# Deep Transfer Learning for Natural Language Processing

## Concepts & Hands-on Tutorials

live@Manning

CONFERENCES

# Natural Language Processing

Dipanjan Sarkar

– "Deep Transfer Learning for Natural Language Processing"

The intent of this session is to journey through the recent advancements in deep transfer learning for NLP by taking a look at various state-of-the-art models and methodologies. These will include: Pre-trained embeddings for Deep Learning Models (FastText with CNNs\Bi-directional LSTMs + Attention), Universal Embeddings (Sentence Encoders, NNLMs), and Transformers.

We will also look at the power of some of these models, especially transformers, to solve diverse problems like summarization, entity recognition, question-answering, sentiment analysis, and classification. Plus lots of hands-on examples leveraging, Python, TensorFlow, and the famous transformers library from HuggingFace.

# Slide Deck and Code

https://bit.ly/lmnlp_ds20

About Me

# Dipanjan Sarkar

**Data Science Lead, Author, Google Developer Expert - ML**

Experts

Machine Learning

# Session Agenda

Transfer Learning Brief

Deep Transfer Learning for NLP

Modeling Approaches

Hands-on Tutorials


Transfer Learning Brief

# Transfer Learning

The ability to utilize knowledge learnt in prior tasks to new and novel tasks

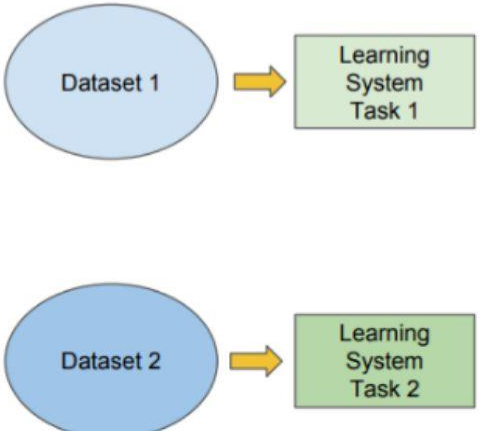
- Know how to ride a motorbike ➦ Learn how to ride a car
- Know how to play classic piano ➦ Learn how to play jazz piano
- Know math and statistics ➦ Learn machine learning

https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a

# Traditional ML vs. Transfer Learning

## Traditional ML

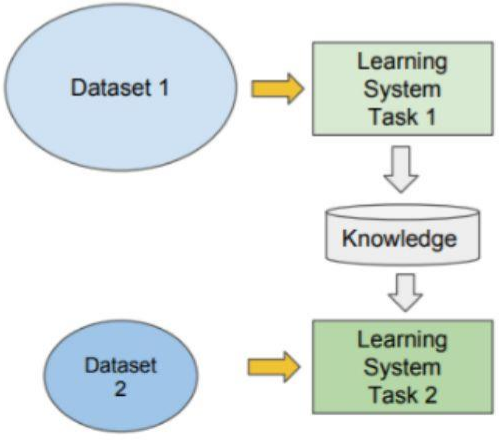
- Isolated, single task learning:
  - Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks



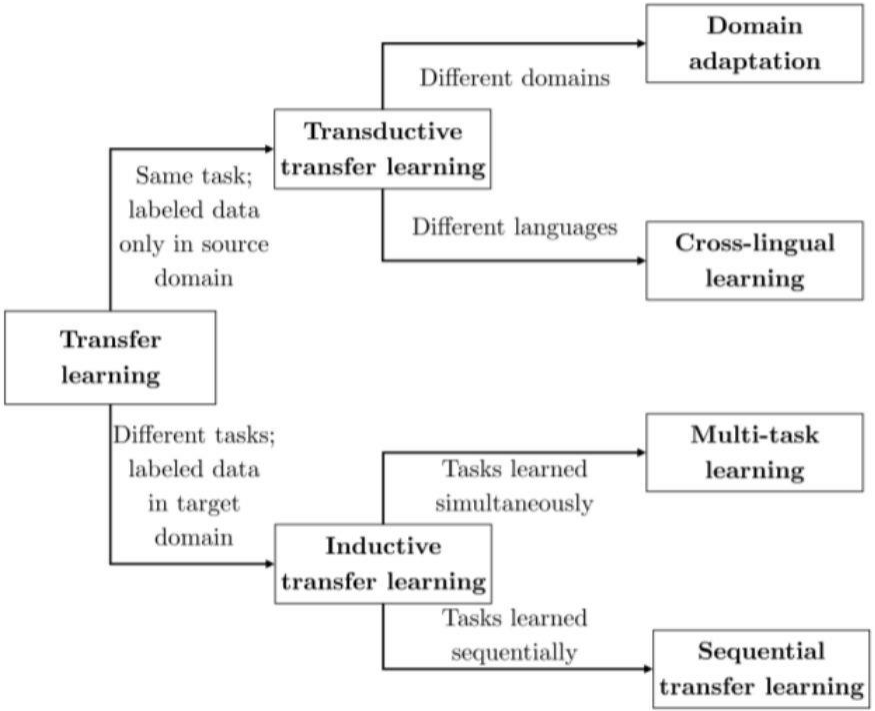

vs

## Transfer Learning

- Learning of a new tasks relies on the previous learned tasks:
  - Learning process can be faster, more accurate and/or need less training data



# Transfer Learning Scenarios for NLP

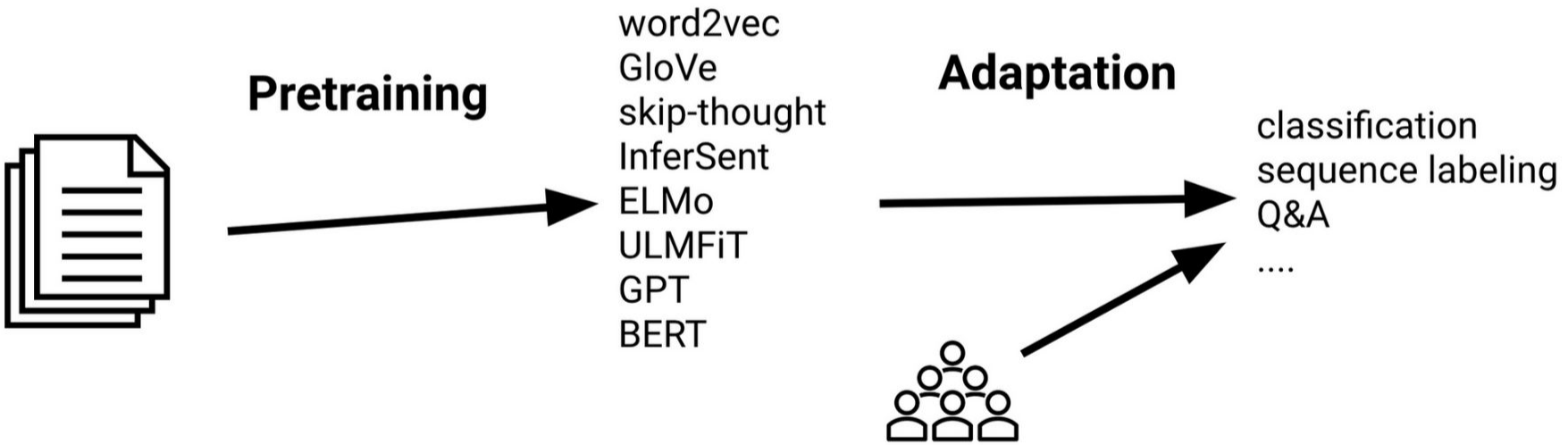



RUDER, 2019


Deep Transfer Learning for NLP

# Sequential Transfer Learning for NLP

Sequential transfer learning has led to the biggest improvements so far in the field of NLP




