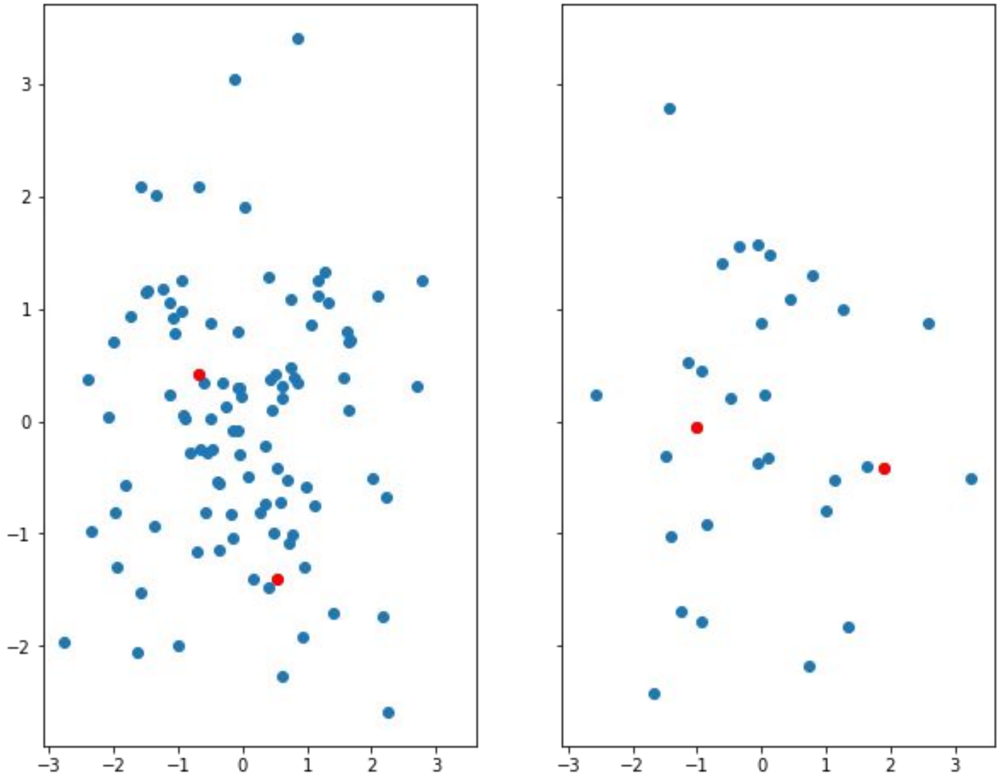
Week 4: Attention and Transformers

Text Analytics and Natural Language Processing
Instructor: Benjamin Batorsky

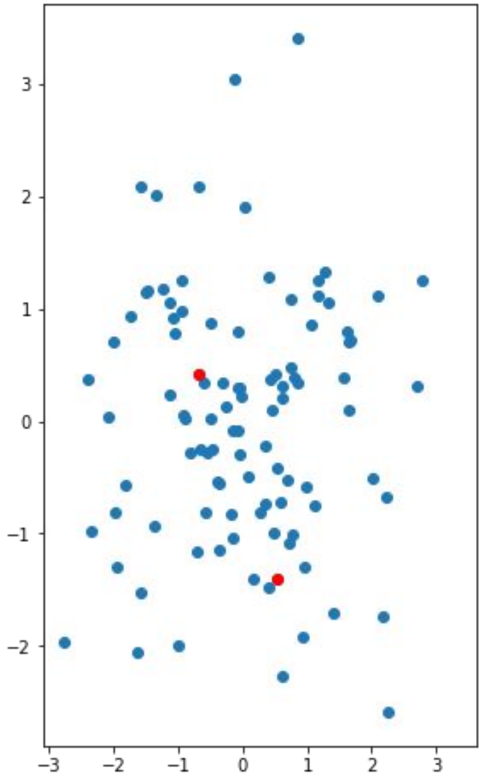
Using/comparing decompositions

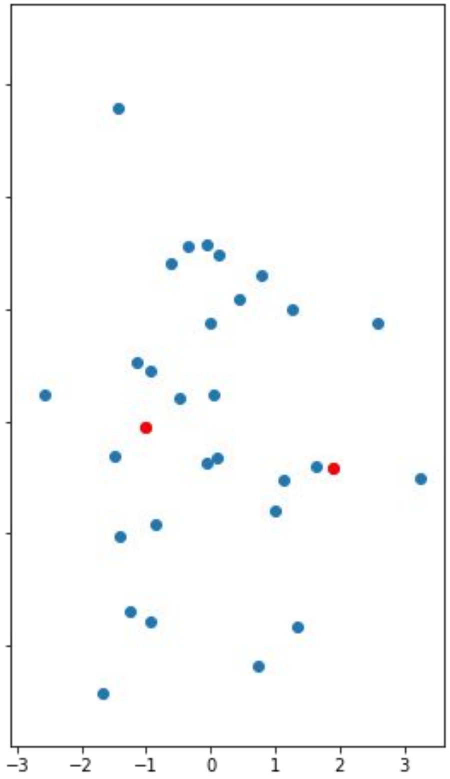
- Same data, different subsets, different PCAs
- Are these two red dots closer in the left or in the right?
- If these are two related documents, which has done a better job of creating a representation?



Using/comparing decompositions

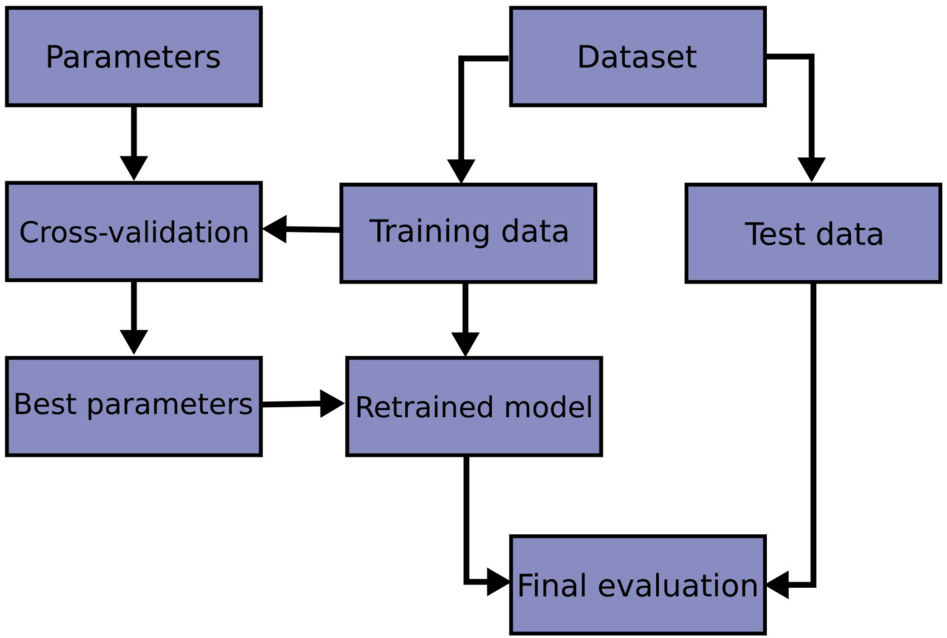
- Same data, different subsets, different PCAs
- Are these two red dots closer in the left or in the right?
- If these are two related documents, which has done a better job of creating a representation?
  - It’s not clear!
  - The dimensions being displayed are different!
- However: Within-decomposition, we can assess their relatedness and compare THOSE





Hyperparameter tuning

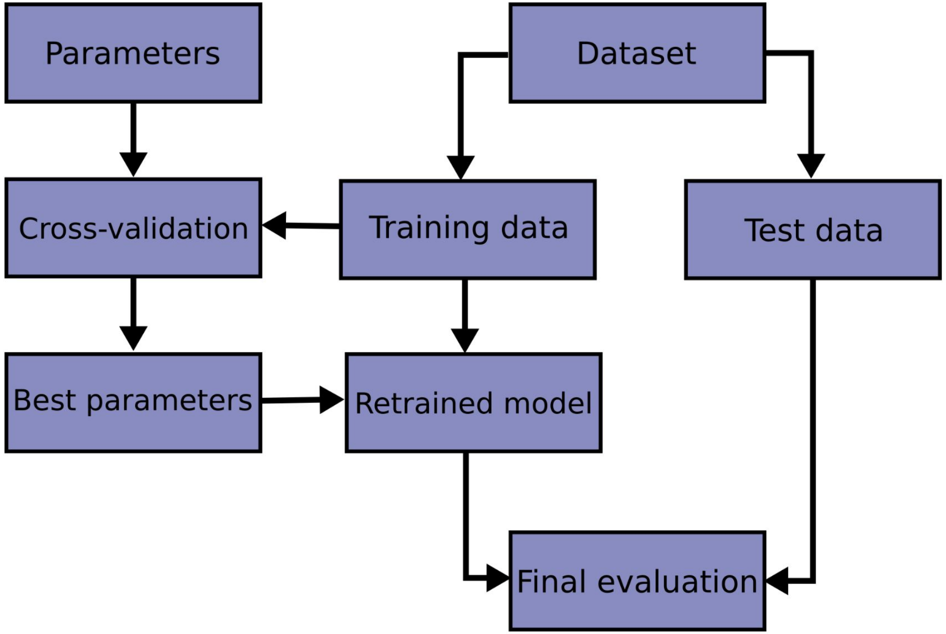
- Training set
- Validation set
- Test set



https://amueller.github.io/ml-training-intro/slides/03-cross-validation-grid-search.html#18

Hyperparameter tuning

- Training set
  - Subset of data on which the model is trained
- Validation set
  - Subset of data on which you monitor performance when designing/tweaking your model
- Test set
  - Subset of data for final evaluation



https://amueller.github.io/ml-training-intro/slides/03-cross-validation-grid-search.html#18

Online midterm (to be posted Wednesday)

- Multiple-choice section
  - Important terms/methods, what do they mean, what do they do, why are they important
- Short answer section
  - Multi-part questions on application/evaluation
  - Requires writing out some code
- Application section
  - Specific question, specific application
  - Open-ended application

Collaboration during midterm

- You're all welcome to make online study groups! Let us know and we can make a Zoom room for you.
- You are NOT permitted to work together on the exam.
  - (I also don't think it'd help much!)
- Do NOT post on Piazza about the midterm until the end of Week 5
  - If you have questions, ask via Inbox or via private message on Piazza

Final project outline (Assignment 4)

- Now on Canvas!
- Due (**latest!**): Week 6, Monday, 7/27, 6:30pm ET
- Earlier submission = earlier feedback
  - NOTE: Covering transformer models this week and scoping next week
- Must include
  - Names of project members (no more than 3!)
  - Overview of sections
- Can include
  - Any questions you have

Outline must detail your strategy around the following sections of the project

- A clear research question or problem to address
- A description of the dataset and justification of why it is appropriate to solve the problem
- Exploratory analyses of the dataset
- A description of the methodology to be applied and justification of its use
- Deployment strategy
  - (optional, if appropriate)

