# COMS W4701: Artificial Intelligence

## Lecture 27: Neural Language Models

Some images adapted from Jurafsky and Martin

Tony Dear, Ph.D.

Department of Computer Science

School of Engineering and Applied Sciences

# Topics

- Natural language processing
- Word embeddings
- Self-attention and transformers
- Large language models

# Natural Language Processing

- Language can be used to describe intelligent behavior, including knowledge representation, planning, perceiving and acting

- **Natural language processing** allow computers to *communicate* (with us and each other) and *learn* new knowledge

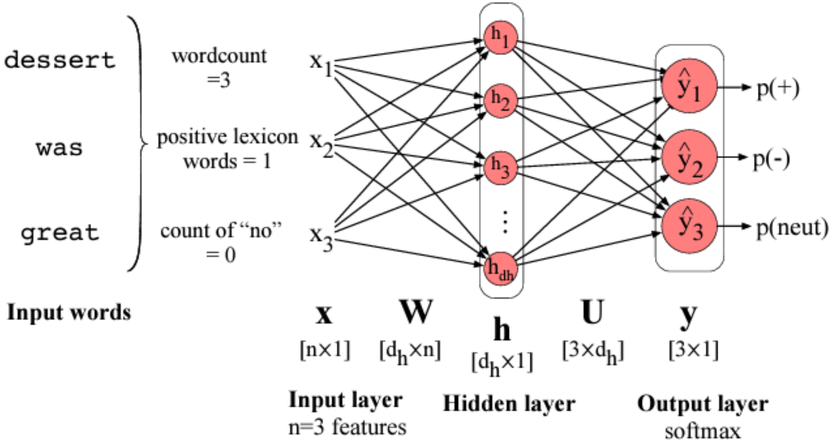
- A **language model** is a probability distribution over possible strings, which can be used for text prediction and translation

# NLP Tasks

- Speech recognition (sound to text)
- Text-to-speech, including synthesis of different voices, intonations, dialects, imitations of specific people
- Machine translation between different languages
- Information extraction (summarization, feature extraction), information retrieval (finding documents), question answering

# Sentiment Analysis

- **Sentiment analysis**: *Classify* input text by positive or negative orientation
- Applications from marketing (e.g., consumer reviews) to politics
- One idea: We can *associate* different words with positive or negative lexicons and combine them with other *features*



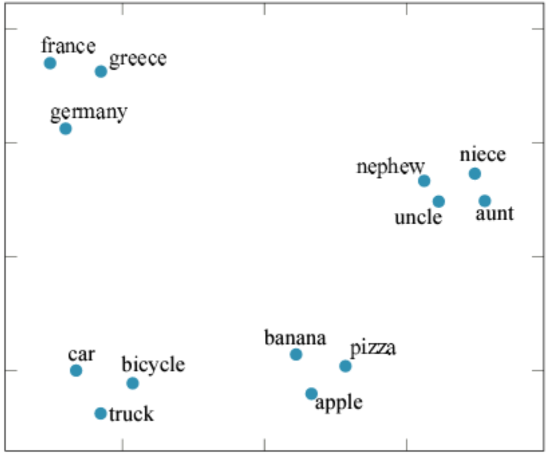
- Given training data, we can then learn a neural network for this task
- Similar issues as in computer vision: Hand-designing good features is hard!

# Word Embeddings

- We need a word representation that captures *context* and *relationships*
- Words may be related by sentiment, syntactically, and/or semantically

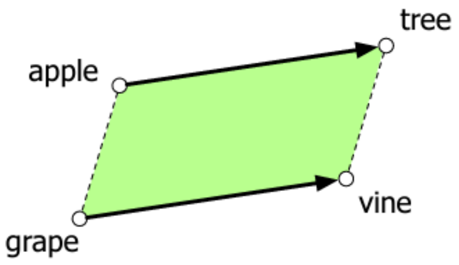
- **Word embedding**: A dense, low-dimensional vector representation of a word
- Number of components in the hundreds, much fewer than a vocabulary size or number of documents

