COMP 3225

Natural Language Processing

Revision

Stuart E. Middleton

sem03@soton.ac.uk

University of Southampton

Sections

- Past Exam Questions
- Landscape of Content
- Non-Neural NLP Revision
- Neural NLP Revision

This lecture is supporting material to help you focus your revision.

All core examinable content has already been covered in previous lectures. Re-watch the videos to recap the content.

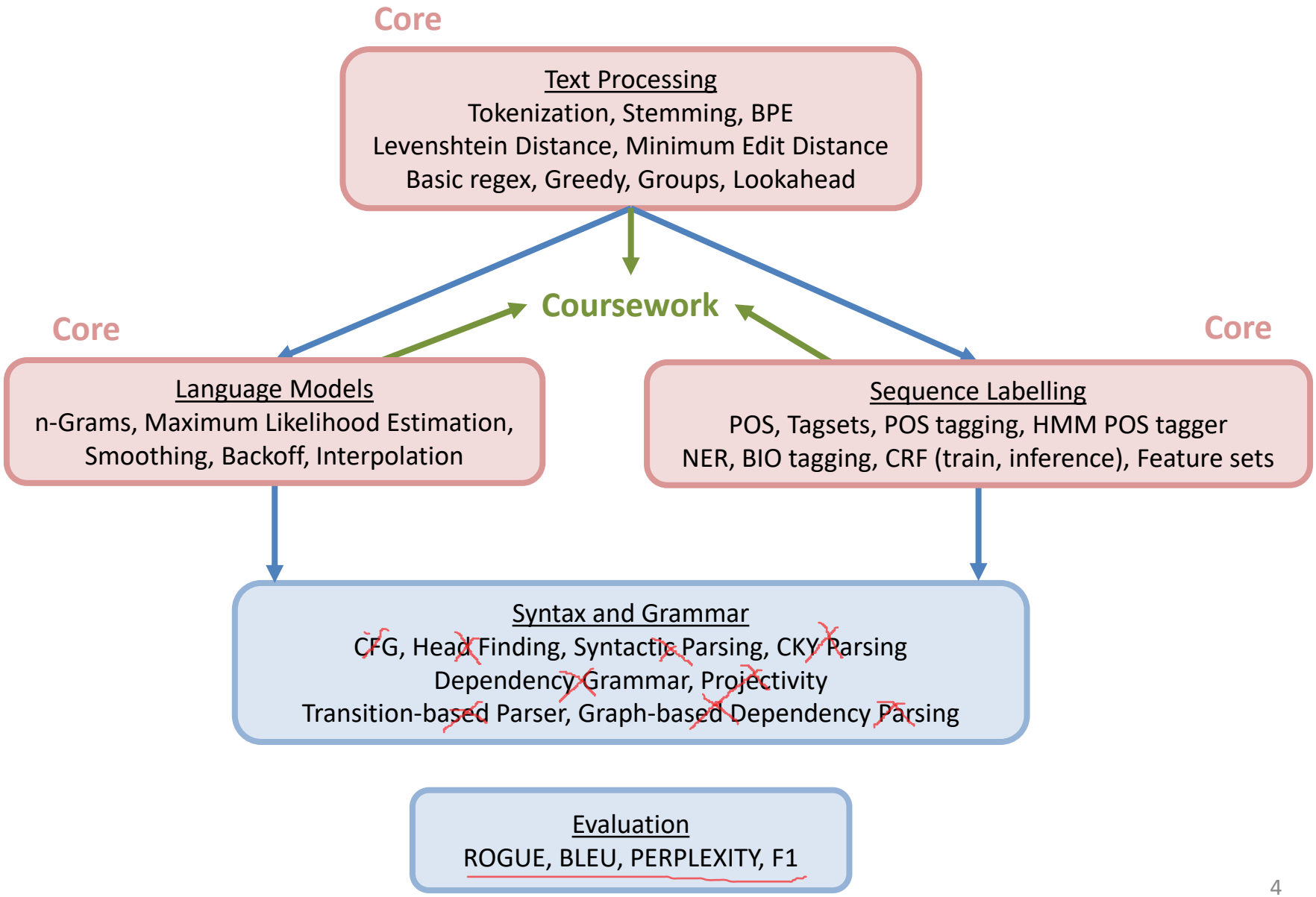
Past Exam Questions

- Exam guidance on module wiki
  - https://secure.ecs.soton.ac.uk/notes/comp3225/exam/COMP3225-exam-paper-guidance.pdf
  - Part A (answer 2 of 2) - one classic NLP topic and one neural NLP topic
  - Part B (answer 1 of 3) - any topic
  - <look at guidance>

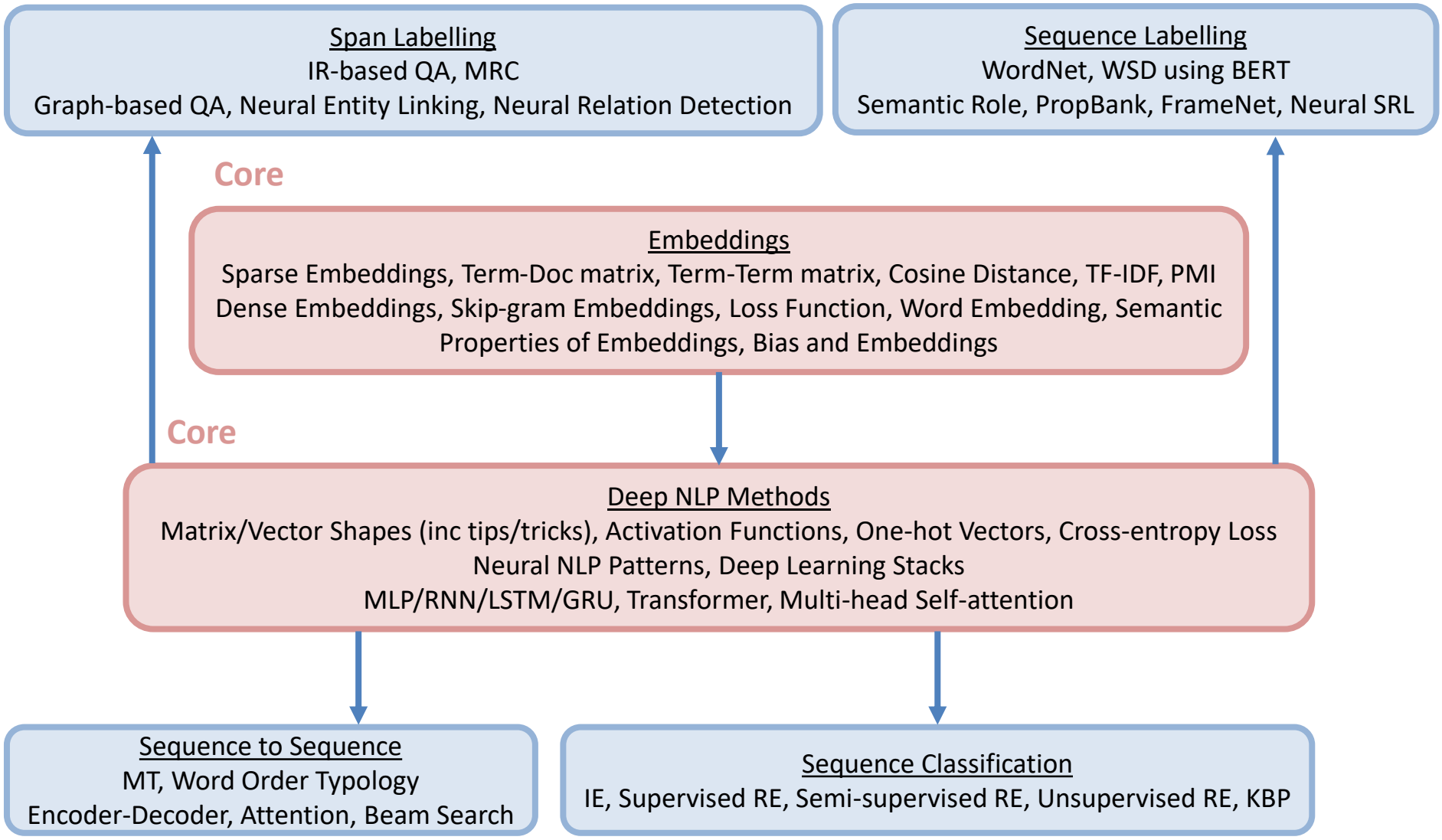
- Typical exam question structure
  - 5 marks assessing recall / application of lectures
  - 10 marks assessing recall / application of lectures & book
  - 15 marks assessing understanding of lectures & book & wider reading

- Past exam paper on module wiki
  - https://secure.ecs.soton.ac.uk/notes/comp3225/exam/COMP3225-past-exam-paper-2021.pdf
  - In 2021 (past exam paper) it was an online 24 hour
  - In 2022 exam will be a 2 hour in-person examination
  - <look at past exam paper>

Landscape of Non-Neural Content



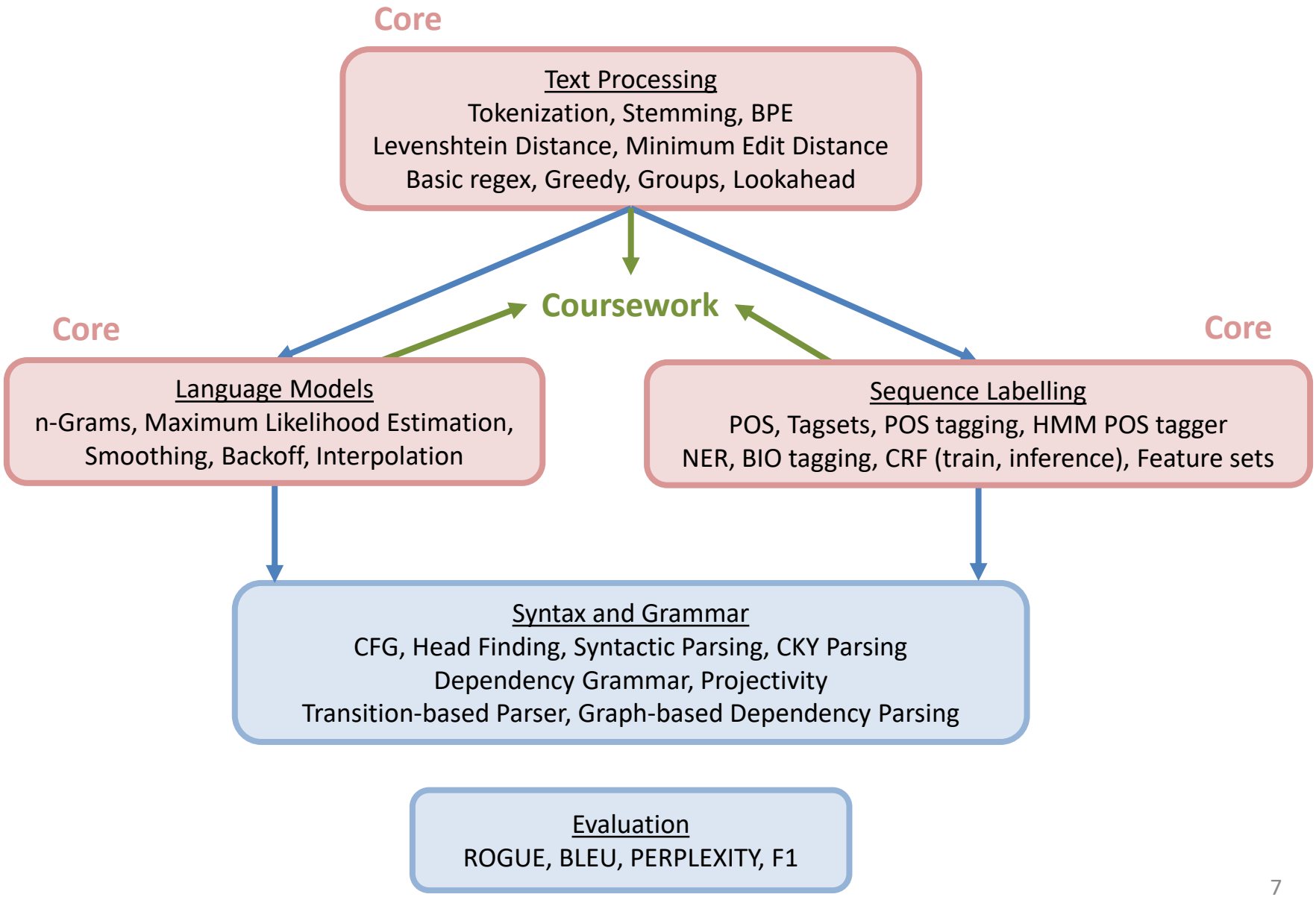
Landscape of Neural Content



Overview - Non-Neural NLP Revision

- Landscape of Non-Neural Content
- Words (2)
- Regular Expressions (3)
- Training, Evaluation & Linguistic Resources (4)
- N-Grams (5)
- Parts of Speech Tagging (6)
- Named Entity Recognition (7)
- Constituency Grammars (14)
- Syntactic Parsing (15)
- Dependency Parsing (16)

