MLE (Maximum Likelihood Estimation)

$P_{MLE}(w_1 \dots w_n) = \frac{C(w_1 \dots w_n)}{N}$

$P_{MLE}(w_n \mid w_1 \dots w_{n-1}) = \frac{C(w_1 \dots w_n)}{C(w_1 \dots w_{n-1})}$

- No probability mass for unseen events
- Data sparseness, Zipf's Law
- Unsuitable for NLP (widely used, though)

Zipf's Laws (1929)

- Word frequency is inversely proportional to its rank (speaker/hearer minimum effort) $f \sim 1/r$
- Number of senses is proportional to frequency root $m \sim \sqrt{f}$
- Frequency of intervals between repetitions is inversely proportional to the length of the interval $F \sim 1/I$
- Frequency based approaches are hard, since most words are rare
  - Most common 5% words account for about 50% of a text
  - 90% least common words account for less than 10% of the text
  - Almost half of the words in a text occur only once

Linear Discounting (discount a proportion α of counts)

General rule:

$$P_{LIN}(X=x) = \begin{cases} (1-\alpha)\frac{C(X=x)}{N} & \text{if } C(X=x) > 0 \\ \alpha/N_0 & \text{otherwise} \end{cases}$$

N_0: *number of possible values for X observed 0 times*

N-gram probability:

$$P_{LIN}(w_1 \dots w_n) = \begin{cases} (1-\alpha)\frac{C(w_1 \dots w_n)}{N} & \text{if } C(w_1 \dots w_n) > 0 \\ \alpha/N_0 & \text{otherwise} \end{cases}$$

N_0: *number of possible n-grams observed 0 times*

N-gram conditional probability:

$$P_{LIN}(w_n \mid w_1 \dots w_{n-1}) = \begin{cases} (1-\alpha)\frac{C(w_1 \dots w_n)}{C(w_1 \dots w_{n-1})} & \text{if } C(w_1 \dots w_n) > 0 \\ \alpha/N_0 & \text{otherwise} \end{cases}$$

N_0: *number of possible values for w_n observed 0 times*

Laplace's Law (adding one count)

General rule:

$$P_{LAP}(X=x) = \frac{C(X=x)+1}{N+B}$$

N: *number of observations of X*
B: *number of potentially observable values for X*

N-gram probability:

$$P_{LAP}(w_1 \dots w_n) = \frac{C(w_1 \dots w_n)+1}{N+B}$$

N: *total number of n-gram observations*
B: *number of potentially observable different n-grams*

N-gram conditional probability:

$$P_{LAP}(w_n \mid w_1 \dots w_{n-1}) = \frac{C(w_1 \dots w_n)+1}{C(w_1 \dots w_{n-1})+B}$$

B: *number of potentially observable w_n values*

For large values of B too much probability mass is assigned to unseen events.

N = num training examples, words or characters
B = set of words or characters to the power of the grams, i.e. bigram B^2, trigram B^3 etc.
N_0 = B-N all possible combinations minus the ones that were observed

Lidstone's Law (adding λ counts, with $\lambda < 1$)

General rule:

$$P_{LAP}(X=x) = \frac{C(X=x)+\lambda}{N+B\lambda}$$

N: *number of observations of X*
B: *number of potentially observable values for X*

N-gram probability:

$$P_{LAP}(w_1 \dots w_n) = \frac{C(w_1 \dots w_n)+\lambda}{N+B\lambda}$$

N: *total number of n-gram observations*
B: *number of potentially observable different n-grams*

N-gram conditional probability:

$$P_{LAP}(w_n \mid w_1 \dots w_{n-1}) = \frac{C(w_1 \dots w_n)+\lambda}{C(w_1 \dots w_{n-1})+B\lambda}$$

B: *number of potentially observable w_n values*

Equivalent to linear interpolation between MLE and uniform prior:
with $\mu = N/(N+B\lambda)$; $P_{LID}(X=x) = \mu\frac{C(X=x)}{N} + (1-\mu)\frac{1}{B}$

Absolute Discounting (discount δ counts, with $0 < \delta < 1$)

General rule:

$$P_{ABS}(X=x) = \begin{cases} \frac{C(X=x)-\delta}{N} & \text{if } C(w_1 \dots w_n) > 0 \\ \frac{(B-N_0)\delta/N_0}{N} & \text{otherwise} \end{cases}$$