- Algorithms like **Word2vec** automatically *learn* embeddings, e.g. by considering how often words appear near each other in a training corpus



# Semantic Relationships

- Embeddings can also learn *analogies* between words, beyond simply noting when they appear near each other

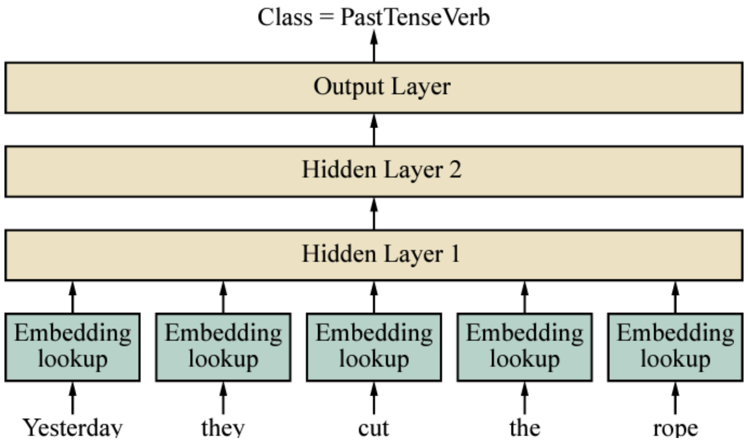


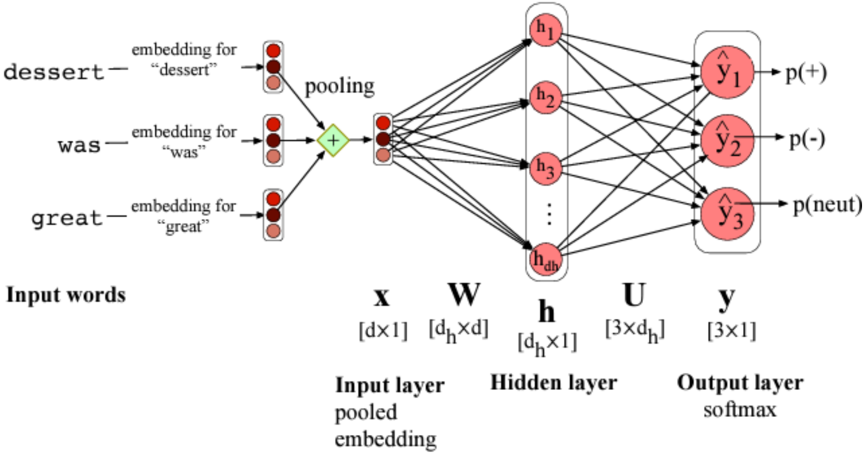
- Vector differences for pairs of words satisfying the same semantic relationships tend to be similar as well

- Social implications: Embeddings can learn biases and stereotypes latent in text and even amplify them! (Bolukbasi et al. 2016)
- Example embeddings from word2vec: “Man” – “programmer” ≈ “woman” – “homemaker”, “father” – “doctor” ≈ “mother” – “nurse”

# Classification with Embeddings

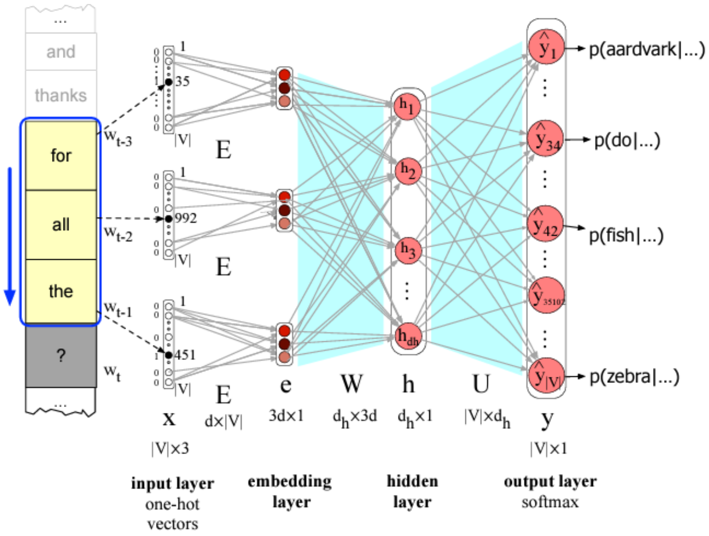
- Once we have a pretrained set of embeddings, we can convert text into these and feed them as inputs into a neural network classifier
- Can also **pool** embeddings together, e.g. via summation, mean, etc.
- Can perform tasks such as part-of-speech tagging and sentiment analysis





