Language

Arthur J. Redfern
axr180074@utdallas.edu
Nov 28, 2018

Outline

- Motivation
- Word embedding
- Language modeling
- Translation
- Text classification
- Summarization
- Part of speech tagging
- Question answering
- Dialogue agents
- References

Not included in this lecture

Motivation

Why This Abbreviated Choice Of Topics

- Word embeddings allow words to be mathematically worked with conveniently and are needed for most all language processing tasks
  - It's a foundational component that affects the accuracy of all subsequent language processing
- Language modeling was previously used to improve speech to text accuracy and it's used here to improve language translation
  - Since it comes up in so many applications it's worthwhile to look at it in some more detail
  - Efficient language modeling impacts applications
- Language translation builds on the sequence related models previously used for speech and is an excellent success story within deep learning
  - Seeing similar models in different settings helps with the understanding
  - This can be used with previous methods to build larger applications (more on this in a bit)

Disclaimer

- There's a lot of language related stuff not included here
  - Different methods within the categories of problems included here
  - Problems that are not included here
- Possibly some of this will be addressed in future versions of the slides
- Regardless of whether it is or not, hopefully these slides provide enough of a base from which to branch off and learn more on your own

An apology within the disclaimer

I don't spend much time with language

This series of lectures will likely not be as beautiful in organization or presentation as some of the other lectures

Word Embedding

Goal

- From a mathematical perspective it’s cumbersome to directly work with words

- So instead assign a vector to each word (embed a word into a vector)
  - Not practical to use a 1 hot encoding because there are so many words and the vector would be too long (~ 13M for English)
  - Instead use a length 100 – 1000 vector of dense real values (implies mapping does dimensionality reduction)
  - Past some length there are diminishing returns in accuracy improvements in the applications that use them

- It’s useful if the word to vector assignment exhibits some language understanding
  - E.g., the cosine of the angle between vectors corresponding to similar words should be close to 1

- Why?
  - Subsequent processing on the vectors is going to do feature extraction, classification and generation
  - Consider feature extraction: it makes sense that it’s easier to extract features if the feature extractor only needs to learn meaning (or whatever the task) vs learning both meaning and a random mapping
  - Consider generation: it makes sense that if the generator is off by a little and generates a synonym to the target word vs a word with no relationship

The Distributional Hypothesis

- The methods used to embed words in dense vectors in this section all rely on the distributional hypothesis
  - Words that are used in the same context tend to have similar meanings
  - A word is described by the company it keeps
  - Per the previous slide this meaning should be accounted for in the assignment of vectors

- This leads to a typical set of parameters, the choice of which helps determine the assignment
  - Context type
  - Context window
  - Frequency weighting
  - Dimensionality reduction
  - Similarity measure

- The choice of the embedding determines how easy or difficult it is for the extrinsic / downstream task to extract the information that it needs from the embedding

List from https://en.wikipedia.org/wiki/Distributional_semantics

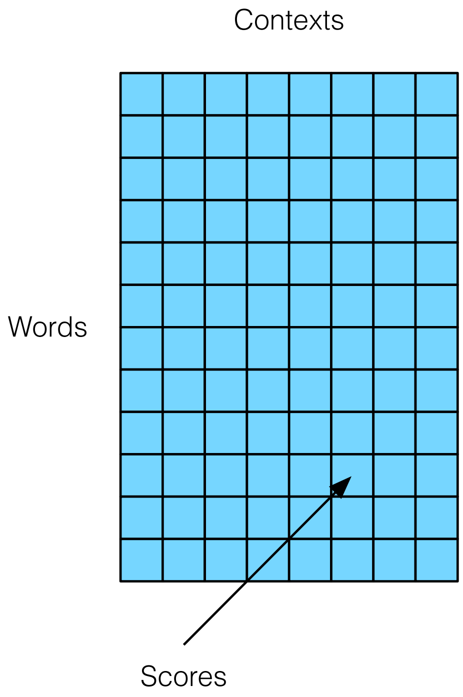
History

- For a nice history of word embeddings see
  - An overview of word embeddings and their connection to distributional semantic models
  - http://blog.aylien.com/overview-word-embeddings-history-word2vec-cbow-glove/

- Side note
  - I believe that this is the 1st machine learning algorithm we've looked at this semester that relies on unsupervised learning

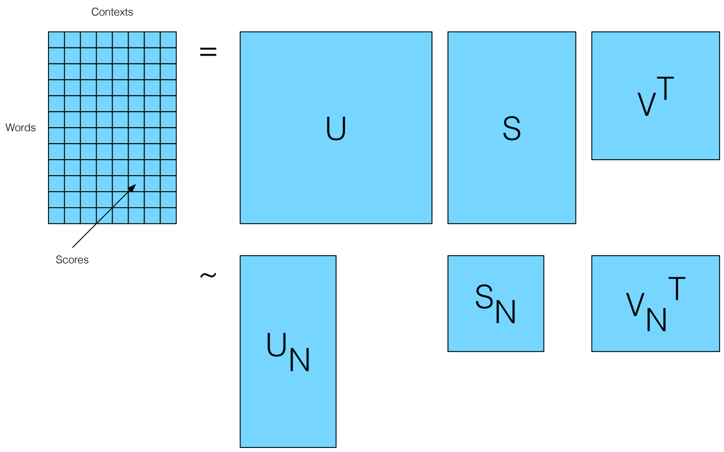
SVD Based Word Embedding

- Idea
  - Use the context of words around a word to describe a word
  - Ideally have a large text with billions of words to learn this from
- Parse the text into pairs D of size |D|
  - Words (W unique)
  - Contexts (C unique)
    - Ex: context can be the proceeding word (bigram)
    - Ex: context can be the whole document (latent semantic analysis)
- Form a matrix X
  - Unique words as rows
  - Unique contexts as cols
  - Entries are counts, probabilities, normalized probabilities, positive point wise mutual information (a normalized measure of the word – context pair likelihood); sometimes these are raised to a power like 3/4 to introduce some smoothing



SVD Based Word Embedding

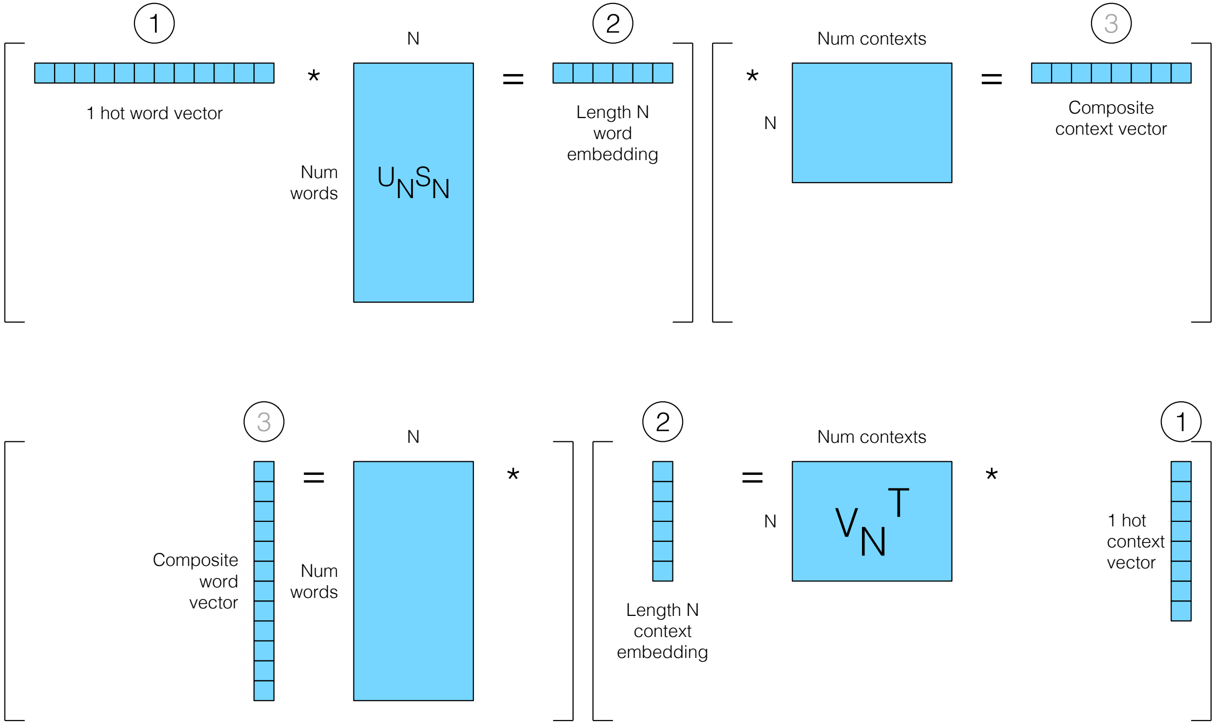
- Take the SVD of X and keep the cols of U, rows and cols of Σ and rows of V corresponding to the N largest singular values
  - $X = U \Sigma V^T \approx U_N \Sigma_N V_N^T$
- Embedding
  - Each row of $U_N \Sigma_N$ is a length N embedding for the corresponding word
  - Each col of V_N^T is a length N embedding for the corresponding context
- Comments
  - The linear map from 1 hot input word vector to length N word embedding can implement all mappings
    - Just make the row the desired mapping for each word
    - So the question really is what it the best linear mapping
  - The linear mapping is a result of 2 choices
    - The matrix representing words and context likelihoods
    - The choice of the factorization method applied to this matrix to create word mapping and context mapping matrices



Note the asymmetry of the word vs context embedding; sometimes this is modified such that Σ_N is distributed partially to both the word and context embedding as Σ_N^p and Σ_N^{1-p}

SVD Based Word Embedding

Step 3 is not part of the embedding, just interesting to think about



Word2Vec

- Word to vector mapping strategy
  - Assign a vector to a word that's a function of the words likely to be around it
  - Don't have to explicitly know the meaning of the word just "the company that it keeps”
  - Learn an embedding matrix and a bias (which could later be combined into the embedding matrix)
- 2 methods for creating the mapping
  - Continuous bag of words
  - Skip gram (more popular)

