1 Introduction

Definitions

Deep Learning = neural networks, sub-field of machine learning

Natural Language Processing (NLP) = Analysis of language

History

- Early booming (1950s – early 1960s)
- Setback (mid 1960s late 1970s)
- Renewed enthusiasm (1980s)
- Out-of-fashion – again (1990s to mid 2000s)
- Since mid 2000: huge progress for "deep learning"

Today enough compute power and large datasets are available.

Perceptron

Simplest form of a neural network (1943)

$\hat{y} = \sigma(\boldsymbol{x} \cdot \boldsymbol{W} + b)$, $\boldsymbol{x}, \boldsymbol{w} \in \mathbb{R}^n$, σ = threshold

Optimization problem

$\min_{\boldsymbol{w}} \sum_{j=1}^{N} (\hat{y}_j - y_j)^2$

NLP Tasks Overview

Sentence or text classification

- Perceptron, MLP, CNN, RNN

Sequence tagging, word tagging

- RNN, LSTM, GRU

Seq2Seq, auto-regressive (generative) LM, name entity recognition

- Encoder-decoder, RNN, Transformer

2 Deep Learning Basics

Feed Forward Neural Network

Change of multiple linear layers with activation function.

$\text{NN}(\boldsymbol{x}; \theta)) = g(f(\boldsymbol{x} \cdot \boldsymbol{W}) \cdot \boldsymbol{V})$, $\theta = \{\boldsymbol{W}, \boldsymbol{V}\}$

Neural network loss function (empirical risk minimization)

$\arg\min_\theta \mathcal{L}_\theta = \arg\min_\theta \sum_{(\boldsymbol{x}, \boldsymbol{y})} (\text{NN} - \boldsymbol{y})$

Activation Functions

Sigmoid $\sigma(x) \frac{1}{1+\exp\{-x\}}$

Hyperbolic tangent $\tanh x = \frac{\exp\{2x\}-1}{\exp\{2x\}+1}$

Rectified linear unit $\text{ReLU}(x) = \max(0, x)$

Leaky ReLU $\text{LeakyReLU}(x) = \max(\alpha x, x)$

Softmax $\text{Softmax}(\boldsymbol{x})_i \frac{\exp\{x_i\}}{\sum_j \exp\{x_j\}}$

Definitions

Generalization Model performs strong on non-training data

Overfitting Remembering the training data (no generalization)

Underfitting Model is to simple to represent the data

Hyperparameters Parameters which are not learned directly

Core components Data, model, and learning (training)

Supervised learning Learning with labels

Cross-validation Different train and validation splits

Validation Metrics

TP = True Positive FP = False Positive

TN = True Negative FN = False Negative

Precision = $\frac{\text{TP}}{\text{TP+FP}}$

Recall = $\frac{\text{TP}}{\text{TP+FN}}$

F1 = $\frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$

sum up the individual terms (TP, TN, ...) sum up the individual terms (TP, TN, ...)

Macro F1 average over all classes

BLEU (machine translation) score based in n-gram matches between the candidate and reference sentences

ROUGE-L (summarizing) longest common sub-sequences between the candidate and references

Other metrics cosine distance, squared distance, correlation (continues), edit distance (sequences), accuracy (classification)

Loss Functions

Mean squared loss $\mathcal{L}_{\text{MSE}} = \mathbb{E}\left((\hat{\boldsymbol{x}} - \boldsymbol{x})^2\right)$

Absolute error loss $\mathcal{L}_{\text{L1}} = \mathbb{E}(|\hat{\boldsymbol{x}} - \boldsymbol{x}|)$

Cross entropy $\mathcal{L}_{\text{XE}} = \mathbb{E}\left(-\sum_i y_i \log(\hat{y}_i)\right)$

Other loss functions hinge-loss, multi-class hinge-loss, binary cross entropy, Barron loss and many more

Regularization

L_2 regularization L_2 on model weights

L_1 regularization L_1 on model weights

Dropout drop randomly entries of the feature maps

Batch normalization noise in statistics of features due to batch-wise approximation of statistics