RUDER, 2019

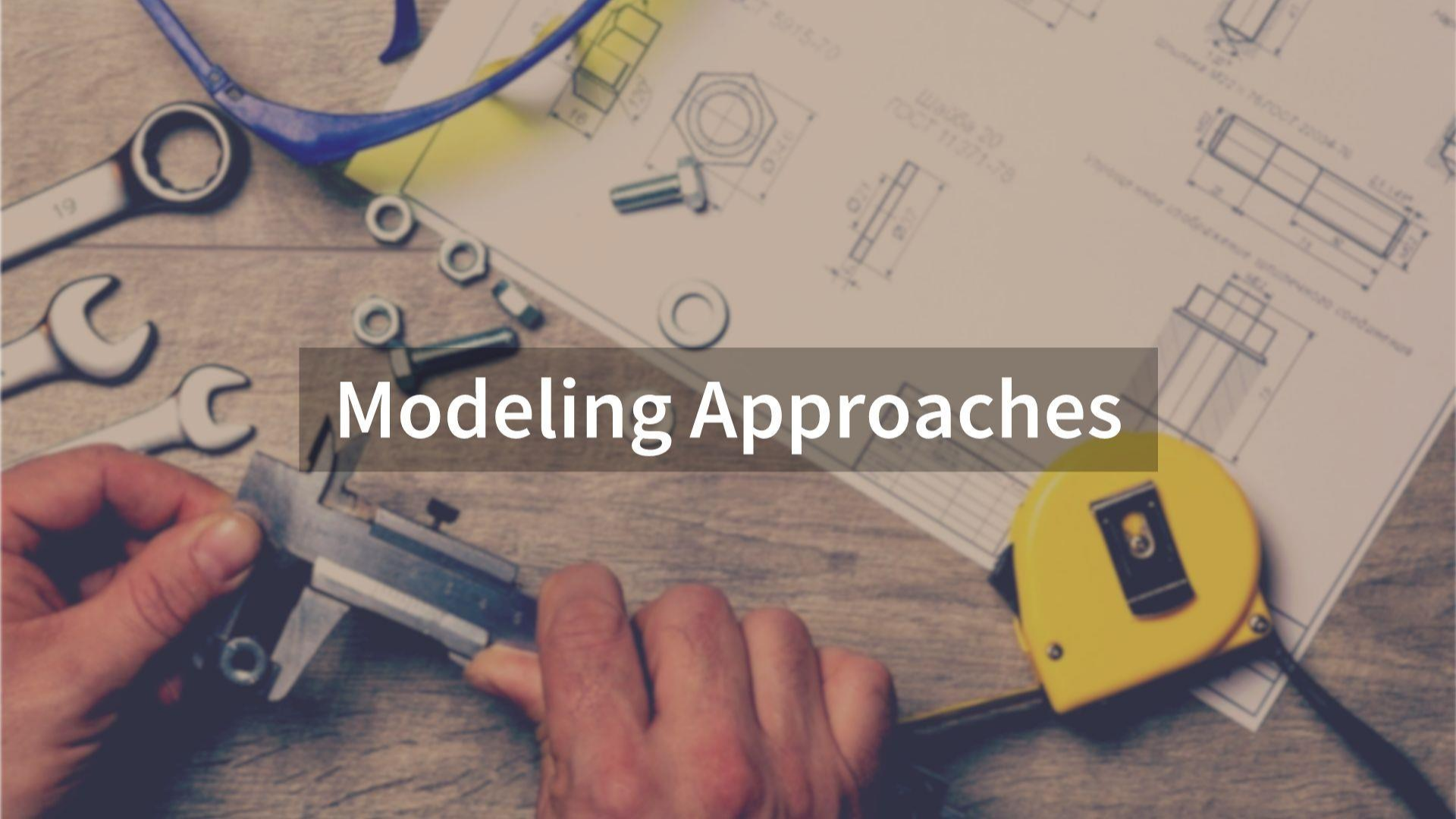
# Sequential Transfer Learning for NLP

## 1 Pre-training

- Word and context-based representations (word2vec, fasttext, elmo)
- Simple Language Models (RNN, LSTMs, Bi-LSTMs)
- Generalized Language Modeling (ULMFiT, GPT, BERT)