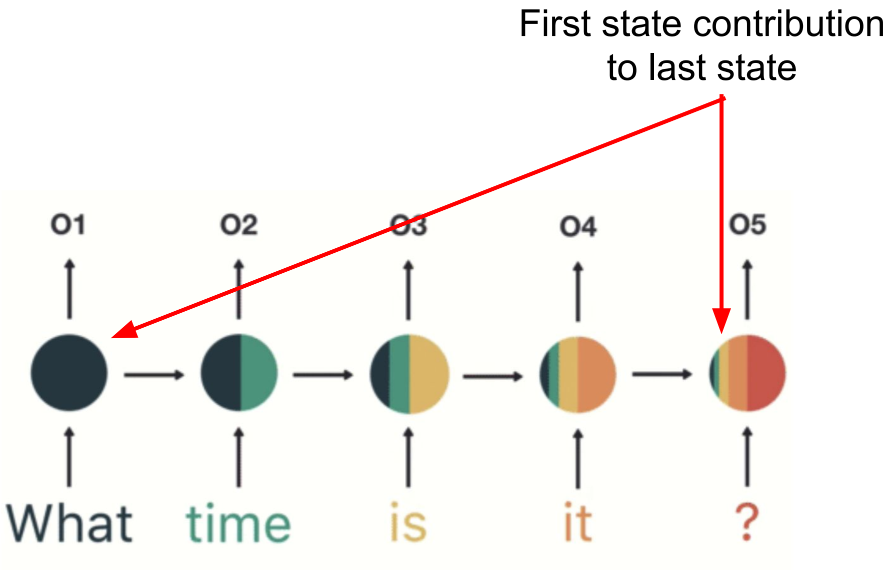
Exciting stuff: Tentative speaker schedule

- Weeks 5, 6 and 7 will have some speakers talking about real-world NLP
- Week 5:
  - Monday @ 7pm-8pm: Andrew Thierrault, former Chief Data Officer, City of Boston
  - Wednesday **@ 6:30-7pm**: Matthew Honnibal, SpaCy author, Explosion.ai founder
- Week 6:
  - Wednesday: Mady Mantha, Senior Technical Evangelist, Rasa AI
- Week 7:
  - Monday: Maryam Jahanshahi, Research Scientist, TapRecruit

NEW readings will be posted ahead of time (will post for Week 5)

Review: Recurrent Neural Networks

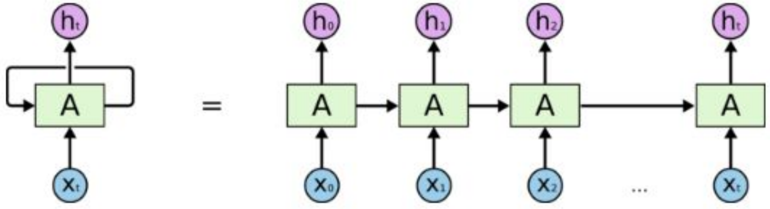
- Information from previous states maintained in “hidden state”
- Single “cell”, single set of weights
  - Updated from loss function backward
- Major issue: Long-term dependencies
  - Smaller updates for earlier states in long sequence
- Adapt structure to deal with these issues



https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9

Major correction: RNN/LSTMs handle variable-length inputs

- Previous lecture: # of cells = # of states
- This is NOT the case
    - RNN cell is "recurrent", it is reused each state in sequence
    - Cell "A" here = 1 cell, 1 vector of weights, 1 bias
    - Backward pass = updates weight based on state contribution
- Can handle variable-length inputs
- However:
    - Training on individual observations is slow/not typical
    - Batch training: Need to have all the same sequence length
    - Usually use padding/truncation
- **NOTE:** Gradients on padded elements are zero-ed out (doesn't learn from the pads themselves)

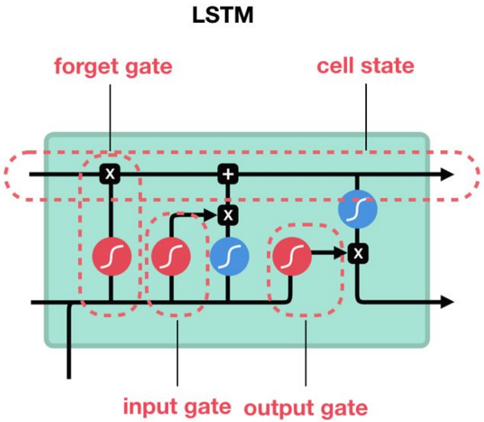


An unrolled recurrent neural network.

https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e

Review: Long Short-Term Memory (LSTM)

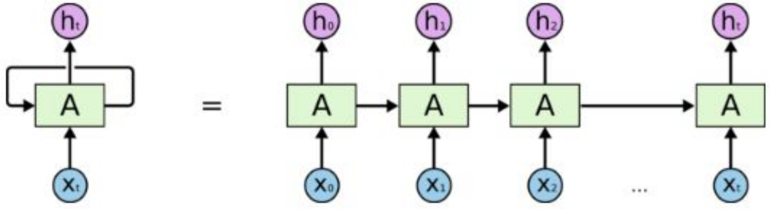
- LSTM networks
  - Added complexity to recurrent cell structure
    - “Cell state” along with “hidden state”
  - Weight information according to importance (learned from training)
  - Four sets of weights + bias
    - Forget, Update, Input, Output
- Still may have difficulty with longer passages



https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

Major correction: RNN/LSTMs handle variable-length inputs

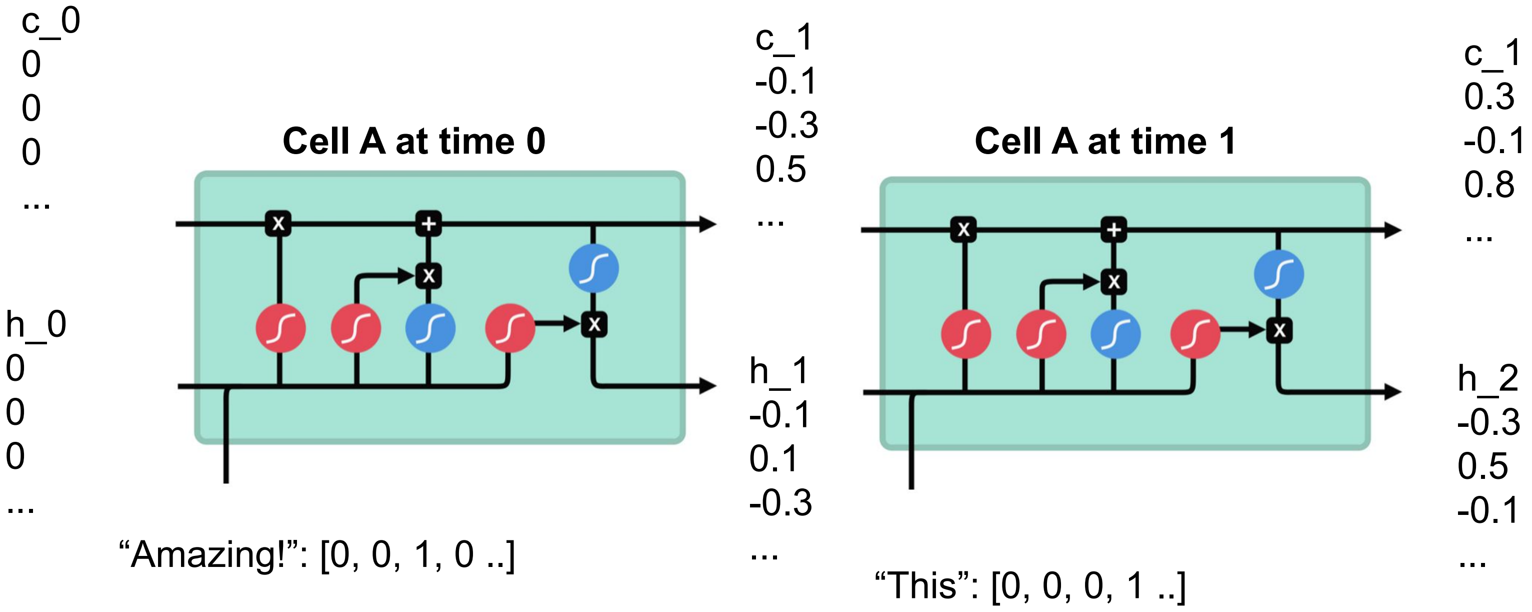
- Previous lecture: # of cells = # of states
- This is NOT the case
  - RNN cell is “recurrent”, it is reused each state in sequence
  - Cell “A” here = 1 cell, 1 vector of weights, 1 bias
  - Backward pass = updates weight based on state contribution
- Can handle variable-length inputs
- However:
  - Training on individual observations is slow/not typical
  - Batch training: Need to have all the same sequence length
  - Usually use padding/truncation
- **NOTE:** Gradients on padded elements are zero-ed out (doesn’t learn from the pads themselves)



An unrolled recurrent neural network.

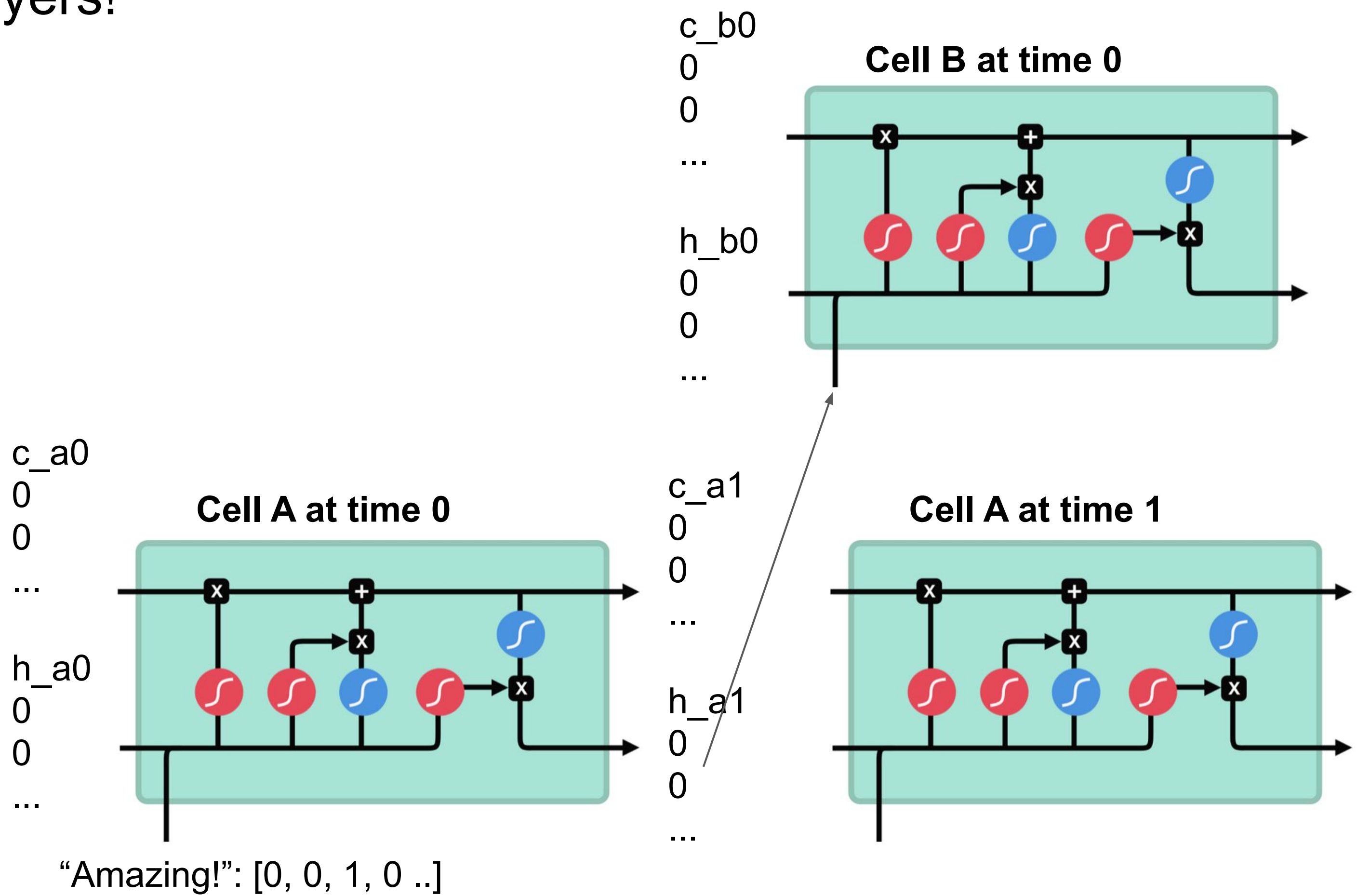
https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e