N_0: *number of possible values for X observed 0 times*

N-gram probability:

$$P_{ABS}(w_1 \dots w_n) = \begin{cases} \frac{C(w_1 \dots w_n)-\delta}{N} & \text{if } C(w_1 \dots w_n) > 0 \\ \frac{(B-N_0)\delta/N_0}{N} & \text{otherwise} \end{cases}$$

N-gram conditional probability:

$$P_{ABS}(w_n \mid w_1 \dots w_{n-1}) = \begin{cases} \frac{C(w_1 \dots w_n)-\delta}{C(w_1 \dots w_{n-1})} & \text{if } C(w_1 \dots w_n) > 0 \\ \frac{(B-N_0)\delta/N_0}{C(w_1 \dots w_{n-1})} & \text{otherwise} \end{cases}$$

N_0: *number of possible values for w_n observed 0 times*

- Overlap.

$$sim_{ovl}(X,Y) = \frac{|X \cap Y|}{\min(|X|,|Y|)} = \frac{a}{\min(a+b, a+c)}$$

- Cosine.

$$sim_{cos}(X,Y) = \frac{|X \cap Y|}{\sqrt{|X|} \cdot \sqrt{|Y|}} = \frac{a}{\sqrt{(a+b)}\sqrt{(a+c)}}$$

- Matching Coefficient

$$sim_{mc}(X,Y) = \frac{|X \cap Y| + |(\Omega - X) \cap (\Omega - Y)|}{|\Omega|} = \frac{a+d}{a+b+c+d}$$

- Dice.

$$sim_{dic}(X,Y) = \frac{2 \cdot |X \cap Y|}{|X| + |Y|} = \frac{2a}{2a+b+c}$$

- Jaccard.

$$sim_{jac}(X,Y) = \frac{|X \cap Y|}{|X \cup Y|} = \frac{a}{a+b+c}$$

similarity to distance and distance to similarity

$$sim(A,B) = \frac{1}{1+d(A,B)};$$

$$d(A,B) = \frac{1}{sim(A,B)} - 1$$

Distances:

Dot sim

$$sim_{dot}(\vec{x}, \vec{y}) = \vec{x} \cdot \vec{y} = \sum_i x_i y_i$$

Minkowski r=1 L1 and r=2 L2

$$\left(\sum_{i=1}^{N} |x_i - y_i|^r\right)^{\frac{1}{r}}$$

Cosine sim

$$\frac{\vec{x} \cdot \vec{y}}{|\vec{x}| \cdot |\vec{y}|} = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2} \cdot \sqrt{\sum_i y_i^2}}$$

Camberra distance:

$$\sum_{i=1}^{N} \frac{|x_i - y_i|}{|x_i + y_i|}$$

Feature Templates

Features for classification

- Type 1: The entity is word a:

$$f_{1,l,a}(x_{1:n}, i, j, y) = \begin{cases} 1 & \text{if } i = j \text{ and } x_i = a \text{ and } y = l \\ 0 & \text{otherwise} \end{cases}$$

- Type 2: All entity words are capitalized:

$$f_{2,l}(x_{1:n}, i, j, y) = \begin{cases} 1 & \text{if } \forall k : i \le k \le j : \mathit{capitalized}(x_k) \text{ and } y = l \\ 0 & \text{otherwise} \end{cases}$$

- Type 3: The entity contains word a:

$$f_{3,l,a}(x_{1:n}, i, j, y) = \begin{cases} 1 & \text{if } \exists k : i \le k \le j : x_k = a \text{ and } y = l \\ 0 & \text{otherwise} \end{cases}$$

- Type 4: The entity has at least three words, the first is a and the second is b:

$$\{\ 1 \quad j \ge i+2 \text{ and } x_i = a \text{ and } x_{i+1} = b \text{ and } y = l$$

factored linear models of the previous exercises:

Viterbi to check features bigrams

$$f(\mathbf{x}_{1:n}) = \operatorname*{argmax}_{\mathbf{y}_{1:n} \in \mathcal{Y}^n} \sum_{i=1}^{n} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i)$$

In order to compute $f(\mathbf{x}_{1:n})$ we can use the Viterbi algorithm as follows:

- *Define $\delta_i(a)$ to be the score of optimal sequence for $\mathbf{x}_{1:i}$ ending with $a \in \mathcal{Y}$:*

$$\delta_i(a) = \max_{\mathbf{y}_{1:i} \in \mathcal{Y}^i : \mathbf{y}_i = a} \sum_{j=1}^{i} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, j, \mathbf{y}_{j-1}, \mathbf{y}_j)$$

- *Use the following recursions, for all $a \in \mathcal{Y}$:*

$$\begin{aligned} \delta_1(a) &= \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, 1, \text{START}, a) \quad (\textit{initialization, assuming } y_0 = \text{START}) \\ \delta_i(a) &= \max_{b \in \mathcal{Y}} \delta_{i-1}(b) + \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, b, a) \quad (\textit{recursion, } \forall i > 1) \\ \gamma_i(a) &= \operatorname*{argmax}_{b \in \mathcal{Y}} \delta_{i-1}(b) + \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, b, a) \quad (\textit{backtrace, } \forall i > 1) \end{aligned}$$

- *The optimal score for $\mathbf{x}$ is $\max_{a \in \mathcal{Y}} \delta_n(a)$*
- *The optimal sequence $\hat{\mathbf{y}}$ can be recovered through the backpointers γ*

Viterbi to check features trigrams

$$\begin{aligned} \delta_1(START, a) &= \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, 1, \text{START}, \text{START}, a) \quad (\text{initialization, } y_{-1} = y_0 = \text{START}) \\ \delta_i(b, a) &= \max_{c \in \mathcal{Y}_{i-2}} \delta_{i-1}(c, b) + \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, c, b, a) \quad (\text{recursion, } \forall i > 1) \\ \gamma_i(b, a) &= \operatorname*{argmax}_{c \in \mathcal{Y}_{i-2}} \delta_{i-1}(c, b) + \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, c, b, a) \quad (\text{backtrace, } \forall i > 1) \end{aligned}$$

PCFG
get probability of dep tree by just multiplying all probabilites

CNF
right-hand side exactly 2 non-terminals or one terminal

Tree Parsing

(b) CKY chart:

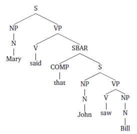
				15 0.00448 S → $N_{11}VP_{25}$ (0.35 × 0.4 × 0.032)
			14	25 0.032 VP → $V_{22}ADVP_{35}$ (0.4 × 0.5 × 0.16)
		13	24	35 0.16 ADVP → $ADV_{33}NP_{45}$ (0.65 × 1.0 × 0.246) 0.048 VP → $V_{33}NP_{45}$ (0.39 × 0.5 × 0.246)
	12 0.0308 NP → $N_{11}N_{22}$ (0.385 × 0.4 × 0.2)	23	34	45 0.246 NP → $D_{44}N_{55}$ (0.615 × 1.0 × 0.4)
11 0.4 N → time	22 0.2 N → flies 0.5 V → flies	33 1.0 ADV → like 0.5 V → like	44 1.0 D → an	55 0.4 N → arrow
time	flies	like	an	arrow

(Blue rule in cell 35 indicates the most likely subtree selected in that cell)

chart[0]	chart[1]	chart[2]	chart[3]	chart[4]	chart[5]
					[0,5] S → $NP_{01}VP_{15}$• S → $NP_{02}VP_{25}$•
				[0,4]	[1,5] VP → $V_{12}ADVP_{25}$•
			[0,3]	[1,4]	[2,5] ADVP → $ADV_{23}NP_{35}$• VP → $V_{23}NP_{35}$•
		[0,2] NP → $N_{01}N_{12}$• S → NP_{02} • VP	[1,3]	[2,4]	[3,5] NP → $D_{34}N_{45}$•
	[0,1] N → time• NP → N_{01} • N NP → N_{01}• S → NP_{01} • VP	[1,2] N → flies• V → flies• VP → V_{12} • NP VP → V_{12} • ADVP	[2,3] ADV → like• V → like• ADVP → ADV_{23} • NP VP → V_{23} • NP VP → V_{23} • ADVP	[3,4] D → an• NP → D • N	[4,5] N → arrow•
[0,0] γ → •S S → •NP VP NP → •N N NP → •D N NP → •N	[1,1] VP → •V NP VP → •V ADVP	[2,2] NP → •N N NP → •D N NP → •N ADVP → •ADV NP VP → •V NP VP → •V ADVP	[3,3] NP → •N N NP → •D N NP → •N ADVP → •ADV NP	[4,4]	[5,5]
0	1	2	3	4	5
	time	flies	like	an	arrow