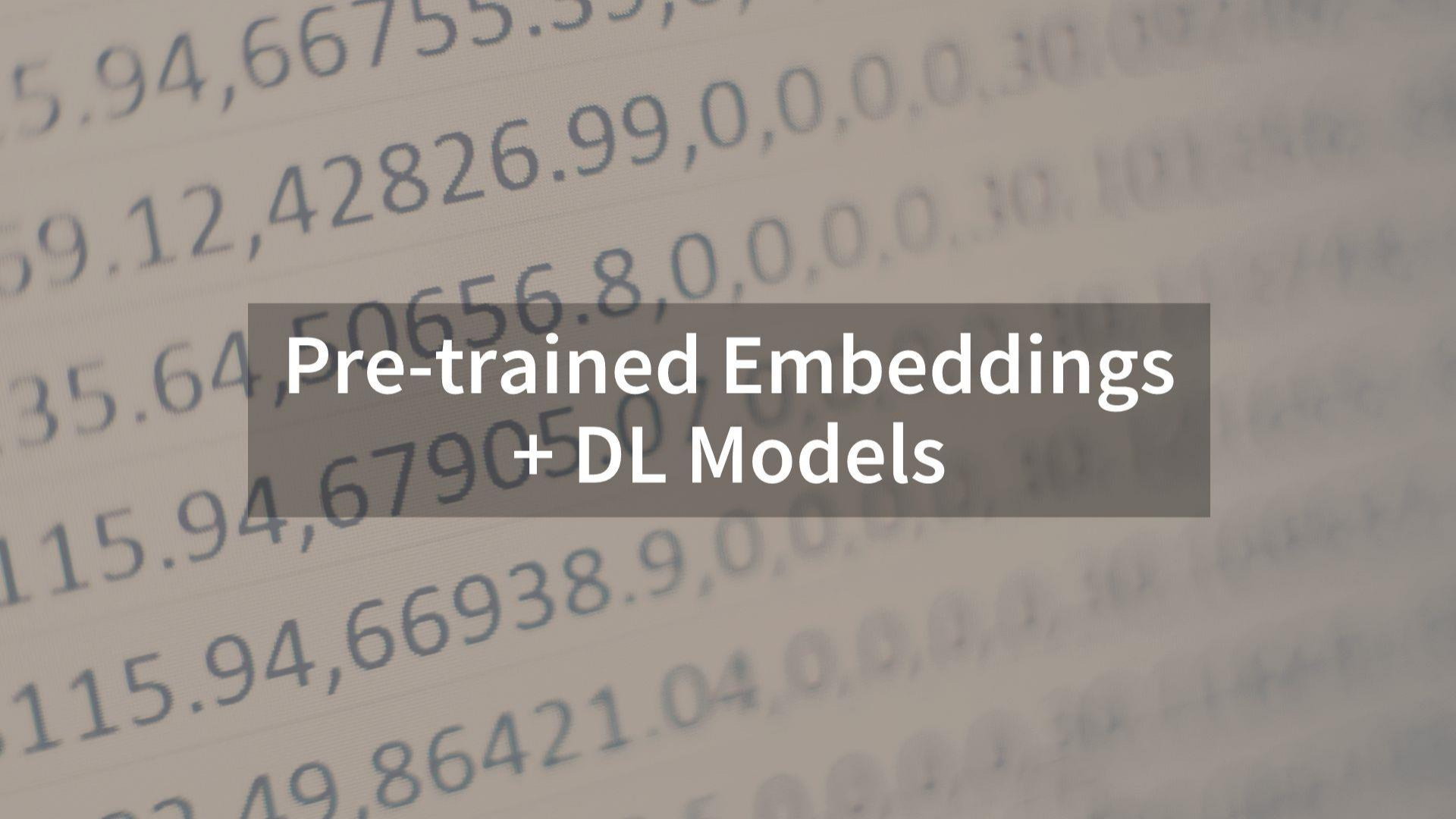
## 2 Adaptation

- Keep the model architecture and weights unchanged
- Change and fine-tune model architecture and weights