Landscape of Non-Neural Content



Metrics - ROUGE

- used to evaluate text summarization
  - a good machine summary is one that includes many of the same sequences of words as a human-generated (reference) summary
  - ROUGE-n is the proportion of the significant word sequences (n-grams, see lecture 5) in the machine summaries matching those in the reference summaries
    - Variations for N=1, 2, longest common subsequence, etc
  - recall-oriented
    - depends on quantity of material that matches

- ROUGE: A Package for Automatic Evaluation of Summaries, by Chin-Yew Lin (ISI)
  - https://www.microsoft.com/en-us/research/wp-content/uploads/2016/07/was2004.pdf

Metrics - BLEU

see chapter 11.8.2

- used to evaluate machine translations
  - a good machine translation is one that includes many of the same sequences of words as a human-generated (reference) translation

- Precision-based
  - how many n-word sequences from the machine-generated set also appear in the reference set for n=1,2,3,4
  - Proportion of all common sequences matching those in the reference set, compared to the total number of sequences

- BLEU ignores recall-based factors (how much of the material did it translate) and focuses only on precision (how much of the material that it translated did it translate well).
  - One solution is to combine both kinds of metric (see F1 score)
  - Instead, BLEU penalizes translations that are shorter than the reference translations

Source

la verdad, cuya madre es la historia, émula del tiempo, depósito de las acciones, testigo de lo pasado, ejemplo y aviso de lo presente, advertencia de lo por venir.

Reference

truth, whose mother is history, rival of time, storehouse of deeds, witness for the past, example and counsel for the present, and warning for the future.

Candidate 1

truth, whose mother is history, voice of time, deposit of actions, witness for the past, example and warning for the present, and warning for the future

Candidate 2

the truth, which mother is the history, émula of the time, deposition of the shares, witness of the past, example and notice of the present, warning of it for coming

Figure 11.16 Intuition for BLEU: One of two candidate translations of a Spanish sentence shares more n-grams, and especially longer n-grams, with the reference human translation.

$$\text{prec}_n = \frac{\sum_{C \in \{Candidates\}} \sum_{n\text{-}gram \in C} \text{Count}_{\text{match}}(n\text{-}gram)}{\sum_{C' \in \{Candidates\}} \sum_{n\text{-}gram' \in C'} \text{Count}(n\text{-}gram')}$$

Equation 11.23

Metrics - PERPLEXITY

- used to evaluate language models
  - how good a vocabulary, or a list of word sequences (n-grams), is at "predicting" a target text
  - based on the probability of all the words in the text appearing in that order
    - inverted and normalized by the number of words
- minimizing perplexity is equivalent to maximizing the test set probability according to the language model.

$$\begin{aligned} \text{PP}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}} \end{aligned}$$

Equation 3.14

	Unigram	Bigram	Trigram
Perplexity	962	170	109

We trained unigram, bigram, and trigram grammars on 38 million words (including start-of-sentence tokens) from the Wall Street Journal, using a 19,979 word vocabulary. We then computed **the perplexity** of each of these models on a test set of 1.5 million words. The blue table above shows the perplexity of a 1.5 million word WSJ test set according to each of these grammars. (see page 37)

Metrics – Precision, Recall & F1

- used in text classification, searching etc
- **Recall** = proportion of relevant items that were actually selected from the set of all relevant items
  - Items that were classified compared to all the items that should have been classified
- **Precision** = proportion of genuinely relevant items that were selected compared to all the items that were selected
  - Items that were correctly classified compared to all the items that were classified
- Trade-off: easy to have 100% precision but 1% recall or 100% recall and 1% precision

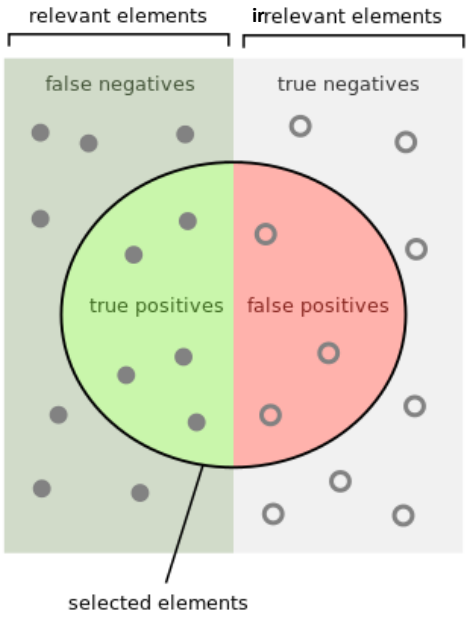


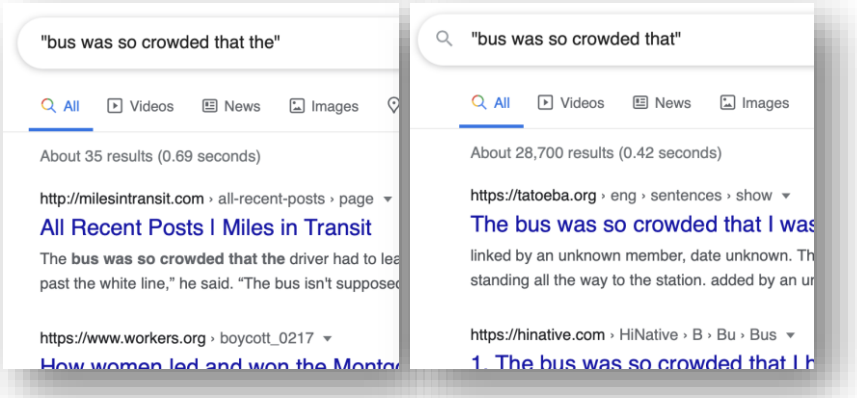
Diagram sourced from Wikipedia

N-grams

- An n-gram model informs us of the probability of the *next word* in the text, given the *previous n-1 words* in the text.
  - P(w | h) the probability of word *w* given the previous history *h*, where *h* is is sequence of words
  - Out of the times that *h* occurred, how may times was it followed by *w*

$$P(\text{the} \mid \text{bus was so crowded that}) = \frac{Count(\text{bus was so crowded that the})}{Count(\text{bus was so crowded that})}$$

$$= \frac{35}{28700} = 0.12\% \textit{ (Google)}$$



Maximum Likelihood Estimation (MLE)

- The process of choosing the right set of bigram parameters to make our model correctly predict (maximise the likelihood of) the nth word in the text is called *maximum likelihood estimation*.

- the MLE estimate for the parameters of an n-gram model are obtained by
  - observing the n-gram counts from a representative corpus
  - normalizing them (dividing by a total count) to lie between 0 and 1

Bigram Parameter Estimation

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

Equation 3.11

N-gram Parameter Estimation

$$P(w_n|w_{n-N+1:n-1}) = \frac{C(w_{n-N+1:n-1}\ w_n)}{C(w_{n-N+1:n-1})}$$

Equation 3.12

This is the process we saw on slide 7 "the bus was so crowded that"

Laplace smoothing

- Apply Laplace smoothing to unigrams.
  - the unsmoothed MLE of word w_i is its count c_i normalized by N, the total number of tokens
    - $P(w_i) = c_i / N$
  - Since there are V words in the vocabulary and each one was incremented, the denominator is also adjusted to take into account the extra V observations
    - $LP(w_i) = (c_i + 1) / (N + V)$
  - Instead of changing both the numerator and denominator, it is convenient to describe how a smoothing algorithm affects the numerator, by defining an adjusted count c^* which is easier to compare directly with the MLE counts
    - $c^*_i = (c_i + 1) N / (N + V)$
  - Normalising by N yields the same expression as $LP(w_i)$ above
  - Also consider the discount rate $d_c = c^* / c$

Backoff & Interpolation

- If data is sparse about the appearance of higher-order n-grams, we can fall back to information about lower order n-grams
- to compute $P(w_n \mid w_{n-2}w_{n-1})$ without counts of the trigram $w_{n-2}\ w_{n-1}\ w_n$
  - estimate its probability by using the bigram probability $P(w_n \mid w_{n-1})$
  - if there are no counts of the bigram, estimate using the unigram $P(w_n)$.

- In the absence of detailed information, using less context can be a good strategy. Allows generalization to more for contexts that the model hasn't been trained on about.

- There are two ways to use this n-gram "hierarchy".
  - **Backoff**: only use lower-order n-gram if we have zero evidence for a higher-order n-gram
  - **Interpolation**: always mix the probability estimates from all the n-gram estimators, weighing and combining the trigram, bigram, and unigram counts.

Introduction to Parts of Speech (POS)

	Tag	Description	Example
Open Class	**ADJ**	Adjective: noun modifiers describing properties	*red, young, awesome*
	ADV	Adverb: verb modifiers of time, place, manner	*very, slowly, home, yesterday*
	NOUN	words for persons, places, things, etc.	*algorithm, cat, mango, beauty*
	VERB	words for actions and processes	*draw, provide, go*
	PROPN	Proper noun: name of a person, organization, place, etc..	*Regina, IBM, Colorado*
	INTJ	Interjection: exclamation, greeting, yes/no response, etc.	*oh, um, yes, hello*
Closed Class Words	**ADP**	Adposition (Preposition/Postposition): marks a noun's spacial, temporal, or other relation	*in, on, by under*
	AUX	Auxiliary: helping verb marking tense, aspect, mood, etc.,	*can, may, should, are*
	CCONJ	Coordinating Conjunction: joins two phrases/clauses	*and, or, but*
	DET	Determiner: marks noun phrase properties	*a, an, the, this*
	NUM	Numeral	*one, two, first, second*
	PART	Particle: a preposition-like form used together with a verb	*up, down, on, off, in, out, at, by*
	PRON	Pronoun: a shorthand for referring to an entity or event	*she, who, I, others*
	SCONJ	Subordinating Conjunction: joins a main clause with a subordinate clause such as a sentential complement	*that, which*
Other	**PUNCT**	Punctuation	; , ()
	SYM	Symbols like $ or emoji	$, %
	X	Other	asdf, qwfg

- Closed Classes - fixed membership
  - Typically function words used for structuring grammar (of, it, and, you)
- Open Classes - open membership
  - Noun (including proper noun), verb, adjective, adverb, interjection