Early algorithm: start with all possible rules until the last non-terminal.
put point on the first right side constituent diagonally up [0,1], check/scan first terminal word and compare it to input, what fits of the terminal elements.
put a point to the next constituent on the right side,
go down [1,1] and predict all constituents with a point to their left.
Second row bottom up is always for terminal symbols.
above is the rules that are combined, below the ones that are predicted and then to be scanned

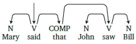
Given the sentence *Mary said that John saw Bill*, with the following parse tree:



And the following grammar rules (where the superscript + indicates the head):

$S \rightarrow NP\ VP^+$
$NP \rightarrow N$
$VP \rightarrow V^+\ NP$
$VP \rightarrow V^+\ SBAR$
$SBAR \rightarrow COMP^+\ S$

from the conversion using the given head rules.

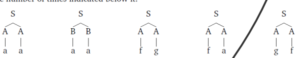


invalid parses if node has two arrows coming in (except root node) and is fully connected and no cycles
projective parses all those that have no crossing lines

transitive vs intransitive

$Vi \rightarrow sleeps$
$Vt \rightarrow saw$

Consider that you have as a training corpus a treebank containing the following trees. Each tree was observed the number of times indicated below it.



75 10 325 8 428

1. What PCFG would one get from this treebank (using MLE)?

Levenshtein

```
subst = 0 if s[i-1] == t[j-1] else 1   # substitution cost

d[i][j] = min(d[i-1][j] + 1,           # deletion
              d[i][j-1] + 1,           # insertion
              d[i-1][j-1] + subst)     # substitution
```

S → A A	$1 \times 75 + 0 \times 10 + 1 \times 325 + 1 \times 8 + 1 \times 428 = 836$
S → B B	$0 \times 75 + 1 \times 10 + 0 \times 325 + 0 \times 8 + 0 \times 428 = 10$
S → anything	$1 \times 75 + 1 \times 10 + 1 \times 325 + 1 \times 8 + 1 \times 428 = 846$
B → a	$0 \times 75 + 2 \times 10 + 0 \times 325 + 0 \times 8 + 0 \times 428 = 20$
B → anything	$0 \times 75 + 2 \times 10 + 0 \times 325 + 0 \times 8 + 0 \times 428 = 20$
A → a	$2 \times 75 + 0 \times 10 + 0 \times 325 + 1 \times 8 + 0 \times 428 = 158$
A → f	$0 \times 75 + 0 \times 10 + 1 \times 325 + 1 \times 8 + 1 \times 428 = 761$
A → g	$0 \times 75 + 0 \times 10 + 1 \times 325 + 0 \times 8 + 1 \times 428 = 753$
A → anything	$2 \times 75 + 0 \times 10 + 2 \times 325 + 2 \times 8 + 2 \times 428 = 1672$

Thus, the MLE probability for each rule would be:

$P(S \rightarrow A\ A)$	=	$P(AA \mid S)$	=	$\#(S \rightarrow A\ A)/\#(S \rightarrow \text{anything})$	=	$836/846 = 0.988$
$P(S \rightarrow B\ B)$	=	$P(BB \mid S)$	=	$\#(S \rightarrow B\ B)/\#(S \rightarrow \text{anything})$	=	$10/846 = 0.012$
$P(B \rightarrow a)$	=	$P(a \mid B)$	=	$\#(B \rightarrow a)/\#(B \rightarrow \text{anything})$	=	$20/20 = 1.000$
$P(A \rightarrow a)$	=	$P(a \mid A)$	=	$\#(A \rightarrow a)/\#(A \rightarrow \text{anything})$	=	$158/1672 = 0.095$
$P(A \rightarrow f)$	=	$P(f \mid A)$	=	$\#(A \rightarrow f)/\#(A \rightarrow \text{anything})$	=	$761/1672 = 0.455$
$P(A \rightarrow g)$	=	$P(g \mid A)$	=	$\#(A \rightarrow g)/\#(A \rightarrow \text{anything})$	=	$753/1672 = 0.450$

Count based tasks:

We have a sample of 1,500 annotated product descriptions. 200 are classified as ELEC, 350 as COMP, 650 as FASH, and 300 as TOOL.

