Language ambiguity (has multiple precise meanings):
- word ‘bring me the file’(resolve w/ POS)
- syntactic ‘I shot an elephant in my pjs’
- semantic ‘the rabbit is ready for lunch’
- referential ‘Pavarotti is a big opera star’
- non-literal ‘it’s raining cats and dogs’

Deep Learning learns✓ abstract functions instead of rules based on maintenance of intuitive linguistic rules.

Lexicon – *morph(eme)*ological analysis (stem and affix e.g. ‘cat’+’s’)
Word segmentation (tokenization)
Word normalization (case/acronyms/spelling)
Lemmatization ‘sing, sung, sang’ → ‘sing’
Stemming (common root, above ‘s’)
Part-Of-Speech (tag words with noun, verb...)

Context-Free Grammar: Derive sentence structure through a parse tree
$\mathrm{S} \to \mathrm{NP}\ \mathrm{VP}, \mathrm{NP} \to \mathrm{Det}\ \mathrm{N}, \mathrm{VP} \to \mathrm{V}\ \mathrm{NP}, \mathrm{VP} \to \mathrm{V}, \mathrm{VP} \to \mathrm{V}\ \mathrm{PP}, \mathrm{PP} \to \mathrm{P}\ \mathrm{NP}$
Discourse: meaning of a text (relationship between sentences) Pragmatics: intentions/commands
Corpus: a collection of documents Document: one item of corpus (sequence) Token: atomic word unit Vocabulary: unique tokens across coropus. **One-Hot Encoding:** sparse (wasted space), orthogonal vectors (every word is equidistant), cannot represent out of vocab well

Sigmoid (binary class.) $\frac{1}{1+e^{-x}}$, ReLU: $\max(0,x)$, Tanh: $\frac{e^x-e^{-x}}{e^x+e^{-x}}$, Softmax (k-class): $\frac{e^{z_i}}{\sum_k e^{z_k}}$
MSE (regression) $\frac{1}{N}\sum_{i=1}^{N}(y_i-\hat{y}_i)^2$, Binary cross-entropy: $-\frac{1}{N}\sum_{i=1}^{N}(y^{(i)}\log(\hat{y}^{(i)})+(1-y^{(i)})\log(1-\hat{y}^{(i)}))$ Categorical cross entropy (k-class): $-\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_c^{(i)}\log(\hat{y}_c^{(i)})$

Euclidean Distance: $\sqrt{\sum_{i=1}^{n}(q_d-d_i)^2}$
Cosine Similarity: $\cos(\theta)=\frac{p_1 \cdot p_2}{||p_1|| \times ||p_2||}$

Analogy Recovery: offset of the vectors reflect their relationship.
$a-b \approx c-d \iff d \approx c-a+b$

Byte-Pair Encoding: Instead of manually specfying rules for lemmatisation or stemming, lets learn from data which character sequences occur together frequently. 1) Start with a vocabulary of all individual characters 2) Split the words in your training corpus also into individual characters + ‘_’ at end 3) Find which two vocabulary items occur together most frequently in the trianing corpus 4) Add that combination as a new vocabulary item 5) Merge all occurrences of that combination in your corpus 6) Repeat until a desired number of merges has been performed. For unknown words follow above and apply replacements in order discovered.

Window: window consists of target and context (surrounding), Window size = radius **Continuous Bag Of Words**: context → target, **Skip-gram**: target → context (give as one-hot, get word representation, map embedding to target words using weight matrix, apply softmax). Train with list of pairs (target, context) by sliding window over input. Loss: $p(w_{t+j}|w_t)=\frac{\exp(u_{w_{t+j}}^\top h_{w_t})}{\sum_{w'=1}^{W}\exp(u_{w'}^\top h_{w_t})}$, the aim: $\max \prod_t \prod_j p(w_{t+j}|w_t)$ → $\min_\theta -\sum_t \sum_j \log p(w_{t+j}|w_t;\theta)$ → $\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t;\theta)$ over all elems in corpus. However, the bottom term in the $p(w_{t+j}|w_t)$ is inefficient to compute across the entire corpus vocabulary. Therefore, train a Negative Sampling model to predict whether a word appears in the context of another: $\log p(D=1|w_t,w_{t+1})+k\mathbb{E}_{\tilde{c}\sim P_{noise}}[\log p(D=0|w_t,\tilde{c})]$ where $p(D=1|w_t,w_{t+1})$ is a binary logistic regression probability of seeing the word w_t in the context w_{t+1}. Approximate the expectation by drawing random words from vocabulary, and on left choose positive pairs. Thus the equation is replaced: $p(D=1|w_t,w_{t+1})=\frac{1}{1+\exp -u_{w_{t+1}}^\top h_{w_t}}$. We can sample k (5-10 words) with frequency or random sampling.