# Language Modeling

- **Language modeling** is the task of *predicting* upcoming words from prior words
- A **neural language model** is a neural network that takes in some number of words and outputs a probability distribution over possible next words
- **Forward inference**: Compute the embeddings of a *moving window* and feed them through a network to obtain prediction probabilities
- Embeddings may be pretrained, or trained simultaneously with the network classifier

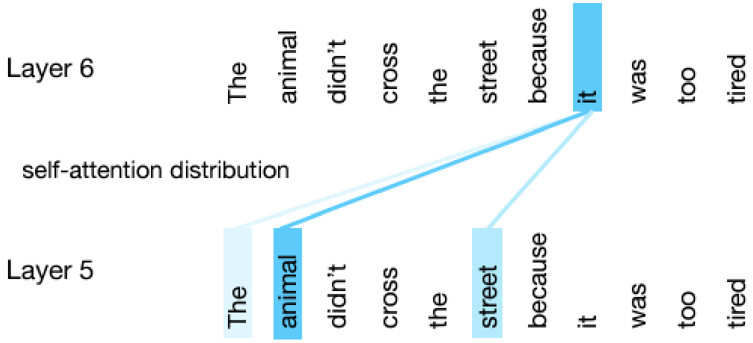


# Self-Attention

- In language, words are often related despite *not* being in close proximity
  - Ex: “The *animal* didn’t cross the street because *it* was too tired.”
- Similar “non-proximity” context also appears in machine translation

- **Self-attention** computes, for a **query** word in a given sequence, a *weighted average* over all prior **key** words and their context **values**

- The same embedding may have different *projections* depending on the role that it is playing in a sequence

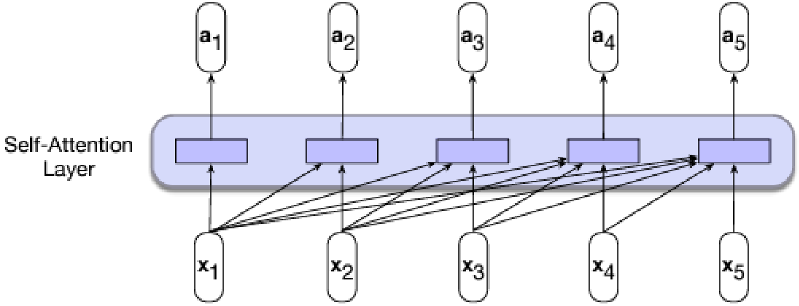


# Self-Attention

- We can learn *weight matrices* $\mathbf{W}_q$, $\mathbf{W}_k$, $\mathbf{W}_v$ to project an embedding $\mathbf{x}_i$ into its **query** $\mathbf{q}_i = \mathbf{x}_i\mathbf{W}_q$, **key** $\mathbf{k}_i = \mathbf{x}_i\mathbf{W}_k$, and **value** $\mathbf{v}_i = \mathbf{x}_i\mathbf{W}_v$ representations

