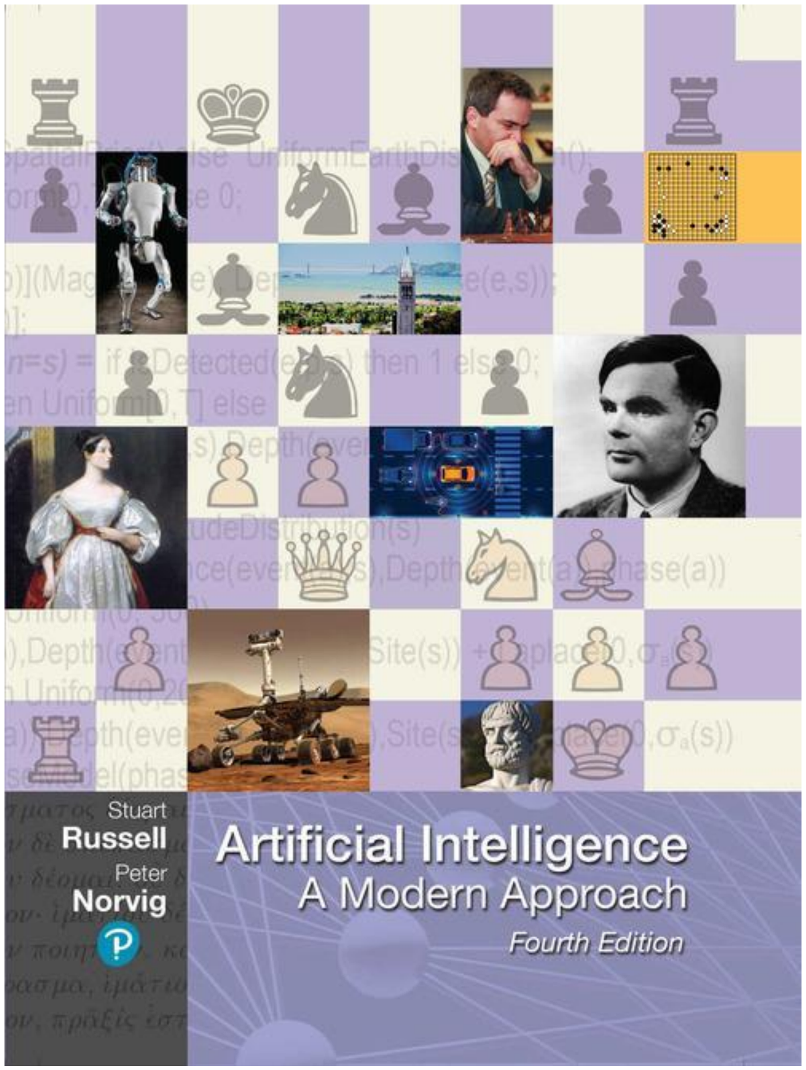
Artificial Intelligence: A Modern Approach

Fourth Edition



Chapter 24

Deep Learning For Natural Language Processing

Outline

- ♦ Word Embeddings
- ♦ Recurrent Neural Networks for NLP
- ♦ Sequence-to-Sequence Models
- ♦ The Transformer Architecture
- ♦ Pretraining and Transfer Learning
- ♦ State of the art

Word Embeddings

- Desire a representation of words that does not require manual feature engineering, but allows for generalization between related words

- “You shall know a word by the company it keeps.” John Firth 1957. [also applied to images!]

- **word embedding**: a low-dimensional vector representing a word.
  - learned automatically from the data.

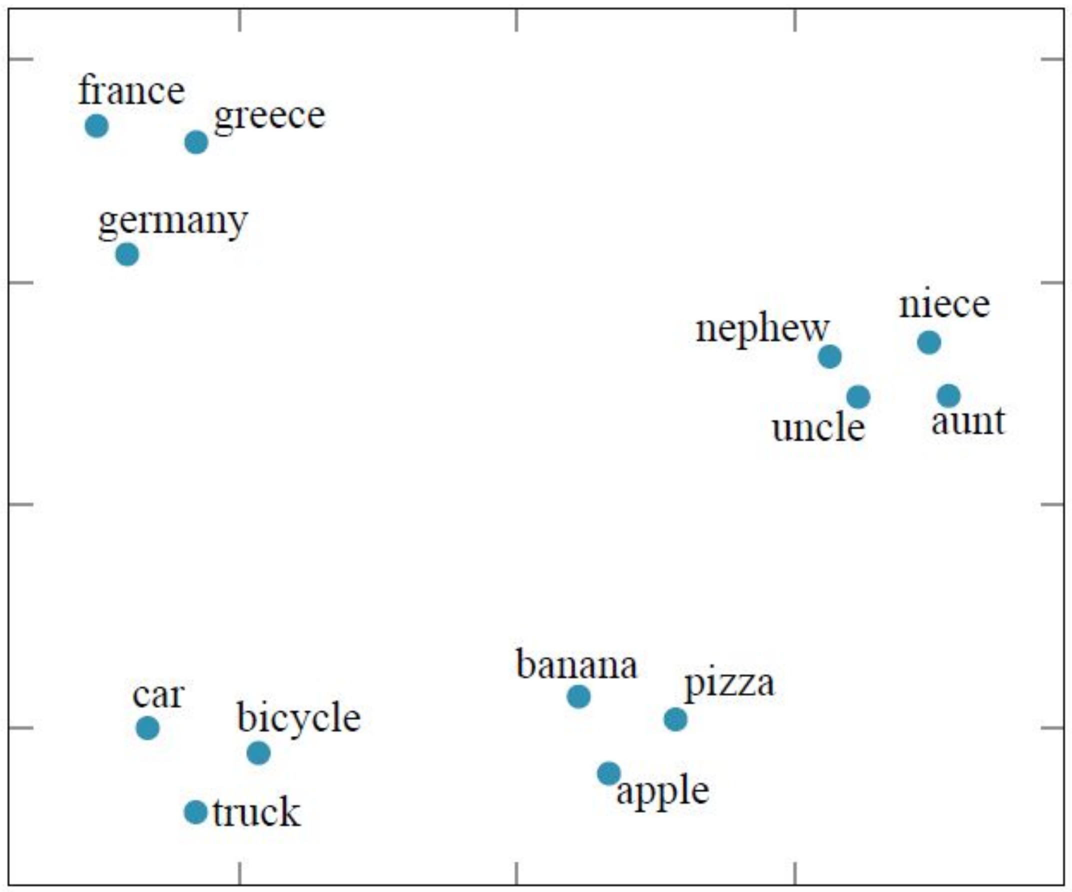
  - have additional properties beyond mere proximity for similar words