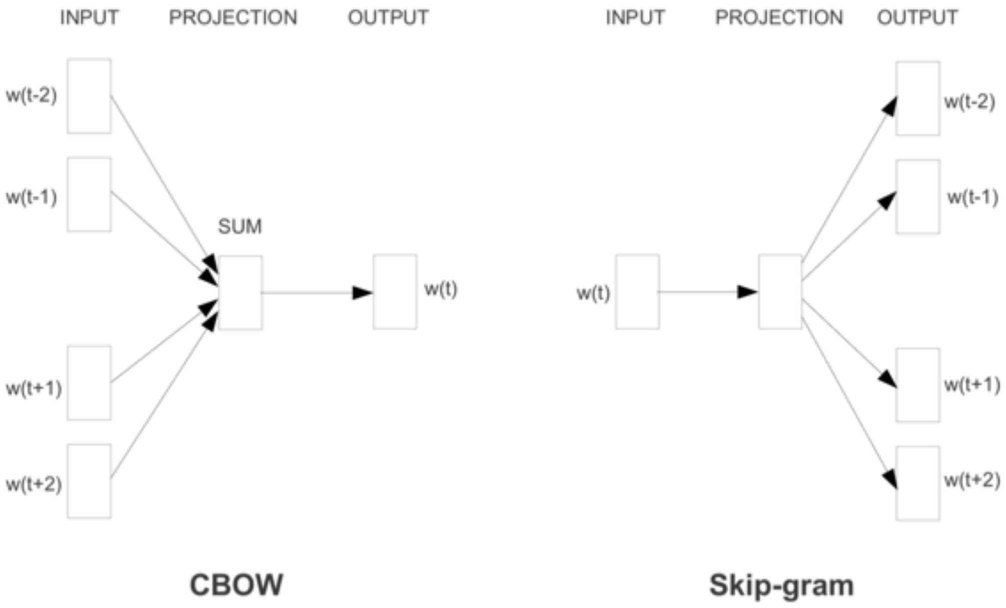


Figure from https://arxiv.org/abs/1301.3781

Word2Vec

- Continuous bag of words
  - Model predicts the current word from surrounding context of words
  - Ordering of context words is not considered
  - Slightly better with small data sets
  - Slightly worse with large data sets
  - Context window size of ~ 5

- Objective
  - Training minimizes the negative log likelihood of the current word given the context

$J_\Theta = -(1/T)\ \Sigma_t \log p(w_t \mid w_{t-n}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+n})$

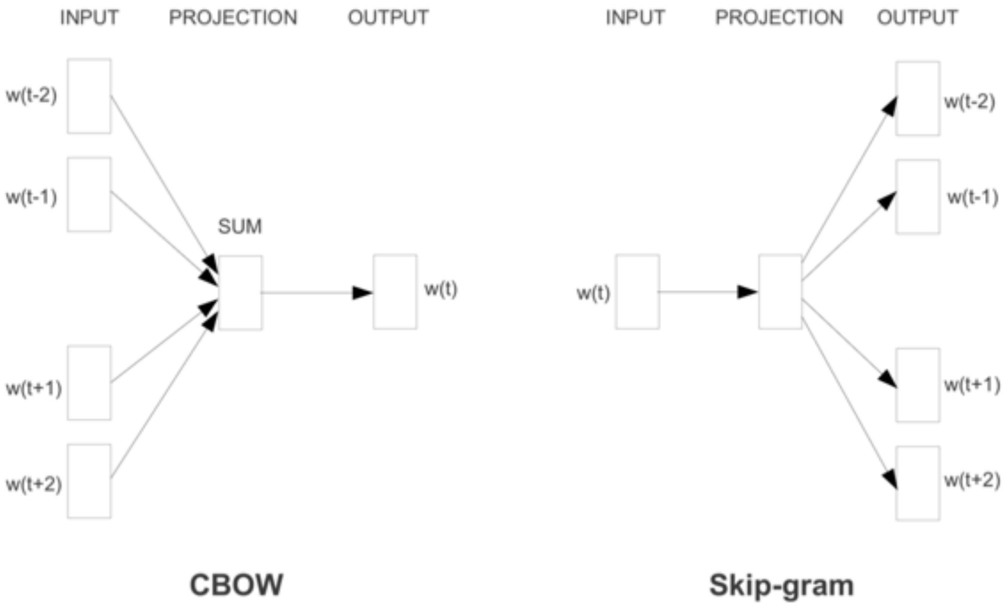


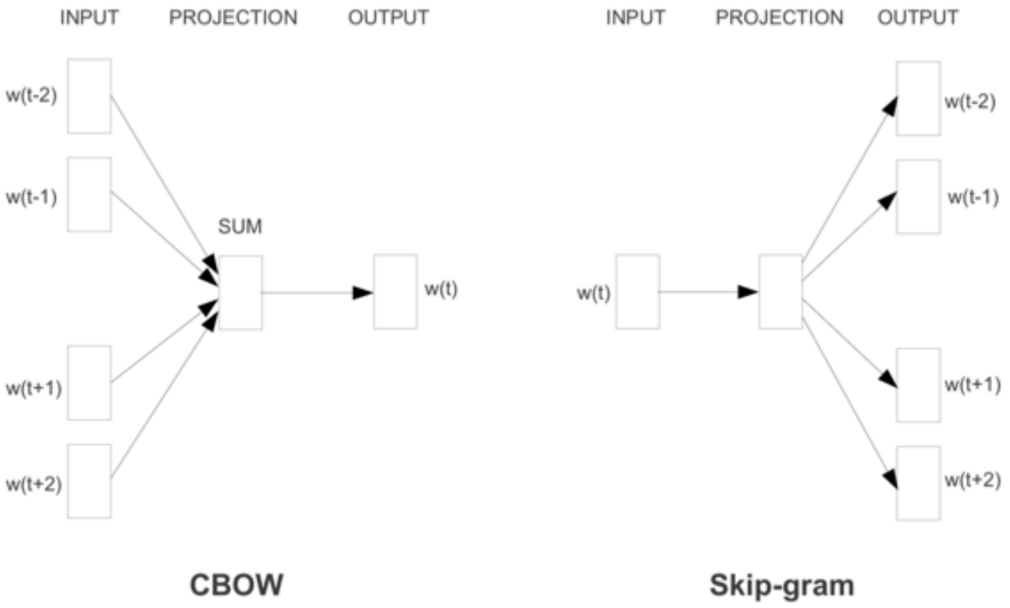
Figure from https://arxiv.org/abs/1301.3781

Word2Vec

- Skip gram
  - Model predicts surrounding context of words from the current word
  - Near words are weighted more heavily than far words based on context window range selection during training, does a better job with infrequent words
  - Slightly worse with small data sets
  - Slightly better with large data sets
  - Maximum context window size of ~ 10

- Objective
  - Training minimizes the negative log likelihood of the context given the current word

$$J_{\Theta} = -(1/T)\ \Sigma_t\ \Sigma_{-n\le j\le n, \ne 0} \log p(w_{t+j} \mid w_t)$$



Using 1 input word to predict multiple context words feels imbalanced from a mapping perspective (it is); however, this is only needed during training and it's handled by turning the 1 to many grouping into multiple 1 to 1 pairs / training samples (potentially each for use with more negative pairs depending on the specific training strategy selected)

Figure from https://arxiv.org/abs/1301.3781

Word2Vec

- Various methods for reducing training complexity are used
    - Hierarchical softmax
    - Negative sampling (more common)
- It turns out that the training method has a relatively large impact on performance
- Various methods for improving accuracy are used such as sub sampling high frequency words

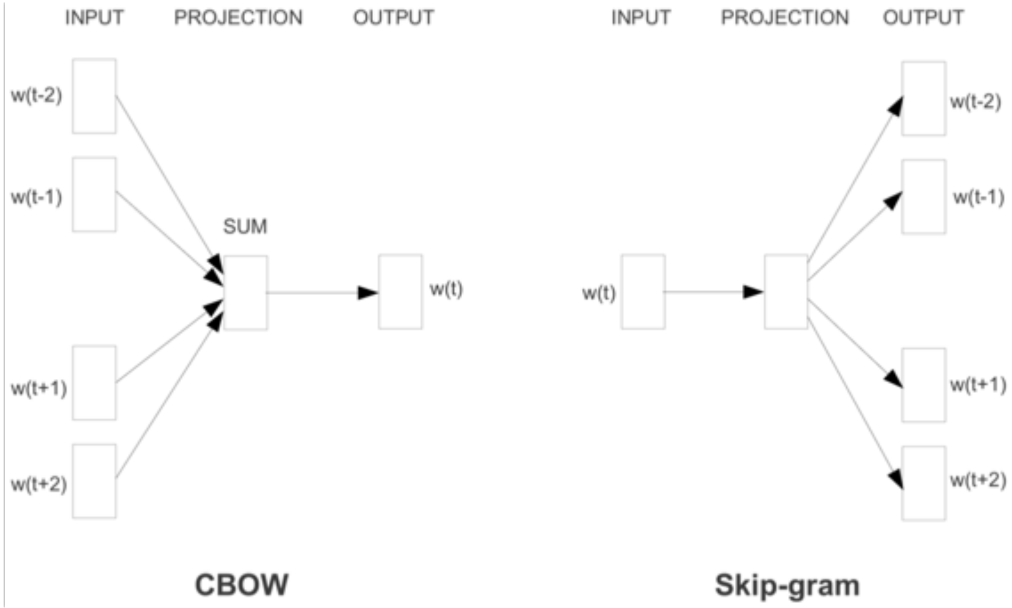


Figure from https://arxiv.org/abs/1301.3781

Word2Vec

Simple vector addition on the resulting word embeddings leads to many reasonable results (implying some basic language understanding)