Tagsets

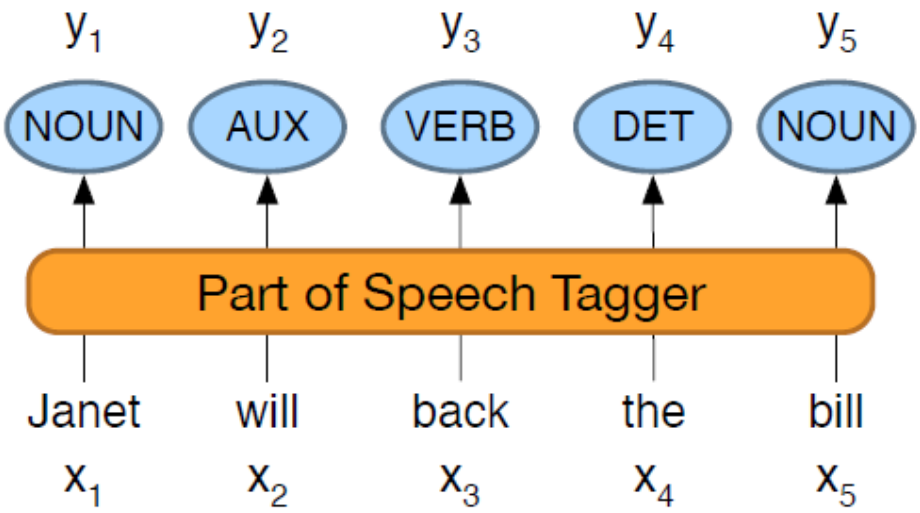
- A list of POS labels is called a Tagset
- Tagsets come in different shapes and sizes
  - Penn Treebank (45 labels)
  - Brown Corpus (87 labels)
  - C7 Tagset (146 labels)
- Penn Treebank

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coord. conj.	*and, but, or*	NNP	proper noun, sing.	*IBM*	TO	"to"	*to*
CD	cardinal number	*one, two*	NNPS	proper noun, plu.	*Carolinas*	UH	interjection	*ah, oops*
DT	determiner	*a, the*	NNS	noun, plural	*llamas*	VB	verb base	*eat*
EX	existential 'there'	*there*	PDT	predeterminer	*all, both*	VBD	verb past tense	*ate*
FW	foreign word	*mea culpa*	POS	possessive ending	*'s*	VBG	verb gerund	*eating*
IN	preposition/ subordin-conj	*of, in, by*	PRP	personal pronoun	*I, you, he*	VBN	verb past participle	*eaten*
JJ	adjective	*yellow*	PRP$	possess. pronoun	*your, one's*	VBP	verb non-3sg-pr	*eat*
JJR	comparative adj	*bigger*	RB	adverb	*quickly*	VBZ	verb 3sg pres	*eats*
JJS	superlative adj	*wildest*	RBR	comparative adv	*faster*	WDT	wh-determ.	*which, that*
LS	list item marker	*1, 2, One*	RBS	superlatv. adv	*fastest*	WP	wh-pronoun	*what, who*
MD	modal	*can, should*	RP	particle	*up, off*	WP$	wh-possess.	*whose*
NN	sing or mass noun	*llama*	SYM	symbol	+,%, &	WRB	wh-adverb	*how, where*

Cite: Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn treebank. Computational Linguistics 19(2), 313–330

POS Tagging

- POS tagging is the process of assigning a POS tag to each word in a text
  - Input sequence >> X >> x_1; x_2; ::: ; x_n of (tokenized) words
  - Output sequence >> Y >> y_1; y_2; ::: ; y_n of POS tags
  - Each output y_i corresponding exactly to one input x_i



Hidden Markov Model (HMM) POS tagger

- Hidden Markov Model >> POS tags are hidden states, which we must infer from observed words

Markov Assumption: $P(q_i|q_1,...,q_{i-1}) = P(q_i|q_{i-1})$

Output Independence: $P(o_i|q_1,\ldots q_i,\ldots,q_T,o_1,\ldots,o_i,\ldots,o_T) = P(o_i|q_i)$

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} \ldots a_{ij} \ldots a_{NN}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{N} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \ldots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, ..., v_V$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state q_i
$\pi = \pi_1, \pi_2, ..., \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

Named Entity Recognition (NER)

- BIO tagging is a common approach for sequence labelling requiring span-recognition
  - Begin, Inside, Outside >> BIO
  - Begin, Inside, Outside, End, Single >> BIOES
- Append NE type to BIO tag to create a token label

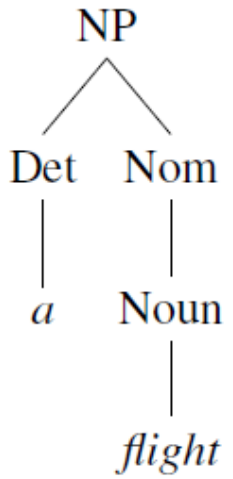
Words	IO Label	BIO Label	BIOES Label
Jane	I-PER	B-PER	B-PER
Villanueva	I-PER	I-PER	E-PER
of	O	O	O
United	I-ORG	B-ORG	B-ORG
Airlines	I-ORG	I-ORG	I-ORG
Holding	I-ORG	I-ORG	E-ORG
discussed	O	O	O
the	O	O	O
Chicago	I-LOC	B-LOC	S-LOC
route	O	O	O
.	O	O	O

Context Free Grammar

- Example productions (rules) for Noun Phrase

$$
\begin{aligned}
NP &\rightarrow Det\ Nominal \\
NP &\rightarrow ProperNoun \\
Nominal &\rightarrow Noun \mid Nominal\ Noun
\end{aligned}
\qquad
\begin{aligned}
Det &\rightarrow a \\
Det &\rightarrow the \\
Noun &\rightarrow flight
\end{aligned}
$$

- Given <left symbol> generate <right set of symbols>
- NP >> Det Nominal >> Det Noun >> a flight = one derivation
- Derivations are usually represented as a parse tree



- Leaf nodes are terminal nodes (words from lexicon)
- Non-terminal nodes define lexical categories (POS)
- A node is said to dominate its child nodes
- The root node is the start symbol (usually 'S')

CKY Parsing

- Dynamic Programming is useful to address ambiguity when using context free rules
- Cocke-Kasami Younger (CKY) algorithm is a classic dynamic programming approach to parsing
- Dynamic Programming = chart parsing
- CKY requires grammars in Chomsky Normal Form (CNF)
  - Right side must be (a) two non-terminal nodes or (b) single terminal node
  - Any Context Free Grammar (CFG) can be converted to CNF
  - If a CFG has a single non-terminal node (unit production) then add a dummy terminal node that itself leads to a non-terminal node

  CFG: INF-VP → to VP

  CNF: INF-VP → TO VP; TO → to
  - Use dummy terminal nodes to break up rules with three or more nodes

  CFG: S → AUX NP VP

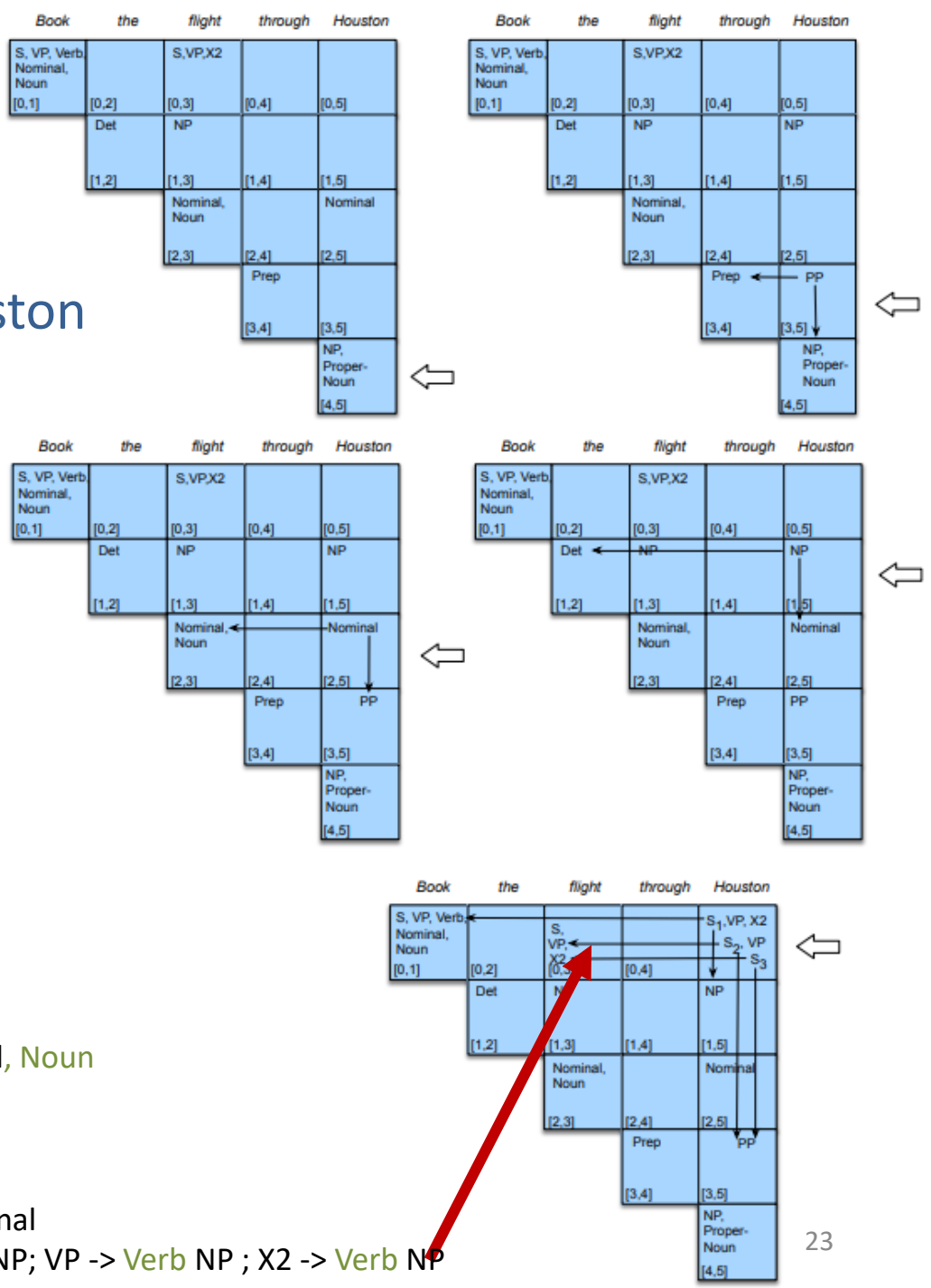
  CNF: S → X1 VP; X1 → AUX NP

CKY Parsing

- Example parse table for sent
  Book the flight through Houston

$\mathscr{L}_1$ in CNF
$S \rightarrow NP\ VP$
$S \rightarrow X1\ VP$
$X1 \rightarrow Aux\ NP$
$S \rightarrow book \mid include \mid prefer$
$S \rightarrow Verb\ NP$
$S \rightarrow X2\ PP$
$S \rightarrow Verb\ PP$
$S \rightarrow VP\ PP$
$NP \rightarrow I \mid she \mid me$
$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$
$VP \rightarrow X2\ PP$
$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$