  - have proven to be a good representation for downstream language tasks (such as question answering or translation or summarization)

  - possible to use generic pretrained vectors

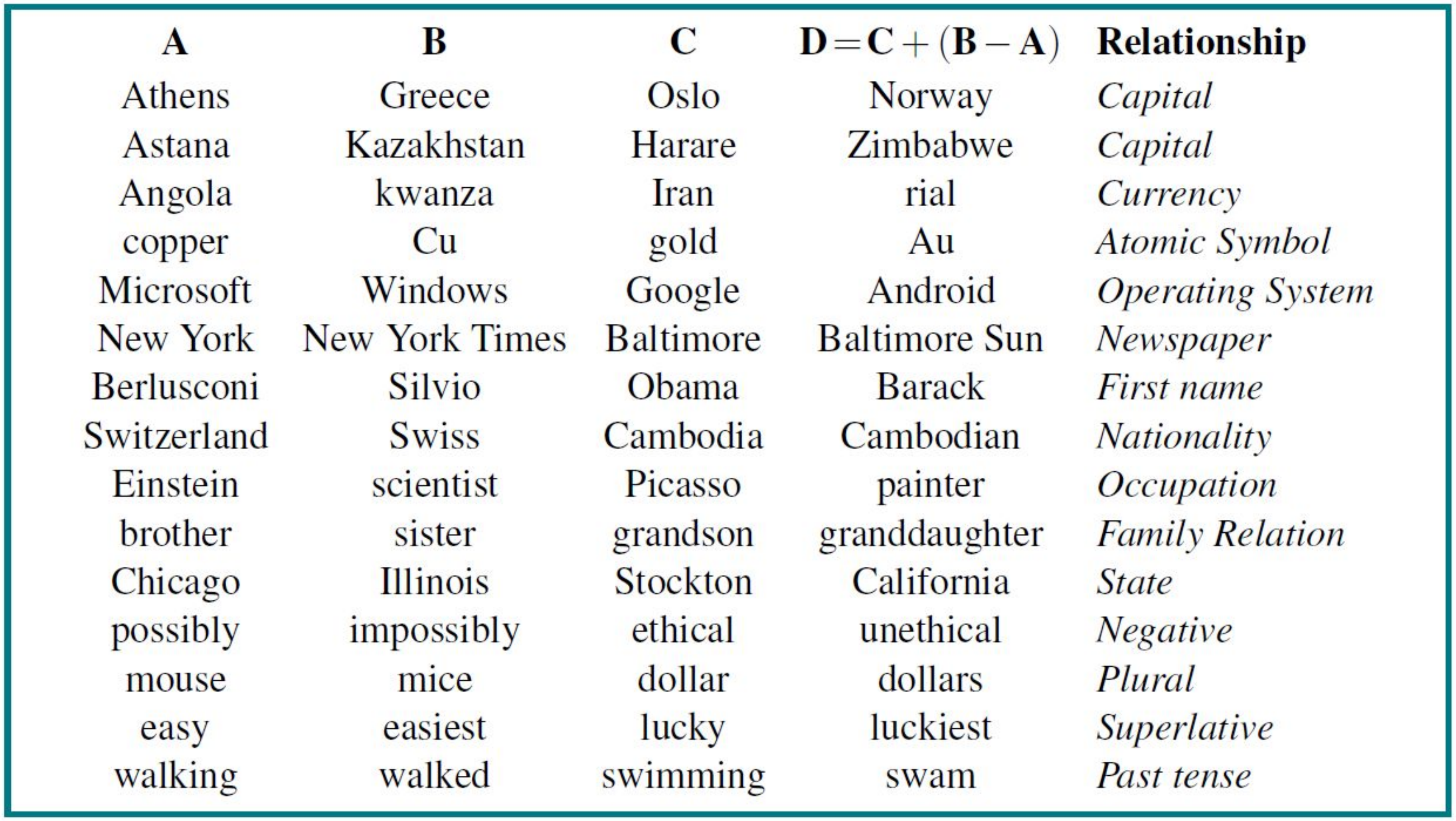
  - vector dictionaries include WORD2VEC, GloVe (Global Vectors), and FASTTEXT, which has embeddings for 157 languages.

Word Embeddings



Word embedding vectors computed by the GloVe algorithm trained on 6 billion
words of text. 100-dimensional word vectors are projected down onto two dimensions in this visualization. Similar words appear near each other.