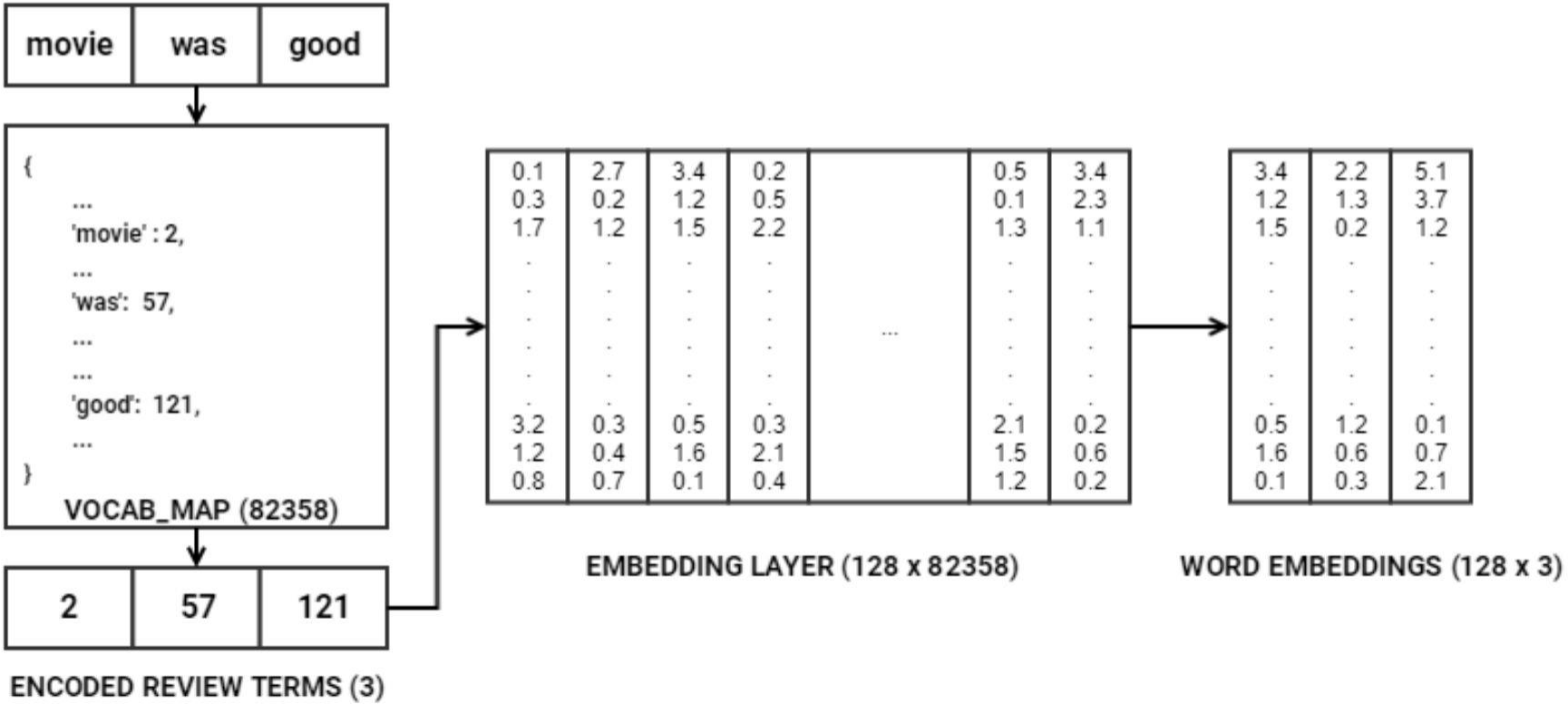

Modeling Approaches

# Model Architectures

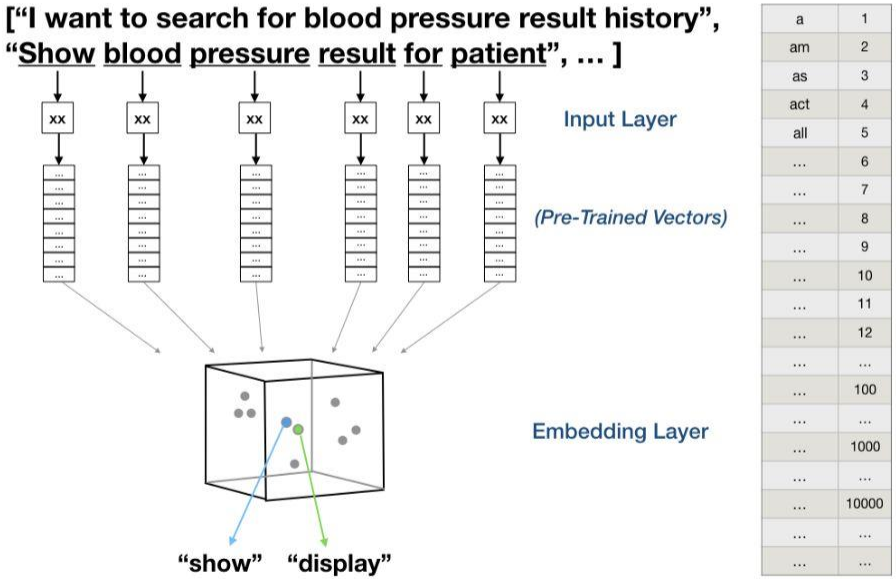
1. **Pre-trained Embeddings + Downstream DL Model**
   - CNNs
   - Bi-directional LSTMs + Attention
2. **Universal Embeddings**
   - Neural Network Language Model
   - Universal Sentence Encoders
3. **Transformers**
   - BERT
   - DistilBERT and more.