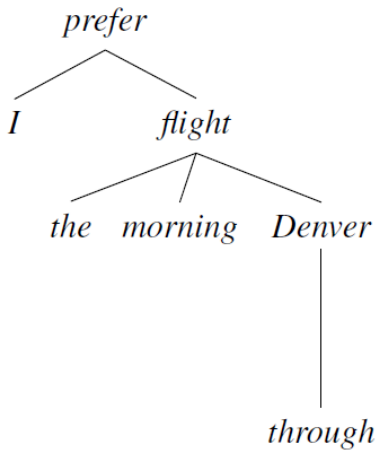
and POS tags

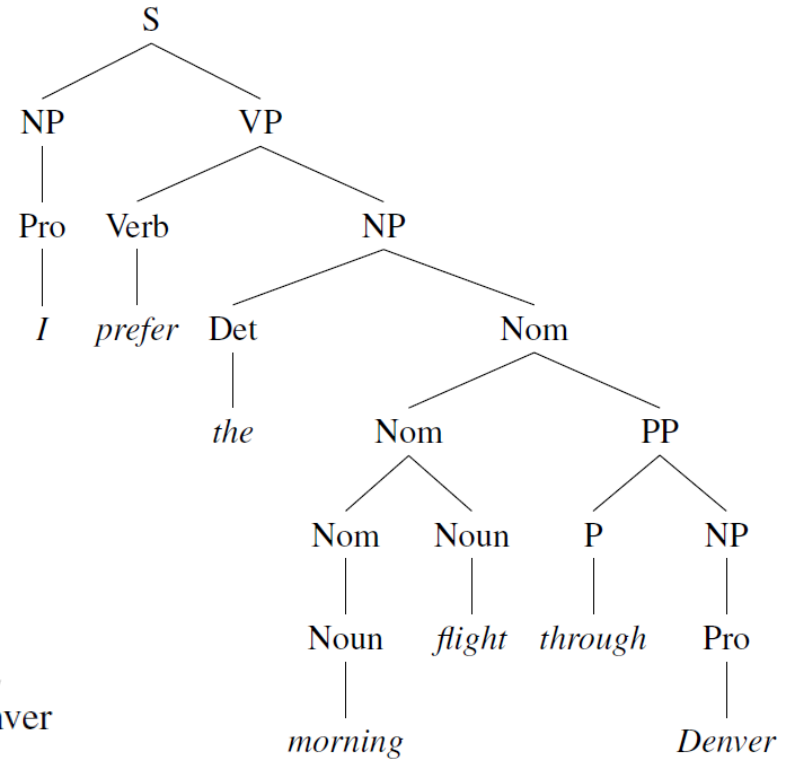


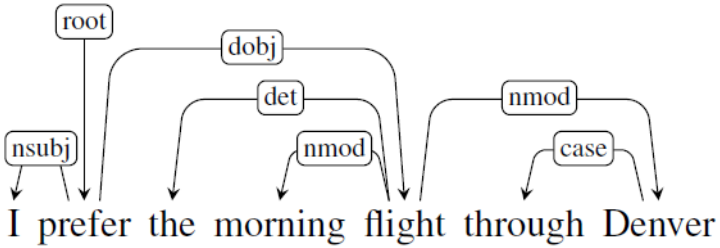
Book = span(0,1) >> productions >> S, VP, Verb, Nominal, Noun
the = span(1,2) >> productions >> Det
Book the = span(0,2) >> productions >> <none>
flight = span(2,3) >> productions >> Nominal, Noun
the flight = span(1,3) >> productions >> NP -> Det Nominal
Book the flight = span(0,3) >> productions >> S -> Verb NP; VP -> Verb NP ; X2 -> Verb NP

Dependency Grammars

- Context free grammars provide a constituent-based structure
- Typed Dependency structures use grammatical relations
  - Grammatical relations defined by linguists
  - Labels for edges are grammatical relations
  - Nodes are words
- Dependency grammars have a free word order

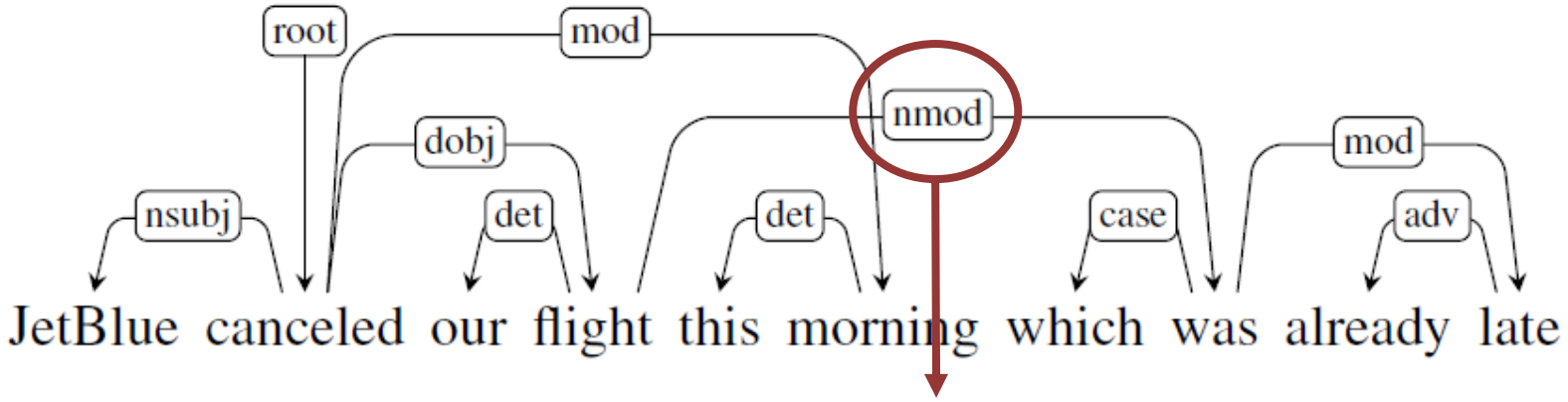






Dependency Formalisms

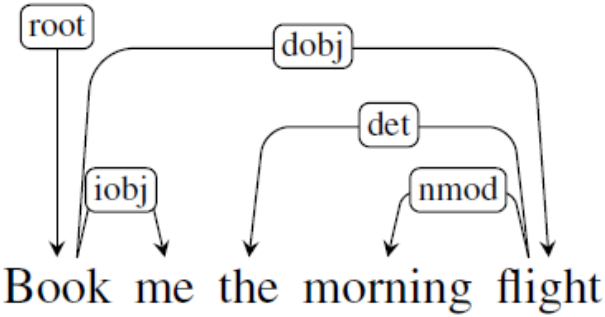
- Dependency tree is a directed graph
  - G = (V,A)
  - V = Vertices = nodes = words or sometimes stems/affixes
  - A = Arcs = grammatical function relationships
  - Root node has no incoming A
  - Each V has one incoming A
  - Root node has a path to connect to every V
- Projectivity
  - An arc is 'projective' if for a pair(head, dependent) there is a path from the head to all words in-between the head and dependent word.



nmod(flight, was) >> in-between words this morning not connected to flight >> not projective

Transition-based Dependency Parsing

- Basic Transition-based Parser
  - Stack = graph so far = root node at start
  - Input buffer = words to try = words in sentence at start
  - Set of relations = dep tree = empty set at start

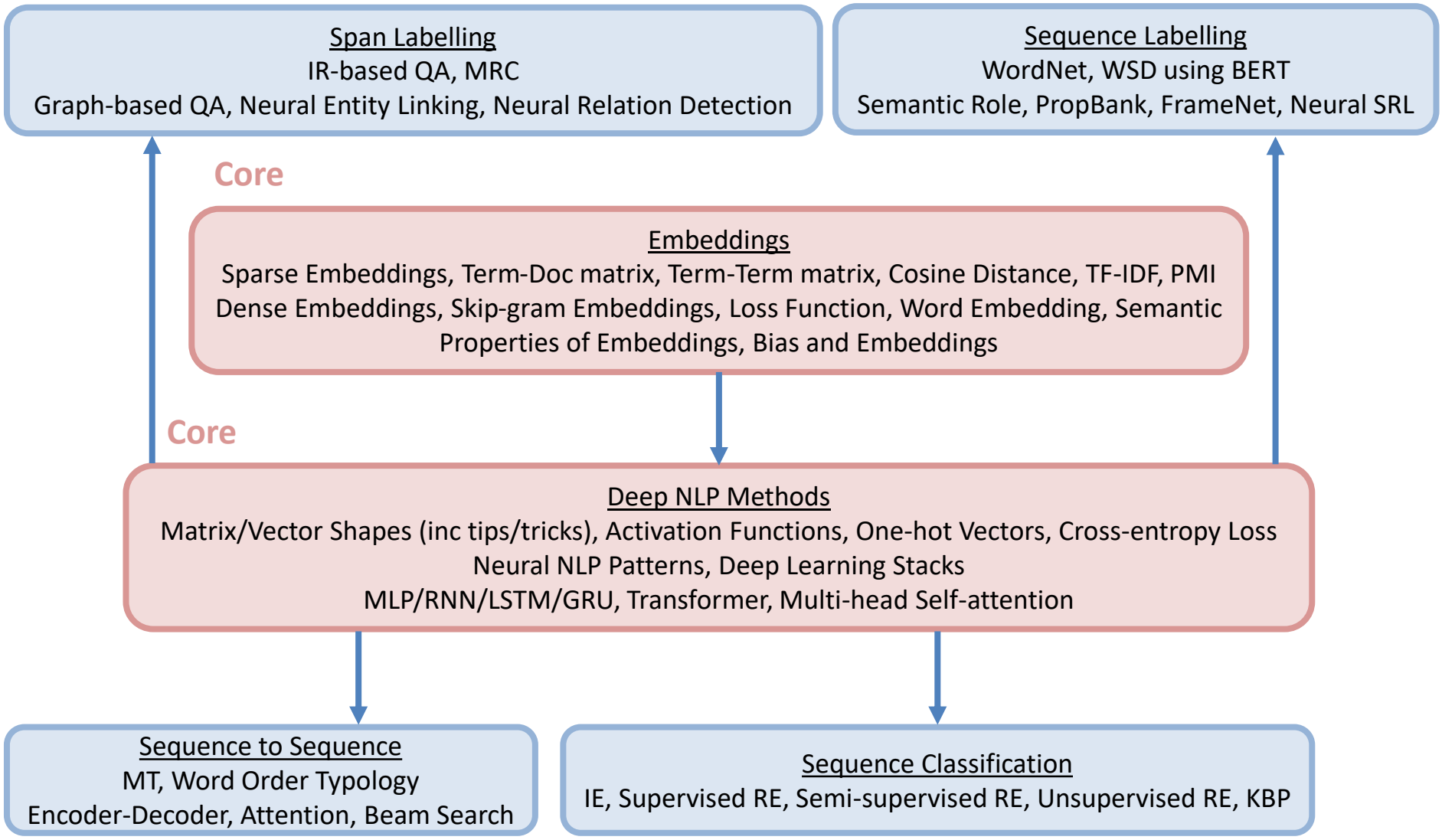


Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning ← flight)
7	[root, book, the, flight]	[]	LEFTARC	(the ← flight)
8	[root, book, flight]	[]	RIGHTARC	(book → flight)
9	[root, book]	[]	RIGHTARC	(root → book)
10	[root]	[]	Done	

Overview - Neural NLP Revision

- Landscape of Neural Content
- Vector Sementics (8)
- Word2Vec (10)
- RNN (11)
- Sequence Processing (12)
- Machine Translation (13)
- Word Sense and WordNet (17)
- Information Extraction (18)
- Semantic Role Labelling (19)
- Question Answering (20)

Landscape of Neural Content



Vector Semantics

- Representational learning is the automated learning of useful representations of text (as opposed to hand-crafted features)
- Vector semantics is the use of embeddings to represent word meaning
  - Embeddings are vectors represent words in a multidimensional space
  - Embeddings can be sparse (TF-IDF) or dense (word2vec)

2D projection of N dimensional embedding for sentiment analysis

Words and Vectors

- Term-document matrix
  - Row = word
  - Column = document
- Vector space model
  - Vector = array of numbers (word frequencies)
  - Vector space = collection of vectors (term-document matrix)
  - Dimension = size of vector (number of words in model vocabulary)

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Words and Vectors

- Term-term matrix
  - Row = word
  - Column = word occurring in same context
  - Context = document; N word window around word (left and/or right)

is traditionally followed by **cherry** pie, a traditional dessert
often mixed, such as **strawberry** rhubarb pie. Apple pie
computer peripherals and personal **digital** assistants. These devices usually
a computer. This includes **information** available on the internet

4 word window (left) 4 word window (right)

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

Word co-occurrence matrix ±4 word window using Wikipedia (term-term matrix)

Cosine for Measuring Similarity

- Dot product to find similarity (distance) between two vectors

$$\text{dot product}(\mathbf{v},\mathbf{w}) = \mathbf{v}\cdot\mathbf{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \ldots + v_N w_N$$

- Problem >> dot product favours longer vectors
- Normalized dot product by dividing by vector length (same as cosine of angle between vectors)

$$\mathbf{a}\cdot\mathbf{b} = |\mathbf{a}||\mathbf{b}|\cos\theta$$

$$\frac{\mathbf{a}\cdot\mathbf{b}}{|\mathbf{a}||\mathbf{b}|} = \cos\theta$$

$$\text{cosine}(\mathbf{v},\mathbf{w}) = \frac{\mathbf{v}\cdot\mathbf{w}}{|\mathbf{v}||\mathbf{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