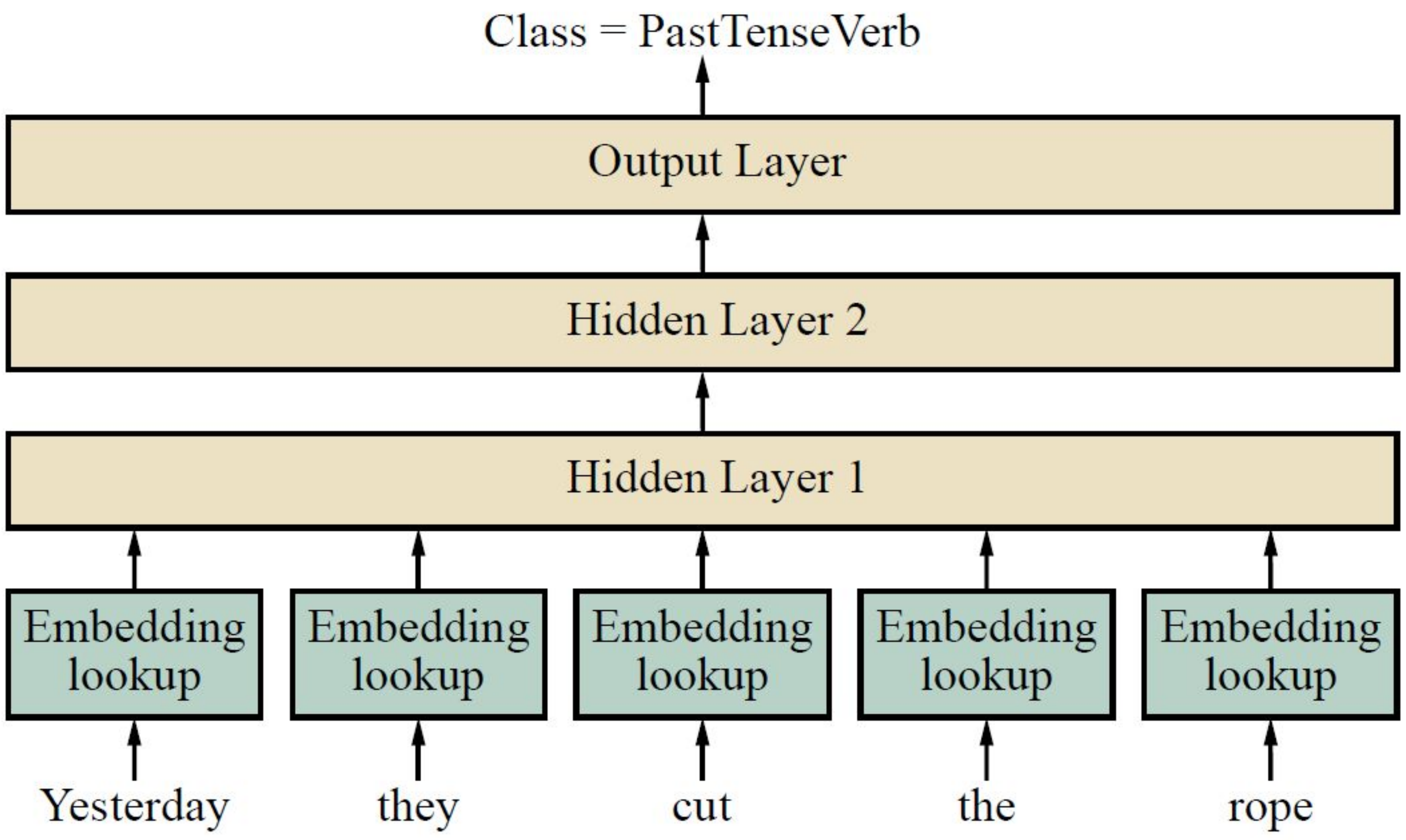
Word Embeddings



A	**B**	**C**	$\mathbf{D} = \mathbf{C} + (\mathbf{B} - \mathbf{A})$	**Relationship**
Athens	Greece	Oslo	Norway	*Capital*
Astana	Kazakhstan	Harare	Zimbabwe	*Capital*
Angola	kwanza	Iran	rial	*Currency*
copper	Cu	gold	Au	*Atomic Symbol*
Microsoft	Windows	Google	Android	*Operating System*
New York	New York Times	Baltimore	Baltimore Sun	*Newspaper*
Berlusconi	Silvio	Obama	Barack	*First name*
Switzerland	Swiss	Cambodia	Cambodian	*Nationality*
Einstein	scientist	Picasso	painter	*Occupation*
brother	sister	grandson	granddaughter	*Family Relation*
Chicago	Illinois	Stockton	California	*State*
possibly	impossibly	ethical	unethical	*Negative*
mouse	mice	dollar	dollars	*Plural*
easy	easiest	lucky	luckiest	*Superlative*
walking	walked	swimming	swam	*Past tense*

A word embedding model can sometimes answer the question “**A** is to **B** as **C** is to [what]?” with vector arithmetic: given the word embedding vectors for the words **A**, **B**, and **C**, compute the vector $\mathbf{D} = \mathbf{C} + (\mathbf{B}\ \mathbf{A})$ and look up the word that is closest to **D**

Word Embeddings



Feedforward part-of-speech tagging model. This model takes a 5-word window as input and predicts the tag of the word in the middle—here, cut. The model is able to account for word position because each of the 5 input embeddings is multiplied by a different part of the first hidden layer. The parameter values for the word embeddings and for the three layers are all learned simultaneously during training.

Word Embeddings

- Given a corpus of sentences with POS tags, we learn the parameters for the word embeddings and the POS tagger simultaneously

- Process steps:
  - Choose the width *w* (an odd number of words, typical=5) for the prediction window to be used
  - Create a vocabulary of all of the unique word tokens that occur more than, say, 5 times in the training data. Denote the total number of words in the vocabulary as *v*.
  - Sort this vocabulary in any arbitrary order (perhaps alphabetically).
  - Choose a value *d* as the size of each word embedding vector.
  - Create a new *v*-by-*d* weight matrix called **E** which is randomly initialized or from pretrained vectors
  - Set up a neural network that outputs a part of speech label
  - To encode a sequence of w words into an input vector, simply look up the embedding for each word and concatenate the embedding vectors
  - Train the weights **E** and the other weight matrices $\mathbf{W}_1$, $\mathbf{W}_2$, and $\mathbf{W}_{out}$ using gradient descent.

Recurrent Neural Networks for NLP

- feedforward network has the problems of asymmetry and of too many parameters

- RNN helps solve these problems

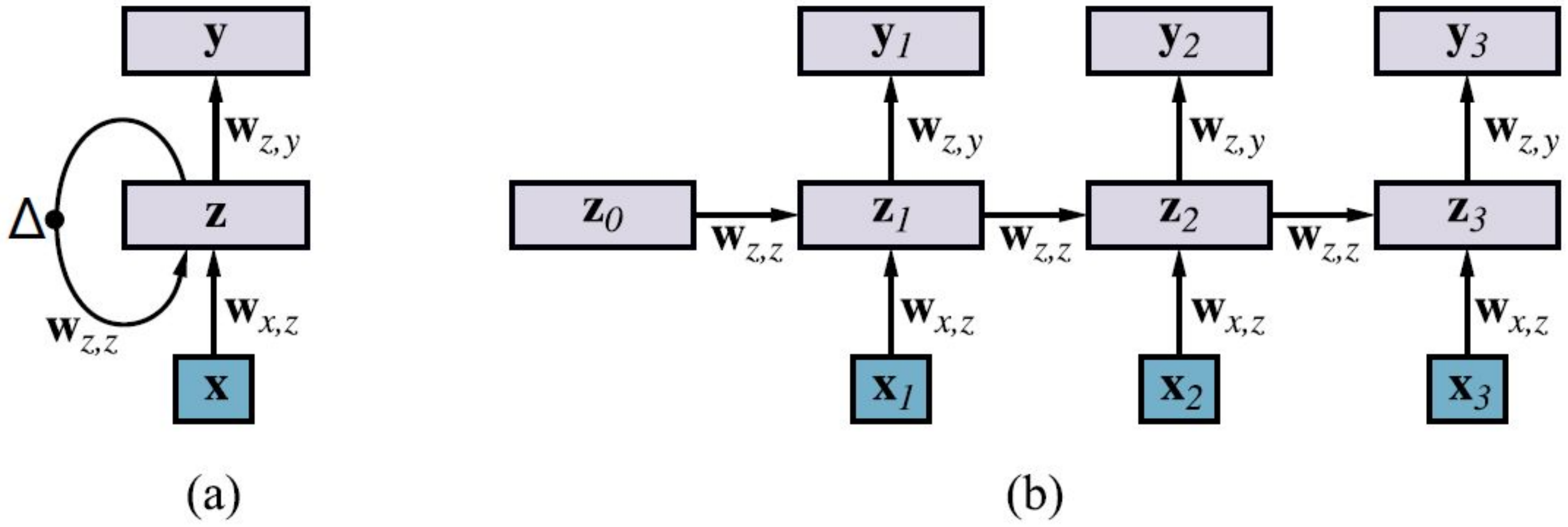
- In an RNN language model each input word is encoded as a word embedding vector, $\mathbf{x}_i$. There is a hidden layer $\mathbf{z}_t$ which gets passed as input from one time step to the next.