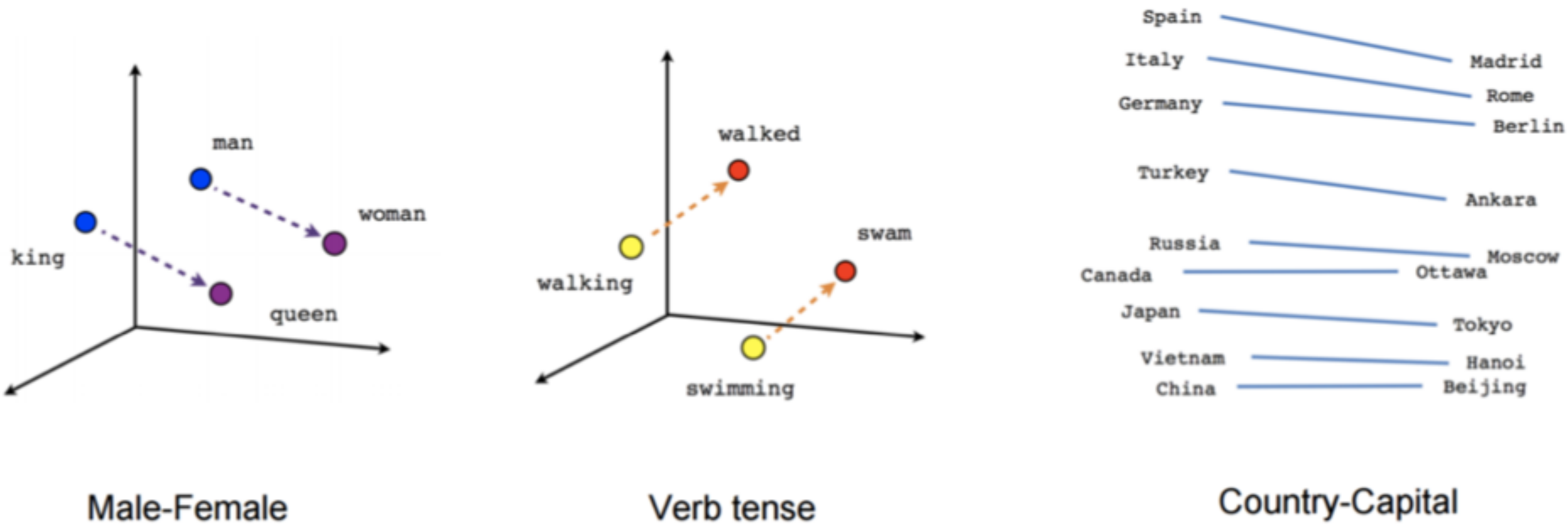


Figure from https://thegradient.pub/nlp-imagenet/

Glove

- Considering the 2 approaches we've seen so far
  - SVD based methods use full data statistics but tend to do poorly on word analogies
  - CBOW and skip gram methods do well on word analogies but poorly use full data statistics

- Global vectors for word embedding
  - Purpose is to combine statistical benefits of global matrix factorization methods with analogy benefits of context window based methods

- Objective
  - Let X be a word / word (context) count matrix
  - Let wi and bi be the word vector and bias of the ith word
  - Let wj and bj be the word vector and bias of the jth word
  - Let f() be a weighting function that assigns lower weights to both rare and frequent word occurrences

$$J_\Theta = \Sigma_{i,j} f(X_{ij}) (w_i^T w_j + b_i + b_j - \log(X_{ij}))^2$$

Visualization Via t-SNE

- Word embeddings can be vectors of ~ 1000 dimensions
    - So how were they plotted in 2 or 3 dimensions a few slides prior to this?
    - t–SNE is a common method for mapping high dimension vectors to 2 or 3 dimensions for plotting

- Compute probabilities for high dimension vectors that are ∝ to the similarities of pairs of objects
    - Let x_i be a vector in the high dimension input space

$$p_{j|i} = \exp(-||x_i - x_j||^2/(2\sigma_i^2)) / \Sigma_{k \neq i} \exp(-||x_i - x_k||^2/(2\sigma_i^2))$$

$$p_{ij} = (p_{j|i} + p_{i|j}) / (2N)$$

- Compute similarities for low dimension vectors
    - Let y_i be the vector in the low dimension output space

$$q_{ij} = \exp(1 + ||y_i - y_j||^2)^{-1} / \Sigma_{k \neq l} \exp(1 + ||y_i - y_l||^2)^{-1}$$

- Adapt y_i to minimize the KL divergence using gradient descent

$$KL(P||Q) = \Sigma_{i \neq j}\, p_{ij} \log (p_{ij} / q_{ij})$$

Evaluation

- All the embeddings discussed in this section start with a word and end up with a dense vector
  - So which is better?
- Some intrinsic metrics used for comparisons based on the original intent of capturing language meaning
  - Word similarity
    - Data is composed of pairs of words with similarity scores assigned by humans, goal is be to match score with something like cosine between vectors
    - Ex: SimLex999, MEN, WordSimilarity353 and RareWords
  - Word analogy
    - Data is composed of quadruples generated by humans such as king queen man woman, goal is be to determine 1 from the other 3
    - Ex: WordRep
  - Sentence
    - Data is composed of sentences with scores assigned by humans, goal is to match score
    - Ex: Stanford Sentiment Tree-bank and News20
  - Single word
    - Data is composed of single words with classes assigned by humans, goal is to match class
    - Ex tasks: POS tagging, sentiment, color, WordNet synset
- Word embedding benchmarks
  - https://github.com/kudkudak/word-embeddings-benchmarks

Evaluation criteria from https://arxiv.org/abs/1702.02170

Evaluation

- But really the main thing that matters is extrinsic to the embedding
  - Does the subsequent downstream language task perform better or worse with 1 mapping or another

- This highlights that the cost function optimized to find the mapping is not the final cost function that matters

Evaluation criteria from https://arxiv.org/abs/1702.02170

Task Specific Optimization

- The output of the previously discussed word embedding methods is a matrix that maps very large 1 hot vectors to a much smaller length ~ 100 – 1000 dense vectors

- This matrix is typically generated via an exceedingly large offline text and is well optimized for that text

- However, this mapping choice is not necessarily optimal for the subsequent downstream application
    - 1 of the keys to xNN success in many applications that we've looked at is end to end training

- Task specific optimization seeks to refine an existing mapping or optimize a new mapping from scratch by incorporating the mapping as a trainable component in the task specific network structure

Language Modeling

Goal

- Assign a probability to a sequence of words
  - $P(w_0, w_1, \ldots, w_{n-2}, w_{n-1}) = P(w_0)\ P(w_1 \mid w_0) \ldots P(w_{n-1} \mid w_{n-2}, \ldots, w_1, w_0)$
    - Remember the chain rule of conditional probability from the probability lecture
  - So can create from learning to predict the next word for all different length sequences $P(w_{n-1} \mid w_{n-2}, \ldots, w_1, w_0)$

- Next word prediction is a fundamental component of other language tasks
  - We'll see it in text to text translation
  - We previously saw it in speech to text transduction

Data

- Penn Treebank
- Billion word corpus
- WikiText

Count Based N Gram Models

- Predict the Nth word in a N word sequence from the previous N – 1 words
- Strategy: estimate probabilities from counting in text and building up recursively
  - $P(w_{n-1} \mid w_{n-2}, \ldots, w_1, w_0) = P(w_0, w_1, \ldots, w_{n-2}, w_{n-1}) / P(w_0, w_1, \ldots, w_{n-2})$
- Problem: what if a word doesn't occur in the training text
  - Solution: smoothing
  - Assign a small non 0 probability to all words
- Problem: what if the N – 1 sequence never occurs
  - Solution: back off
  - Use N – 2 sequence instead or more generally interpolate
  - $P(w_{n-1} \mid w_{n-2}, \ldots, w_1, w_0) = c_0 P(w_{n-1} \mid w_{n-2}, \ldots, w_1, w_0) + c_1 P(w_{n-1} \mid w_{n-2}, \ldots, w_1) + \ldots + c_{n-1} P(w_{n-1})$
  - Where $c_0 + c_1 + \ldots + c_{n-1} = 1$

Neural Language Models

Green arrows represent matrix vector multiplication + bias, dotted are optional, first is table lookup word embedding

- Why use something other than a N gram language model?
  - N gram models are very large, sometimes implementations prevent their use
  - xNNs good at predicting stuff, perhaps can find a more compact representation that also allows for larger values of N
- Can think of at least 2 types of neural language models
  - NN to embed N length history into a continuous space
  - RNN to embed the full history
  - CNNs to embed a window
- Basic idea for NN based method
  - Individually encode each of N – 1 words into vectors via an embedding layer
  - Concatenate the vectors and (potentially) have multiple intermediate layers
  - Classifier head to predict a pdf of the Nth word (this softmax complexity is large)

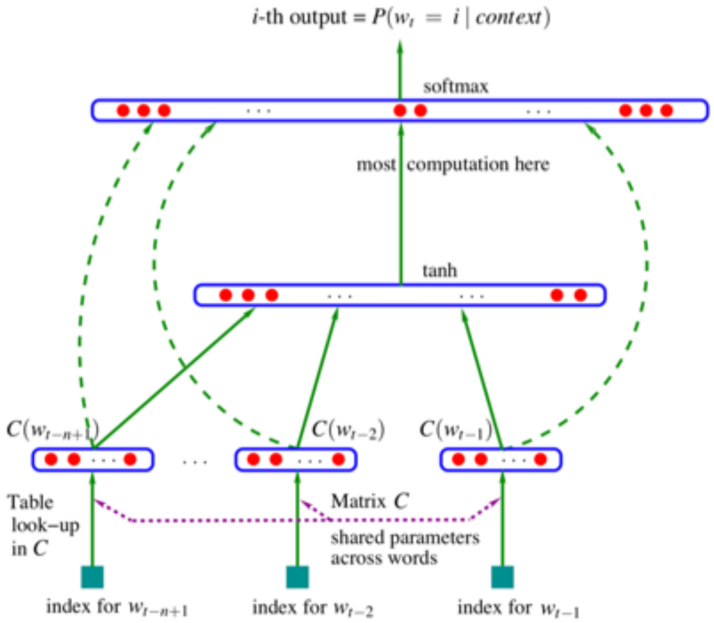


Figure from http://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf

NN Language Model Example

Example of standard neural network language model training predicting the next word from the previous 2 words