Pre-trained Embeddings
+ DL Models

# Embedding Layer



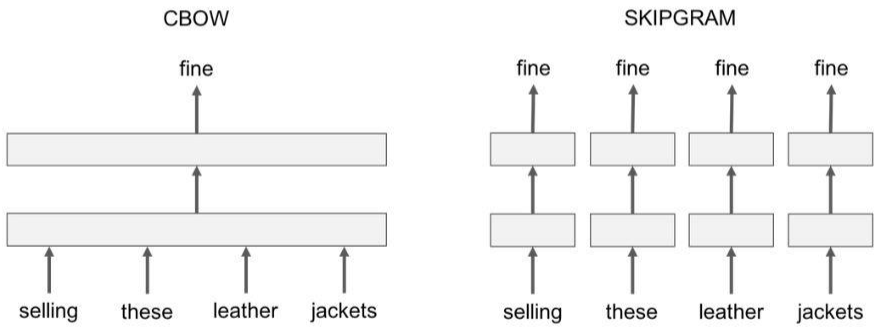
# Embedding Layer




| | |
|---|---|
| a | 1 |
| am | 2 |
| as | 3 |
| act | 4 |
| all | 5 |
| ... | 6 |
| ... | 7 |
| ... | 8 |
| ... | 9 |
| ... | 10 |
| ... | 11 |
| ... | 12 |
| ... | ... |
| ... | 100 |
| ... | ... |
| ... | 1000 |
| ... | ... |
| ... | 10000 |
| ... | ... |
| ... | ... |

- Don't initialize layer with random weights
- Initialize embedding layer with pre-trained embeddings
- Use pre-trained embeddings like FastText, Word2Vec, GloVe
- Model may perform better than random initialization