Example with product review (LSTM)



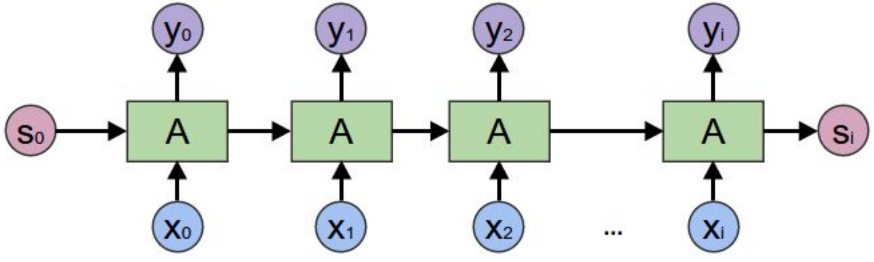
https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

More layers!


c_b0
0
0
...
h_b0
0
0
...
Cell B at time 0
c_a0
0
0
...
h_a0
0
0
...
Cell A at time 0
c_a1
0
0
...
h_a1
0
0
...
Cell A at time 1
"Amazing!": [0, 0, 1, 0 ..]

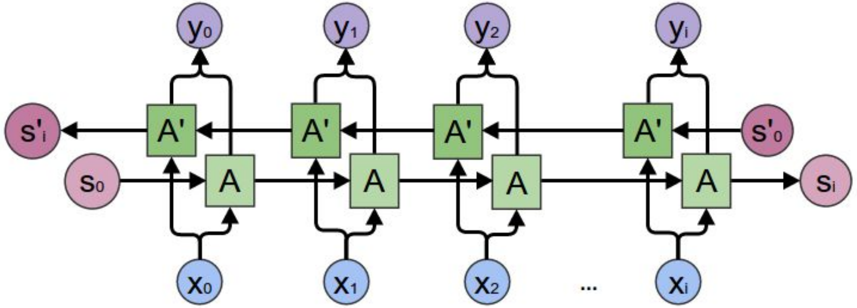
Bi-directional LSTMs

Single directional LSTM




```
LSTM(embedding_dim, hidden_dim, n_layers,
     dropout=dropout_prob, batch_first=True,
     bidirectional=False)
```

Bi-directional LSTM



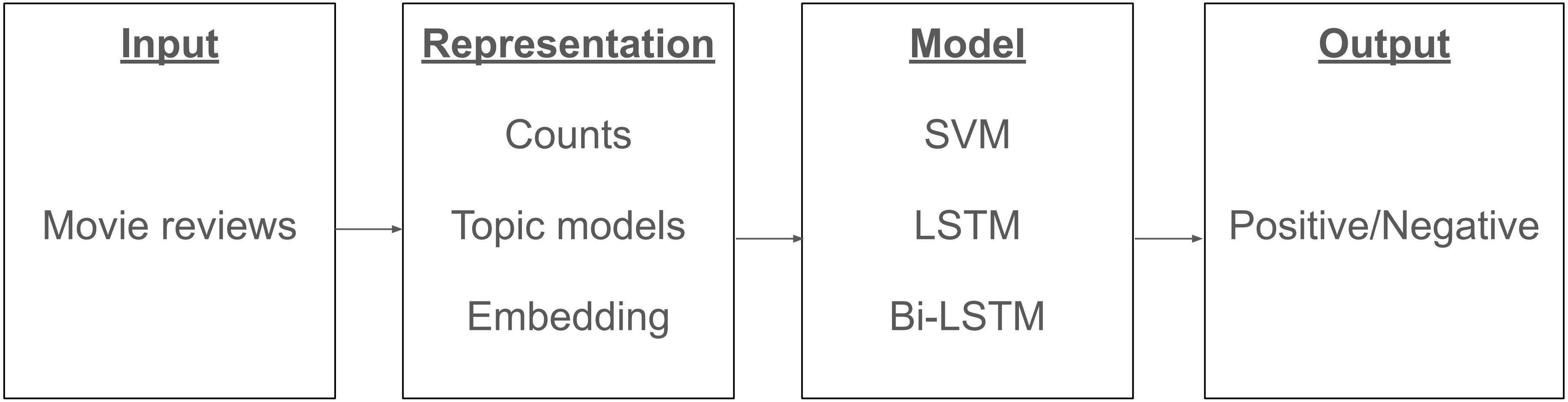

```
LSTM(embedding_dim, hidden_dim, n_layers,
     dropout=dropout_prob, batch_first=True,
     bidirectional=True)
```

http://colah.github.io/posts/2015-09-NN-Types-FP/

Review: History of NLP

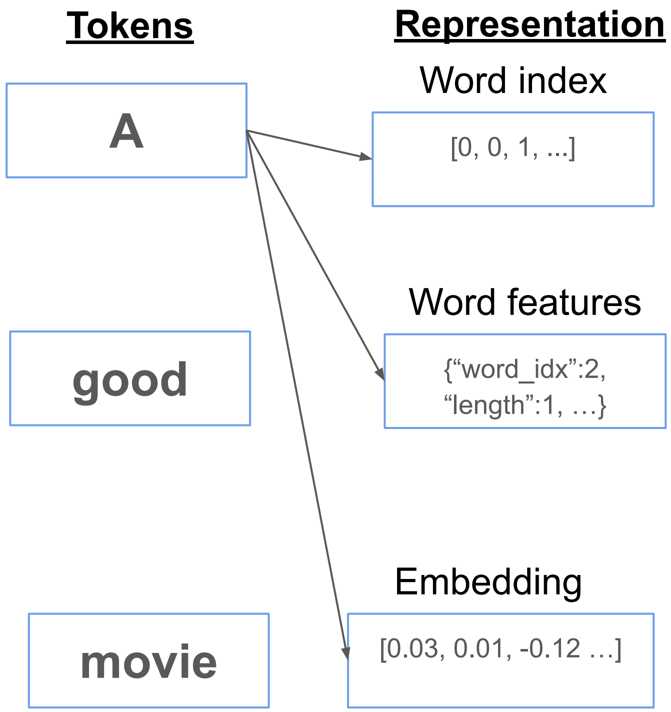
- 40s-50s: Machine translation era
- 60-70s: Shift towards semantic-driven processing
- 70s to 80s: Community expansion
- 90s-00s: Probabilistic/Statistical models
  - Also expansion of available data
- 2000s: Neural Language models
- 2008: Multi-task learning
- 2013: Word embeddings
- 2014: Expansion of Neural models
- 2015: Attention
- 2018 and beyond: Language model advancements

High-level overview of supervised NLP pipeline

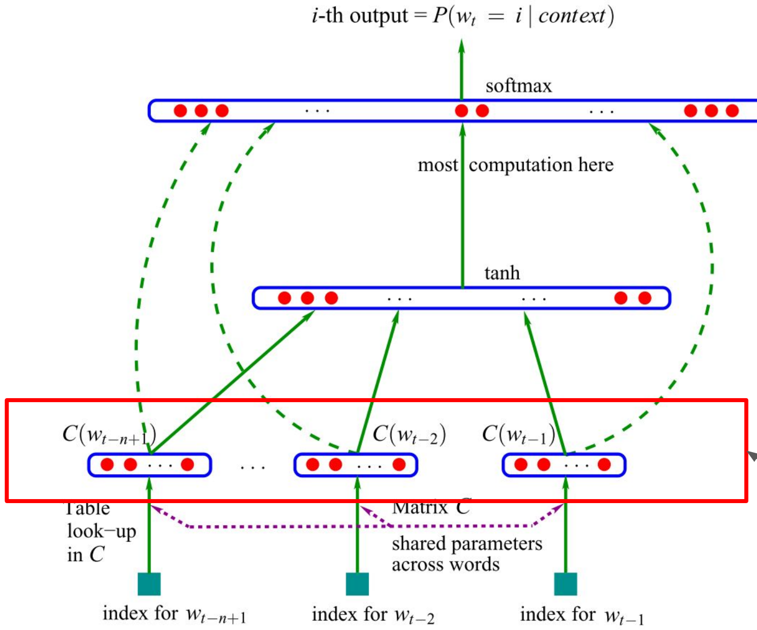


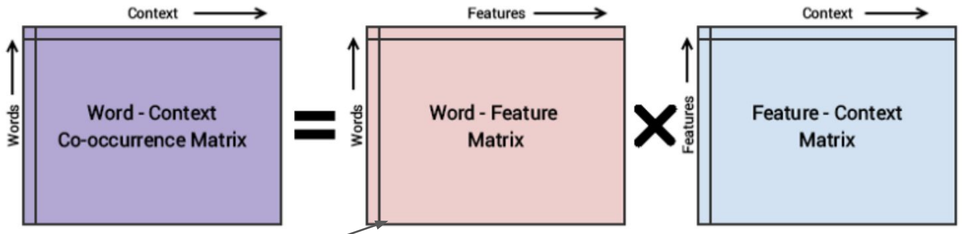
Review: How to represent words

- In notebook: Words as sparse vectors (one-hot encoded)
- "Car" vs "automobile" totally different vectors
- Model needs to learn weights for every word in vocabulary
- Condensed, informative representation
    - Word-level representations from topic models
    - Embeddings



Language model vs GloVe





Conceptual model for the GloVe model's implementation

Embeddings

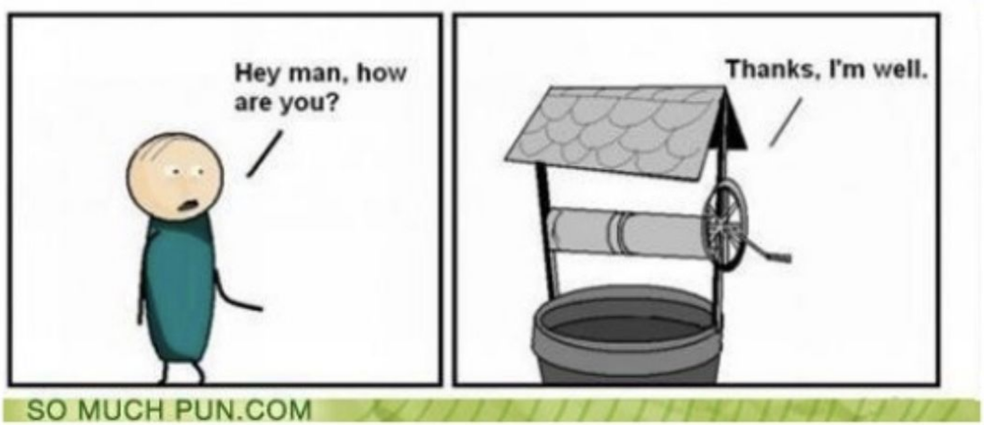
A Neural Probabilistic Language Model
(https://papers.nips.cc/paper/1839-a-neural-probabilistic-language-model.pdf)

https://arxiv.org/pdf/1301.3781.pdf

Importance of context in language

- Negation
  - "It was good" vs "It was not good"
- Coreferences
  - "I saw the movie. It was bad"
- Homonyms
  - "Doing well" vs "wishing well"
- Entailment/Contradiction
  - "I liked the actors. I didn't like the story"
- Directionality/Causation
  - "Dog bit man" vs "Man bit dog"

Does GloVe solve these problems?