Training as Optimization

DNN are trained with first-order optimization methods (**gradient decent**)

Dataset to large for memory → **mini-batch stochastic gradient decent**

Typically advanced gradient decent optimizers used with an **adaptive learning rate** (RMSProp or Adam)

Gradients are computed by performing **backpropagation** through the compute graph (applying the chain rule)

Language Models

Language modeling is the task of assigning a probability to a sentence in a language

$p(w_{1:n}) = p(w_1)\, p(w_2|w_1) \ldots p(w_n|w_{n-1})$

$p(w_{1:n}) = \prod_{i=1}^{n} p(w_1|w_{i-k:i-1})$

3 Text Representation

Terminologies

Letters smallest unit in a language

Tokens output of a tokenizer (interchangeable with word)

Lemma the dictionary entry of a word

Stem a shorter version of a word

Lexical resource information about a word and their relations

Unit of representation

- Characters
- Morphemes (sub-words)
- words
- sentences
- etc. (phrases, windows, documents)

Textual Features

Direct features

- Letters of a word
- Length of a word
- Capitalized letters
- Word is a hyphen

Bag-of-Words (BoW) the count of each word in a text

Term Frequency - Inverse Document Frequency (TF-IDF) kind of normalized frequency n-grams can be performed

Types of Features

- Numerical features
- Categorical features

Categorical features encoded as one-hot or dense embeddings

Word Embedding

General goal embed typically a token into a vector representation

Representation to be expressed semantics, syntax, morphology, grammar, word similarity

One-hot encoding vector all zero except for one entry (very sparse)

Dense encoding map a categorical feature to a continuous space (many different approaches available)

Representation Approaches (unsupervised)

General problem How to obtain good dense encodings

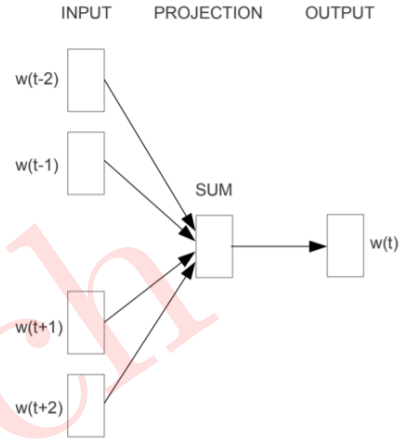
Static word embeddings can not capture polysemy and generate the same embedding for the same word in different contexts

Random initialization most basic representation but typically only used as an initialization

Word-context-matrix matrix with counts how often a word has occurred in a context (variable window size)

word2vec implements both CBoW and Skip-Gram

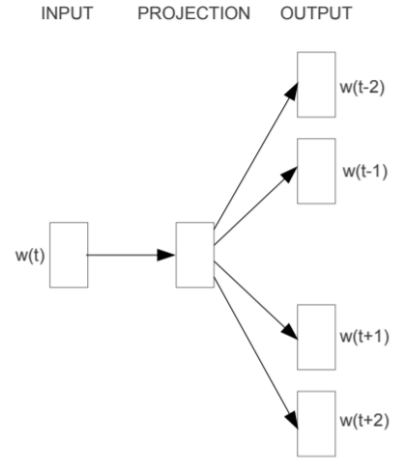
Continuous Bag of Words (CBoW) neural architecture which is given an context and tries to predict the missing target word



CBoW schematic.

Window size is a crucial hyperparameter of CBoW

Skip-Gram neural architecture which is given a word and tries to predict the context words



Skip-Gram schematic.

Window size is also a crucial hyperparameter of Skip-Gram

GLoVe advanced representation approach which aims at reconciling the advantages of corpus-wide co-occurrence counts and local context windows

Hierarchical softmax aims to make the softmax activation more efficent by using a binary classifier tree followed by a softmax on the leaves

Negative sampling don't compute softmax, use multiple binary classification with k negative samples instead.

