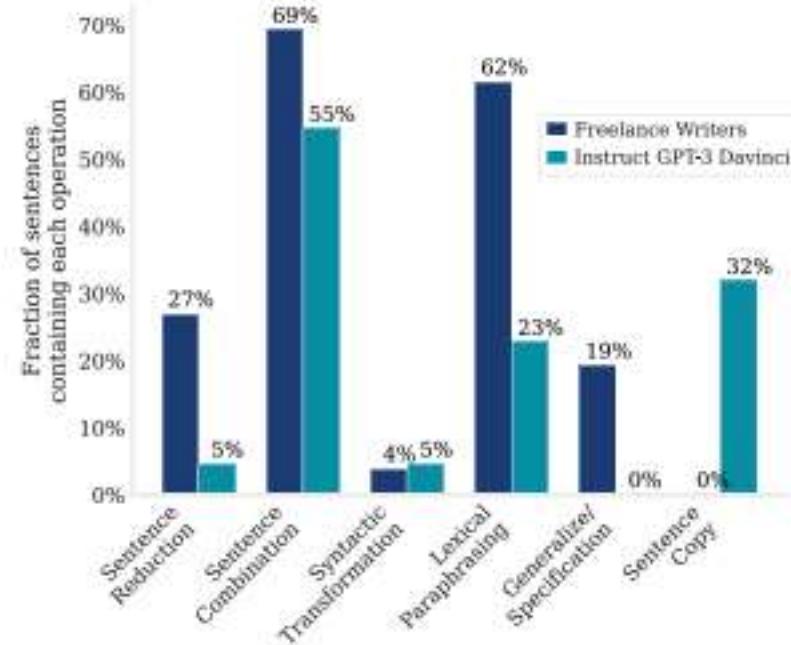


Figure 4: Distributions of cut and paste operations in the summaries written by freelance writers and by Instruct Davinci. By comparison, human-written summaries contain more lexical paraphrasing and sentence reduction whereas the Instruct Davinci model has more direct copying from the article.

Retrieval-Augmented Generation (Gao et al. 2023)