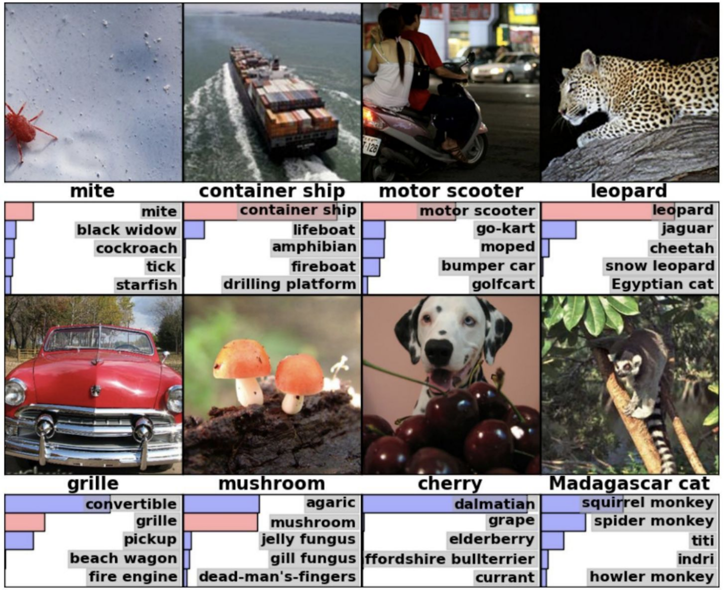
Review: History of NLP

- 40s-50s: Machine translation era
- 60-70s: Shift towards semantic-driven processing
- 70s to 80s: Community expansion
- 90s-00s: Probabilistic/Statistical models
  - Also expansion of available data
- 2000s: Neural Language models
- 2008: Multi-task learning
- 2013: Word embeddings
- 2014: Expansion of Neural models
- 2015: Attention
- 2018 and beyond: Language model advancements

Learning from computer vision

ImageNet Challenge

IMAGENET



- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.

https://ruder.io/nlp-imagenet/

Large, labelled datasets in NLP

- Stanford Question Answering Dataset (SQuAD)
    - 100k question-answer pairs
- Stanford Natural Language Inference Corpus
    - 570k entailment/contradiction pairs
- Machine translation
    - Lots of available resources here
- Constituency parsing
    - Parse trees, also available in a lot of forms
- Language modelling
    - Essentially any text dataset can be used here
    - Example: WikiText-2, all wikipedia articles

List of public NLP datasets:

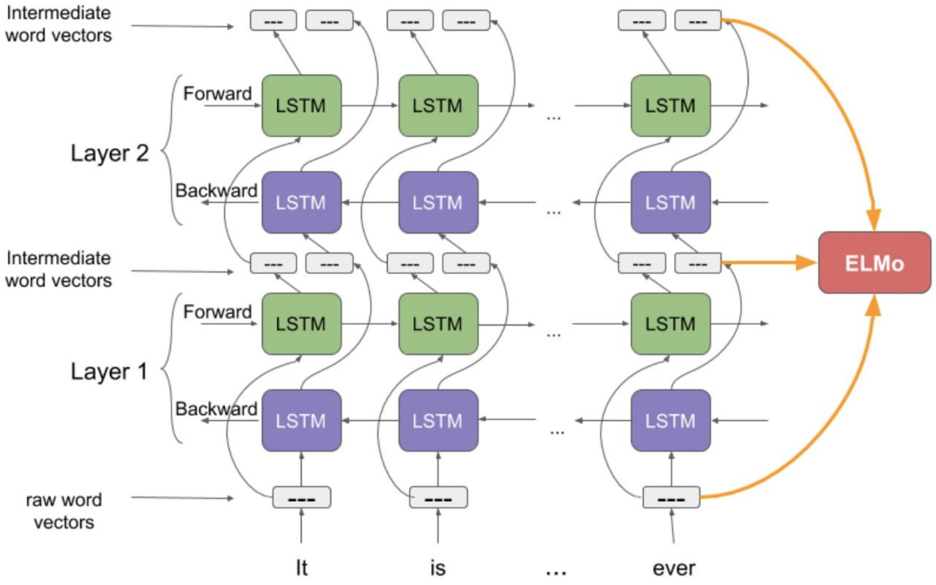
https://github.com/niderhoff/nlp-datasets

Example NLP tasks

https://demo.allennlp.org/

Embeddings from Language Models (ELMo)

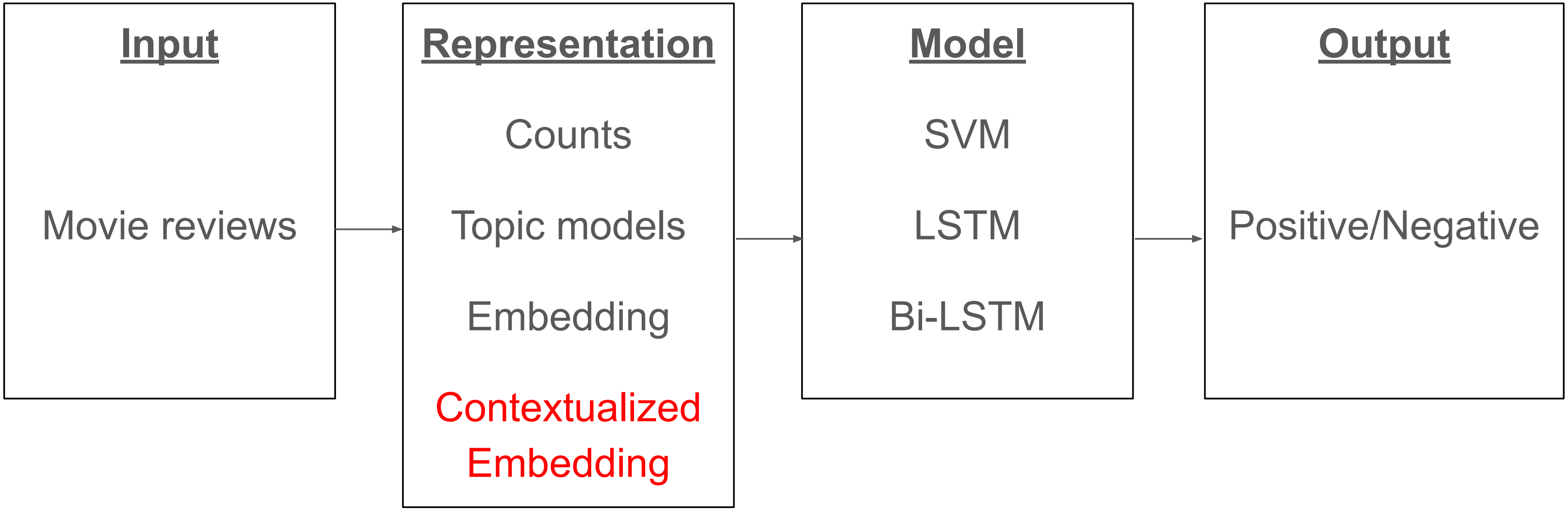
- Raw word vectors
  - Trained as part of the model (not the same as GloVe, etc)
- Two-layer Bi-LSTM with a FC layer combining all information
  - Similar logic to Computer Vision: Each layer learns different “aspects”
- Objective: Predict word given context
  - Has context from previous words + subsequent words
- Final output = word-level vector
  - Incorporates context information and meaning information



https://www.analyticsvidhya.com/blog/2019/03/learn-to-use-elmo-to-extract-features-from-text/

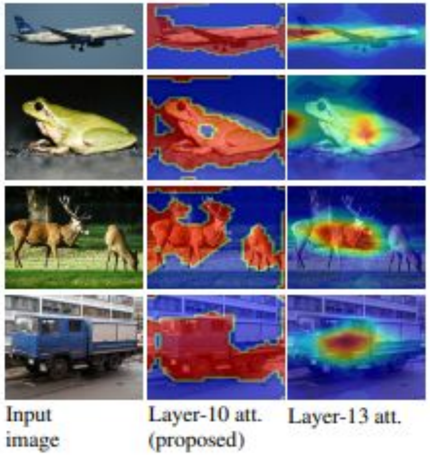
Using ELMo embeddings (notebook)

Back to the NLP pipeline



Again: Learning from computer vision

- Image data
  - Sequence of pixels with color channel intensities
  - Neural models learn weights on pixels
    - Later layers learn from representations of previous
  - Image region weights can be visualized
    - Sometimes referred to as “attention”
- Text data
  - Sequence of tokens
  - Models need to learn dependencies between different tokens

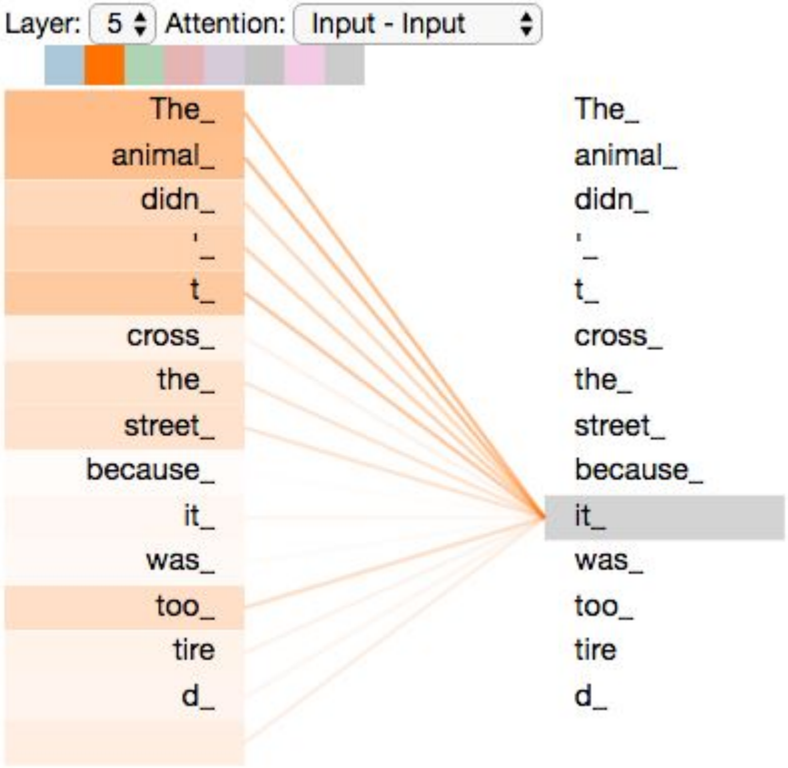


“I watched this movie today. It was bad.”