- the output $\mathbf{y}_t$ will be a softmax probability distribution over the possible values of the next word in the sentence
- The number of parameters in the weight matrixes $w_{,z,z}$, $w_{,x,z}$ $w_{,z,y}$ stay constant
- Weights are the same for every word position (solve asymmetry)

- Training an RNN to do classification like this is done the same way as with the language model.
    - require labels—part of speech tags or reference indications.

- To capture the context on the right, we can use **a bidirectional RNN**, which concatenates a separate right-to-left model onto the left-to-right model.

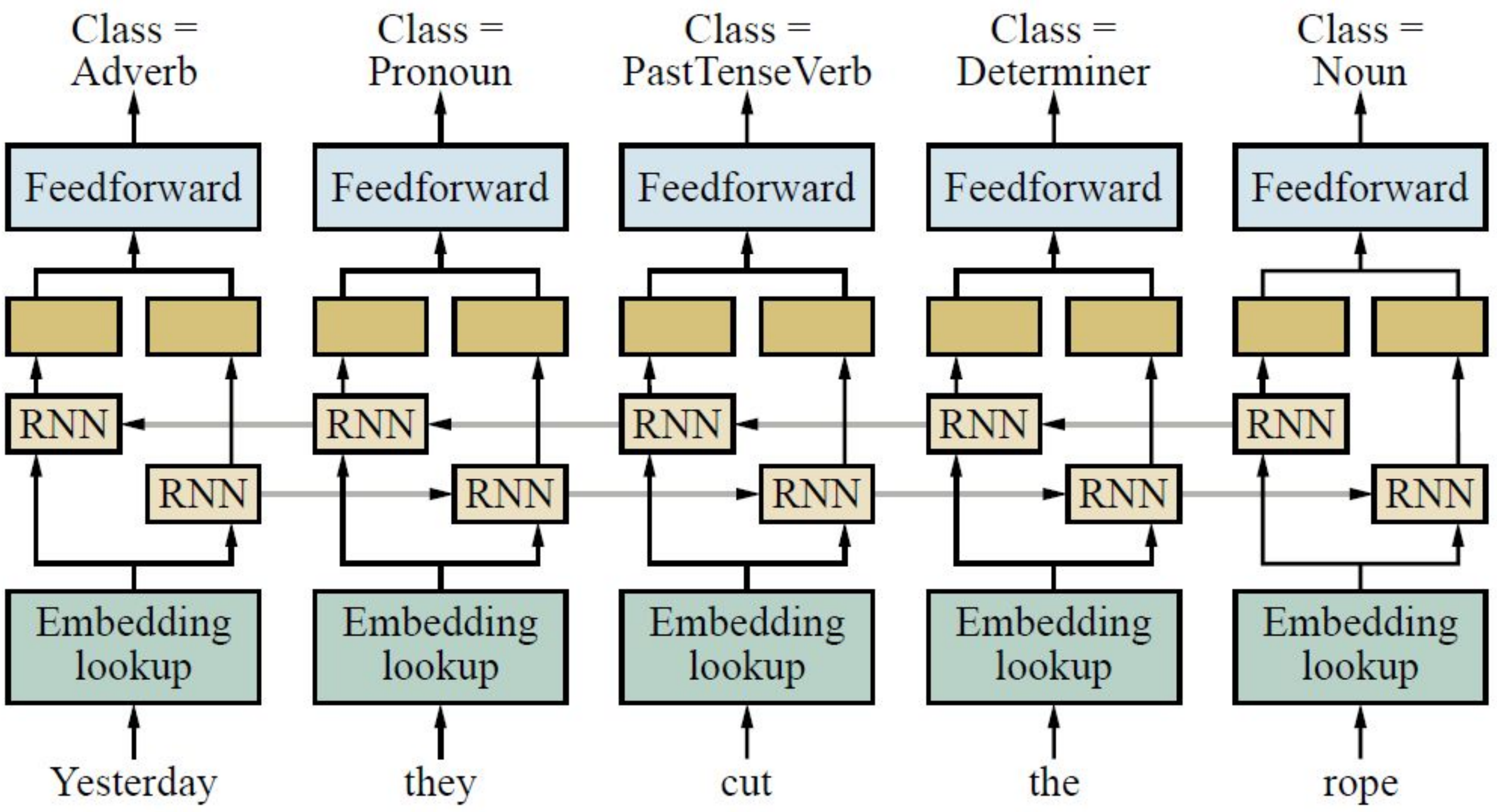
Recurrent Neural Networks for NLP



(a) Schematic diagram of an RNN where the hidden layer **z** has recurrent connections; the Δ symbol indicates a delay.

(b) The same network unrolled over three timesteps to create a feedforward network

Recurrent Neural Networks for NLP



A bidirectional RNN network for POS tagging.