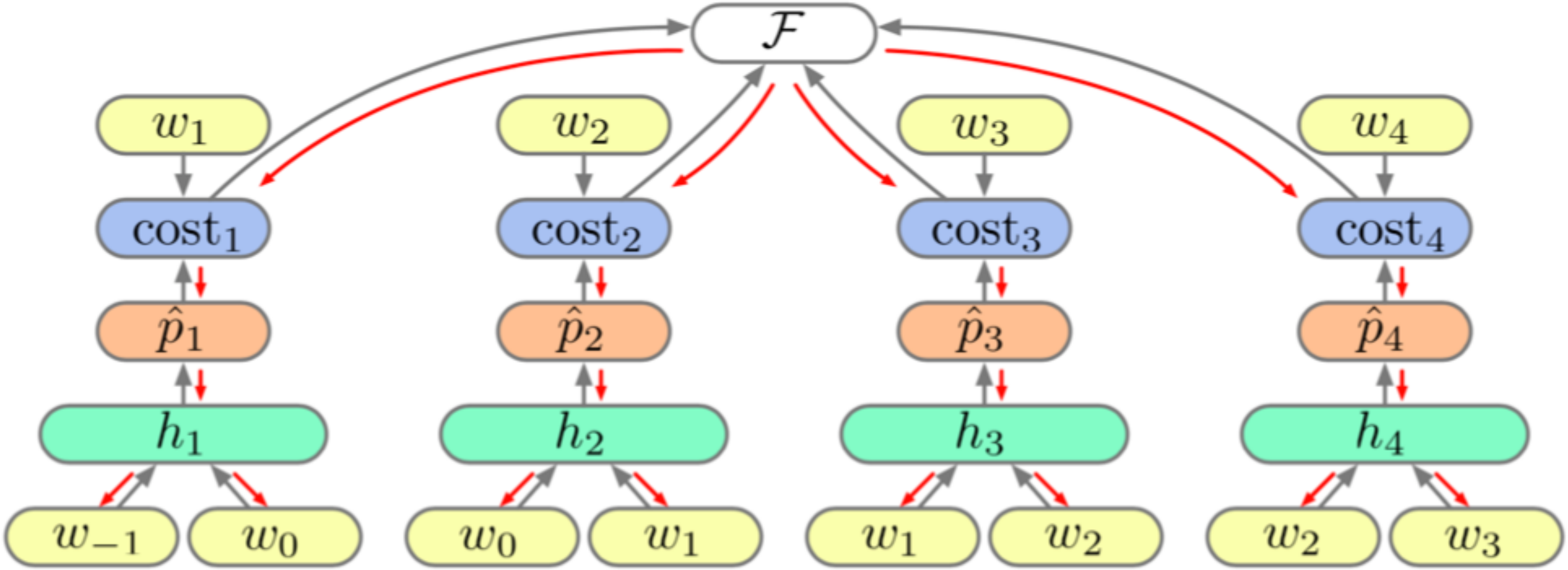


Figure from https://github.com/oxford-cs-deepnlp-2017/lectures/blob/master/Lecture%203%20-%20Language%20Modelling%20and%20RNNs%20Part%201.pdf

RNN Language Model Example

Example of recurrent neural network language model training predicting the next word from the previous history of words

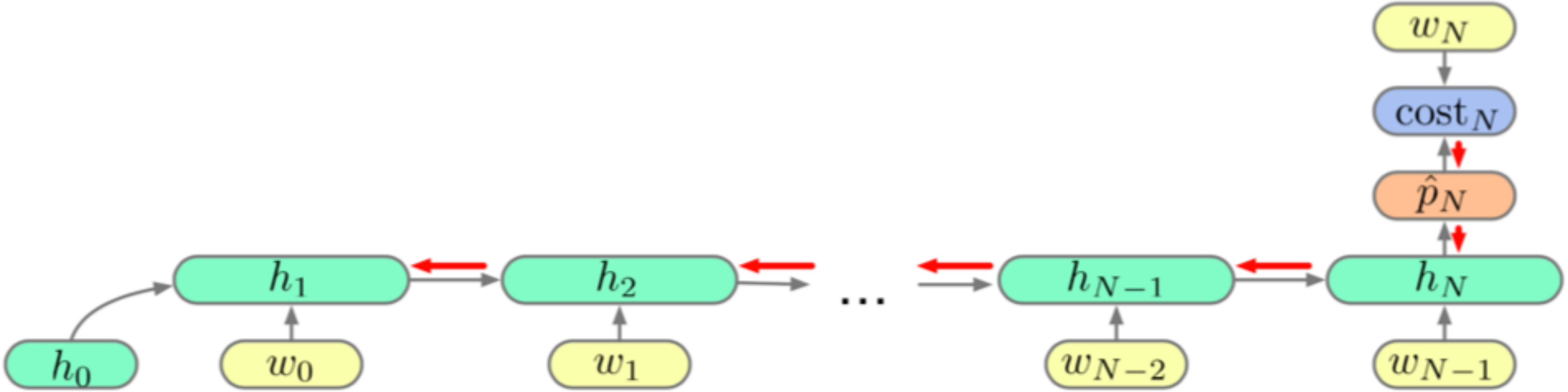


Figure from https://github.com/oxford-cs-deepnlp-2017/lectures/blob/master/Lecture%204%20-%20Language%20Modelling%20and%20RNNs%20Part%202.pdf

CNN Language Model Example

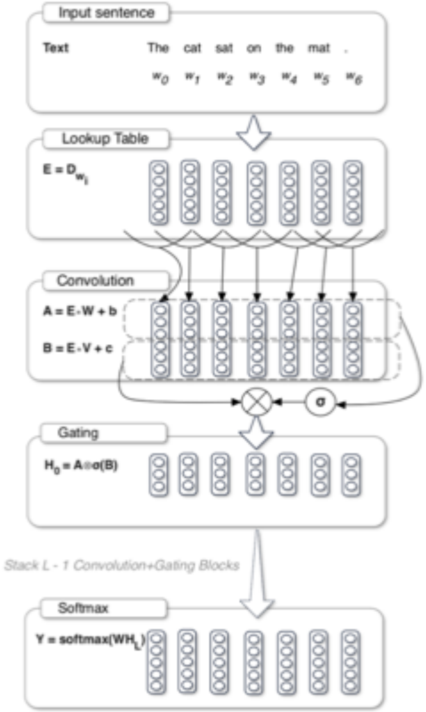


Figure from https://arxiv.org/pdf/1612.08083.pdf

Evaluation

- Want a measure of how well the language model predicts a sample
  - Remember from info theory that cross entropy $H(p, q) = -\Sigma_x\, p(x) \log_2(q(x))$ fell out of Kullback–Leibler divergence as a method of comparing pmfs
  - Let p(x) be the true distribution of words and q(x) be the distribution of words predicted by the language model
  - Unfortunately, the true distribution p(x) is unknown so cross entropy is estimated as $H(\text{training data}, q) = -\Sigma_{xi}\, (1/N) \log_2(q(x_i))$ where N is the size of the testing data

- Perplexity is defined as $2^{H(\text{training data, q})}$
  - Interpretation of how many different equally probable words can follow as the next word
  - A low perplexity is the goal (implying a low cross entropy, requires a good pmf match)

- However, just like for the case of word embeddings, the better evaluation of a language model is how well it helps with the subsequent downstream task
  - How much does it improve speech to text transduction accuracy?
  - How much does it improve text to text translation accuracy?

Character Based

- A challenge of language modelling is the size of the vocabulary (lots of words)
- Instead of building a word based language model, it’s possible to create a character (or other smaller element) based language model
  - Potentially a much much smaller model (number of characters is << number of words)
  - Able to capture sub word relationships
  - But perhaps not as good at word relationships as a word optimized model
  - Note: this was used in the RNN transformer for speech to text
- Just something to consider if you’re thinking about highly resource constrained applications

A Little Bit Of Fun

- You can train a language model using specific text to learn to predict words in the style of that text

- You can then use a language model to generate new text via feeding in it's previous outputs
  - For a N gram language model, use the Nth predicted word output as the most recent input to predict the N + 1 th word (similar strategy for a RNN language model)

- Things tend to go off the rails after a bit but it's interesting to think about

Translation

Goal

- Translate a sentence from 1 language to another
  - Initial strategy was a bilingual dictionary but that doesn't work very well

- Specifically: find best sentence y given sentence x (x in language 0, y in language 1)
  - $y^* = \arg\max_y P(y \mid x) = \arg\max_y P(x \mid y) P(y)$
    - $P(x \mid y)$ is the translation model learned from the parallel text
    - $P(y)$ is the language model learned from the target text only (how to write good target text)
    - Note that this is not exactly Bayes rule but may still be correct
  - The previous section considered language models
  - Translation can be viewed as a conditional language model
    - Predicts the next word given the previous words
    - Conditioned on the input sentence

Complications

- A laundry list
  - Input output sentence alignment
  - 1 to many mapping of input sentence to valid translation
  - How to evaluate the translation
  - Out of vocabulary words
  - Train set vs test set domain mismatch
  - Maintaining context over long text
  - Low resource language pairs

3 Encoder Decoder Model Types

- Sequence to sequence
  - Source words mapped to vectors
  - Encoded uses a RNN
  - Decoder uses a RNN initialized by the final hidden state of the encoder to recursively generate target word vectors
  - End of sentence output token allows learning variable length translations

- Recurrent neural machine translation with attention
  - Decoder uses attention over all the encoder hidden states
  - Combines with the hidden state of the decoder RNN and previous output word vector to predict the next output word vector

- Transformer neural machine translation with attention and self attention
  - Organized as a stack of encoders / decoders
  - Uses self attention to compute the encoder and decoder embeddings
  - Includes a positional encoding to track the order of the input and output sequences (since no recurrence)

Sequence 2 Sequence

During training the shifted by 1 target sentence is used as the decoder input, during testing the previous decoder output is used as the input; in the paper a 4 layer LSTM with length 1000 vectors was used for the encoder and decoder RNN and the input word order was reversed, 80k / 160k output / input words