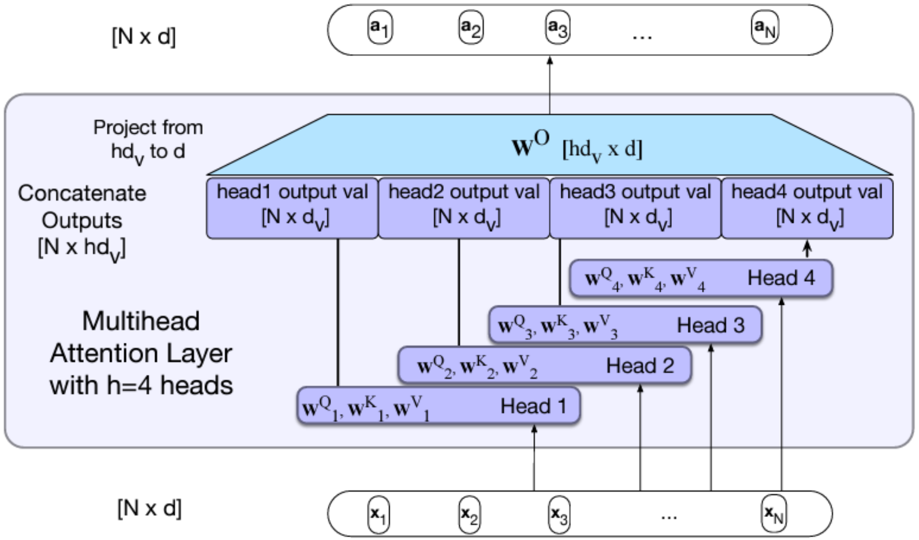
- Compute a normalized score $\alpha_{ij} = \mathrm{softmax}(\mathbf{q}_i \cdot \mathbf{k}_j)$, for all $j \leq i$
- The output is the weighted sum of all previous values: $\mathbf{a}_i = \sum_{j \leq i} \alpha_{ij}\mathbf{v}_j$

- Self-attention is generally *asymmetric* ($\alpha_{ij} \neq \alpha_{ji}$) since the context of words depends on their roles



# Multihead Attention

- One set of projection matrices for each word may not adequately capture all semantic, syntactic, and discourse relationships among them
- **Multihead self-attention** learns distinct sets of parameters for extracting different context information
- Self-attention outputs from each "head" are then concatenated and then projected back into a sequence with the same size as the input

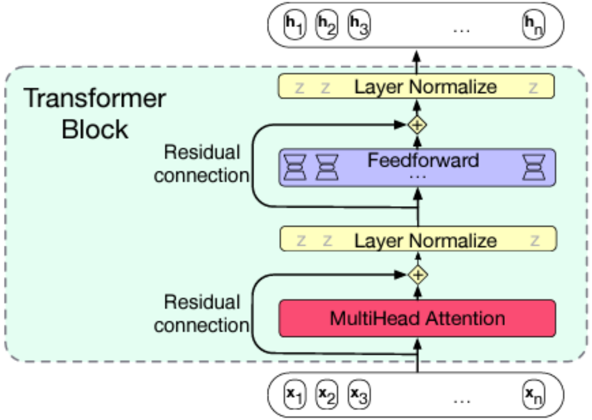

[N x d]
$a_1$ $a_2$ $a_3$ … $a_N$
Project from $hd_V$ to d
$W^O$ $[hd_V \times d]$
Concatenate Outputs $[N \times hd_V]$
head1 output val $[N \times d_V]$
head2 output val $[N \times d_V]$
head3 output val $[N \times d_V]$
head4 output val $[N \times d_V]$
$W^Q_4, W^K_4, W^V_4$ Head 4
$W^Q_3, W^K_3, W^V_3$ Head 3
$W^Q_2, W^K_2, W^V_2$ Head 2
$W^Q_1, W^K_1, W^V_1$ Head 1
Multihead Attention Layer with h=4 heads
[N x d]
$x_1$ $x_2$ $x_3$ … $x_N$

# Transformer Blocks

- A **transformer** architecture consists of multiple self-attention layers interleaved with normalization and feedforward (nonlinear) layers

- Like ConvNets, successive transformer blocks extract new and higher levels of context given outputs of previous blocks