Recurrent Neural Networks for NLP

- LSTMs for NLP tasks

long short-term memory (LSTM), form of RNN with gating units

- don’t suffer from the problem of imperfectly reproducing a message from one time step to the next

- choose to remember some parts of the input, copying it over to the next timestep, and to forget other parts

- can learn to create a latent feature
  - `copy feature forward without alteration until it is needed to make a choice

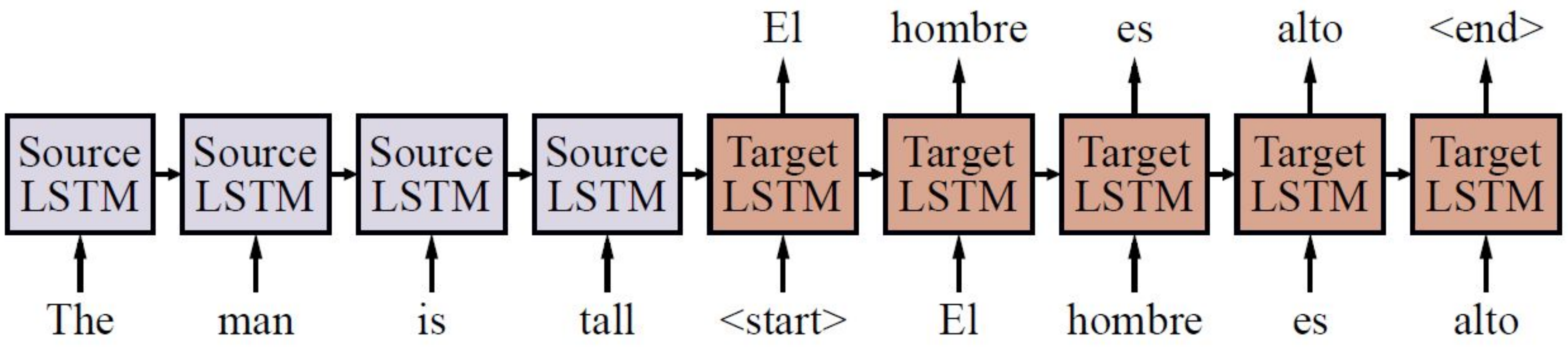
Sequence-to-Sequence Models

- **Machine translation** (MT) goal: to translate a sentence from a source language to a target language (eg: Spanish to English)

Sequence-to-sequence model

- Use two RNNs, one for the source and one for the target.
- Run the source RNN over the source sentence and then use the final hidden state from the source RNN as the initial hidden state for the target RNN.
- Target word is implicitly conditioned on both the entire source sentence and the previous target words.
- Used for translation, automatically generating text caption, summarization of image, rewriting long text to shorter ones.
- Major weaknesses:
    - Nearby context bias
    - Fixed context size limit
    - Slower sequential processing

Recurrent Neural Networks for NLP



Basic sequence-to-sequence model. Each block represents one LSTM timestep.
(For simplicity, the embedding and output layers are not shown.)

Sequence-to-Sequence Models

Attention

- when the target RNN is generating the target one word at a time, it is likely that only a small part of the source is actually relevant to each target word

- the target RNN must pay attention to different parts of the source for every word.

$$
\begin{aligned}
r_{ij} &= \mathbf{h}_{i-1} \cdot \mathbf{s}_j \\
a_{ij} &= e^{r_{ij}} / (\sum_k e^{r_{ik}}) \\
\mathbf{c}_i &= \sum_j a_{ij} \cdot \mathbf{s}_j
\end{aligned}
$$

$\mathbf{h}_{i-1}$ is the target RNN vector, predicting at timestep, i, and $\mathbf{s}_j$ is the output of the source RNN vector for the source word (or timestep) j. Both $\mathbf{h}_{i-1}$ and $\mathbf{s}_j$ are d-dimensional vectors, where d is the hidden size. The value of r_{ij} is therefore the raw "attention score" between the current target state and the source word j. These scores are then normalized into a probability a_{ij} using a softmax over all source words. $\mathbf{c}_i$ is the weighted average of the source RNN vectors,

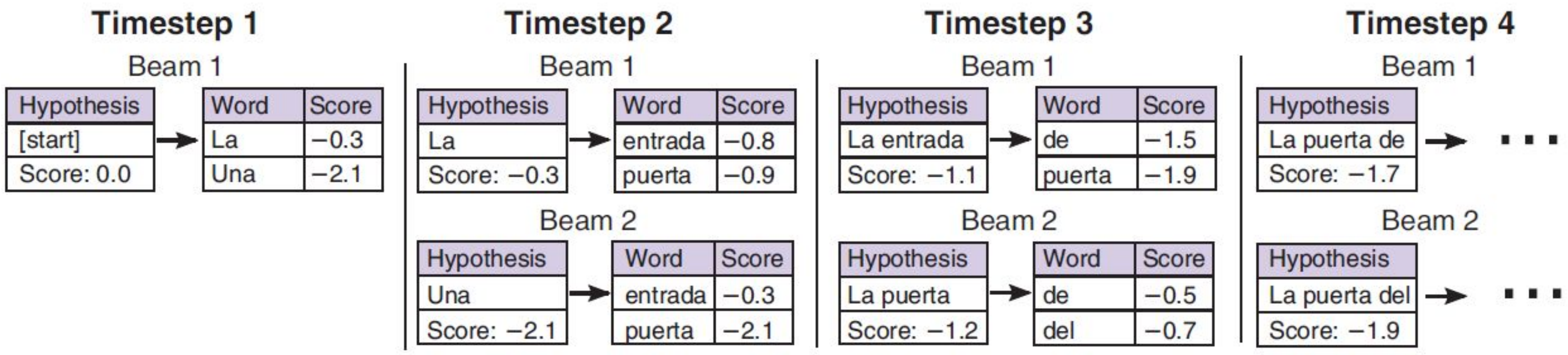
Sequence-to-Sequence Models

Decoding
- our goal is to generate the corresponding target sentence
- generate the target one word at a time, and then feed back in the word that we generated at the next timestep.

Greedy decoding
- select the highest probability word at each timestep and then feed this word as input to the next timestep
- may not achieve maximizing the probability of the entire target sequence
- better approach is to search for an optimal decoding

Beam search
- **For MT decoding,** keeps the top *k* hypotheses at each stage, extending each by one word using the top *k* choices of words,
- chooses the best *k* of the resulting k^2 new hypotheses.
- When all hypotheses in the beam generate the algorithm outputs the highest scoring hypothesis
- state-of-the-art neural MT models use a beam size of 4 to 8, (older >100)

Sequence-to-Sequence Models



"The front door is red." ⇒ "La puerta de entrada es roja."

Beam search with beam size of b = 2. The score of each word is the log-probability generated by the target RNN softmax, and the score of each hypothesis is the sum of the word scores. At timestep 3, the highest scoring hypothesis *La entrada* can only generate low-probability continuations, so it "falls off the beam."