General limitations of embedding approches

- Little control over the kind if similarity
- Trivial properties not captured
- Embeddings can be biased due to biased corpus

3.1 Evaluation of Word Representations

Extrinsic look at the performance of a model that uses the word representation

Possible extrinsic evaluation approaches: named entity recognition, MT (BLEU), summarization (ROUGE-L), information retrieval (coverage), compare performance of two models with different word representation

Intrinsic use the representation directly for evaluation

Possible extrinsic evaluation approches: word similarity, analogy, intrusion task, correlation with human judgments

4 Sense Embeddings

General idea train embeddings on sense-disambiguate corpora, different emebddings for each sense, corpora expensive, annotation needed

Parsimonious approach

1. Run word2vec on data
2. Cluster context representations
3. Assign each word's context to a cluster
4. Run word2vec on sense-disambiguate corpus

Bilingual Word Embeddings

General idea embed words of two languages into the same space

Motivation second language may act as an additional signal, direct transfer may feasible

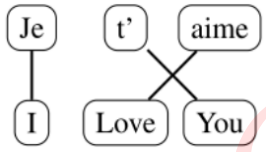
Cross-lingual objective words that are translations of each other should be close in the projected space

Mono-lingual objective words that occur in monolingually similar contexts should be close to each other in vector space

Naive approach given monolingual embeddings and a dictionary, find embedding pairs and combine both (concatenation, average)

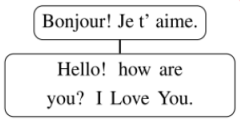
Mikolov (transformation matrix) encodes dictionary into emebddings by learning the matrix $\boldsymbol{W}$ by the objective $\arg\min_{\boldsymbol{W}} \sum_i ||\boldsymbol{x}_i \boldsymbol{W} - \boldsymbol{z}_i||_2^2$, can handle out of dictionary words

BiSkip utilize sentence and word aligned texts and run Skip-Gram with contexts of both languages, need for aligned text



BiSkip example.

BiVCD merge aligned text and shuffle all words, run monolingual approach.



BiVCD example.

Syntactic Word Embeddings

Motivation syntactic relation of words should also be represented in embeddings, word order matters

Solution add positional information to common methods, such as Skip-Gram, highly dependent on the window size (long range dependencies)

5 Convolutional Neural Networks

General properties for NLP

- Can handle variable size inputs (in theory, reality padding is used)
- Relevance of words id dependent of their position in input
- Parameter sharing, less parameters
- Sparse connectivity
- Multiple filters in each layer

1D Convolution

A 1D kernel (vector) is used to convolve over the input vector

$s[i] = (f * g)[i] = \sum_{m=-M}^{M} f[i-m] \cdot g[m]$

$s[i]$ output, f input, $*$ convolution operator, i current position in input, M window size, g kernel (filter)

Often used in NLP since text is most commonly represented as a 1D vector

2D Convolution

Input and kernel is now 2D. Kernel is convolved over both dimensions

Pooling

Pooling is used for downsampling the intermediate activation's, should impose positional invariance.

Max-pooling applies a max filter with a given stride and kernel size (commonly used)

Average-pooling applies a mean filter with a given stride and kernel size

Other pooling layers minimum, detail preserving pooling

What do CNNs learn

This is an active area of research

Experiments have shown that CNNs fail to fully incorporate sequential information

CNNs transform the embeddings into a different manifold to perform tasks such as classification

6 Recurrent Neural Networks

General properties of RNNs

- Can handle sequences of variable length
- Used mostly for sequence prediction
- Hidden state is known also as memory
- RNNs (non-bidirectional) can capture left-to-right inputs

Simple Recurrent Neural Network

$\boldsymbol{h}_t = \sigma_h\left(\boldsymbol{W}_h \boldsymbol{x}_t + \boldsymbol{U}_h \boldsymbol{h}_{t-1} + \boldsymbol{b}_h\right)$

$\boldsymbol{y}_t = \sigma_y\left(\boldsymbol{W}_y \boldsymbol{h}_t + \boldsymbol{b}_y\right)$

$\boldsymbol{x}_t$ input vector, $\boldsymbol{h}_t$ hidden state, $\boldsymbol{y}_t$ output vector, $\boldsymbol{W}$, $\boldsymbol{U}$, $\boldsymbol{b}$ parameters, σ_h, σ_y activation functions.