TF-IDF

- Term frequency inverse document frequency (TF-IDF) is a balance between TF (terms which occur often) and IDF (terms which discriminate between documents well).
  - TF alone does not discriminate well
  - IDF alone picks terms that hardly ever occur (so in practice are useless)

$$w_{t,d} = \mathrm{tf}_{t,d} \times \mathrm{idf}_t$$

TF scores

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

TF-IDF scores

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

Pointwise Mutual Information

$$\text{PPMI}(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P(c)}, 0)$$

- Worked example

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

$$P(\text{w=information,c=data}) = \frac{3982}{11716} = .3399$$

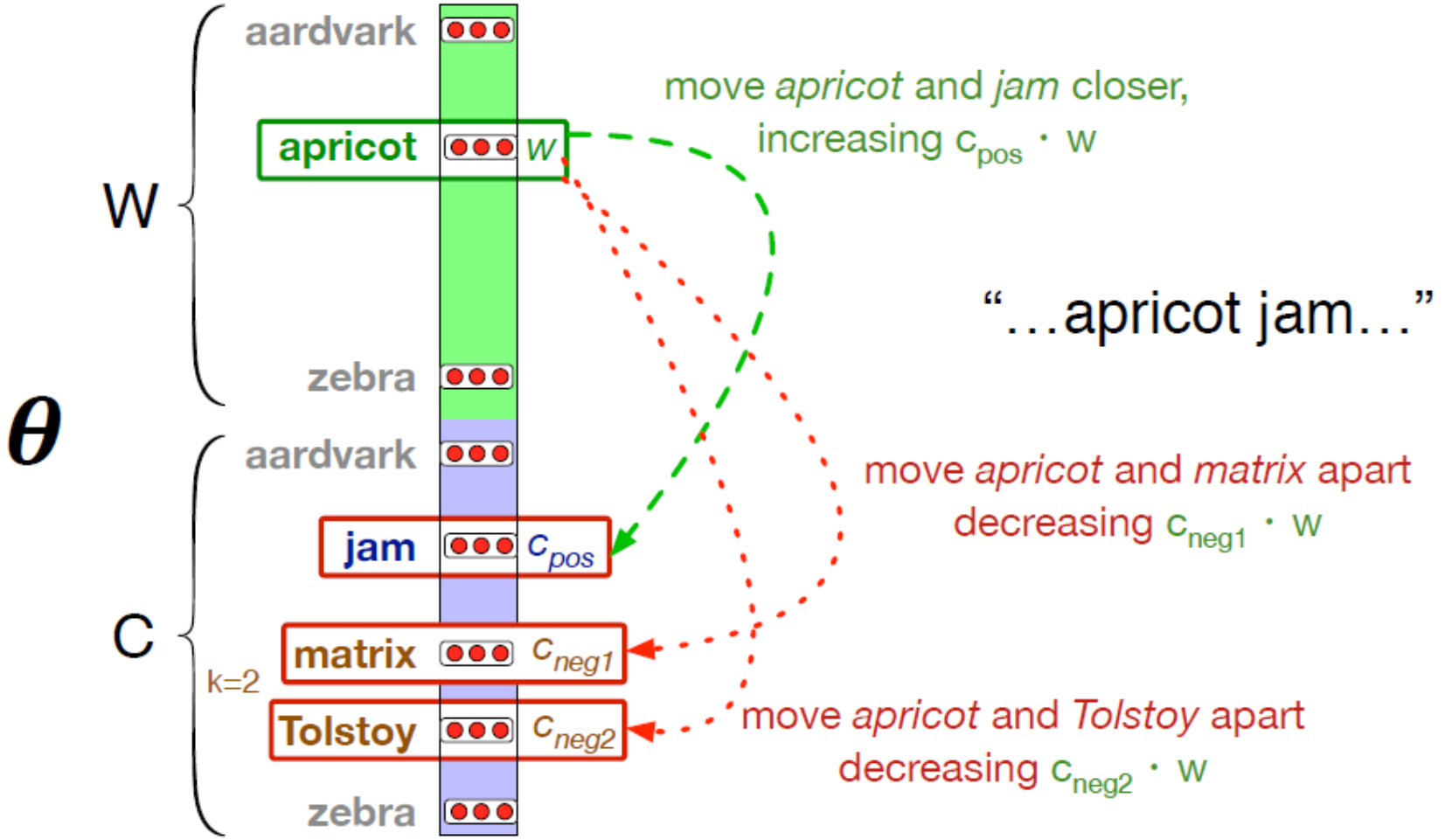
$$P(\text{w=information}) = \frac{7703}{11716} = .6575$$

$$P(\text{c=data}) = \frac{5673}{11716} = .4842$$

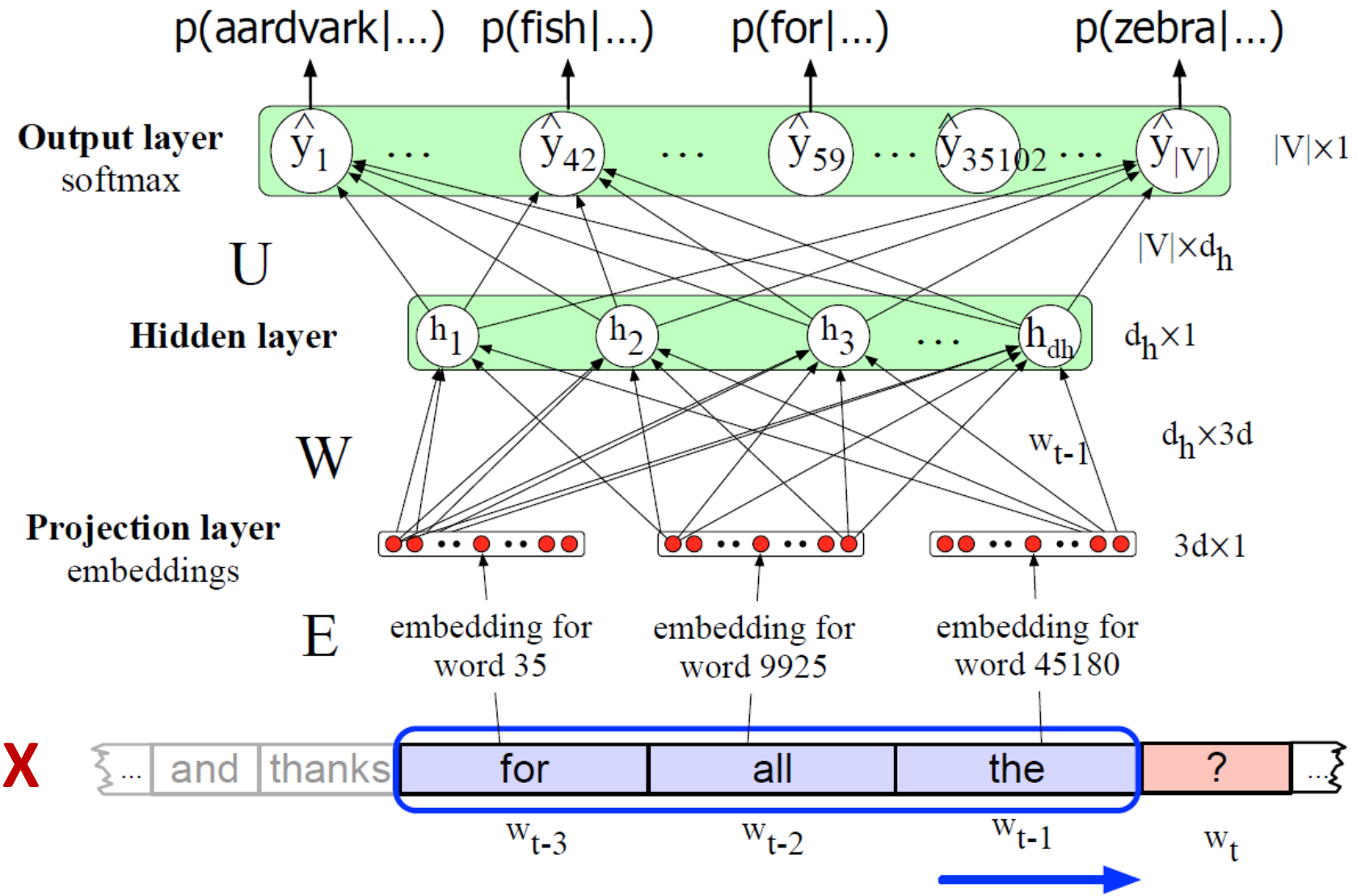
$$\text{ppmi(information,data)} = \log 2(.3399/(.6575 * .4842)) = .0944$$

Word2Vec

- Move 'apricot' weights closer to 'jam' and further from 'matrix'



Simple Recurrent Neural Networks



- MLP (not RNN) with sliding window 3 words and a hidden layer with dim d
  Separate patterns learnt for 'thanks for all', 'for all the' ...

Simple Recurrent Neural Networks

- Activation function g computes hidden layer values h_t
- Output vector is computed using a function f (often softmax)

$$
\begin{aligned}
h_t &= g(U h_{t-1} + W x_t) \\
y_t &= f(V h_t) \\
y_t &= \text{softmax}(V h_t)
\end{aligned}
$$

softmax maps vector of values to a probability distribution

- Weight matrix (current W, previous U) and input vector are multiplied together
- Weight matrix V (size = vocab) is multiplied with hidden layer values

function FORWARDRNN(x, *network*) **returns** output sequence y

$h_0 \leftarrow 0$
for $i \leftarrow 1$ **to** LENGTH(x) **do**
  $h_i \leftarrow g(U\ h_{i-1} + W\ x_i)$
  $y_i \leftarrow f(V\ h_i)$
return y

- Weights U, V and W are shared across time

Applications of RNNs

- Language model using RNN (e.g. predicting next word)
- Input X = sequence of L words in vocab V
  = each word x_t is a one-hot vector (dim = size of V)
  = matrix L x V

```
[0 0 0 0 1 0 0 ... 0 0 0 0]
 1 2 3 4 5 6 7 ...   ... |V|
```

'1' indicates the word is at index 5 within V

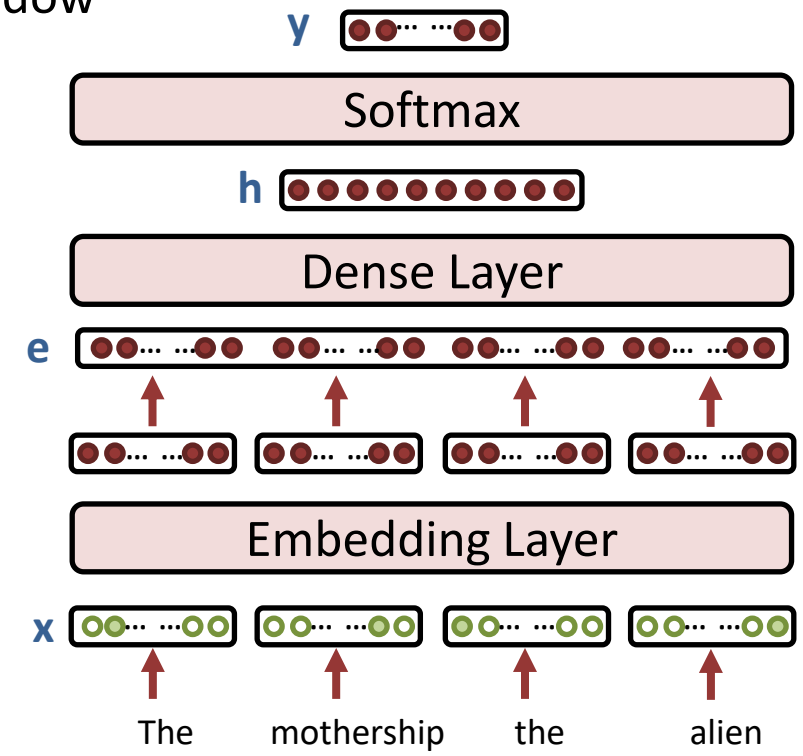
- Output Y = predicted next word in seq = probability dist over V
- E = Word embedding matrix (shape = one-hot dim x hidden dim d)

$$
\begin{aligned}
e_t &= E^T x_t \\
h_t &= g(U h_{t-1} + W e_t) \\
y_t &= \text{softmax}(V h_t)
\end{aligned}
$$