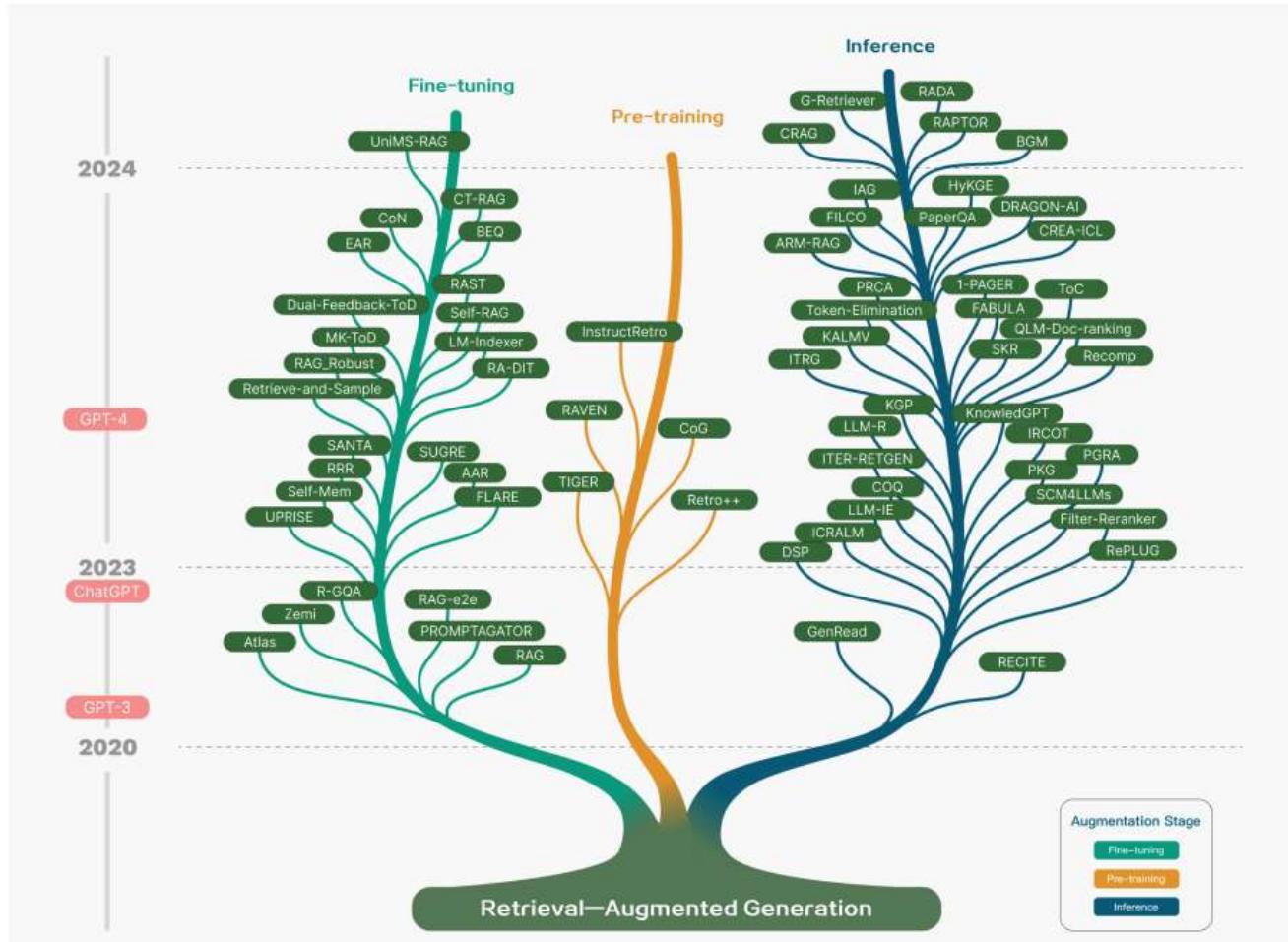


Fig. 1. Technology tree of RAG research. The stages of involving RAG mainly include pre-training, fine-tuning, and inference. With the emergence of LLMs,