Problems of RNNs

Backpropagation through time poses often problems with the gradient flow

Vanishing gradient problem describes the problem of gradients becoming very small, thus RNNs are biased to more recent inputs and training is unstable

Vanishing gradients are caused by backpropagating through activation's in the range between -1 and 1, due to the properties of the chain rule

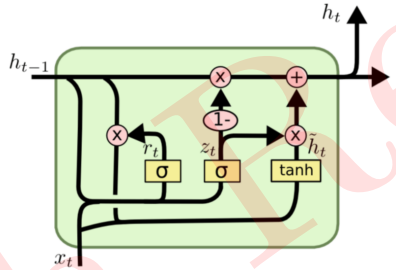
Solution using a ReLU activation over a tanh or sigmoid activation, good weight initialization may also beneficial

Exploding gradients describes the problem of very large gradeints, which may lead to unstable training or even NaNs (not common in simple RNNs)

Solution using gradeint clipping is the most common approach, normalization layers and good weight initialization can be also beneficial

Gated Recurrent Unit

Idea enable the hidden state to capture ling range dependencies by optimizing the gradient flow



GRU schematic.

Reset gate $r_t = \sigma\left(\boldsymbol{U}_r \boldsymbol{x}_t + \boldsymbol{h}_{t-1} \boldsymbol{W}_r\right)$

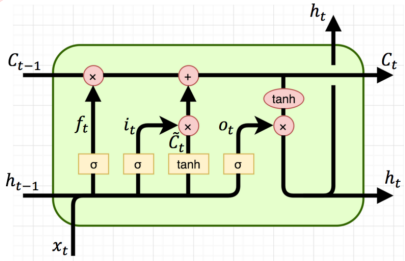
Possible hid. state $\hat{\boldsymbol{h}}_t = \tanh\left(\boldsymbol{U}_h \boldsymbol{x}_t + \boldsymbol{W}_h \boldsymbol{h}_{t-1} \odot r_t\right)$

Update gate $z_t = \sigma\left(\boldsymbol{U}_z \boldsymbol{x}_t + \boldsymbol{W}_z \boldsymbol{h}_{t-1}\right)$

Final hid. state $\boldsymbol{h}_t = (1 - z_t) \odot \boldsymbol{h}_{t-1} + z_t \hat{\boldsymbol{h}}_t$

Long Short-Term Memory

More popular than GRUs, LSTMs contain more parameters than GRUs for similar settings



LSTM schematic.

Input gate (write gate) $i_t = \sigma\left(\boldsymbol{U}_i \boldsymbol{x}_t + \boldsymbol{W}_i \boldsymbol{h}_{t.1}\right)$

Forget gate (reset gate) $f_t = \sigma\left(\boldsymbol{U}_f \boldsymbol{x}_t + \boldsymbol{W}_f \boldsymbol{h}_{t-1}\right)$

Output gate (read gate) $o_t = \sigma\left(\boldsymbol{U}_o \boldsymbol{x}_t + \boldsymbol{W}_o \boldsymbol{h}_{t-1}\right)$

New memory cell $\hat{\boldsymbol{c}}_t = \tanh\left(\boldsymbol{U}_{\hat{c}} \boldsymbol{x}_t + \boldsymbol{W}_{\hat{c}} \boldsymbol{h}_{t-1}\right)$

Final memory cell $\boldsymbol{c}_t = f_t \odot \boldsymbol{c}_{t-1} + i_t \odot \hat{\boldsymbol{c}}_t$

Final hidden state $\boldsymbol{h}_t = o_t \odot \tanh\left(\boldsymbol{c}_t\right)$