An example with coreference resolution

“The animal didn’t cross the road because it was tired”

What is “it” referring to?

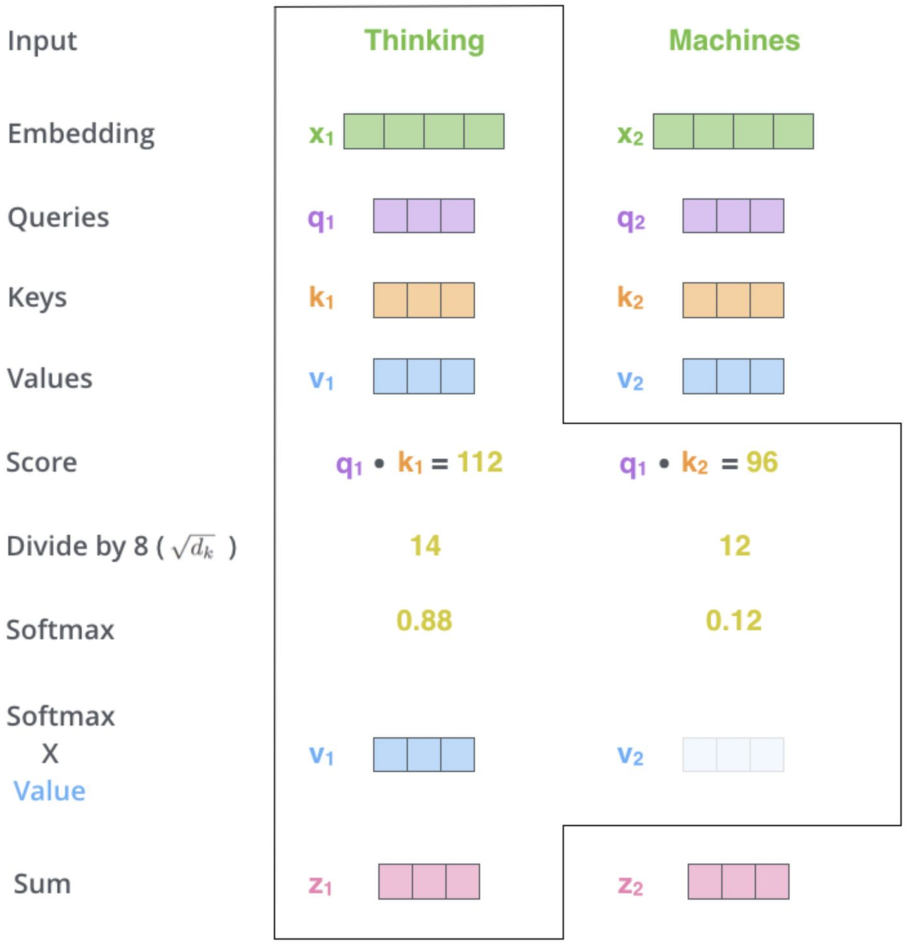


http://jalammar.github.io/illustrated-transformer/

Attention

- Mechanism
  - Calculate weights for a particular model state based on other model states
- Terminology*
  - Given Query, attend to Values based on Keys
  - Query: Target representation
  - Key: Representation in same domain as Query
  - Value: Representation linked to Key
- Example
  - Query: User's search on Youtube
  - Key: Youtube video info (e.g. title)
  - Value: The video itself

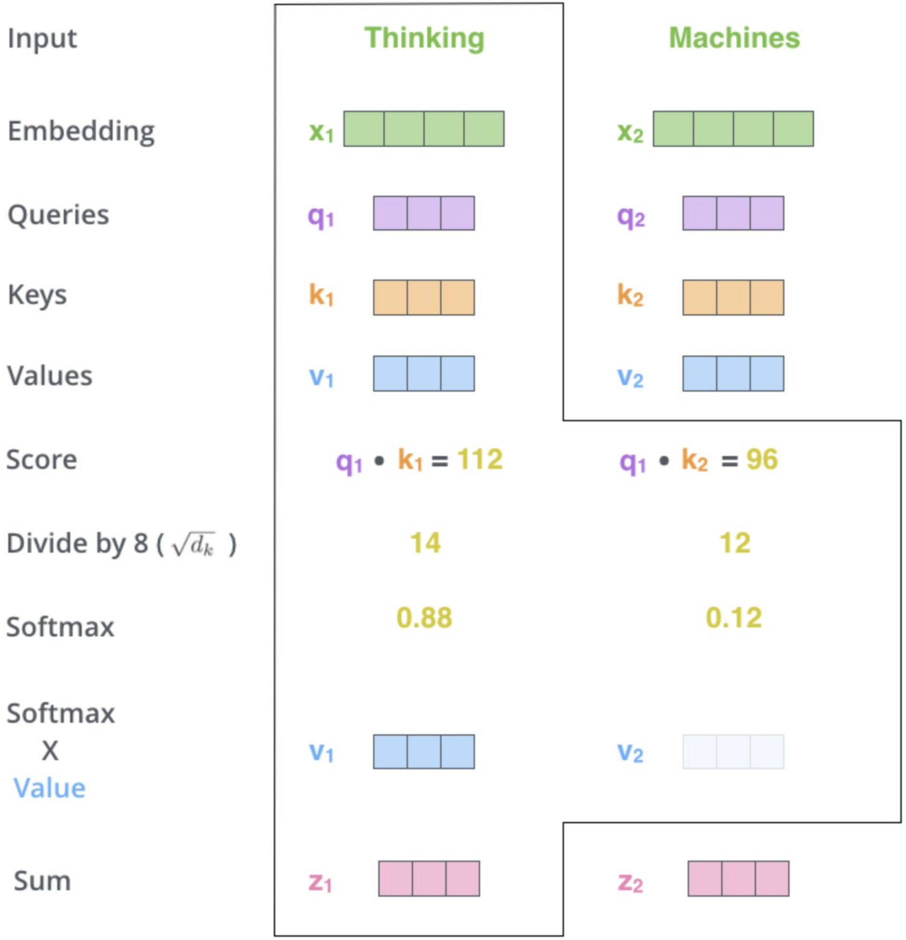
*Note: Terminology vary, definitions are generic



http://jalammar.github.io/illustrated-transformer/

Types of attention

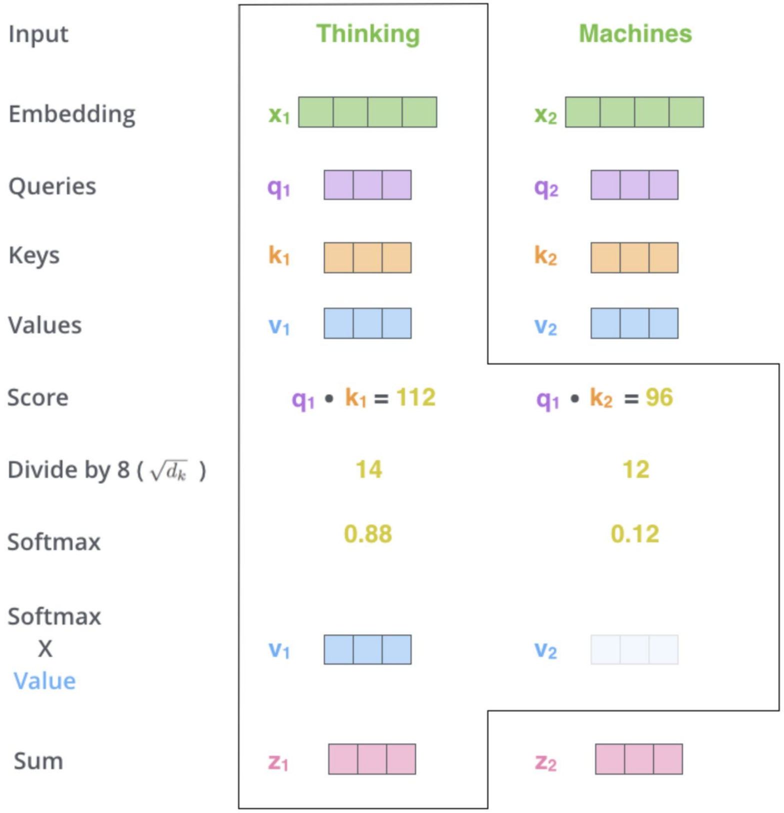
- Self-attention
  - Weights for different states in the same sequence (e.g. different words in the same document)
  - Query for state i, key-values for all other states
- Additive attention
  - Add up the attention between one state and all other states
- Dot-product attention
  - Dot product of queries and keys
    - Queries/keys can be hidden layers, token embeddings, reprojections, etc
  - Sometimes scaled (e.g. by vector length)



http://jalammar.github.io/illustrated-transformer/

Learning attention parameters

- Our implementation: Dot-product between word vectors
  - Not particularly flexible
  - Most attention on same state
- Most implementations: Learn a attention block-specific representation
  - Can be a linear reprojection (e.g. FCNN) from word embedding to representation
  - “Learning” of attention relationships
- Query, Key Values as parameter matrices

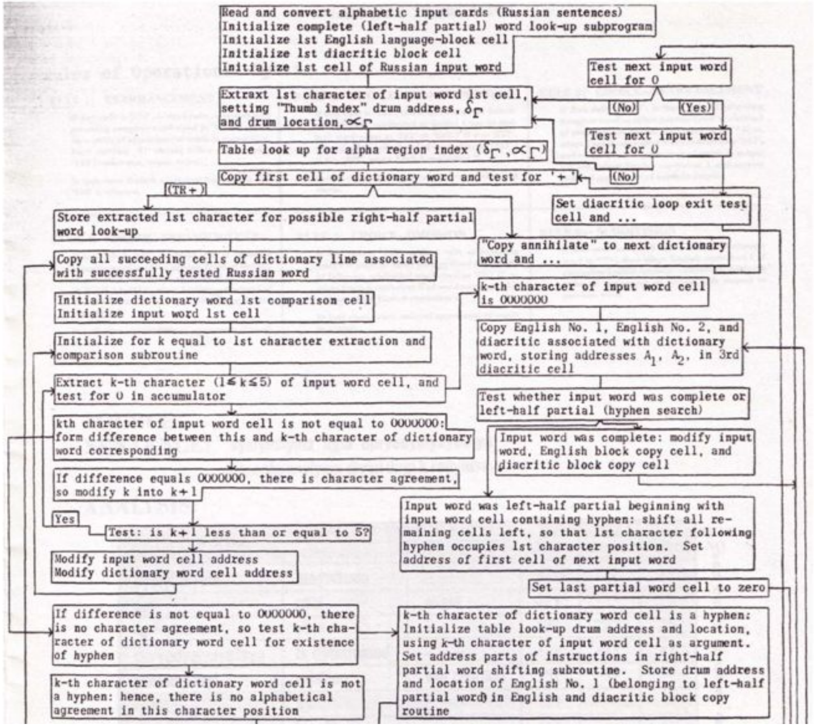


http://jalammar.github.io/illustrated-transformer/

Example implementation of attention (notebook)

Remember machine translation?