160 products contain the word waterproof. 80 are annotated as FASH, 30 as ELEC, and 50 as TOOL.
• 110 products contain the word handmade. 95 are annotated as FASH, and 15 as TOOL.

• Probability that a Computer product contains word display, PMLE(display|COMP)
• Probability that a Computer product contains word handmade, PMLE(handmade|COMP)

$P_{MLE}(display \mid COMP) =$

$P_{MLE}(handmade \mid COMP) =$

$$\frac{\#(COMP \wedge display)}{\#COMP} = \frac{20}{350} \quad (3)$$

$$\frac{\#(COMP \wedge handmade)}{\#COMP} = \frac{0}{350} \quad (3)$$

Word embeddings

One hot vector (dim == vocabulary size)
- Very large vector (millions of words in some applications)
- Sparse, orthogonal representations
- No information about how words are related
- No useful vector distance
- Huge use of memory (if sparse matrices are not used)
- Usual coding of categorical variables for Linear models and SVMs with the standard kernels

Word2Vec:
Cbow: given context words predict target word
Skipgram: given target word predict context words
Dot-product for score vector, softmax output, cross entropy loss, minimize negative log likelihood

<u>TF-IDF</u>

$$TF(t,d) = \frac{|\{x \in d : x = t\}|}{|d|}$$

number occurences divdided by number of words in doc

$$IDF(t,\mathcal{D}) = \log\left(\frac{|\mathcal{D}|}{|\{d \in \mathcal{D} : t \in d\}|}\right)$$

TF*IDF

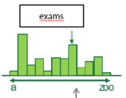
|D| num total Docs, below frac: num doc where this term occurs

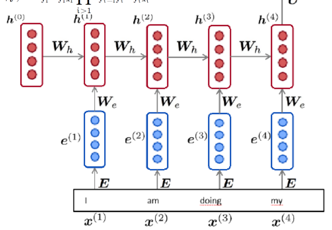
GloVe: Global Vectors for Word Representation:
Co-occurrences matrix probabilites word-word

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

<u>Hidden Markov Model</u>
- π_y : probability of starting with label y
- $T_{yy'}$: probability of transitioning from label y to y'
- O_{yx}: probability of generating symbol x given label y
- Predictions: $p(x, y) = \pi_{y_1} O_{y_1 x_1} \prod_{i>1} T_{y_{i-1} y_i} O_{y_i x_i}$

RNN schema





Word2Vec vs GloVe:
They differ in the way they are trained. GloVe is based on global word to word co-occurrence counts (in the entire corpus). Word2Vec uses co-occurrence within local context (neighbour words). More especifically, GloVe's training objective is to find a feature matrix that factorizes the whole word to word co-ocurrence matrix while keeping most of its variance. Word2Vec is trained using either the skip-gram model, wich tries to predict the context words given a central word; or the cbow model, which tries to predict the central word given all words in its context. Another aspect to consider is the fact that GloVe's training time scales with the vocabulary size whereas Word2Vec scales with the corpus size.

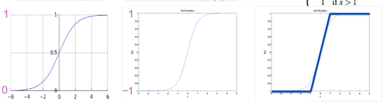
<u>Neural Networks</u>

activation functions

logistic ("sigmoid") $f(z) = \frac{1}{1+\exp(-z)}$

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

hard tanh $\text{HardTanh}(x) = \begin{cases} -1 & \text{if } x < -1 \\ x & \text{if } -1 <= x <= 1 \\ 1 & \text{if } x > 1 \end{cases}$



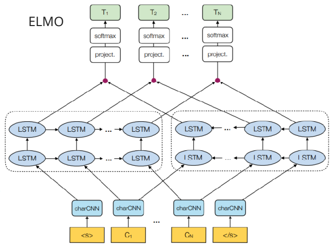
Evaluation:

$$\text{Perplexity}(P) = \exp\left\{\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T_i}\mathcal{L}_t\right\}$$

RNN advantages:
- Can process any length input
- Computation for step t can (in theory) use information from many steps back
- Model size doesn't increase for longer input
- Same weights applied on every timestep, so there is symmetry in how inputs are processed

RNN disadvantages:
- Recurrent computation is slow
- In practice, difficult to access information from many steps back