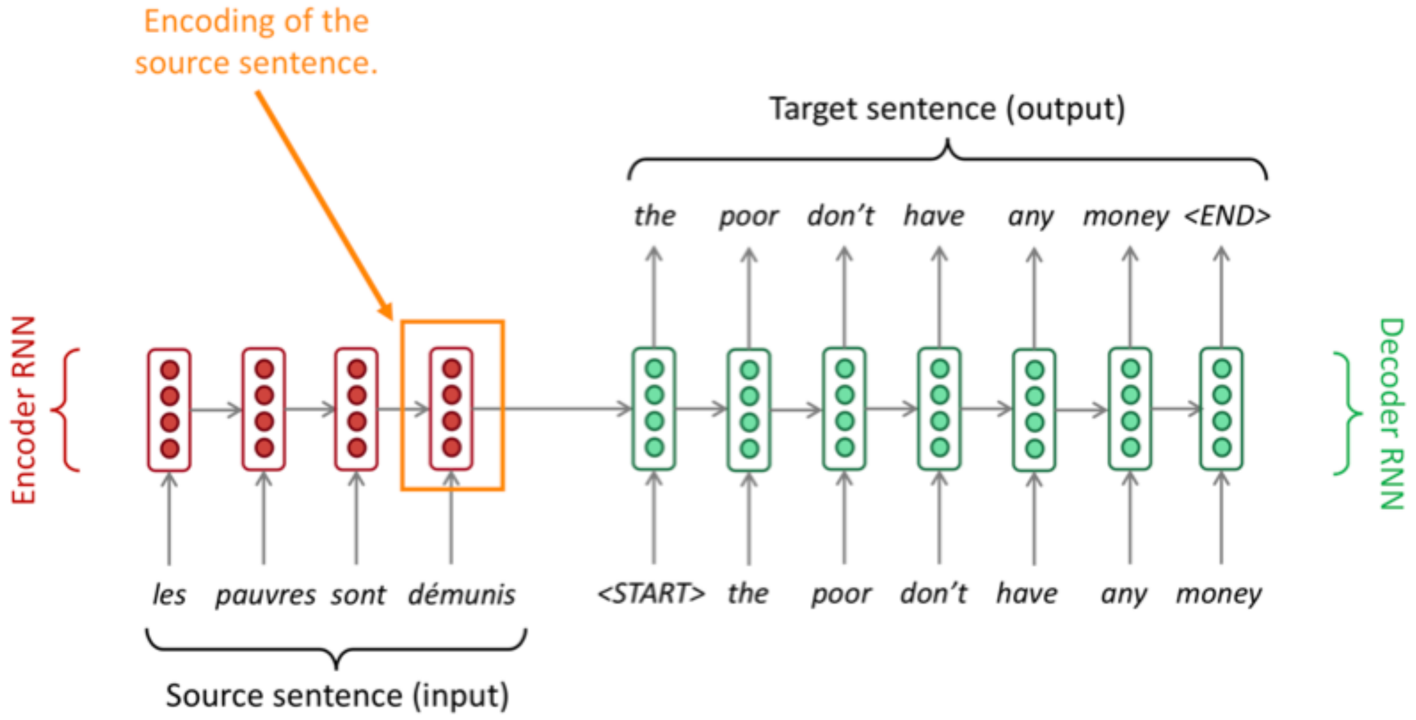


Figure from http://web.stanford.edu/class/cs224n/lectures/lecture10.pdf

Greedy And Beam Search Decoding

Beam search works well, don't use overly large beams

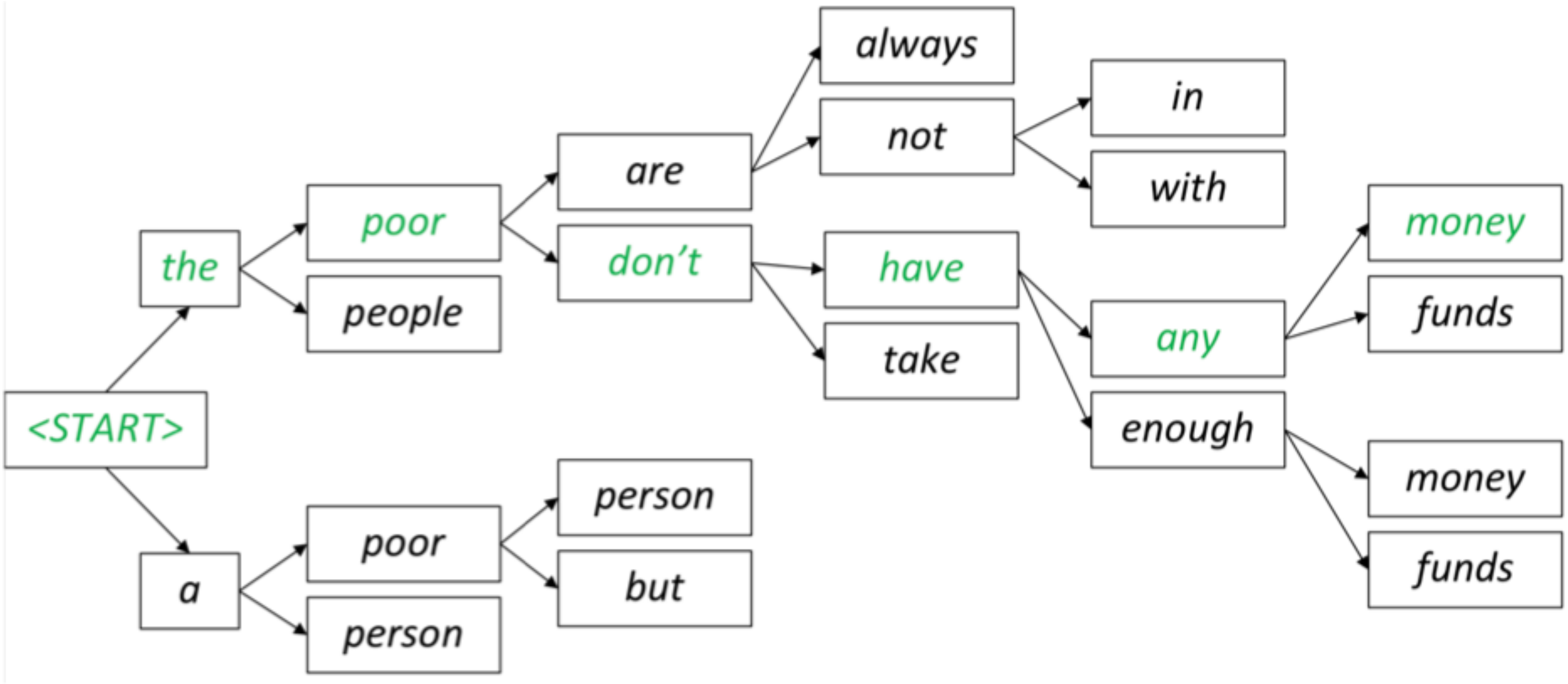


Figure from http://web.stanford.edu/class/cs224n/lectures/lecture10.pdf

Structured Prediction

- Note that test time operation is different than train time operation
  - During train time operation the ground truth is fed back to the decoder input
  - During test time operation the prediction is fed back to the decoder input
  - Want to optimize the sequence performance but the error and operation is optimized for token performance
  - Question: can optimize train time for full sequence vs token at a time
- This paper looks at doing that: Classical structured prediction losses for sequence to sequence learning
  - https://arxiv.org/abs/1711.04956
- Some suggestions from the paper
  - Do token level optimization first
  - Maybe use label smoothing
  - This becomes less important as baseline model improves

Attention

- A problem with seq2seq is that the whole 1st sentence is represented by 1 vector
  - This creates and information bottleneck
  - Becomes a bigger issue as the sentence length becomes longer

- Solution: use attention

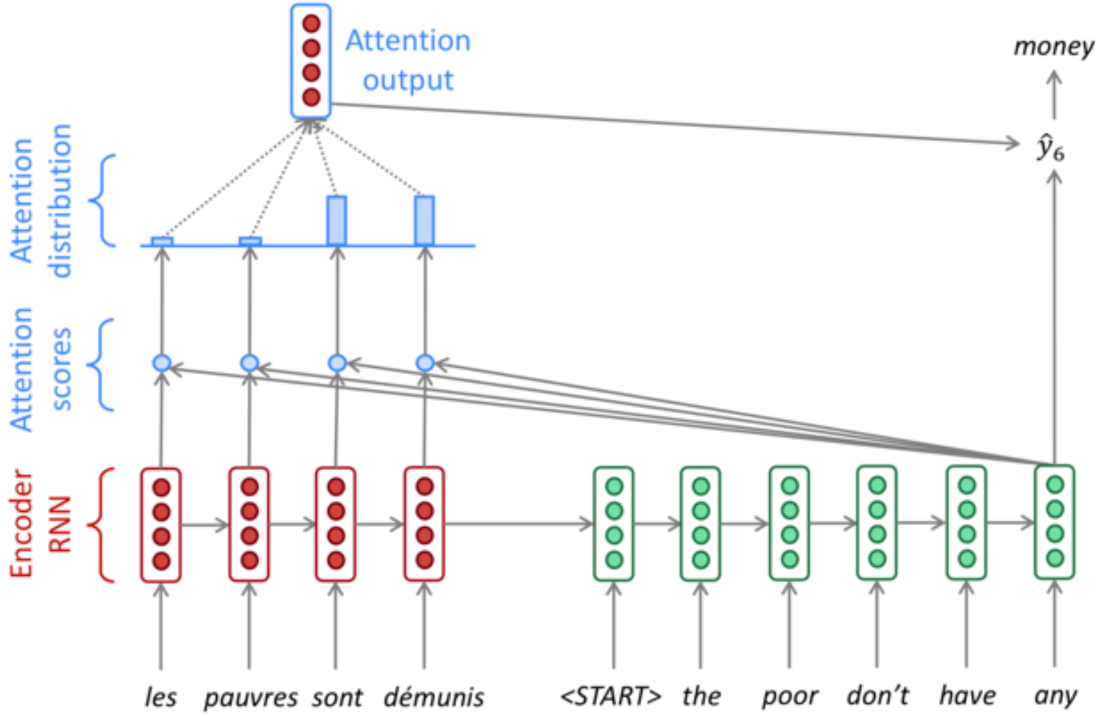


Figure from http://web.stanford.edu/class/cs224n/lectures/lecture10.pdf

Attention

- Attention allows the decoder to use different parts of the input at different time steps
  - Instead of the source sentence being stored as a vector it's now stored as a matrix with 1 vector per word
  - The decoder state for the current output is used with the encoded input vectors followed by soft max to determine a weighting
  - Encoded input vectors are weighted and summed to produce the attention output which is concatenated with the decoder state
  - This concatenated vector is then used to generate the output