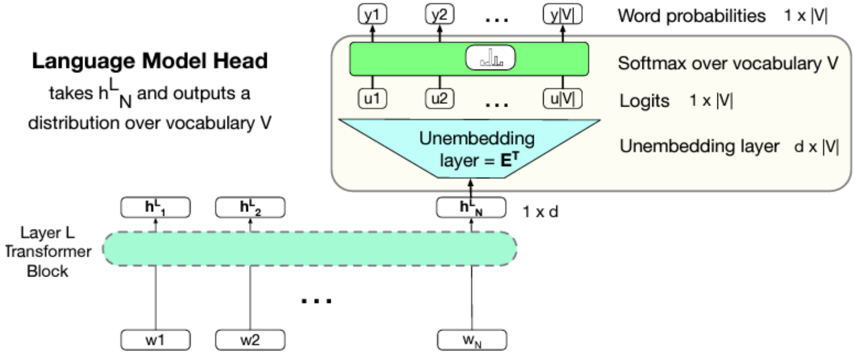
- Each attention layer *contextualizes* a word/token with other tokens to extract linguistic information
- Large language models stack dozens of these blocks together



# Input and Output Layers

- Self-attention is agnostic to word *order*, which provides important context
- Input embeddings typically consist of a sum of the corresponding *word* embedding with an embedding for the *position* in the sequence

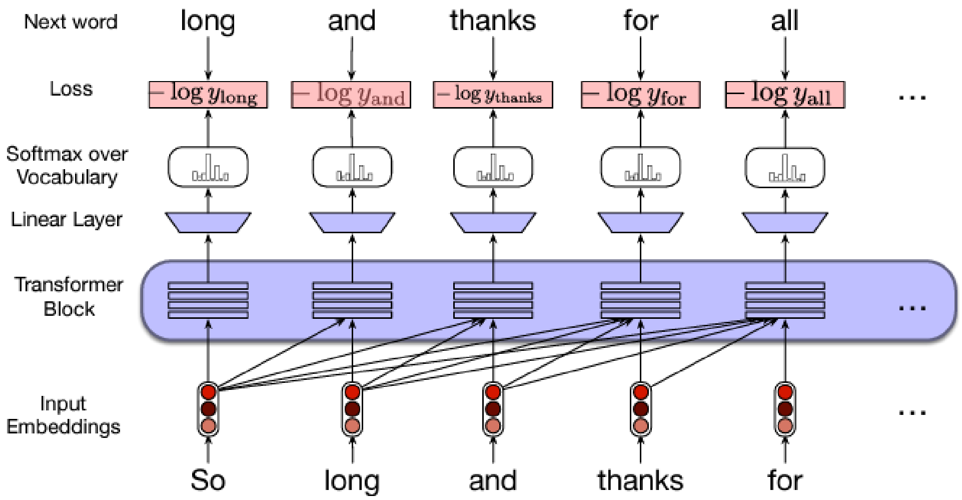
- How to use the output of a transformer for word prediction?
- The **model head** "unembeds" the last token and projects it into a probability distribution over the vocabulary
- A word can then be generated from this distribution



Language Model Head
takes $h^L_N$ and outputs a distribution over vocabulary V
y1
y2
...
y|V|
Word probabilities 1 x |V|
Softmax over vocabulary V
u1
u2
...
u|V|
Logits 1 x |V|
Unembedding layer = $E^T$
Unembedding layer d x |V|
$h^L_1$
$h^L_2$
$h^L_N$
1 x d
Layer L Transformer Block
...
w1
w2
$w_N$