Retrieve-Read Framework for RAG

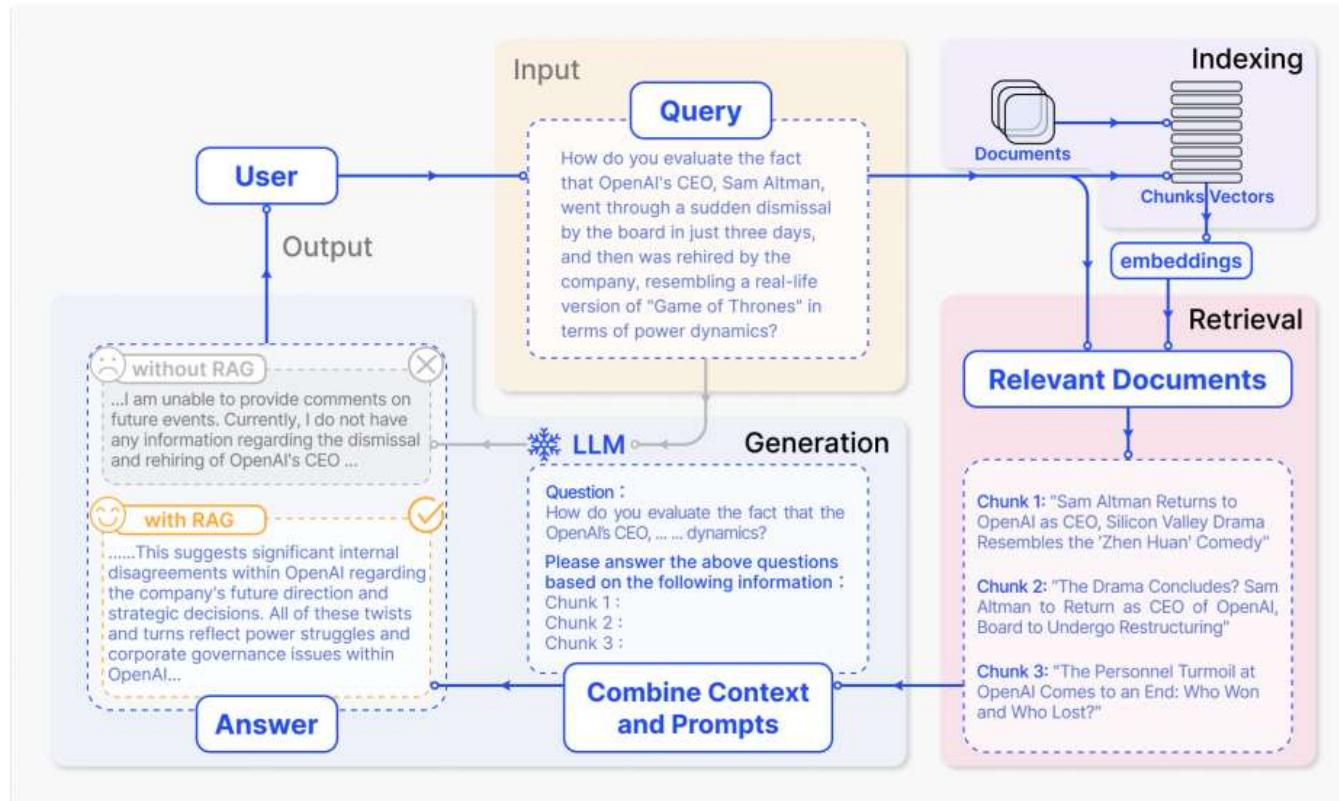


Fig. 2. A representative instance of the RAG process applied to question answering. It mainly consists of 3 steps. 1) Indexing. Documents are split into chunks, encoded into vectors, and stored in a vector database. 2) Retrieval. Retrieve the Top k chunks most relevant to the question based on semantic similarity. 3) Generation. Input the original question and the retrieved chunks together into LLM to generate the final answer.