The Transformer Architecture

Self-attention

- Previously attention was applied from the target RNN to the source RNN
- Self-attention each sequence of hidden states attends to itself
    - Source to source
    - Target to target
    - Able to capture long-distance (and nearby) context within each sequence

projecting the input into three different representations using three different weight matrices

- **query vector** $\mathbf{q}_i = W_q\mathbf{x}_i$ the one being attended from
- **key vector** $\mathbf{k}_i = \mathbf{W}_k\mathbf{x}_i$ the one being attended to
- ***value* vector** $\mathbf{v}_i = \mathbf{W}_v\mathbf{x}_i$ the context that is being generated

$$
\begin{aligned}
r_{ij} &= (\mathbf{q}_i \cdot \mathbf{k}_j)/\sqrt{d} \\
a_{ij} &= e^{r_{ij}}/(\sum_k e^{r_{ik}}) \\
\mathbf{c}_i &= \sum_j a_{ij} \cdot \mathbf{v}_j ,
\end{aligned}
$$

,

The Transformer Architecture

Self-attention

- self-attention mechanism is *asymmetric*, as r_{ij} is different from r_{ji}.
- the scale factor $\sqrt{d}$ was added to improve numerical stability
- encoding for all words in a sentence can be calculated simultaneously
- sometimes important information gets lost, because it is essentially averaged out over the whole sentence.

multiheaded attention

- sentence up into m equal pieces and apply the attention model to each of the m pieces.
- Each piece has its own set of weights
- **concatenating** rather than summing, we make it easier for an important subpiece to stand out

,

The Transformer Architecture

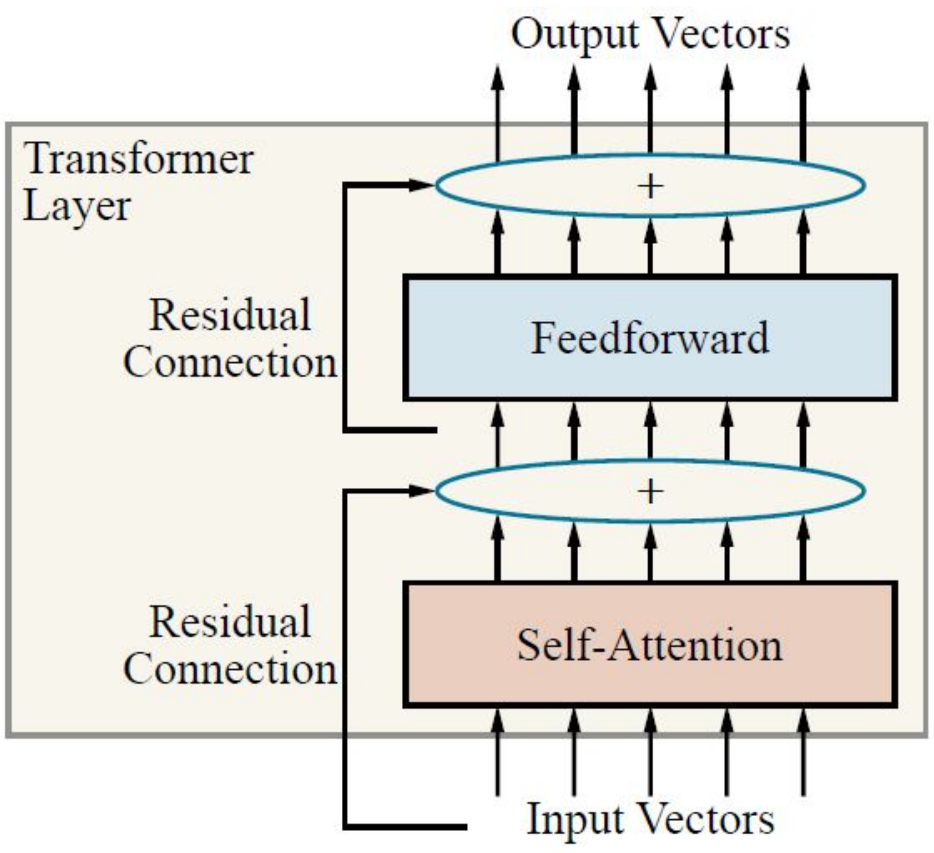
From self-attention to transformer

- Each transformer layer consists of several sub-layers

Transformer encoder:

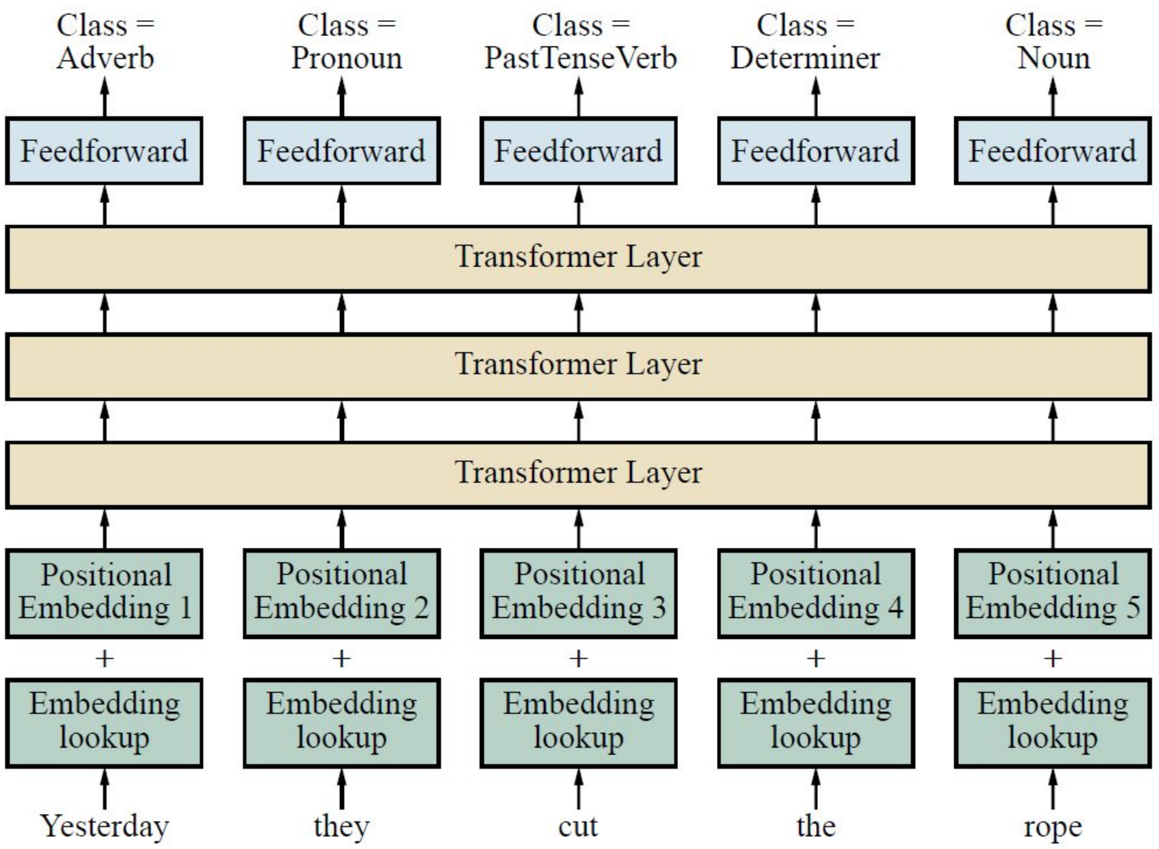
- self-attention is applied first.
- The output of the attention module is fed through feedforward layers, where the same feedforward weight matrices are applied independently at each position nonlinear activation function, typically ReLU, is applied after the first feedforward layer.
- two residual connections are added into the transformer layer.
- Usually have six or more layers
- Transformer does not explicitly capture the order of words. Self attention is agnostic to word order
- Positional embeddings helps to capture ordering
  - our input sequence has a maximum length of *n*, then we learn *n* new embedding vectors—one for each word position
  - Transformer decoder is nearly identical
  - Decoder uses version of self-attention where each word can only attend to the words before it