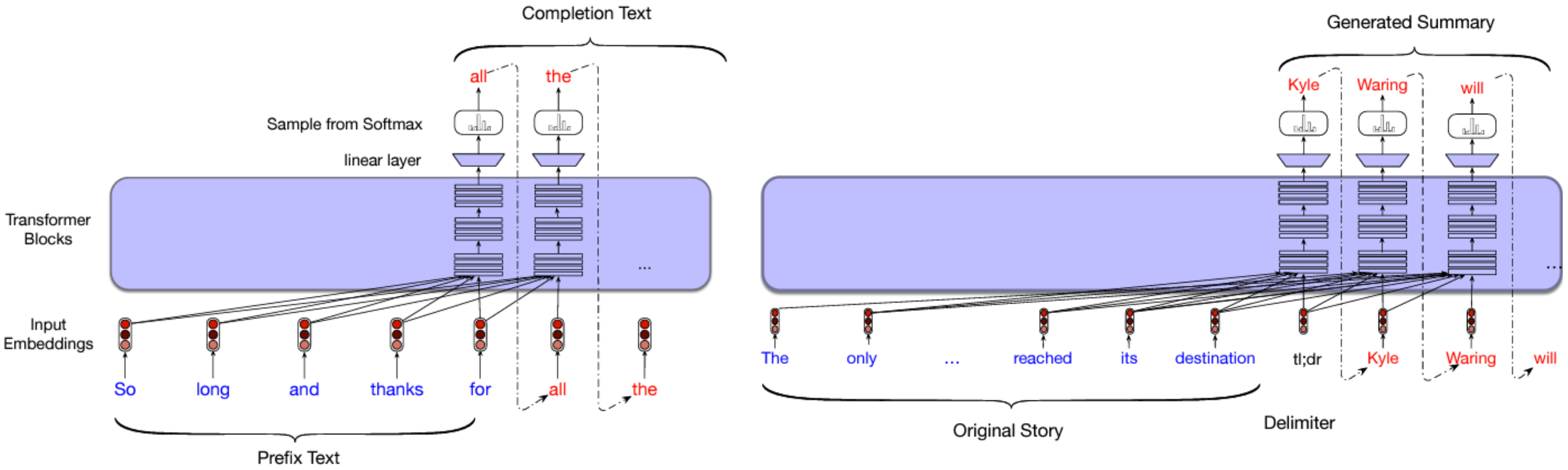
# Self-Supervised Training

- To train on a general unlabeled corpus, the *sequence* of words encountered acts as a "ground truth", allowing for **self-supervised training**
- Data can be easily acquired from crawls of web text, including Wikipedia, news sites, patent docs, books
- Same ideas as general neural net training: Predict outputs from batches of data, compute losses and gradients, update weights



# NLP with Transformers

- Many NLP tasks can be cast as word prediction given input text
- Transformers can accommodate long contexts (e.g., thousands of tokens)
- Textual answers and summaries can be generated by repeatedly feeding back each new word into the transformer



# Large Language Models

- Like ConvNets, transformer architectures have rapidly improved over time (since 2017, “Attention Is All You Need” by Vaswani et al.)

- State-of-the-art models are *huge* and excel in many NLP tasks
- Ex: GPT-3 has 175B parameters, trained with 300B tokens

- Relatively easy to do transfer learning on pretrained embeddings and models
- Training process can also be augmented by prompt engineering
- Performance has been shown to scale up with model size, dataset size, and amount of training computations

# Some LLM Considerations

- Like ConvNets, transformer models are *end-to-end*; no reliance on hand-crafted features or classical knowledge of grammars, parsing, semantics, etc.
- Transformers can more easily adapt to new domains and new languages

- Highly debated whether LLMs actually *understand* the knowledge on which they are trained, or whether they are just “stochastic parrots” (Bender, 2021)
- LLMs are prone to **hallucination**, claims that are coherent but false

- LLMs can perpetuate biases and generate toxic language and misinformation
- LLM training and usage also present security, privacy, and copyright issues

# Summary

- Natural language processing tasks include speech recognition, machine translation, information extraction, information retrieval, Q&A, and more

- Words (tokens) are usually represented using learned embedding vectors
- The self-attention mechanism computes contextual relationships between tokens in a given sequence

- Transformer models use self-attention to extract multiple levels of context
- Form the basis of large language models, which have shown great performance in many NLP tasks, many based on text generation