1950

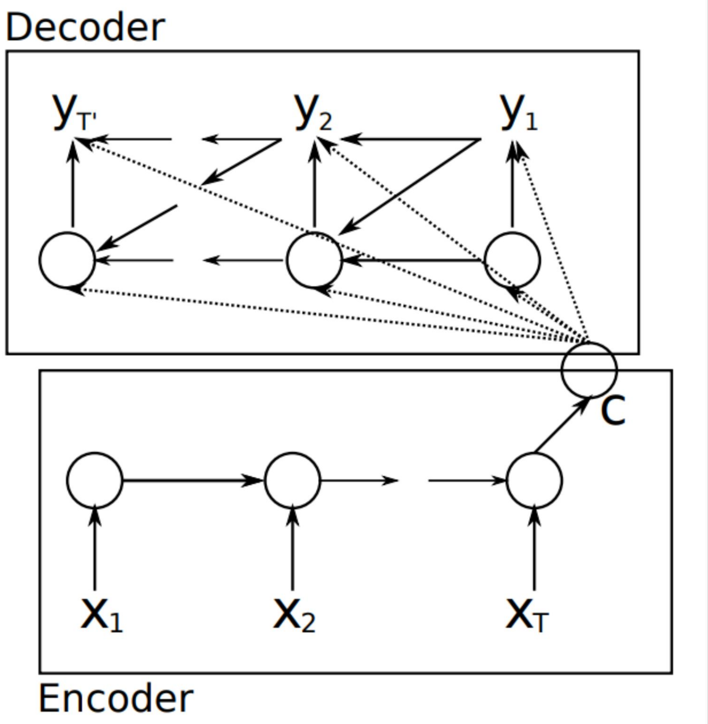


2013

https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Neural Machine Translation: Where attention really shines

- NMT: One of the earliest NLP problems
- Encoder-Decoder models
  - Encoder outputs a final representation (“c” in the diagram)
  - Decoder “decodes” representation
    - Also uses “decoder state”
- Responsible for recent improvement in translation software



[Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation]

Encoder-decoder with attention

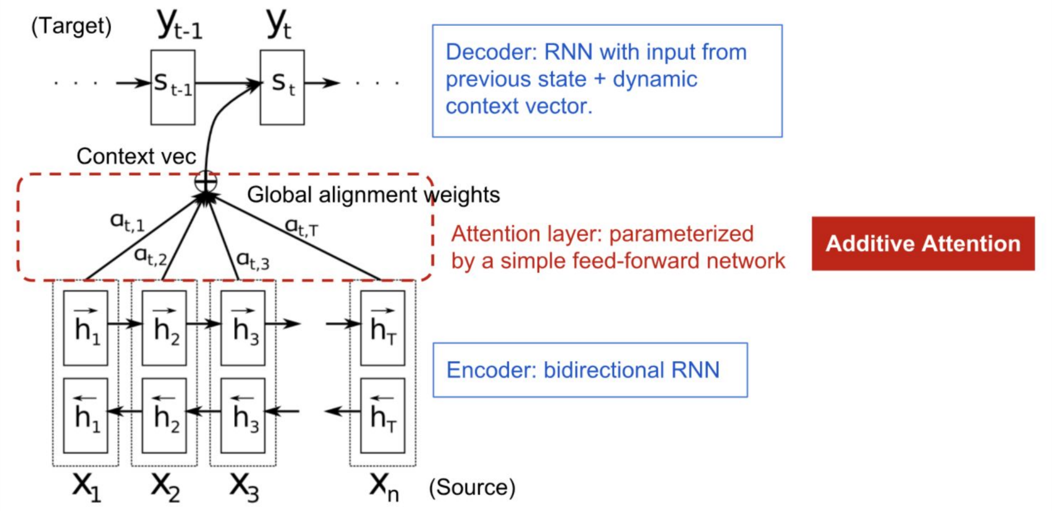
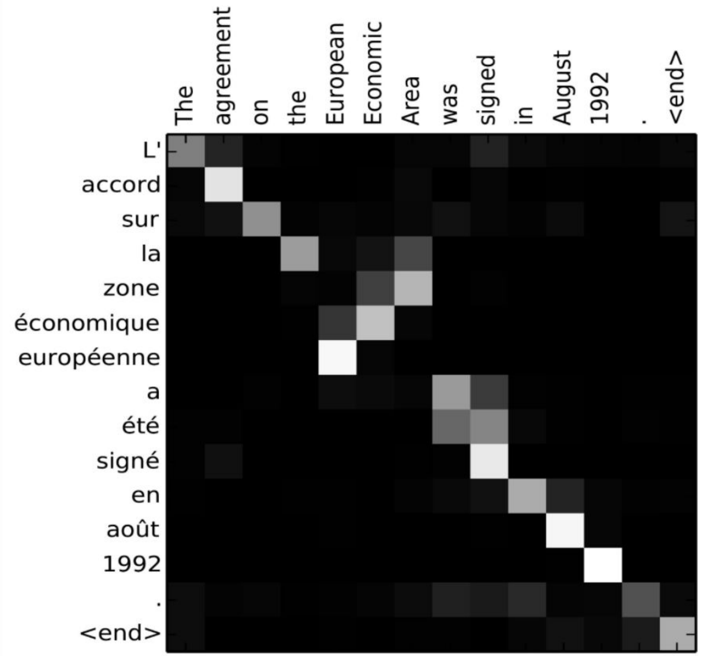
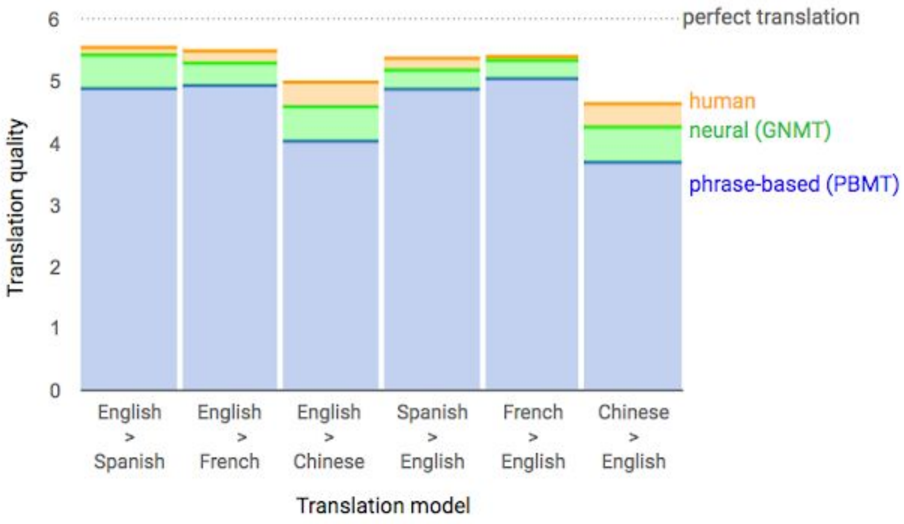


Fig. 4. The encoder-decoder model with additive attention mechanism in Bahdanau et al., 2015.



https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

What it looks like (Google Translate)

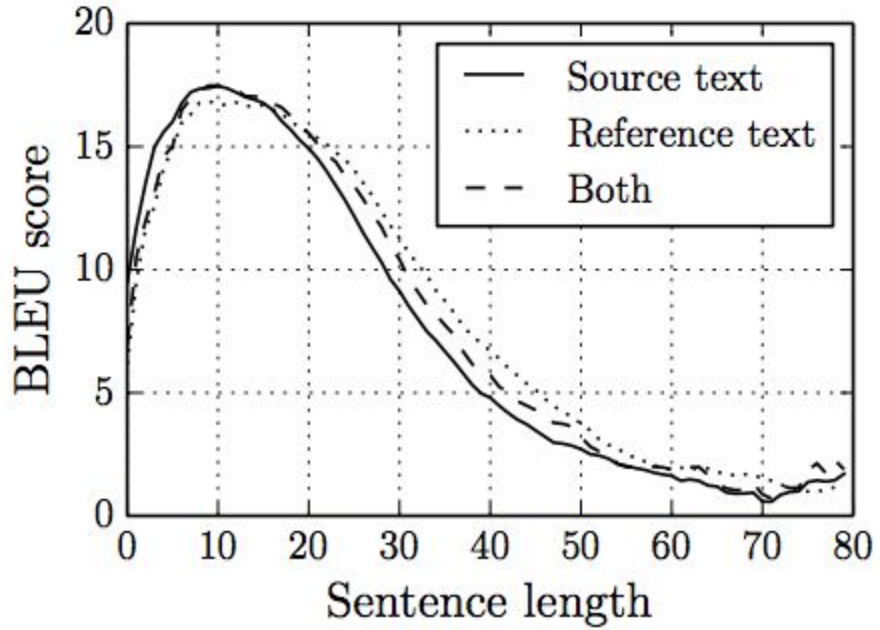


Input sentence:	***Translation (PBMT):***	***Translation (GNMT):***	***Translation (human):***
李克強此行將啟動中加總理年度對話機制，與加拿大總理杜魯多舉行兩國總理首次年度對話。	Li Keqiang premier added this line to start the annual dialogue mechanism with the Canadian Prime Minister Trudeau two prime ministers held its first annual session.	Li Keqiang will start the annual dialogue mechanism with Prime Minister Trudeau of Canada and hold the first annual dialogue between the two premiers.	Li Keqiang will initiate the annual dialogue mechanism between premiers of China and Canada during this visit, and hold the first annual dialogue with Premier Trudeau of Canada.

Spanish->English	Uno no es lo que es por lo que escribe, sino por lo que ha leído.	One is not what is for what he writes, but for what he has read.	You are not what you write, but what you have read.	You are who you are not because of what you have written, but because of what you have read.

https://ai.googleblog.com/2016/09/a-neural-network-for-machine.html

But we still have problems!



[On the Properties of Neural Machine Translation: Encoder-Decoder Approaches]

Is recurrence necessary?

- Handling long-term dependencies
  - LSTMs/GRUs
  - Bi-LSTMs
  - Attention
- Computational complexity
  - Longer sequences/larger vectors = more computation time
  - Difficult to parallelize recurrent structures
- Attention mechanism
  - Can learn “global” relationships
  - More easy to parallelize
- Is attention all we need?