Bidirectional RNNs

Two RNNs with different parameters used to process the sequence from left-to-right as well as from right-to-left

Final output of each step is the typically the concatenation of both outputs

7 Encoder-Decoder Models

Encoder-Decoder models consist as the name suggests of two network paths, an encoder and a decoder

Encoder

A neural model to transform a text into a vector in an embedding space

An encoder can be made of the following components

- Pre-trained word embeddings
- Standard deep learning components (MLPs, CNNs, RNNs, Transformers)

Decoder

A neural model to transform a vector from an embedding space to tokens

Different langauge models can be sued as decoders

- RNN-based LMs
- Transformer-based LMs

RNN-based Encoder-Decoder

General function input sequence is encoded into a latent representation, the decoder predicts on the basis of the latent vector (context vector) multiple output tokens

Different schemes can be utilized to incorporate the latent vector into the RNN-encoder

Issues RNN-based encoder-decoder models may not capture long range dependencies accurately

Auto Encoder

Goal learn the identity mapping $f(\boldsymbol{x}) = \boldsymbol{x}$ and thus learn a rich latent vector (context vector) representation

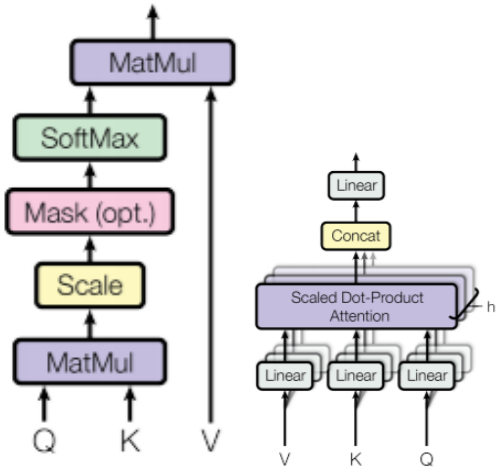
Used for dimensionality reduction or unsupervised representation learning

Encoder or decoder used after training for different tasks

8 Attention

General idea use attention to replace RNNs for NLP tasks

Attention mapping $\text{Att}\left(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}\right) = \text{softmax}\left(\frac{\boldsymbol{q}\boldsymbol{k}^T}{\sqrt{d_k}}\right)\boldsymbol{v}$



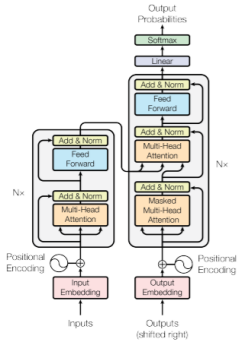
Attention (multi-head right) schematic.

Learn relations (self-attention) between all words

A attention layer can be seen as a dynamic fully connected layer
Other variants beyond standard (multi-head) attention (scale dot-product attention) are available, such as content-based attention, adaptive attention, or location-based attention
Transformer encoder-decoder models overcome the issue with the bottleneck path in RNN-based encoder-decoder models
Limitation classical attention has a memory and computational complexity of $\mathcal{O}\left(n^2\right)$ whereby n is the sequence length, thus classical attention is limited to moderate sequence lengths
Linear attention recently different approaches have been proposed to reduce the complexity of the attention mechanism to a linear complexity

Transformer

Overview a transformer is a attention-based architecture for various learning tasks, transformers also use skip-connections, layer normalization, and MLPs

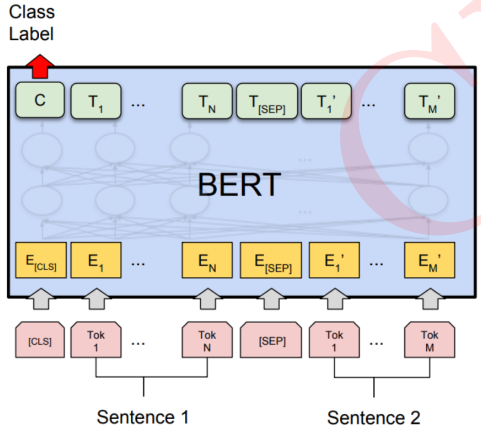


Transformer encoder-decoder.