- There are many variations on this theme

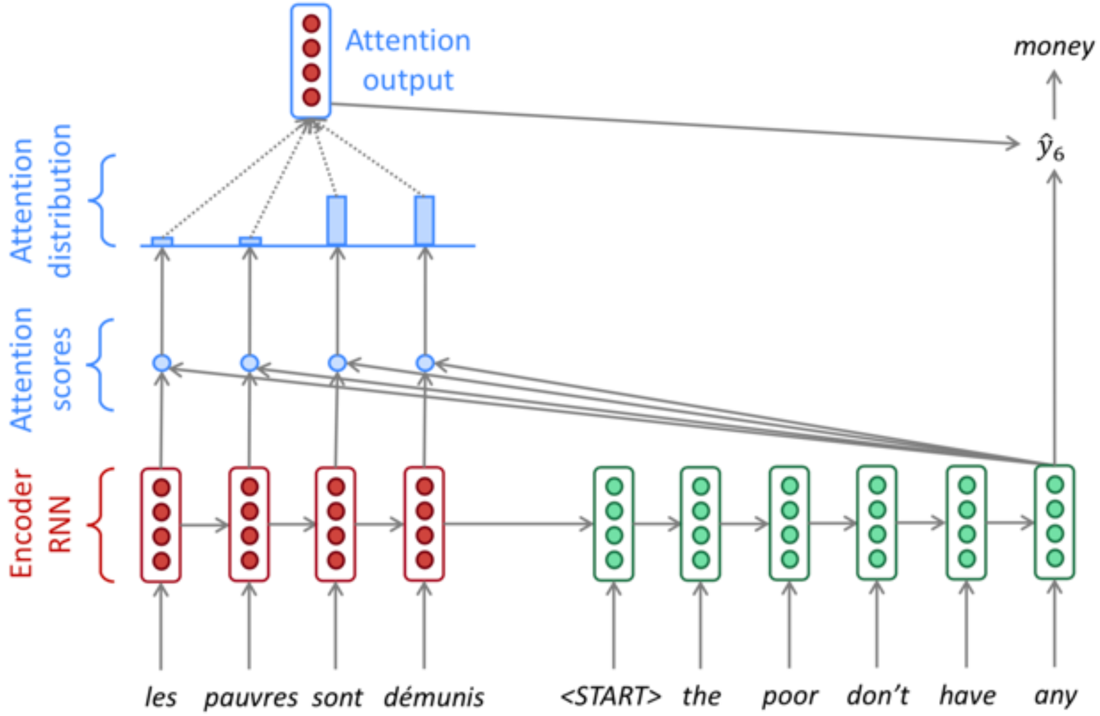


Figure from http://web.stanford.edu/class/cs224n/lectures/lecture10.pdf

Google's Neural Machine Translation System

Encoder attention decoder with a network architecture partitioned across multiple GPUs for performance

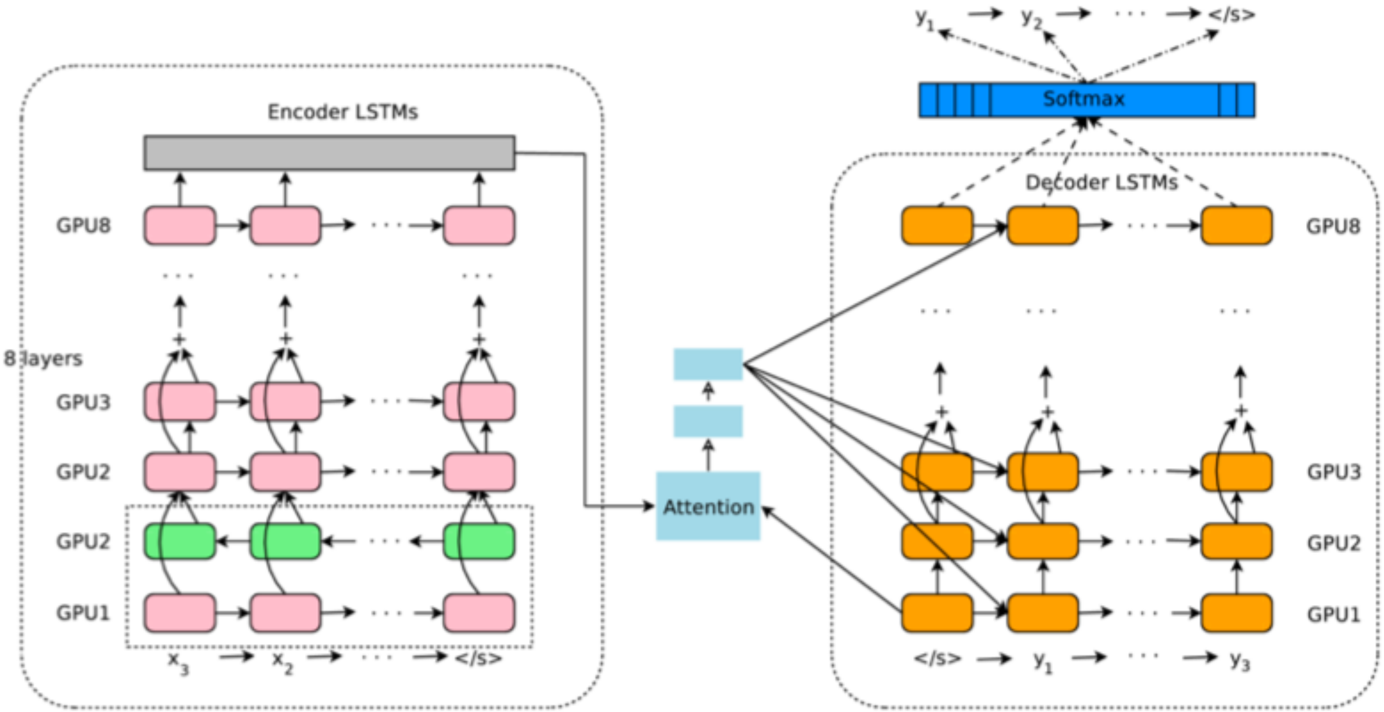
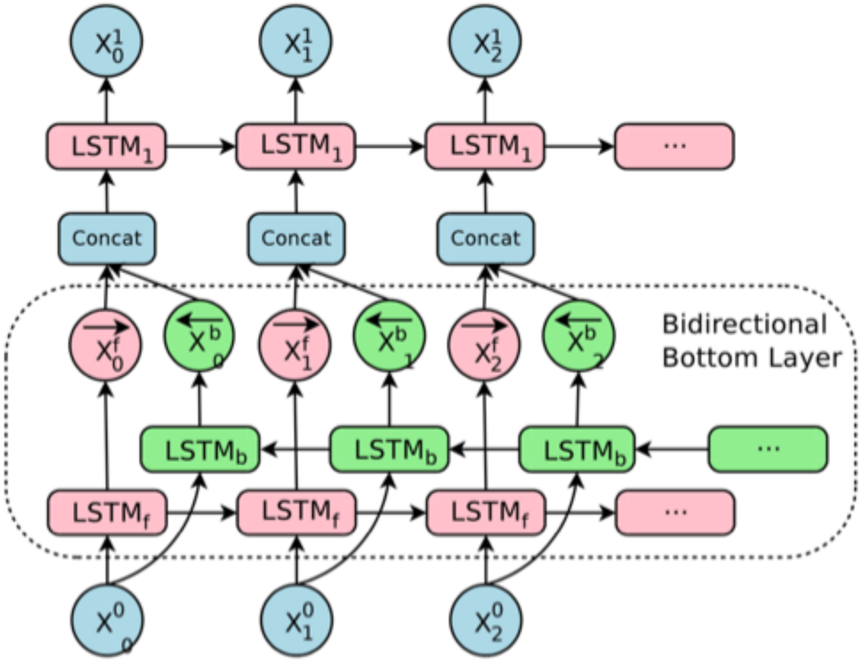


Figure from https://arxiv.org/abs/1609.08144

Google's Neural Machine Translation System

Bottom layer is a bi directional LSTM, subsequent layers are LSTM cells with residual connections



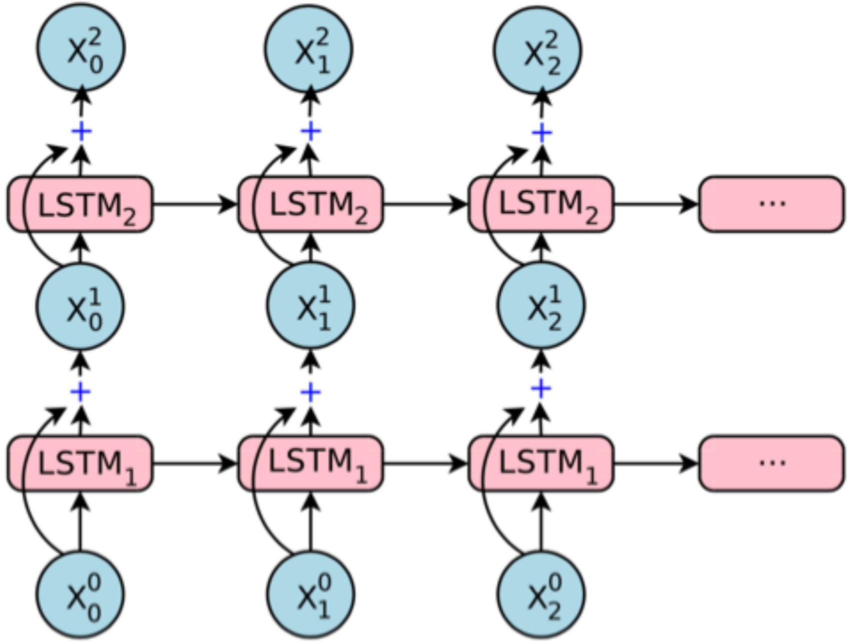


Figure from https://arxiv.org/abs/1609.08144

Architecture Exploration

- Source embedding
  - Performance was relatively agnostic with embedding lengths from 128 – 2048
- RNN cell architecture
  - LSTM was a little better than GRU
  - Standard RNN performed worse
- Encoder and decoder depth
  - Depth from 2 to 8 were explored with and without residual connections
  - Training was difficult with deeper models and needs to be re thought
  - Given current training capabilities, depths of 2 – 4 for the encoder and decoder worked best