# FastText Embeddings

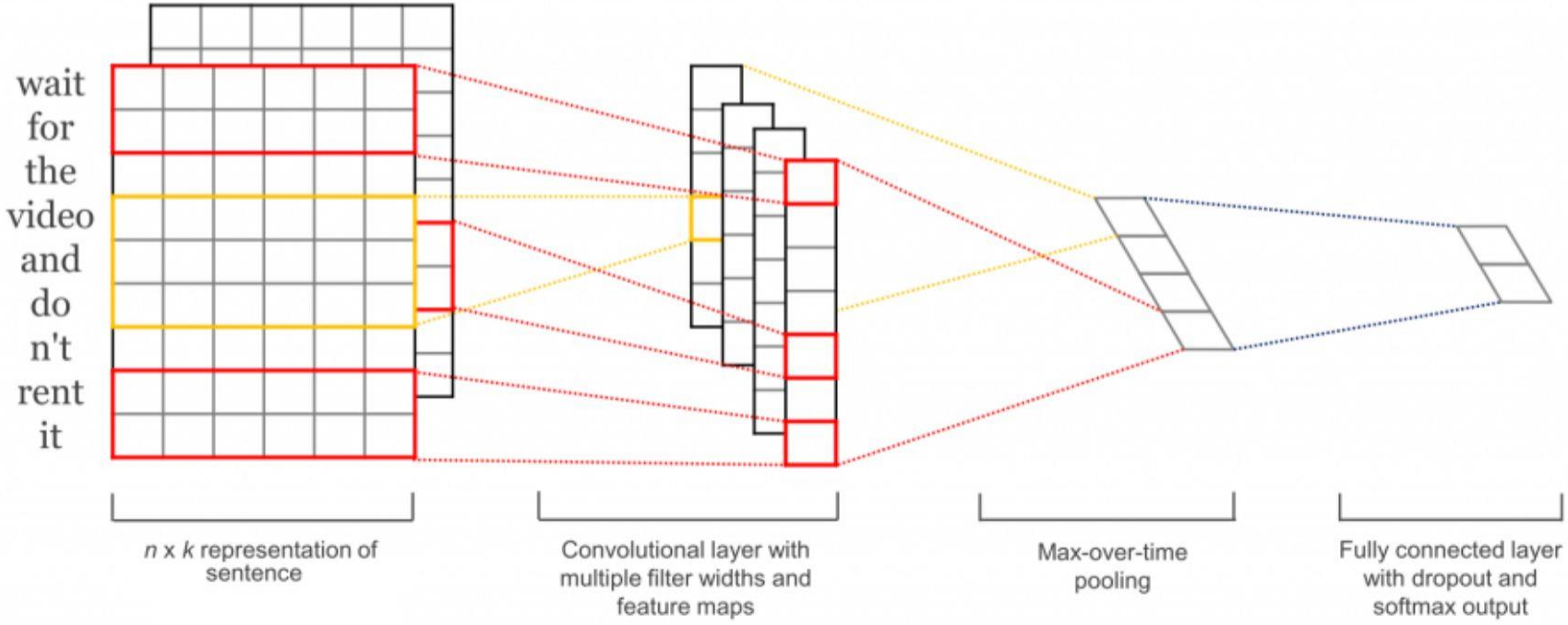



I am selling these fine leather jackets

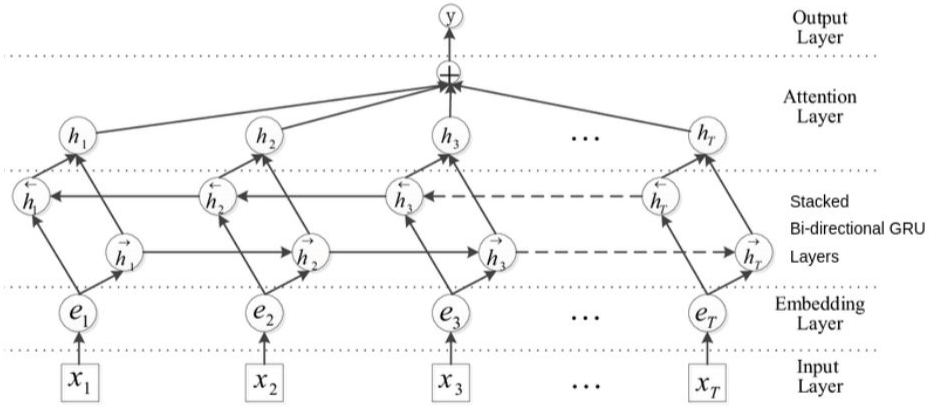
- **FastText has similar training architectures as Word2Vec**
- **Considers each word as a bag of character n-grams**
- **Taking the word where and n=3 (tri-grams) as an example:**

  Word is represented by the character n-grams: **<wh, whe, her, ere, re>** and the special sequence **<where>** representing the whole word