Dot product of matrix E + one-hot vector x_t = hidden layer values e_t
concat this with previous hidden layer values and apply softmax

Matrix shapes in NLP models

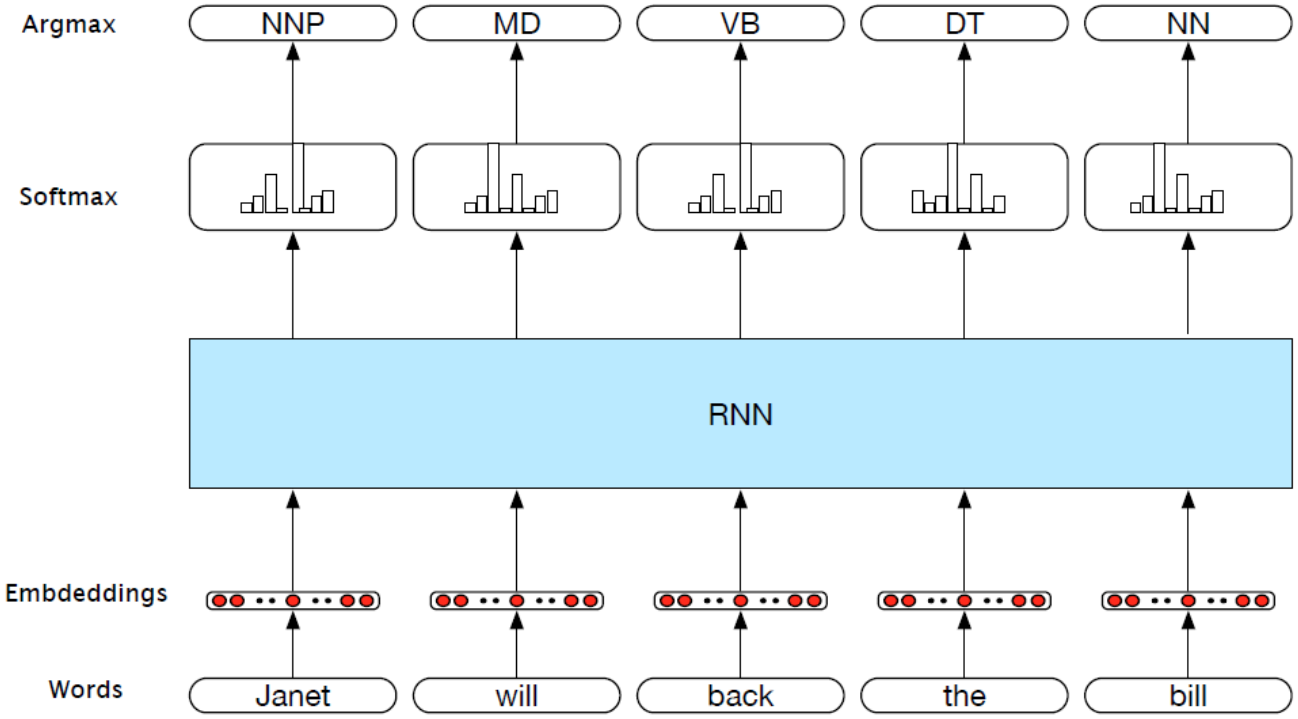
- Fixed-window neural language model (predict next word)
  - Training input = n words in an input window
  - Vocab size = V
  - d_e ; d_h = layer dim size
  - input x = n x vector(1 x V)
  - E = matrix(V x d_e)
  - output e = vector(1 x d_e*n)
  - W_e = matrix(d_e*n x d_h)
  - b_1 = vector(1 x d_h) = bias term
  - output h = vector(1 x d_h)
    >> $h_t = f(W_e e^t + b_1)$
  - U = matrix(d_h x V)
  - output y = vector(1 x V)
    >> y_t = softmax(Uh_t)



- J(θ)= cross entropy loss between predicted word (word probability distribution) and true word (one hot vector)
- Batched input (good idea). 3D matrices (batch_size x _ x _)

Applications of RNNs

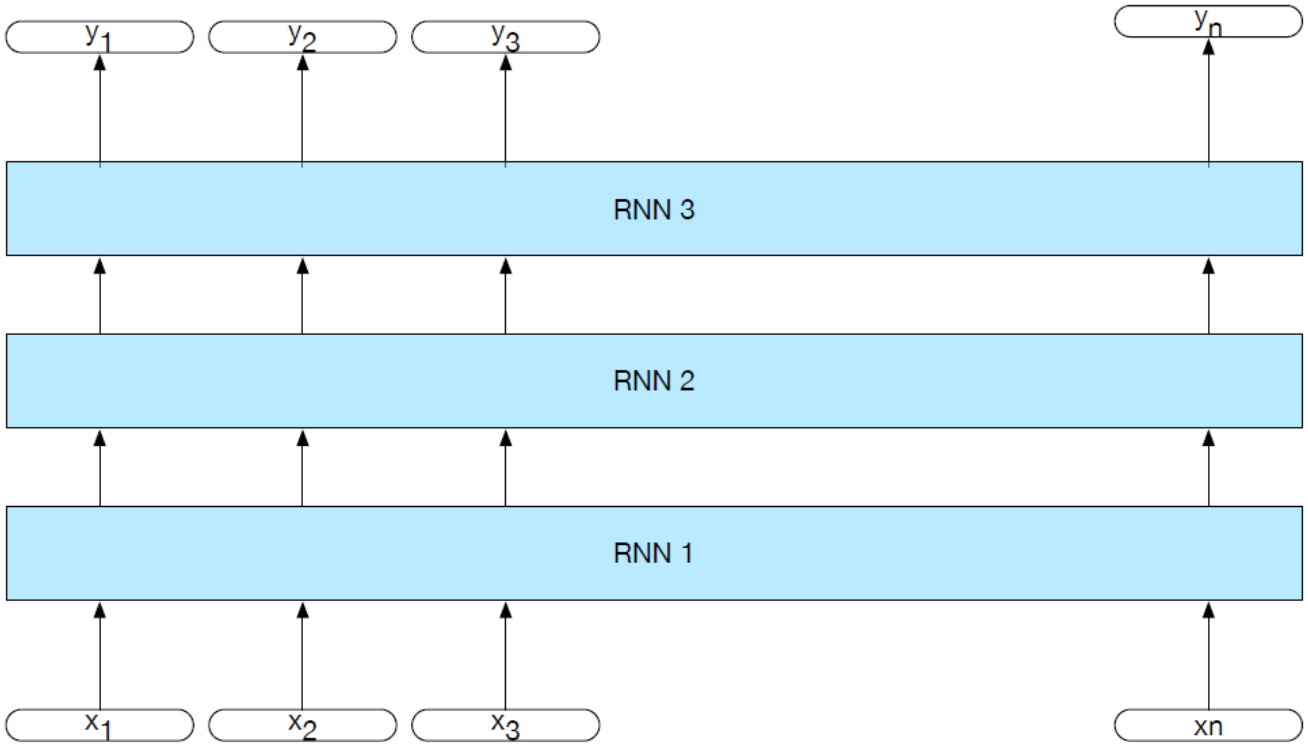
- Sequence labelling using RNN (e.g. POS tagging)



- Input X = sequence of words
- Output Y = POS tag probabilities (argmax chooses most likely)
- Pre-trained word embeddings
- Cross-entropy loss function

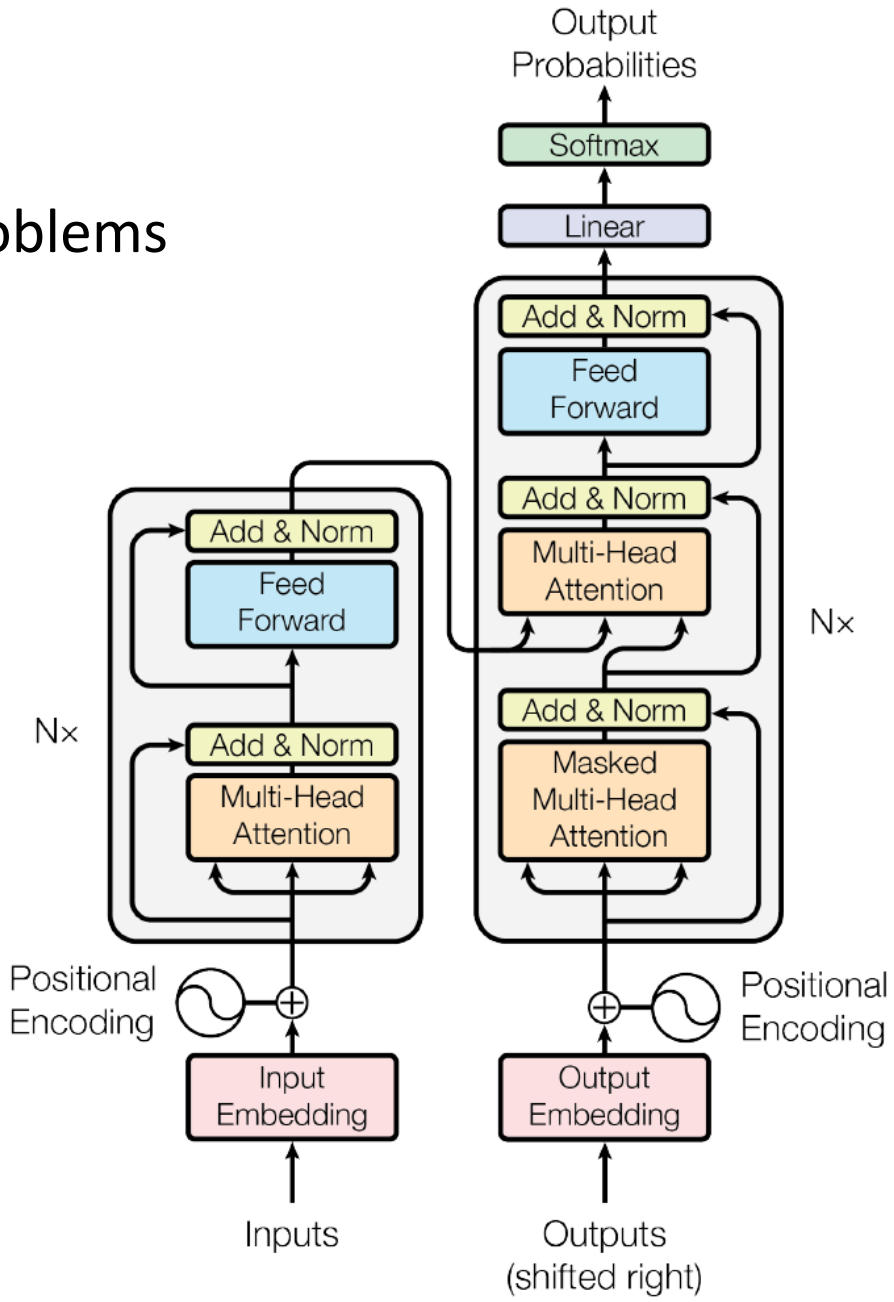
Stacked RNNs

- Stacked RNNs >> Deep Learning
- Entire output sequence of one RNN used as input to another
- Adding RNN layers boosts performance at expense of train time
- RNN layers encode different levels of abstract representations, allowing more sophisticated patterns to be encoded



Transformer

- Even LSTM and GRU's have problems with very long sequences
- Transformer architecture
  - Uses attention layers not sequential RNN layers
  - This architecture has led to many advances in NLP model performance