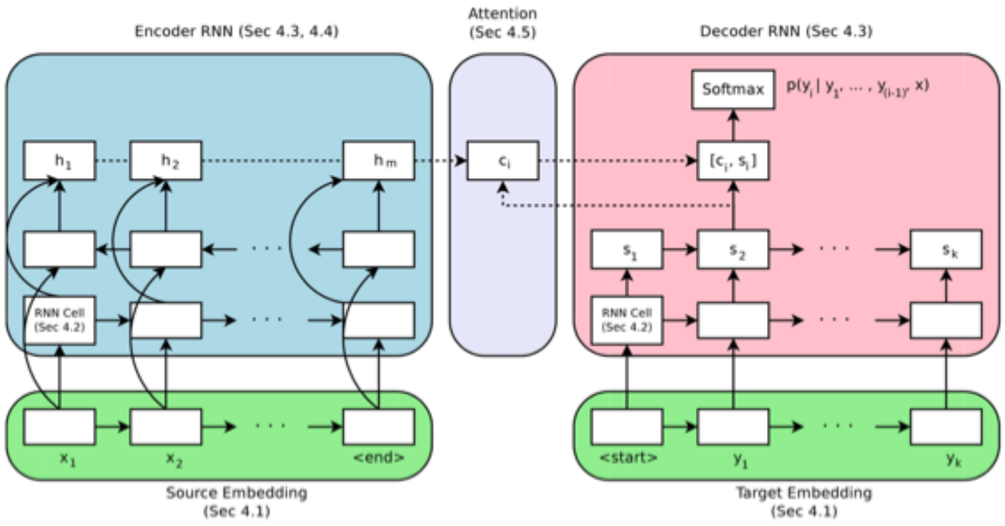


Figure from https://arxiv.org/abs/1703.03906

Architecture Exploration

- Unidirectional vs bi directional encoders
  - Bi directional performed slightly better
- Attention
  - Additive performed slightly better that multiplicative
  - The dimension size from 128 – 1024 didn't matter much
  - Training data indicated that attention played a larger role in the flow of the gradient than allowing the decoder access to the encoded states (as common belief suggests)
  - More thought is needed here
- Beam search
  - Beam width of ~ 10 with a length penalty performed best

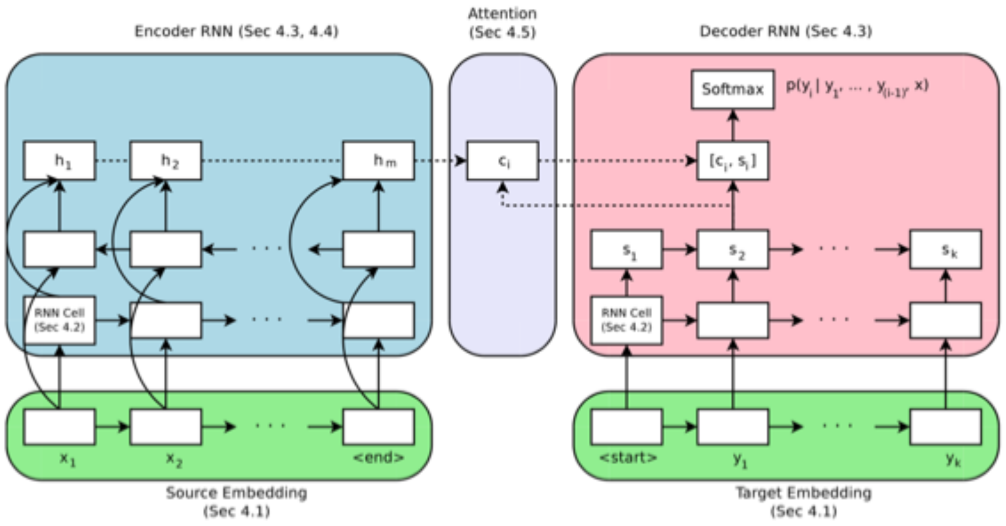


Figure from https://arxiv.org/abs/1703.03906

Self Attention In The Transformer

- Positional encoding
  - No sequential recurrent connections to enforce sequential structure in the output so a positional encoding is added
  - Sin() and cos() vectors with different frequencies are added to the inputs
- Encoder
  - 6 layers identical layers with 2 parts each
  - Part 1 is a multi head self attention mechanism
  - Part 2 is a fully connected layer
  - Each part includes normalization as y = LayerNorm(x + Part1or2(x))
- Decoder
  - 6 identical layers with 3 parts each
  - Part 1 is a multi head self attention mechanism identical to encoder part 1
  - Part 3 is a fully connected layer identical to encoder part 3
  - Part 2 is a multi head attention mechanism applied to the encoder output
  - Each part includes normalization as in the encoder

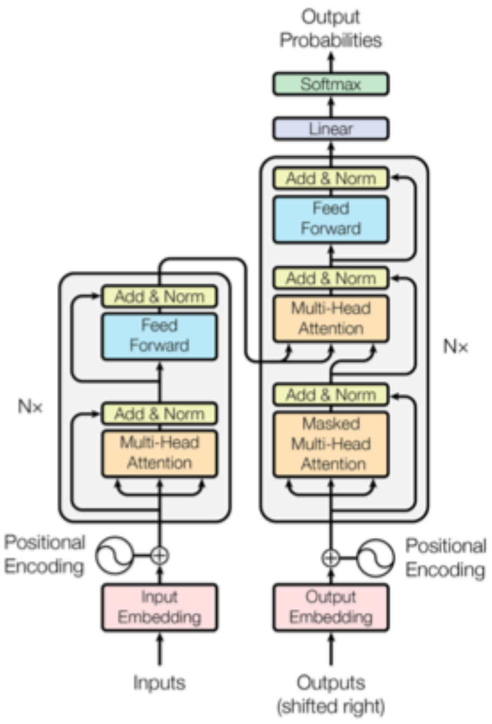


Figure from https://arxiv.org/abs/1706.03762

Self Attention In The Transformer

- Multi head attention equations
  - MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O
  - head_i = Attention(Q W_i^Q, K W_i^K, V W_i^V)
  - Attention(Q, K, V) = softmax(Q K^T / sqrt(d_k)) V

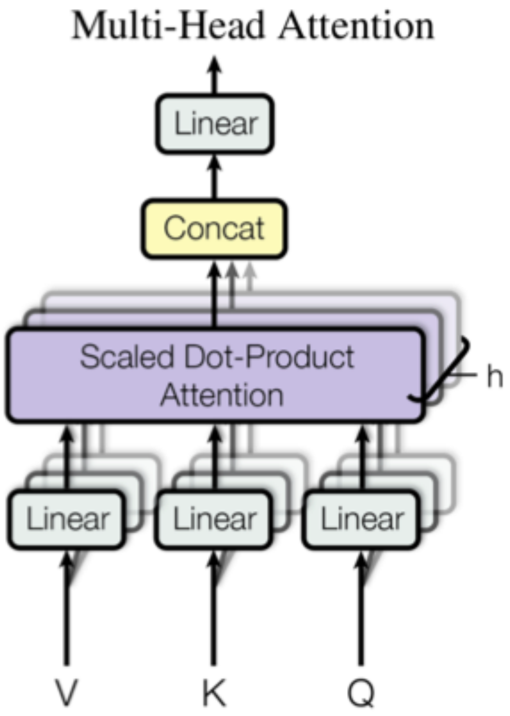


Figure from https://arxiv.org/abs/1706.03762

Convolutional Architectures

- A convolutional encoder model for neural machine translation
  - https://arxiv.org/abs/1611.02344

- Convolutional sequence to sequence learning
  - https://arxiv.org/abs/1705.03122

- Attentive convolution: equipping CNNs with RNN-style attention mechanisms
  - https://arxiv.org/abs/1710.00519

Evaluation

- Bilingual evaluation understudy (BLEU)

- Compares the output of a machine translation with the output of a human translation
    - The closer the match the better the translation
    - Uses N grams and a clipped precision metric with a geometric mean for combining them
    - Adds a penalty term to prevent translations that are too short
    - Output is between 0 and 1

Question

- Combining the material in the speech lecture with the material in the language lecture we have the following capabilities
    - Speech in language 1 to text in language 1
        - CTC, RNN transducer, attention, ...
    - Translation from text in language 1 to text in language 2
        - Sequence to sequence, attention, ...
    - Text language 2 to speech in language 2
        - Text to spectrogram to generative model, ...
- Applying these 3 networks sequentially allows us to translate from speech in language 1 to speech in language 2
- A question: why not design a network to directly map from speech in language 1 to speech in language 2?
    - A partial answer: remember back to the design lecture and the suggestion to keep your problems simple
    - However, it's interesting to think more about this

A quick survey for students that can speak more than 1 language

How do you translate from 1 language to another? Directly from speech 1 to speech 2? Or from speech 1 to a mental text translation to speech 2?

How do you converse with someone in the 2nd language you learned? Do you think and reply all in the 2nd language? Or do you make a round trip through the 1st language for thinking?

Is you deepest thinking language independent or language dependent?

Do the answers to these questions evolve as you become more familiar with the 2nd language?