Recently advanced transformer like architectures has been proposed, such as Linformer, FNet (Fourier trans. instead of attention), MLP mixer (no-attention), etc.

9 BERT

Overview the BERT model consists of bidirectional transformers which are pre-trained in an unsupervised setting



BERT model overview.

Architecture

BERT consists of bidirectional transformers combining the idea of bidirectional LSTMs and left-to-right transformers
The whole BERT model consists of $\sim$ 340M parameters
BERT learns its own token (WordPiece) embeddings
Max WordPiece token length of 512 used

Unsupervised Pre-Training

Overview BERT is trained in a multi-task fashion
Task 1 predict the randomly masked WordPiece unit (multi-class classification 30k classes)
Task 2 classify if the first sentence appears before the second sentence (cls token used, binary classification problem)
Very large dataset (corpus) and a lot of compute used for pre-training

Fine-Tuning

After pre-training BERT can be fine-tuned on a small labeled target dataset
Using pre-trained BERT lead to state-of-the-art results in many different classical NLP tasks (before GPT-2/3)

10 Generative Pretrained Transformer

Overview train (unsupervised) a huge transformer decoder on very large corpus for autoregressive language prediction
GPT models $p\left(x\right) = \prod_{i=1}^{n} p(s_n | s_1, \dots s_{n-1})$
The next word is sampled from the top-k predictions

GPT-2

Architecture GPT-2 consists of a transformer decoder with $\sim$ 1.5G (billion) parameters, left-to-right transformer used
Dataset from 8M web pages resulting in 40GB of text
Applications

- Text generation
- Conversation
- Machine translation
- Text summarization

Different applications are performed by querying the model, TL;DR for summarization
GPT-2 achieved state-of-the-art results in various NLP tasks without any supervision

GPT-3

Overview GPT-3 trains similar to GPT-2, however, a even larger model and a larger dataset is used, additionally GPT-3 is shown to be a few-shot learner
Architecture GPT-3 consists of the same transformer decoder as GPT-2, however, scaled to 175G (billion) parameters
Dataset consists of 570GB of filtered text
Applications similar to GPT-2
Loss decreases logarithmic with the model size (and compute)

11 Representation Learning

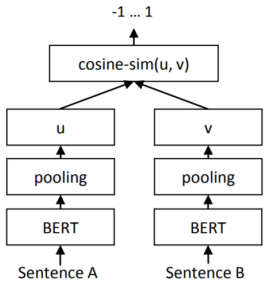
Goal learn a dense representation for a given input ((sub-)words, sentences, images, videos)
Use cases strong representations can be utilized for few or zero-shot (image) classification, search, or mining (find related items)
Global and local structure of the vector space, global structure should capture the relation between random samples, local structure should express the relation between similar sentences

Loss Functions for Representation Learning

Contrastive loss pulls positive pairs together and pushes negative pairs away
Triplet loss uses an anchor, a positive, and a negative sample, positive sample should be close to the anchor and vice versa for the negative sample (requires triplets)
Batch hard triplet loss create dynamically hard triplets in a batch
Multiple negative ranking loss use multiple negative samples in a batch
Multiple Negative Ranking Loss Hard Negatives use large batch size and combine both hard negatives and multiple negatives
Additional losses cosine similarity loss, softmax loss, mega batch margin loss, multiple negative ranking loss, etc.

12 Sentence-BERT

Overview SBERT is a modification of BERT (RoBERTa) and uses a siamese and triplet like structure for semantically meaningful sentence embedding



SBERT at inference.

SBERT is able to semantically embed sentence with a linear complexity, BERT has quadratic complexity for this task

13 Generation Based Conversational AI

Application for conversational AI

- Personal assistants (Siri, Google, Alexa, etc.)
- Education
- Healthcare
- Customer service

Types of dialogue systems

- Task-oriented conversations
- Open-domain chit chat

Issues (automatic) evaluation is non-trivial