The Transformer Architecture



A single-layer transformer consists of self-attention, a feedforward network, and residual connections

The Transformer Architecture



Using the transformer architecture for POS tagging

Pretraining and Transfer Learning

Pretrained word embeddings

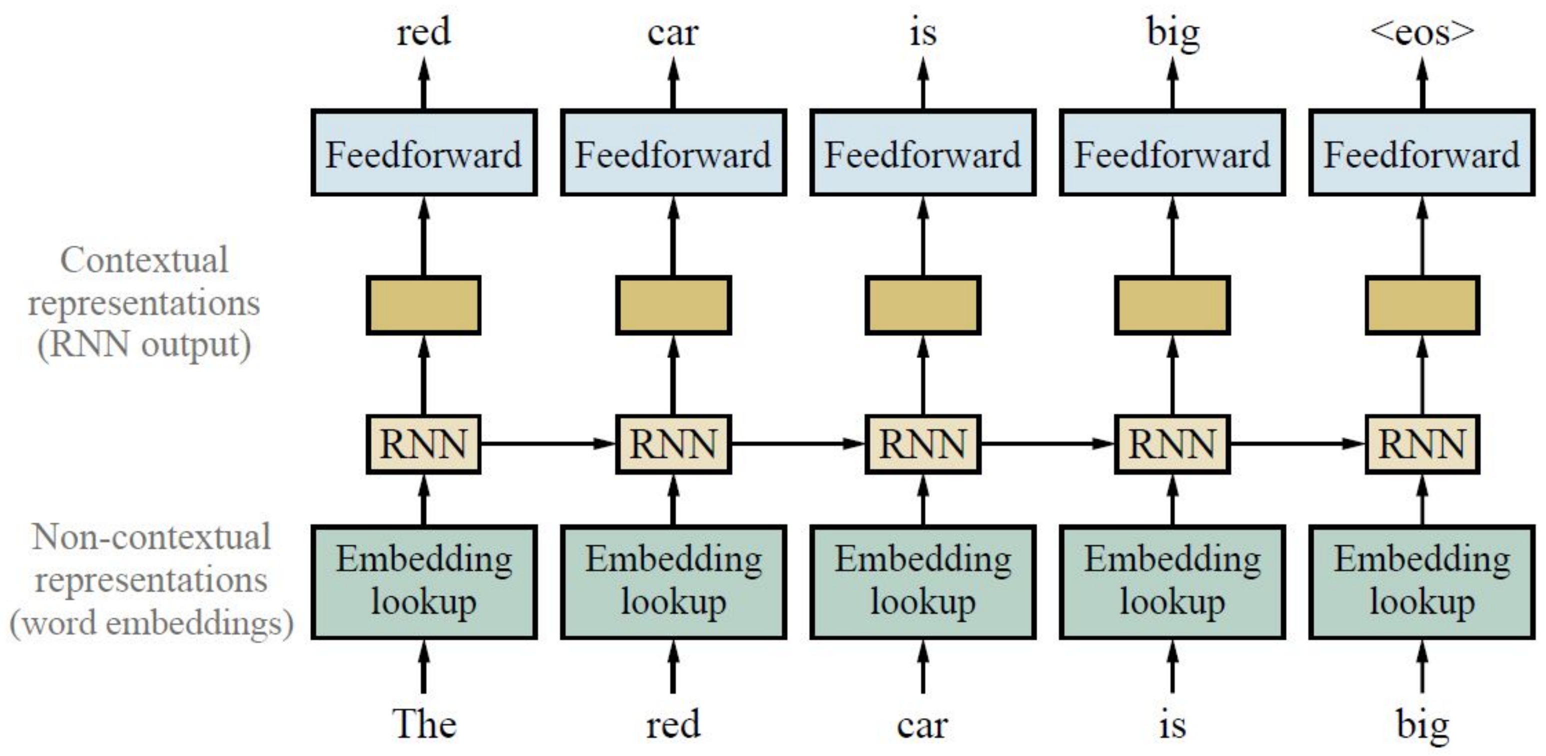
GloVe (Global Vectors) model

- specific model for word embeddings
- relationship between two words can best be captured by comparing them both to other words
- starts by gathering counts of how many times each word appears within a window of another word, similar to the skip-gram model
- choose window size (perhaps 5 words) and let $X_{i\,j}$ be the number of times that words i and j co-occur within a window, and let X_i be the number of times word i co-occurs with any other word.
- Let $P_{i\,j} = X_{i\,j}/X_i$ be the probability that word j appears in the context of word i. As before, let $\mathbf{E}_i$ be the word embedding for word i.

$$\boldsymbol{E}_i \cdot \mathbf{E}'_k = \log(P_{i\,j}) \;.$$

Pretraining and Transfer Learning

Pretrained contextual representations



Training contextual representations using a left-to-right language model.