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

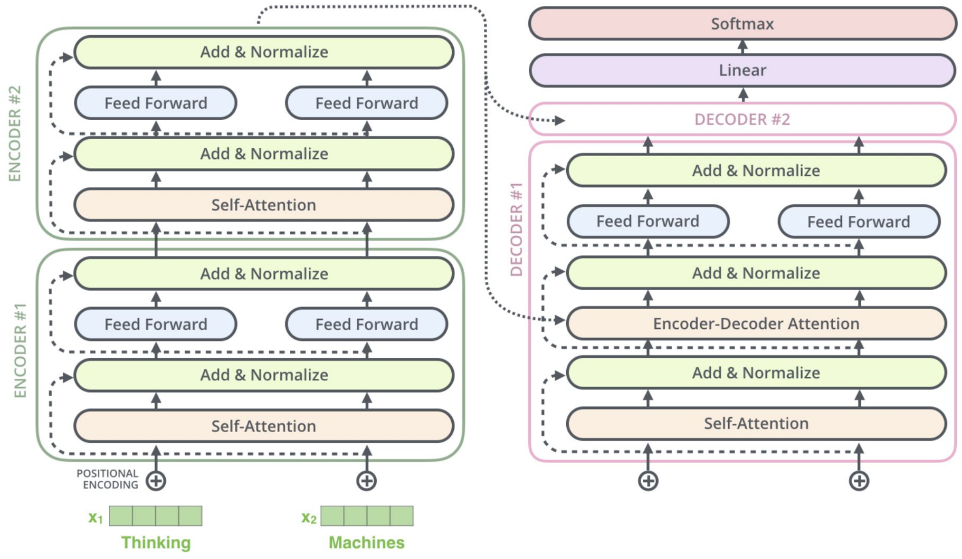
CNN
LSTM
LSTM
CNN+LSTM

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Vaswani, Ashish, et al. "Attention is all you need."
https://arxiv.org/pdf/1706.03762.pdf

Transformer models

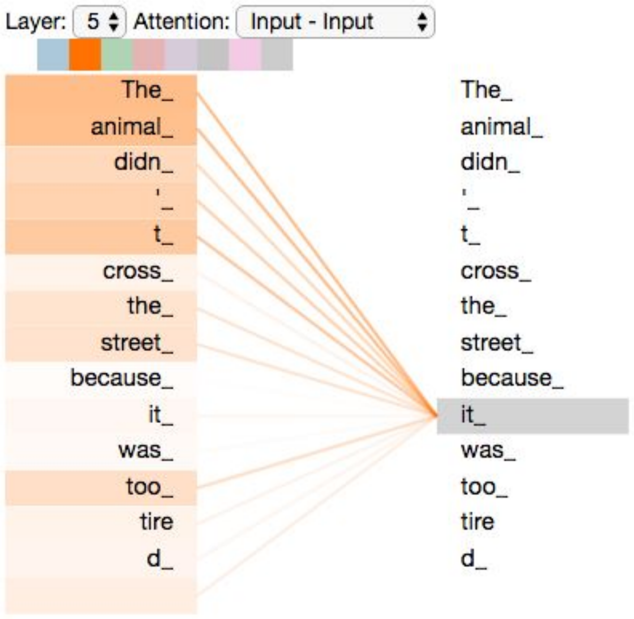
- Encoder-Decoder structure
  - Multiple blocks, output from previous block passed to next
- Input
  - Word embeddings/representations
  - Positional encoding
    - Deterministic function (e.g. cosine) to represent position
- Encoder blocks
  - Self-attention
  - FFNN (aka FCNN) layer
- Decoder blocks
  - Same as encoder, but with added attention layer
    - Attention for decoder relative to encoder



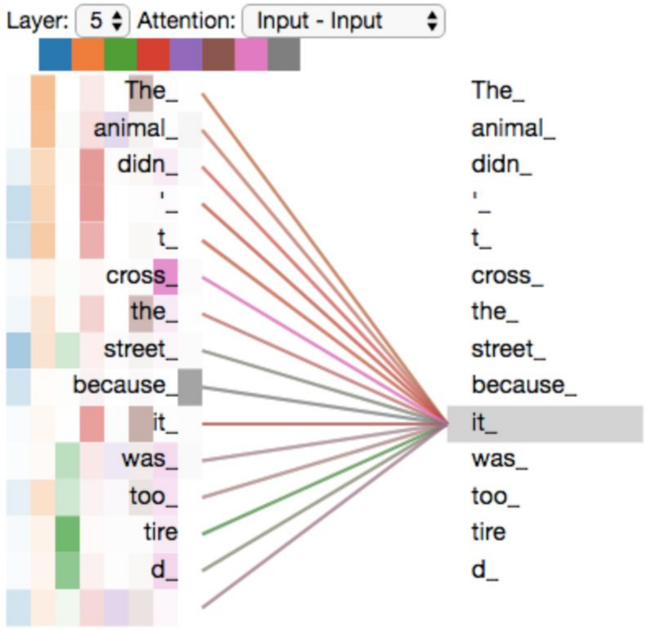
https://jalammar.github.io/illustrated-transformer/

Multi-headed attention

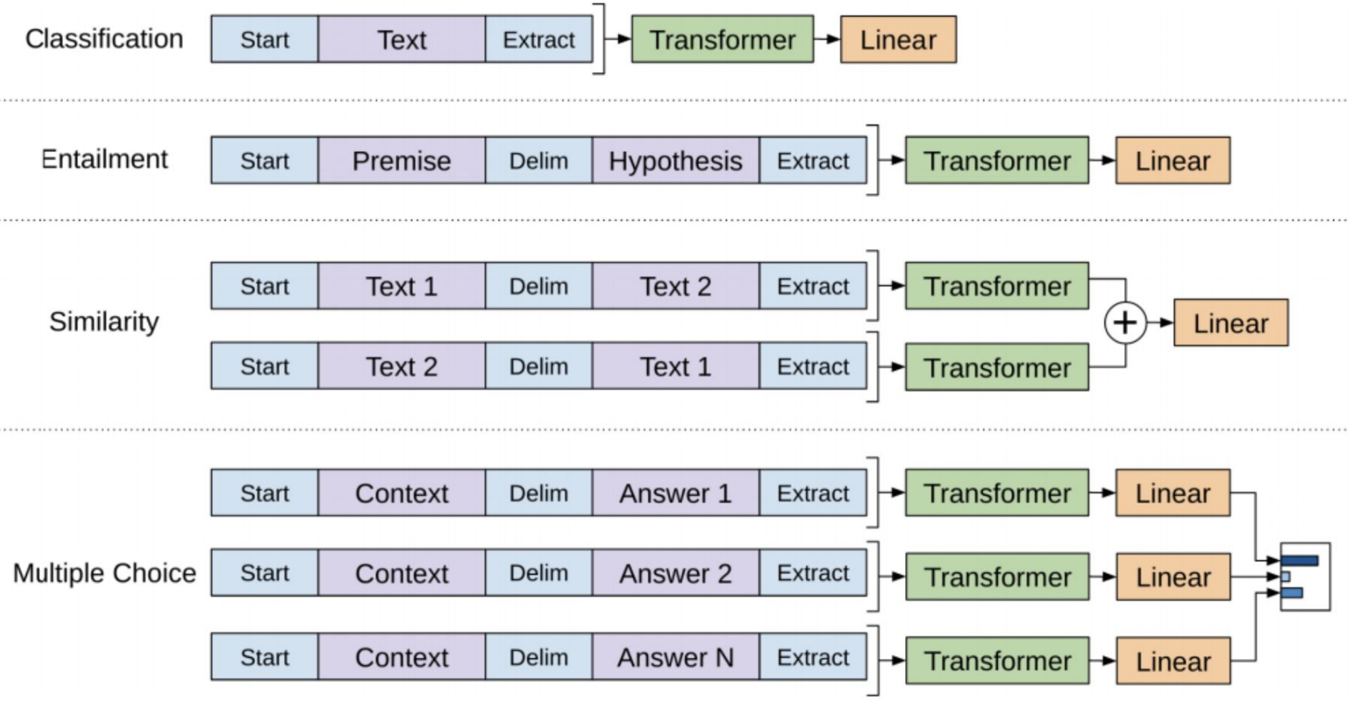
Single-headed attention



Multi-headed attention



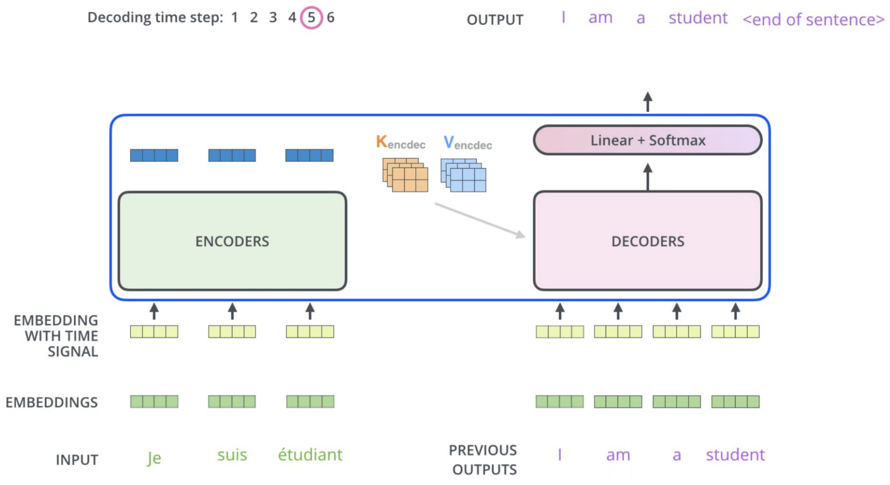
Transfer learning with transformers (Fine-tuning)



http://jalammar.github.io/illustrated-bert/

Issues with vanilla Transformer

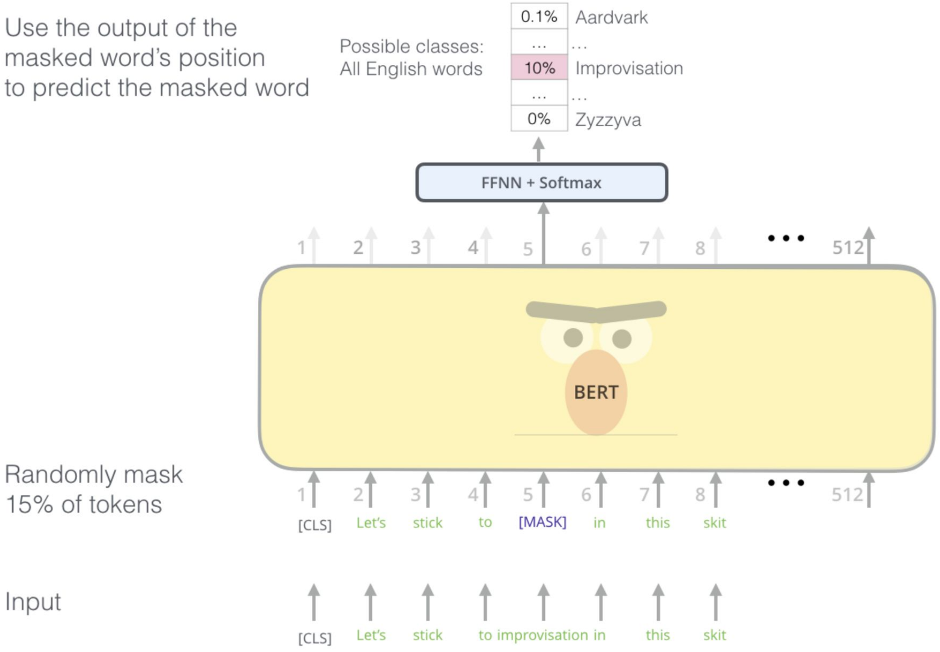
- Language model: Predict the next word
  - Giving the model future words would “leak” the target
- ELMo = Bi-directional LSTM
  - LSTMs are trained jointly
  - No sharing of information from one direction to the other
- Transformer (vanilla)
  - Passes full sequence into encoder
  - Decoder “shifted” one position, each position only builds on information from previous position
- Why can’t we let the decoder look at the whole sequence?
- Is there a way we can remove this limitation?