More Advanced RAG

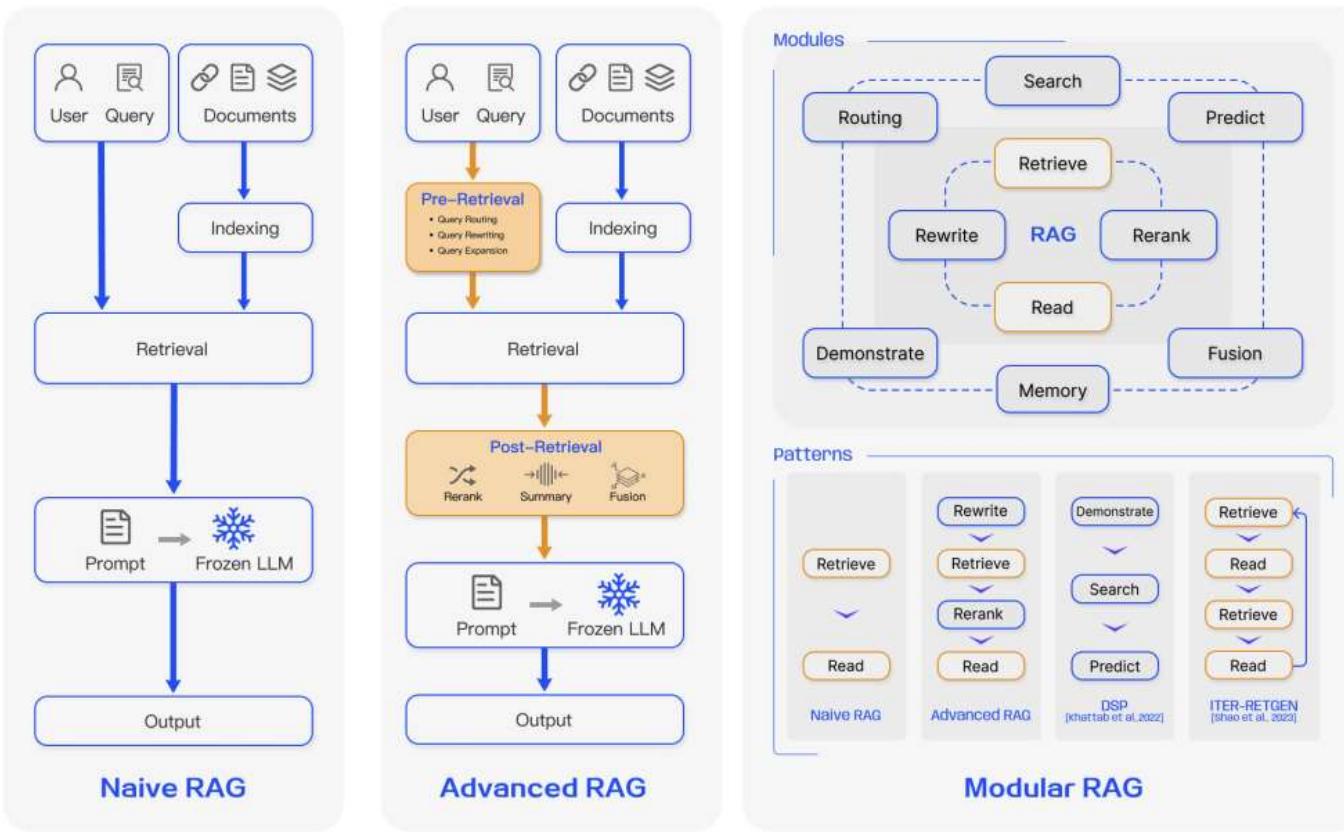


Fig. 3. Comparison between the three paradigms of RAG. (Left) Naive RAG mainly consists of three parts: indexing, retrieval and generation. (Middle) Advanced RAG proposes multiple optimization strategies around pre-retrieval and post-retrieval, with a process similar to the Naive RAG, still following a chain-like structure. (Right) Modular RAG inherits and develops from the previous paradigm, showcasing greater flexibility overall. This is evident in the introduction of multiple specific functional modules and the replacement of existing modules. The overall process is not limited to sequential retrieval and generation; it includes methods such as iterative and adaptive retrieval.

Ethical considerations

- What else do you need to consider before using an advanced LLM system to generate text for you?

Accountability

By Maria Yagoda 23rd February 2024

When Air Canada's chatbot gave incorrect information to a traveller, the airline argued its chatbot is "responsible for its own actions".

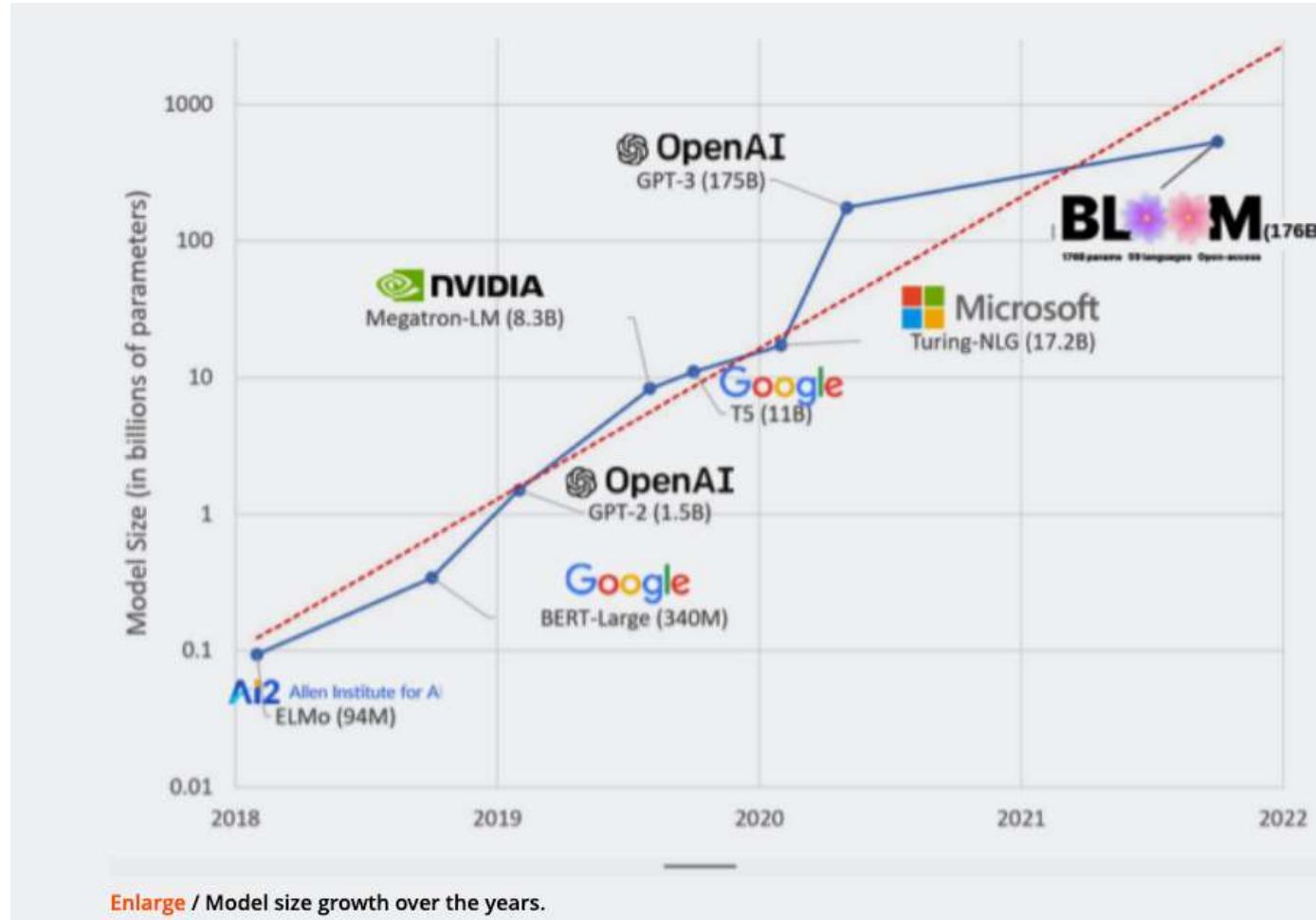
Artificial intelligence is having a growing impact on the way we travel, and a remarkable new case shows what AI-powered chatbots can get wrong – and who should pay. In 2022, Air Canada's chatbot promised a discount that wasn't available to passenger Jake Moffatt, who was assured that he could book a full-fare flight for his grandmother's funeral and then apply for a bereavement fare after the fact.