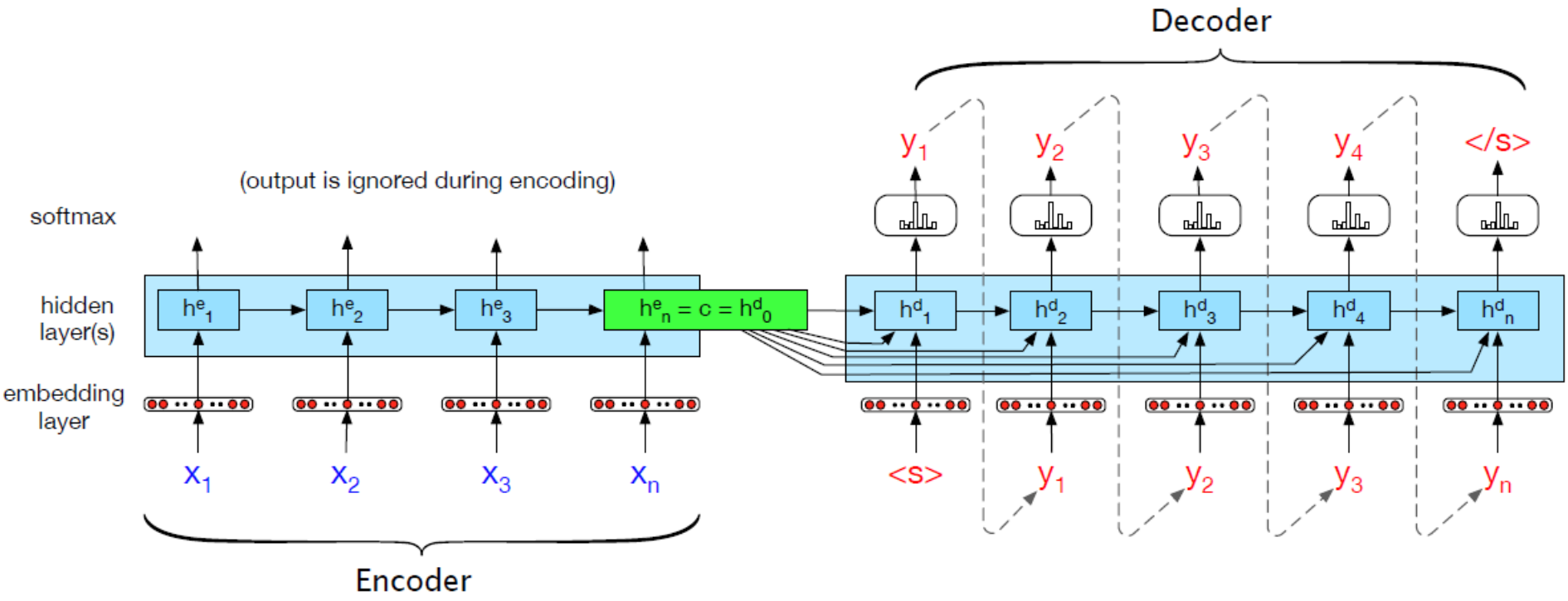
Cite: Vaswani, A. Shazeer, N. Parmar, N. Uszkoreit, J. Jones, L. Gomez, A.N. Kaiser, L. Polosukhin, I. (2017) Attention Is All You Need, arXiv:1706.03762v5

Machine Translation

- Sequence to Sequence (seq2seq) tasks
  - Input >> X >> Sequence of words
  - Output >> Y >> Sequence of words
  - len(X) != len(Y)
- MT can be formulated as a seq2seq task
- seq2seq MT has inspired seq2seq approaches for many NLP tasks
  - Russian text → English text
  - Russian speech → Russian transcript → English transcript
  - Russian Image (e.g. menu) → OCR Russian transcript → English transcript
  - Question → Answer
  - Sentence → Clause (relation + arguments)
  - Document → Abstract

    ... any sequence to sequence problem

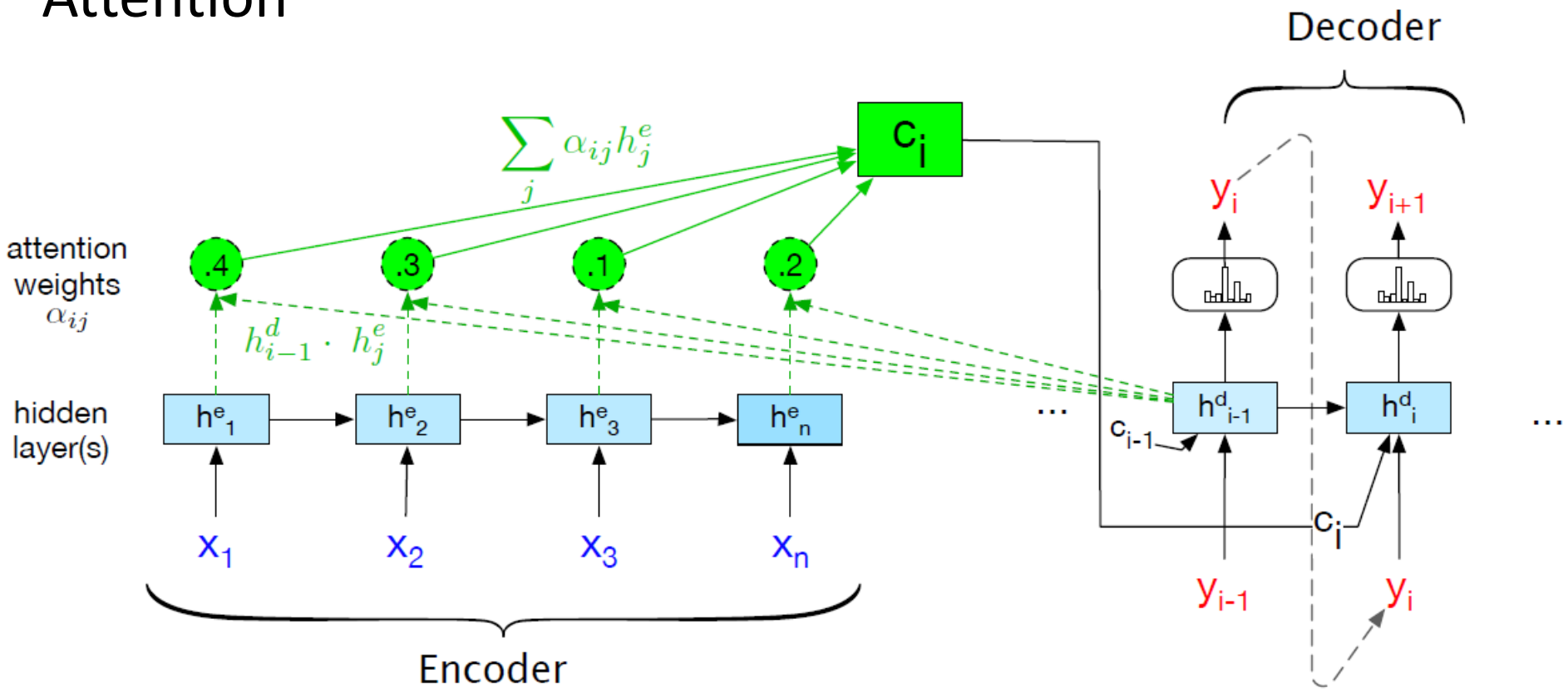
Encoder-Decoder Model



Train encoder hidden layers using X (current word)
encoder hidden layer $h_t = g(h_{t-1},x_t)$, g = activation function (e.g. ReLU)
Final encoder state h_n = context vector c

Train decoder hidden layers using c (<u>at every step</u>) and Y (previous word generated)
decoder hidden layer $h_t = g(c,h_{t-1},y_{t-1})$
... continue until end of sequence predicted

Attention



$$c_i = \sum_j \alpha_{ij} h_j^e$$

Context vector c is computed using a weighted sum of encoder hidden states
α_{ij} is the attention weight for decoder state i and encoder state j
α is defined by the chosen attention mechanism

WordNet

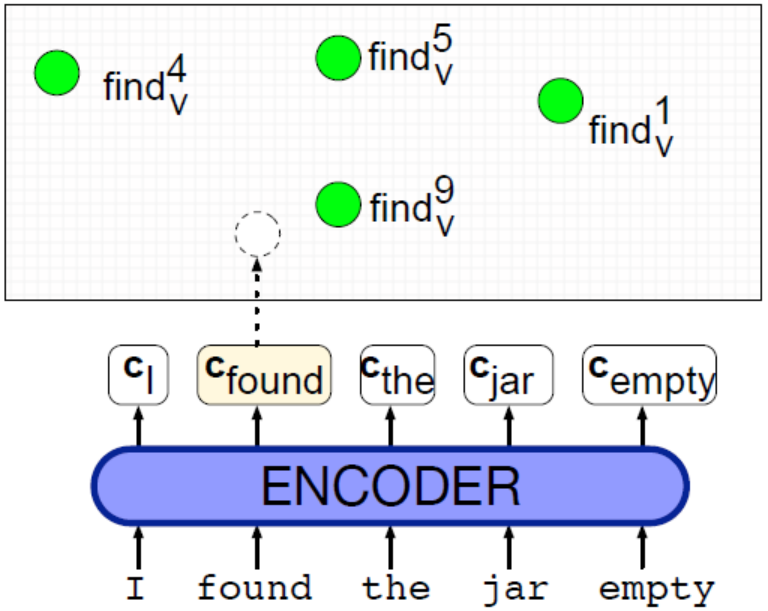
- Synset is a synonym set
- Supersense is a semantic category
  - 26 categories for noun; 15 for verb; 2 for adjective; 1 for adverb

Category	Example	Category	Example	Category	Example
ACT	*service*	GROUP	*place*	PLANT	*tree*
ANIMAL	*dog*	LOCATION	*area*	POSSESSION	*price*
ARTIFACT	*car*	MOTIVE	*reason*	PROCESS	*process*
ATTRIBUTE	*quality*	NATURAL EVENT	*experience*	QUANTITY	*amount*
BODY	*hair*	NATURAL OBJECT	*flower*	RELATION	*portion*
COGNITION	*way*	OTHER	*stuff*	SHAPE	*square*
COMMUNICATION	*review*	PERSON	*people*	STATE	*pain*
FEELING	*discomfort*	PHENOMENON	*result*	SUBSTANCE	*oil*
FOOD	*food*			TIME	*day*

26 supersenses for nouns in WordNet

Word Sense Disambiguation

- Baselines
  - Statistically most frequent sense in corpus
  - One sense per discourse (remember previous sense and use that)
- WSD classifier using contextual embeddings
  - Encoder = BERT embeddings (pre-trained, embedding weights fixed)
  - Training phase
    >> average (across corpus) each token_embedding for a particular word sense to get a sense_embedding
  - Testing phase
    >> nearest neighbour classifier
    >> cosine distance between input token_embedding and learnt sense_embeddings



Information Extraction

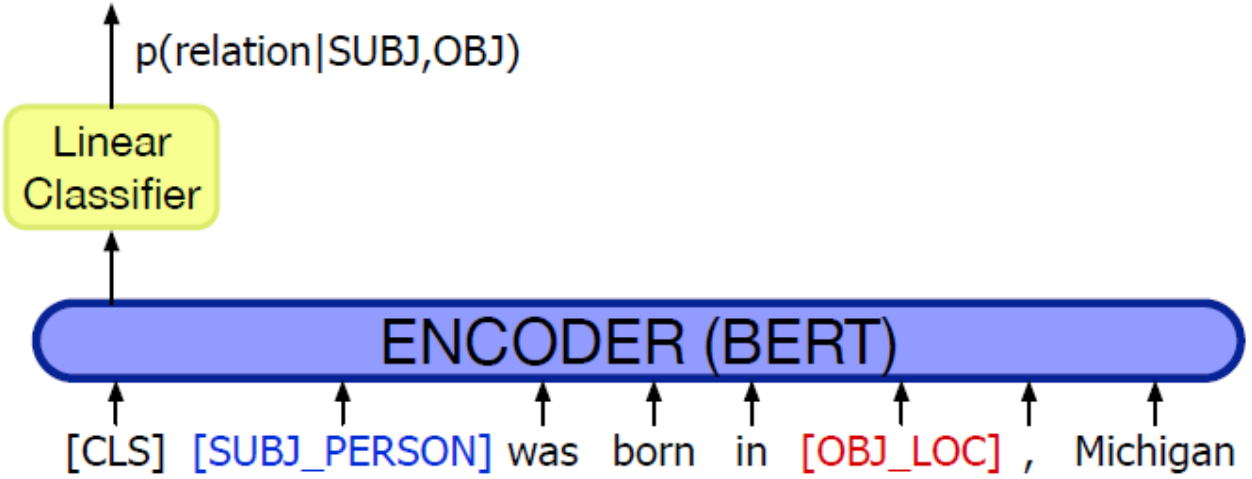
- Information Extraction turns unstructured text into structured data, such as a relational database or set of extracted tuples
- Relation Extraction (RE) finds semantic relations among text entities, such as parent-child, part-whole or geospatial relations
  - Relation words are typically action verb focused and often have noun phrase arguments
  - Knowledge-graphs can be constructed to encode relational information