Pretraining and Transfer Learning

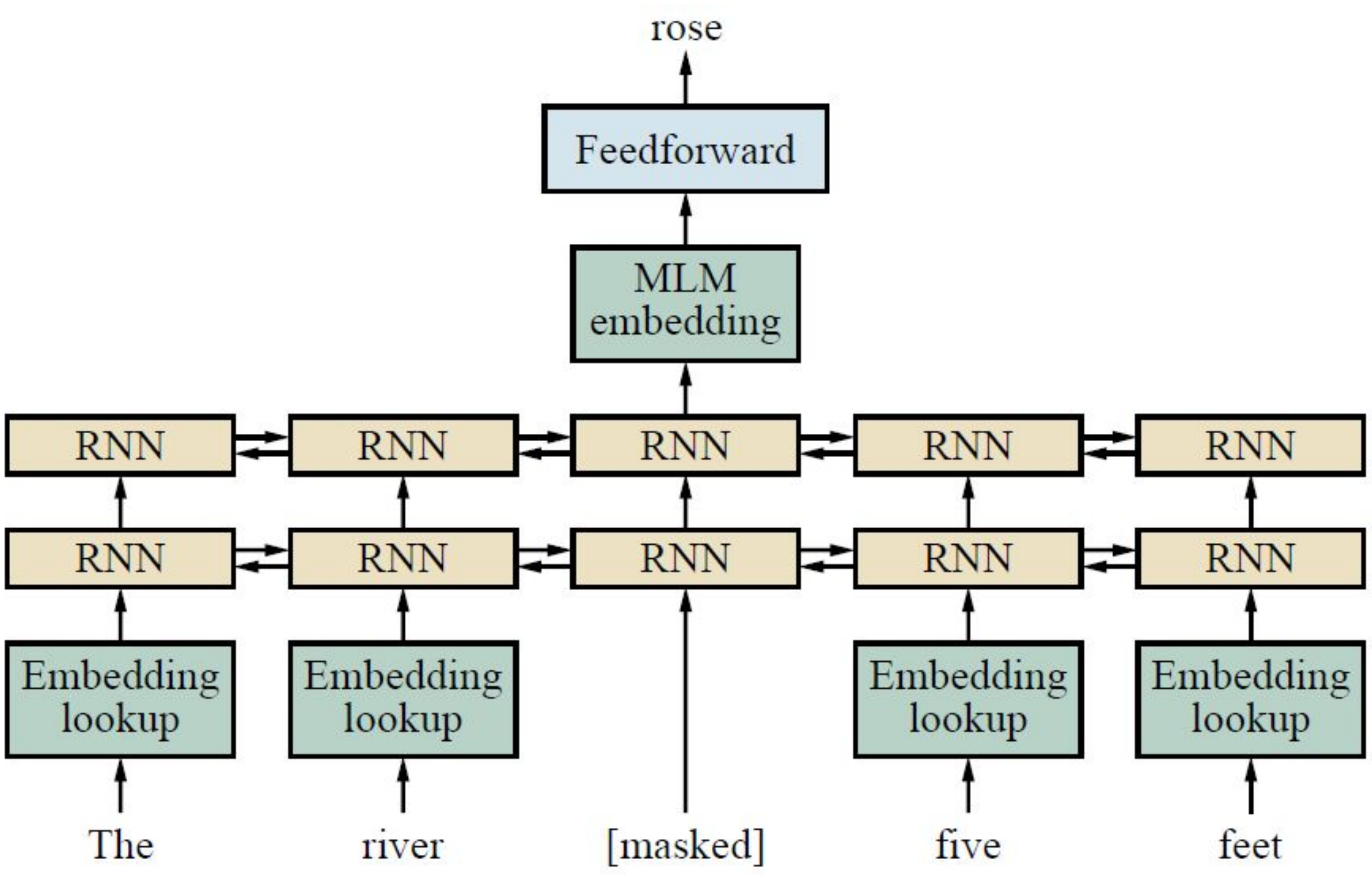
Masked language models

Predictions are made from left to right but sometimes context from later in a sentence

- **Masked language model** (**MLM**) are trained by masking (hiding) individual words in the input and asking the model to predict the masked words.
- For this task, one can use a deep bidirectional RNN or transformer on top of the masked sentence.
- The final hidden vectors that correspond to the masked tokens are then used to predict the words that were masked
- During training a single sentence can be used multiple times with different words masked out.
    - requires no labeled data: the sentence provides its own label for the masked word.

If this model is trained on a large corpus of text, it generates pretrained representations that perform well across a wide variety of NLP tasks (machine translation, question answering, summarization, grammaticality judgments, and others)

Pretraining and Transfer Learning



Masked language modeling: pretrain a bidirectional model—for example, a multilayer RNN—by masking input words and predicting only those masked words

State of the art

Since 2018, new NLP projects typically start with a pretrained transformer model

Although these transformer models were trained to predict the next word in a text, they do a surprisingly good job at other language tasks

A ROBERTA model with some fine-tuningachieves state-of-the-art results in question answering and reading comprehension tests

GPT-2, a transformer-like language model with 1.5 billion parameters trained on 40GB of Internet text, achieves good results on such diverse tasks as translation between French and English, finding referents of long-distance dependencies, and general-knowledge question answering, all without fine-tuning for the particular task.

ARISTO achieved a score of 91.6% on an 8th grade multiple-choice science exam. ARISTO consists of an ensemble of solvers: some use information retrieval (similar to a web search engine), some do textual entailment and qualitative reasoning, and some use large transformer language models.

State of the art

emergence of hybrid approaches that combine the best concepts (grammatical and semantic modeling)

SLING, parses directly into a semantic frame representation, mitigating the problem of errors building up in a traditional pipeline system.

There is certainly room for improvement: not only do NLP systems still lag human performance on many tasks, but they do so after processing thousands of times more text than any human could read in a lifetime.

This suggests that there is plenty of scope for new insights from linguists, psychologists, and NLP researchers

Summary

- Continuous representations of words with word embeddings are more robust than discrete atomic representations, and can be pretrained using unlabeled text data.
- Recurrent neural networks can effectively model local and long-distance context by retaining relevant information in their hidden-state vectors.
- Sequence-to-sequence models can be used for machine translation and text generation problems.
- Transformer models use self-attention and can model long-distance context as well as local context. They can make effective use of hardware matrix multiplication.
- Transfer learning that includes pretrained contextual word embeddings allows models to be developed from very large unlabeled corpora and applied to a range of tasks