According to a civil-resolutions tribunal decision last Wednesday, when Moffatt applied for the discount, the airline said the chatbot had been wrong – the request needed to be submitted before the flight – and it wouldn't offer the discount. Instead, the airline said the chatbot was a "separate legal entity that is responsible for its own actions". Air Canada argued that Moffatt should have gone to the link provided by the chatbot, where he would have seen the correct policy.

The British Columbia Civil Resolution Tribunal rejected that argument, ruling that Air Canada had to pay Moffatt \$812.02 (£642.64) in damages and tribunal fees. "It should be obvious to Air Canada that it is responsible for all the information on its website," read tribunal member Christopher Rivers' written response. "It makes no difference whether the information comes from a static page or a chatbot." The BBC reached out to Air Canada for additional comment and will update this article if and when we receive a response.



Model Growth



Source:

<https://arstechnica.com/gadgets/2023/04/generative-ai-is-cool-but-lets-not-forget-its-human-and-environmental-costs/>

Environmental Impact of LLMs

Model name	Number of parameters	Datacenter PUE	Carbon intensity of grid used	Power consumption	CO ₂ eq emissions	CO ₂ eq emissions × PUE
GPT-3	175B	1.1	429 gCO ₂ eq/kWh	1,287 MWh	502 tonnes	552 tonnes
Gopher	280B	1.08	330 gCO ₂ eq/kWh	<i>1,066 MWh</i>	<i>352 tonnes</i>	380 tonnes
OPT	175B	<i>1.09</i> ²	<i>231gCO₂eq/kWh</i>	<i>324 MWh</i>	70 tonnes	<i>76.3 tonnes</i> ³
BLOOM	176B	1.2	57 gCO ₂ eq/kWh	433 MWh	25 tonnes	30 tonnes

Table 4: Comparison of carbon emissions between BLOOM and similar LLMs. Numbers in *italics* have been inferred based on data provided in the papers describing the models.

Sasha Luccioni, et al.
DOI

- high levels of energy use and CO₂ emissions
- Training GPT-3 and Llama 2 required around 1.3 GWH → 552 tonnes of CO₂ → 2 or 3 full Boeing 767s flying round-trip from New York to San Francisco
- Electricity use of ChatGPT in inference likely surpasses that of training within weeks or days (inference counts for 90% of AI workloads) → more like using a Boeing 767 to transport a single passenger at a time

What should we do?

Part 2 : Revision

Revision questions

1. Name and give examples of different semantic relationships which can hold between *distributionally similar* words.
2. What is Zipf's Law and why is it a problem?
3. What are the main similarities and differences between word2vec and GLoVE?
4. Explain how to use a trigram model to compute the probability of a sentence.
5. What is perplexity?
6. What advantage do LSTMs have over vanilla RNNs in language modelling?
7. How and why might you combine a character-based network with a word-based network in language modelling?
8. For what types of problems are CRFs typically used?
9. What's the difference between a generative statistical classification model and a discriminative statistical classification model?
10. When and why might it be better to use F₁ rather than accuracy as an evaluation metric?

More questions

1. In a multi-class scenario, how would you calculate micro-average F1 and macro-average F1. Which is better and why?
2. Describe 2 different ways word embeddings might be combined to make sentence embeddings
3. Give an example of structural differences between languages.
4. Outline 2 different methods for evaluating machine translation systems.
5. How might an encoder-decoder network be used for MT?
6. What is subword tokenization?
7. In an attention head, what are the 3 different vectors which are created? How are they created and how are they used?
8. What is the input representation used by BERT?
9. What is the difference between masked language modelling and autoregressive language modelling?
10. What is transfer learning? Explain with reference to BERT