https://jalammar.github.io/illustrated-transformer/

Bidirectional Encoder Representations from Transformers

- Two major differences
  - Just encoder stack (no decoder)
  - Method of training
- Token masking and replacement
  - Handles the issue of “leaking” target words by replacing word with mask or random word
- Training tasks
  - Predict the masked/replaced word
  - Given two sentences, predict that they’re sequential
- [CLS] token
  - Essentially a document-level representation



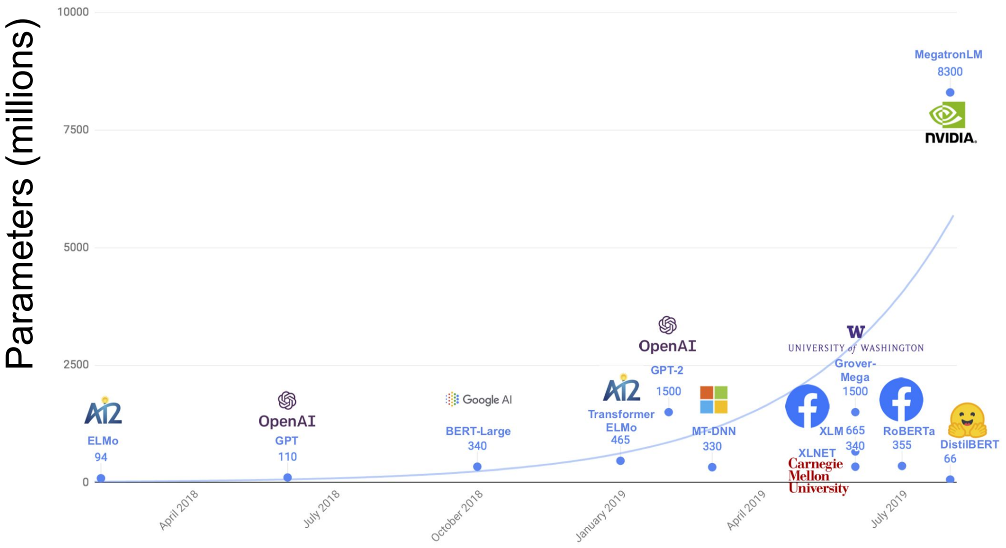
BERT's clever language modeling task masks 15% of words in the input and asks the model to predict the missing word.

https://jalammar.github.io/illustrated-transformer/

Using pre-trained transformer models (notebook)

Brief aside on BERT-ology

- Huge proliferation of BERT-based models
- RoBERTa
  - Optimization on top of base BERT
- DistilBERT
  - ½ the number of parameters, 95% performance of BERT base
- Language-specific (FlauBERT, AlBERTo)
  - Models trained for specific languages



https://medium.com/huggingface/distilbert-8cf3380435b5

Explorations with DistilBERT

Review: History of NLP

- 40s-50s: Machine translation era
- 60-70s: Shift towards semantic-driven processing
- 70s to 80s: Community expansion
- 90s-00s: Probabilistic/Statistical models
- 2000s: Neural Language models
- 2008: Multi-task learning
- 2013: Word embeddings
- 2014: Expansion of Neural models
- 2015: Attention
- 2018 and beyond: Language model advancements

Review: Exploratory/unsupervised techniques

- Tokenization/vocabulary development
  - N-grams
  - Stopwords
  - Non-standard entities/Named-entities
  - Lemmatization/Stemming
- Word count techniques
  - Term frequency
  - Term frequency - inverse document frequency
- Topic modelling
  - Non-negative matrix factorization
  - Latent Dirichlet Allocation
- What’s left out by these approaches to text?

Review: (Some) Methods for including context

- Adding context features
  - N-grams
  - Document-level info (e.g. length)
- Auto-regression models
  - Prediction at state x = info from state x + info from state x-1, etc
- Recurrent models
  - Learned “hidden state” passed to each state
- Word representations learned from context
  - Word2Vec, GloVe

“I liked this movie”

1-grams (Unigrams):

[“I”, “liked”, “this”, “movie”]

2-grams (Bigrams):

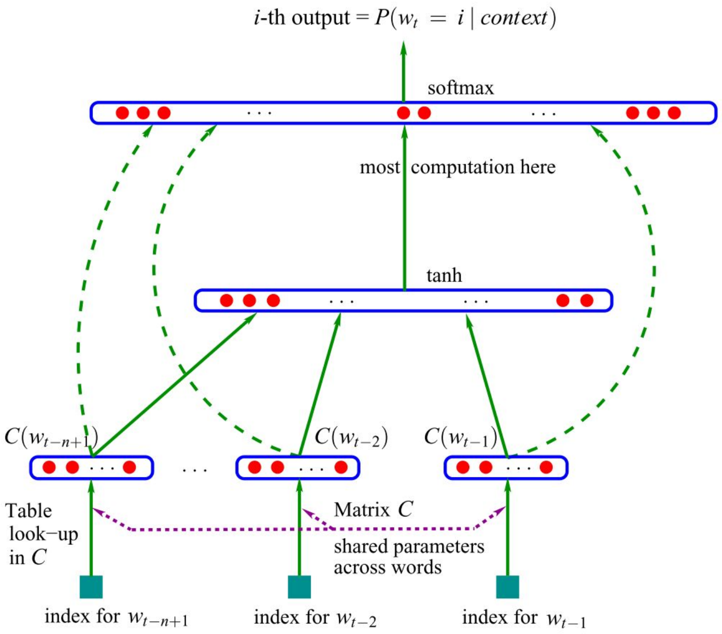
[“I liked”, “liked this”, “this movie”]

4-grams: “I liked this movie”

Review: History of NLP

- 40s-50s: Machine translation era
- 60-70s: Shift towards semantic-driven processing
- 70s to 80s: Community expansion
- 90s-00s: Probabilistic/Statistical models
  - Also expansion of available data
- 2000s: Neural Language models
- 2008: Multi-task learning
- 2013: Word embeddings
- 2014: Expansion of Neural models
- 2015: Attention
- 2018 and beyond: Language model advancements

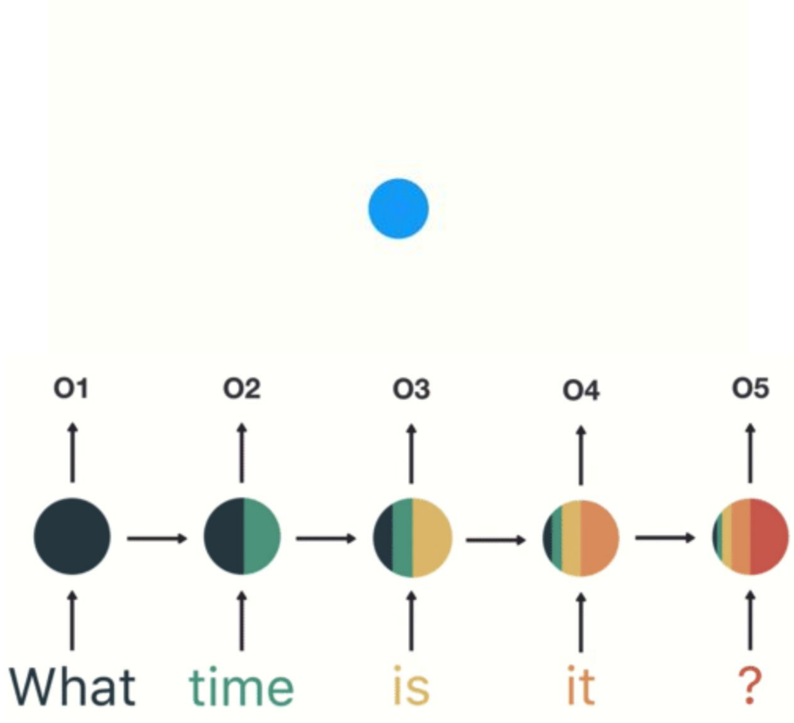
Language modelling



A Neural Probabilistic Language Model
(https://papers.nips.cc/paper/1839-a-neural-probabilistic-language-model.pdf)

Recurrent Neural Networks (RNN)

- Specialized structure for handling sequence information
    - Audio
    - Video
    - Text
- Information from previous states maintained in “hidden state”
- Have representations at each state and representation at the end of the sequence

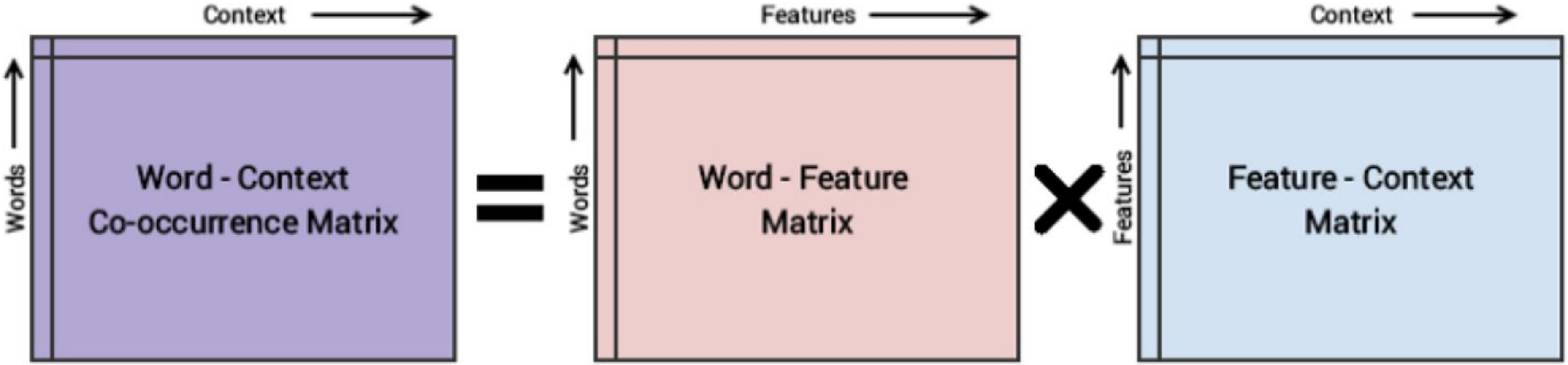


https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9

Review: History of NLP

- 40s-50s: Machine translation era
- 60-70s: Shift towards semantic-driven processing
- 70s to 80s: Community expansion
- 90s-00s: Probabilistic/Statistical models
  - Also expansion of available data
- 2000s: Neural Language models
- 2008: Multi-task learning
- 2013: Word embeddings
- 2014: Expansion of Neural models
- 2015: Attention
- 2018 and beyond: Language model advancements

Global Vectors for Word Representation (GloVe)



Conceptual model for the GloVe model's implementation

Implementing Deep Learning Methods and Feature Engineering for Text Data: The GloVe Model

Review: History of NLP

- 40s-50s: Machine translation era
- 60-70s: Shift towards semantic-driven processing
- 70s to 80s: Community expansion
- 90s-00s: Probabilistic/Statistical models
    - Also expansion of available data
- 2000s: Neural Language models
- 2008: Multi-task learning
- 2013: Word embeddings
- 2014: Expansion of Neural models
- 2015: Attention
- 2018 and beyond: Language model advancements