References

Tutorials

- Stanford CS224n natural language processing with deep learning
  - http://web.stanford.edu/class/cs224n/
- Oxford deep NLP 2017 course
  - https://github.com/oxford-cs-deepnlp-2017/lectures
- An introduction to deep learning (lectures 2 and 3)
  - http://www.cs.toronto.edu/~ranzato/files/ranzato_deeplearn17_lec2_nlp.pdf
  - http://www.cs.toronto.edu/~ranzato/files/ranzato_deeplearn17_lec3_sequences.pdf
- Analyzing and tackling challenges in NMT
  - https://ranzato.github.io/publications/ranzato_harvard_1march18.pdf
- A primer on neural network models for natural language processing
  - https://u.cs.biu.ac.il/~yogo/nnlp.pdf
- Primer on neural network models for natural language processing
  - https://machinelearningmastery.com/primer-neural-network-models-natural-language-processing/

Tutorials

- Neural machine translation (seq2seq) tutorial
  - https://github.com/tensorflow/nmt
- Deep learning for natural language processing: tutorials with Jupyter notebooks
  - https://insights.untapt.com/deep-learning-for-natural-language-processing-tutorials-with-jupyter-notebooks-ad67f336ce3f

Data

- Conference on machine translation
    - http://statmt.org/wmt18/index.html
- Large movie review dataset
    - http://ai.stanford.edu/~amaas/data/sentiment/
- Tagged and cleaned Wikipedia (TC Wikipedia) and its ngram
    - https://nlp.cs.nyu.edu/wikipedia-data/
- The WikiText long term dependency language modeling dataset
    - https://www.salesforce.com/products/einstein/ai-research/the-wikitext-dependency-language-modeling-dataset/
- WIT3 web inventory of transcribed and translated talks
    - https://wit3.fbk.eu/mt.php?release=2016-01
- XNLI: the cross-lingual NLI corpus
    - https://github.com/facebookresearch/XNLI
- Yelp open dataset
    - https://www.yelp.com/dataset
- WordNet a lexical database for English
    - https://wordnet.princeton.edu

Word Embedding

- Word vectors and lexical semantics
  - https://github.com/oxford-cs-deepnlp-2017/lectures/blob/master/Lecture%202a-%20Word%20Level%20Semantics.pdf
- Distributed representations of words and phrases and their compositionality
  - https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf
- Efficient estimation of word representations in vector space
  - https://arxiv.org/abs/1301.3781
- Word2vec
  - https://code.google.com/archive/p/word2vec/
- Word2vec explained: deriving Mikolov et al.'s negative-sampling word-embedding method
  - https://arxiv.org/abs/1402.3722
- GloVe: global vectors for word representation
  - https://nlp.stanford.edu/pubs/glove.pdf
  - https://nlp.stanford.edu/projects/glove/

Word Embedding

- Embedding word similarity with neural machine translation
  - https://arxiv.org/abs/1412.6448
- Learning structured text representations
  - https://arxiv.org/abs/1705.09207
- Non-distributional word vector representations
  - http://www.aclweb.org/anthology/P15-2076
- Deep contextualized word representations
  - https://arxiv.org/abs/1802.05365
- Linguistic regularities in continuous space word representations
  - https://www.aclweb.org/anthology/N13-1090
- What are word embeddings for text?
  - https://machinelearningmastery.com/what-are-word-embeddings/
- Word embedding benchmarks
  - https://github.com/kudkudak/word-embeddings-benchmarks

Word Embedding

- An overview of word embeddings and their connection to distributional semantic models
  - http://blog.aylien.com/overview-word-embeddings-history-word2vec-cbow-glove/
- Vector representations of words
  - https://www.tensorflow.org/tutorials/representation/word2vec
  - https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/tutorials/word2vec/word2vec_basic.py
  - https://github.com/tensorflow/models/blob/master/tutorials/embedding/word2vec.py
- Stop Using word2vec
  - https://multithreaded.stitchfix.com/blog/2017/10/18/stop-using-word2vec/
- Lecture notes: part 1
  - https://cs224d.stanford.edu/lecture_notes/notes1.pdf
- Neural word embedding as implicit matrix factorization
  - https://papers.nips.cc/paper/5477-neural-word-embedding-as-implicit-matrix-factorization.pdf
- Inducing domain-specific sentiment lexicons from unlabeled corpora
  - https://arxiv.org/abs/1606.02820

Word Embedding

- Diachronic word embeddings reveal statistical laws of semantic change
  - https://arxiv.org/abs/1605.09096
- Linguistic regularities in continuous space word representations
  - https://www.aclweb.org/anthology/N13-1090
- How to evaluate word embeddings? On importance of data efficiency and simple supervised tasks
  - https://arxiv.org/abs/1702.02170

Visualization

- Visualizing data using t-SNE
  - http://jmlr.csail.mit.edu/papers/volume9/vandermaaten08a/vandermaaten08a.pdf
- How to use t-SNE effectively
  - https://distill.pub/2016/misread-tsne/

Language Modeling

- Recurrent neural networks and language modelling: part 1
  - https://github.com/oxford-cs-deepnlp-2017/lectures/blob/master/Lecture%203%20-%20Language%20Modelling%20and%20RNNs%20Part%201.pdf
- Recurrent neural networks and language modelling: part 2
  - https://github.com/oxford-cs-deepnlp-2017/lectures/blob/master/Lecture%204%20-%20Language%20Modelling%20and%20RNNs%20Part%202.pdf
- NLP's ImageNet moment has arrived
  - https://thegradient.pub/nlp-imagenet/
- Improving language understanding by generative pre-training
  - https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf
- Universal language model fine-tuning for text classification
  - https://arxiv.org/abs/1801.06146
- Language modeling at scale
  - https://arxiv.org/abs/1810.10045

Language Modeling

- A neural probabilistic language model
  - http://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf
- Neural network based language models for highly inflective languages
  - http://www.fit.vutbr.cz/research/groups/speech/publi/2009/mikolov_ic2009_nnlm_4.pdf
- An estimate of an upper bound for the entropy of English
  - http://www.cs.cmu.edu/~roni/11761/PreviousYearsHandouts/gauntlet.pdf

Translation

- Learning phrase representations using RNN encoder-decoder for statistical machine translation
    - https://arxiv.org/abs/1406.1078
- Neural machine translation by jointly learning to align and translate
    - https://arxiv.org/abs/1409.0473
- Sequence to sequence learning with neural networks
    - https://arxiv.org/abs/1409.3215
- Addressing the rare word problem in neural machine translation
    - https://arxiv.org/abs/1410.8206
- Effective approaches to attention-based neural machine translation
    - https://arxiv.org/abs/1508.04025
- Sequence level training with recurrent neural networks
    - https://arxiv.org/abs/1511.06732
- Achieving open vocabulary neural machine translation with hybrid word-character models
    - https://arxiv.org/abs/1604.00788

Translation

- Sequence-to-sequence learning as beam-search optimization
  - https://arxiv.org/abs/1606.02960
- An actor-critic algorithm for sequence prediction
  - https://arxiv.org/abs/1607.07086
- A convolutional encoder model for neural machine translation
  - https://arxiv.org/abs/1611.02344
- Language modeling with gated convolutional networks
  - https://arxiv.org/abs/1612.08083
- Massive exploration of neural machine translation architectures
  - https://arxiv.org/abs/1703.03906 and https://github.com/google/seq2seq
- Convolutional sequence to sequence learning
  - https://arxiv.org/abs/1705.03122
- Attention is all you need
  - https://arxiv.org/abs/1706.03762

Translation

- Attentive convolution: equipping CNNs with RNN-style attention mechanisms
  - https://arxiv.org/abs/1710.00519
- Classical structured prediction losses for sequence to sequence learning
  - https://arxiv.org/abs/1711.04956
- Universal neural machine translation for extremely low resource languages
  - https://arxiv.org/abs/1802.05368
- Analyzing uncertainty in neural machine translation
  - https://arxiv.org/abs/1803.00047
- Achieving human parity on automatic Chinese to English news translation
  - https://arxiv.org/abs/1803.05567
- A comparison of transformer and recurrent neural networks on multilingual neural machine translation
  - https://arxiv.org/abs/1806.06957
- Pervasive attention: 2d convolutional neural networks for sequence-to-sequence prediction
  - https://arxiv.org/abs/1808.03867
- XNLI: evaluating cross-lingual sentence representations
  - https://arxiv.org/abs/1809.05053

Translation

- Attention and augmented recurrent neural networks
  - https://distill.pub/2016/augmented-rnns/
- Introducing tf-seq2seq: an open source sequence-to-sequence framework in tensorflow
  - https://ai.googleblog.com/2017/04/introducing-tf-seq2seq-open-source.html
- How to configure an encoder-decoder model for neural machine translation
  - https://machinelearningmastery.com/configure-encoder-decoder-model-neural-machine-translation/
- Machine translation
  - https://www.microsoft.com/en-us/translator/business/machine-translation/#nnt
- Sequence to sequence models
  - http://www.cse.iitd.ernet.in/~mausam/courses/col772/spring2018/lectures/21-seq2seq.pdf
- Visualizing a neural machine translation model (mechanics of seq2seq models with attention)
  - https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Translation

- Seq2Seq tutorial
  - https://deeppavlov.ai/examples/tutorials/04_deeppavlov_chitchat.pdf
- A neural network for machine translation, at production scale
  - https://ai.googleblog.com/2016/09/a-neural-network-for-machine.html
- The illustrated transformer
  - https://jalammar.github.io/illustrated-transformer/
- The annotated transformer
  - http://nlp.seas.harvard.edu/2018/04/03/attention.html
- Learning phrase representations using RNN encoder–decoder for statistical machine translation
  - http://emnlp2014.org/papers/pdf/EMNLP2014179.pdf

Translation

- A general-purpose encoder-decoder framework for Tensorflow
  - https://github.com/google/seq2seq/
- Facebook AI Research sequence-to-sequence toolkit written in Python
  - https://github.com/pytorch/fairseq

Translation

- BLEU: a method for automatic evaluation of machine translation
  - https://www.aclweb.org/anthology/P02-1040.pdf