    American Airlines supported the move by chairman Wagner
    American Airlines **supported** the move by chairman Wagner
    supported(American Airlines, the move by chairman Wagner)

- Event Extraction finds events in which entities participate
- Temporal Extraction finds times and dates
- Knowledge Base Population (KBP) populates knowledge bases from unstructured text using extracted information

Pattern-based and Supervised Relation Extraction

- Supervised RE
  - Find pairs for named entities (in same sent) + classify relation word for pair
  - Any supervised ML >> logistic regression; random forest; RNN; Transformer
- Example >> BERT encoder + classifier
  - Source sent >> NER >> select NER pair >> create feature template
    entity 1 (phrase, NER type, context window around entity) features +
    entity 2 (phrase, NER type, context window around entity) features +
    features for words in-between entities
  - X = Feature set for the entity pair (embedding vector or one-hot vector)
  - Y = Prediction of relation (for provided entity pair) from 42 TAC relations (and a class for no relation)



Semantic Roles

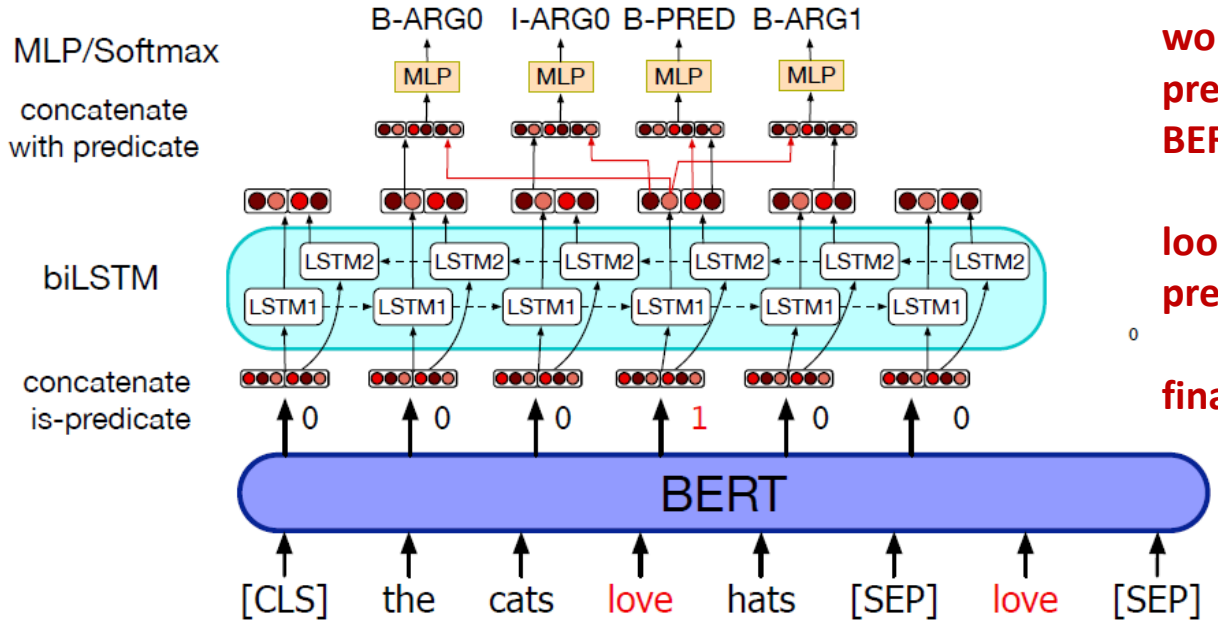
- A semantic role is the role that arguments of a predicate take in an event
- Datasets >> PropBank, FrameNet
- Semantic role labelling is the task of assigning roles to spans of text in sentences
- A thematic role captures the semantic commonality around participants of event
  - No single definition - common examples below (typically around 12 roles)

Thematic Role	Definition	Example
AGENT	The volitional causer of an event	*The waiter* spilled the soup.
EXPERIENCER	The experiencer of an event	*John* has a headache.
FORCE	The non-volitional causer of the event	*The wind* blows debris from the mall into our yards.
THEME	The participant most directly affected by an event	Only after Benjamin Franklin broke *the ice*...
RESULT	The end product of an event	The city built a *regulation-size baseball diamond*...
CONTENT	The proposition or content of a propositional event	Mona asked *"You met Mary Ann at a supermarket?"*
INSTRUMENT	An instrument used in an event	He poached catfish, stunning them *with a shocking device*...
BENEFICIARY	The beneficiary of an event	Whenever Ann Callahan makes hotel reservations *for her boss*.
SOURCE	The origin of the object of a transfer event	I flew in *from Boston*.
GOAL	The destination of an object of a transfer event	I drove *to Portland*.

AGENT → ARG [The waiter] PREDICATE[spilled] ARG[the soup]

Semantic Role Labelling

- Neural SRL can be formulated as a sequence labelling task
  - X = [CLS] sent ... [SEP] predicate [SEP]
  - Y = BIO tags for SRL from PropBank (e.g. B-ARG0; I-ARG0 ...)
  - BiLSTM encoder
    - Concatenate (a) BERT word embedding and (b) predicate flag embedding
    - Could use other layers than BiLSTM (e.g. Transformer)
  - MLP + Softmax decoder
    - Concatenate (a) hidden layer output for word and (b) hidden layer output for predicate
    - Could use other layers than MLP (e.g. CRF layer)



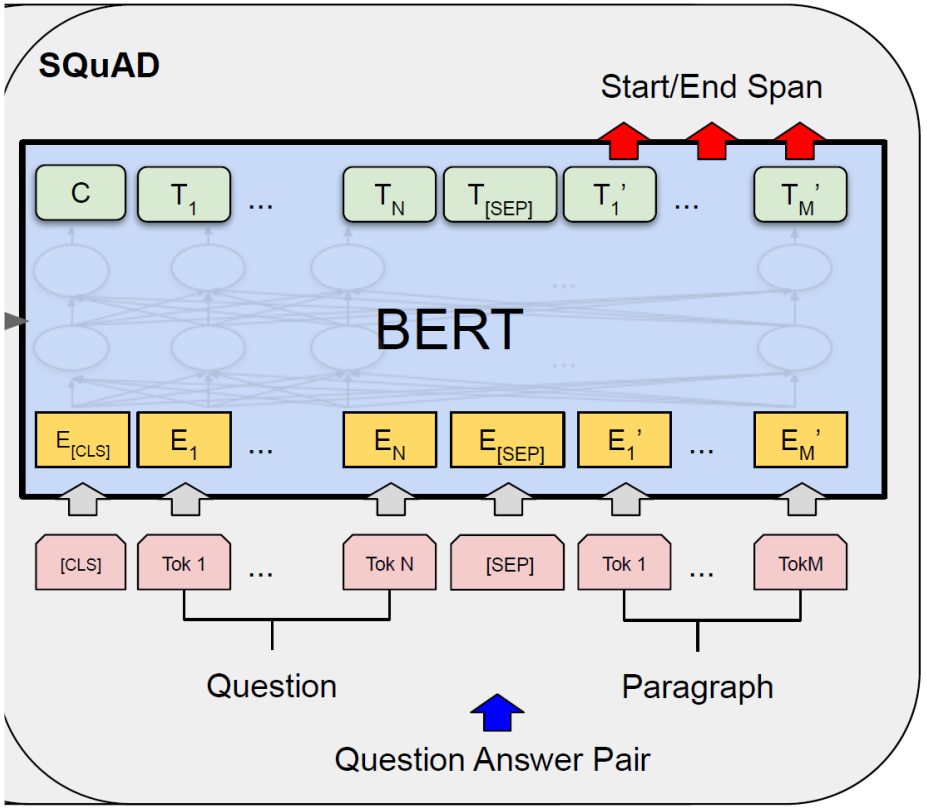
word (seq pos 4) = 'love'
predicate flag = True = 1
BERT vocab size = V

lookup BERT embedding = shape(1,V)
predicate value vector = shape(1,1)

final input embedding = shape(1,V+1)

IR-based factoid QA

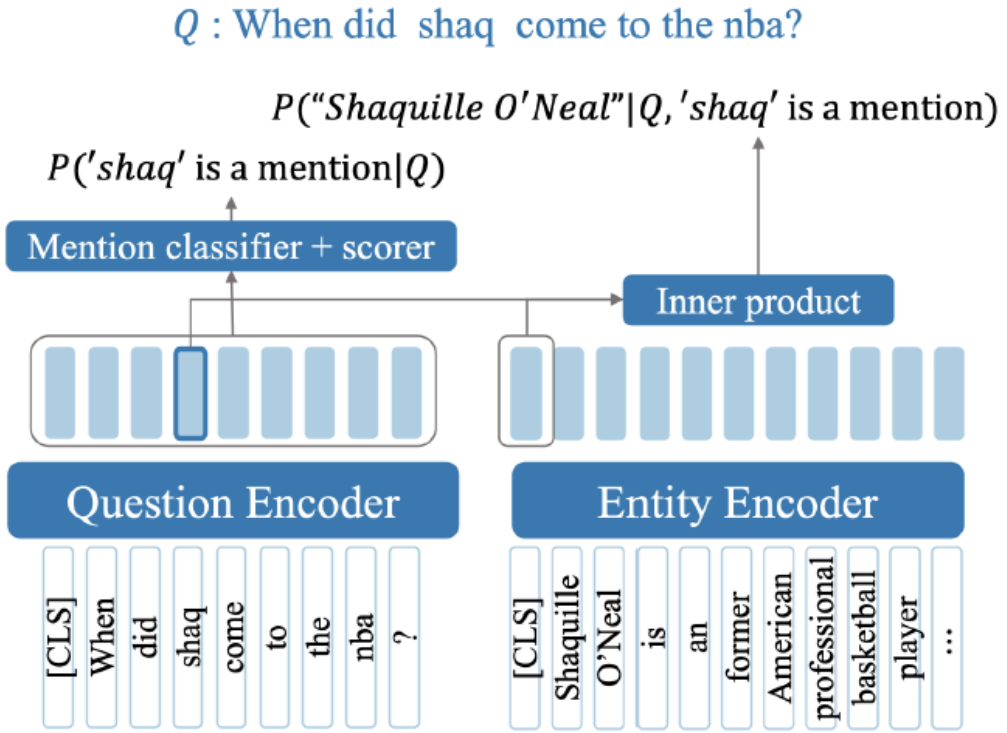
- Machine Reading Comprehension (MRC)
  - Answer extraction is a span labelling task (extractive in this model)
  - X = question and passage = [CLS] question ... [SEP] paragraph ...
  - Y = answer start = is_start (0/1)
  - Y = answer end = is_end (0/1)



Cite: Devlin 2019, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding https://arxiv.org/abs/1810.04805

Knowledge-based QA

- Neural Entity Linking
  EQL model



- X = (question, entity candidate desc)
  question encoder = BERT([CLS] question ... [SEP])
  entity encoder = BERT([CLS] entity title ... [ENT] entity desc ... [SEP])
- Y = entity mention classifier = is_start(0/1); is_end(0/1); is_entity(0/1)
  Y = entity linker = KB_entity (for entity mention in question)
- Jointly train entity mention classifier and entity linker
- The entity linker compares (a) entity mention embedding and (b) entity candidate embedding to provide a disambiguated KB_entity for each question entity text span

Cite: Li, B. Z. et. al. (2020). Efficient one-pass end-to-end entity linking for questions. EMNLP https://www.aclweb.org/anthology/2020.emnlp-main.522/

Be smart - plan your revision - give yourself enough time
Good luck!