ELMO



LSTM gate computations:
- New cell content: $\tilde{\mathbf{c}}^{(t)} = \tanh(\mathbf{W_c}[h^{(t-1)}, x^{(t)}] + \mathbf{b_c})$ this is the new content to be written to the cell
- Cell state: erase ("forget") some content from last cell state, and write ("input") some new cell content:

$$\mathbf{c}^{(t)} = \mathbf{f}^{(t)} \odot \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \odot \tilde{\mathbf{c}}^{(t)}$$

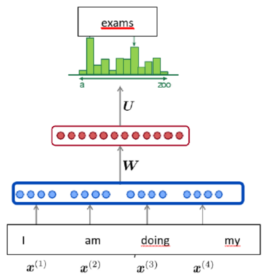
- Hidden state: read ("output") some content from the cell:

$$\mathbf{h}^{(t)} = \mathbf{o}^{(t)} \odot \tanh(\mathbf{c}^{(t)})$$

- $\odot$: Gates are applied using element-wise product
- σ: Sigmoid goes returns values from 0 to 1

- ELMo uses a two-layer bi-directional LSTM network as its architecture
- Each layer has 4096 units and 512-dimensional projections
- The input to the network is a sequence of characters, which are embedded into a 16-dimensional vector
- A convolutional layer with 2048 filters of width 1 to 7 applied to the input character embeddings. The max-pooled output is then re-projected to a 512-dimensional vector.
- The network is pre-trained on a large corpus with the following training objective:
  - Language modeling: predict the next word given the previous words (forward LM) and predict the previous word given the next words (backward LM)

FNN schema

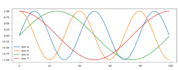


Transformers:

Positional Encoding

We can add positional encodings to the input word vectors:
- Fixed. A usual choice is sine and cosine functions of different frequencies, since it allow the model to attend by relative positions

$$PE(pos, dim) = \sin(\omega_i \cdot pos) \quad \text{if } dim = 2i$$
$$PE(pos, dim) = \cos(\omega_i \cdot pos) \quad \text{if } dim = 2i+1$$
$$\omega_i = \frac{1}{10000^{2i/d_{emb\_encoding}}}$$



- *pos* is the position of the token in the sentence,
- *dim* the dimension of the embeddings
- *i* the position within the embedding.

formular multihead attention block scaled dot product attention

$$MultiHead(Q,K,V) = [head_1, \ldots, head_h]W_0$$
$$\text{where } head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

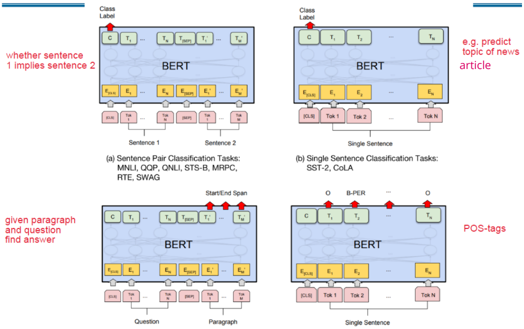
<u>CRF</u>

$$P(\mathbf{y}|\mathbf{x}) = \frac{\exp(\sum_{i=1}^{n} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i))}{Z(\mathbf{x})}$$

Find argmax for CRF

Example of bias case	Application	Ethical concern
COMPAS	predict recidivism risk	Accuracy varies with race: Darker skins, higher risk
Gender Shades	face recognition	Accuracy varies with race: Darker skins, lower accuracy

Contextual word embeddings. BERT



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG
(b) Single Sentence Classification Tasks: SST-2, CoLA
(c) Question Answering Tasks: SQuAD v1.1
(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Resource	Application	Advantatge
GeBioToolkit	Machine Translation	Balanced in gender
MT-DataSheet for Dataset	Machine Translation	Details about the corpus

Positional encoding:

A d-dimensional vector that encodes the wordpiece position in the sentence is added (not concatenated)
to the d-dimensional vector representing the wordpiece. The positional vector uses sinusoidal functions with varying frequences so that when two positions are multiplied, the result
is scales with the distance between such positions.
We need positional encoding because the attention block is not recurrent and does not have any
sense of position/order for each input.
Advantages:
- It is able to encode distances between inputs even for long sequences
- Because it is added and not concatenated, less weights are required
- They are precomputed and don't have to be trained