# Convolutional Neural Networks (CNN)

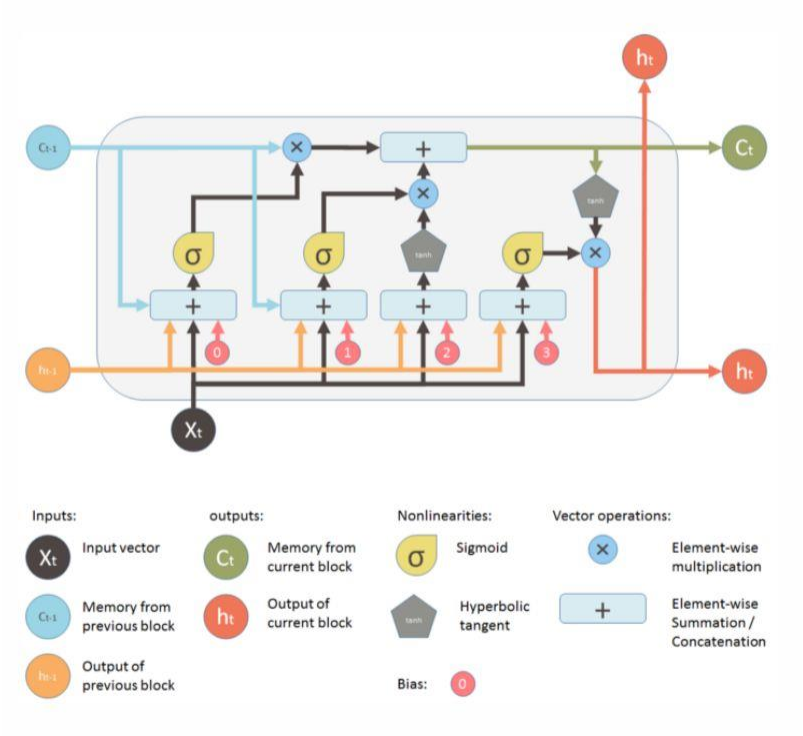


# Bi-directional LSTMs + Attention



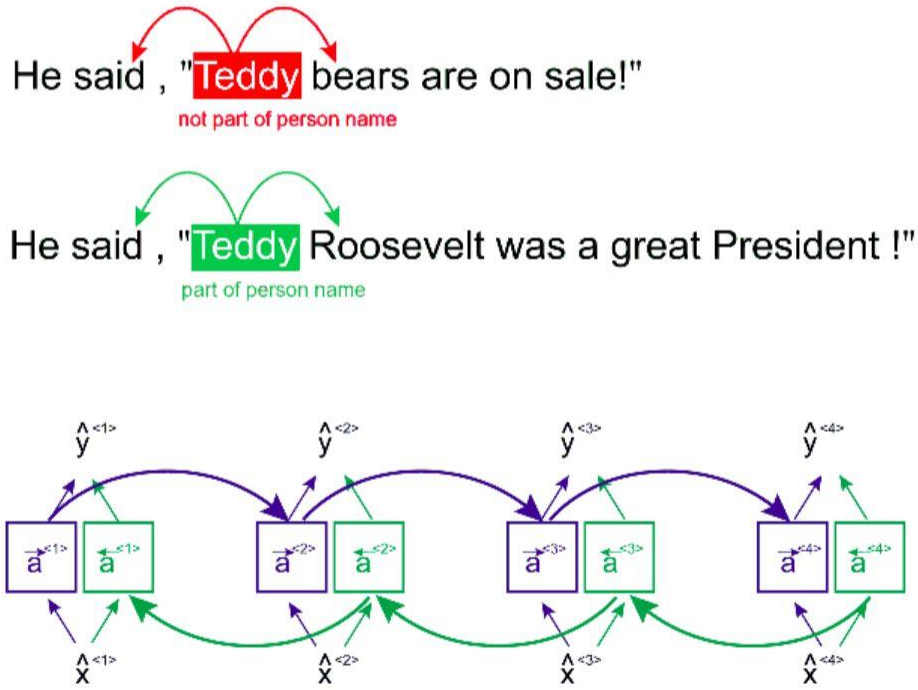

- Embedding Layer populated with pre-trained embeddings
  - FastText Embeddings
- Bi-directional GRUs or LSTMs
- Global Attention Layer
- FC Dense Layers

# Sequential Models - LSTM



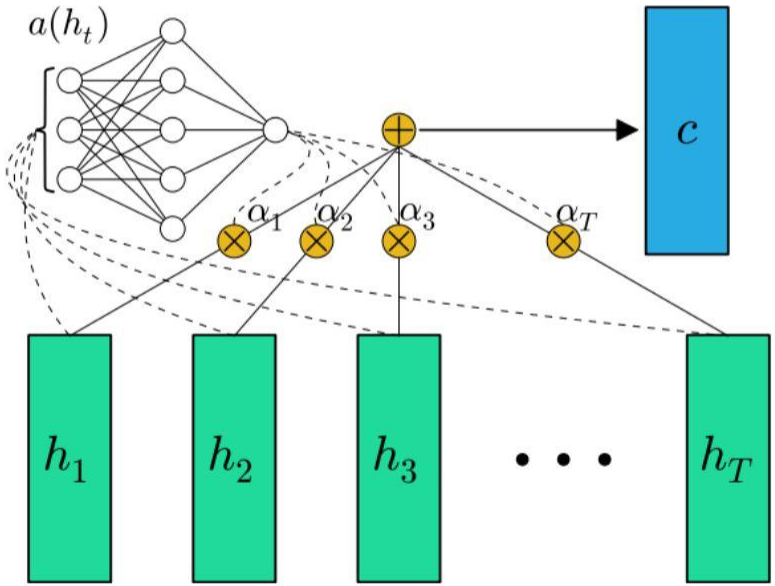

- LSTMs can remember longer sequences of data than RNNs
- The network takes three inputs
  - $X_t$ is the input of the current time step
  - $h_{t-1}$ is the output from the previous LSTM unit
  - $C_{t-1}$ is the "memory" or cell state of the previous unit
- The forget gate decides what is relevant to retain from previous steps
- The input gate decides what information can be added from the current step
- The output gate decides what the next hidden state should be

# The need for Bi-directional LSTMs




- Bi-directional LSTMs are just putting two independent LSTMs together
- The input sequence is fed in forward order for one LSTM, and in reverse order for the other
- The outputs of the two networks are usually concatenated at each time step
- Preserving information from both past and future helps understand context better

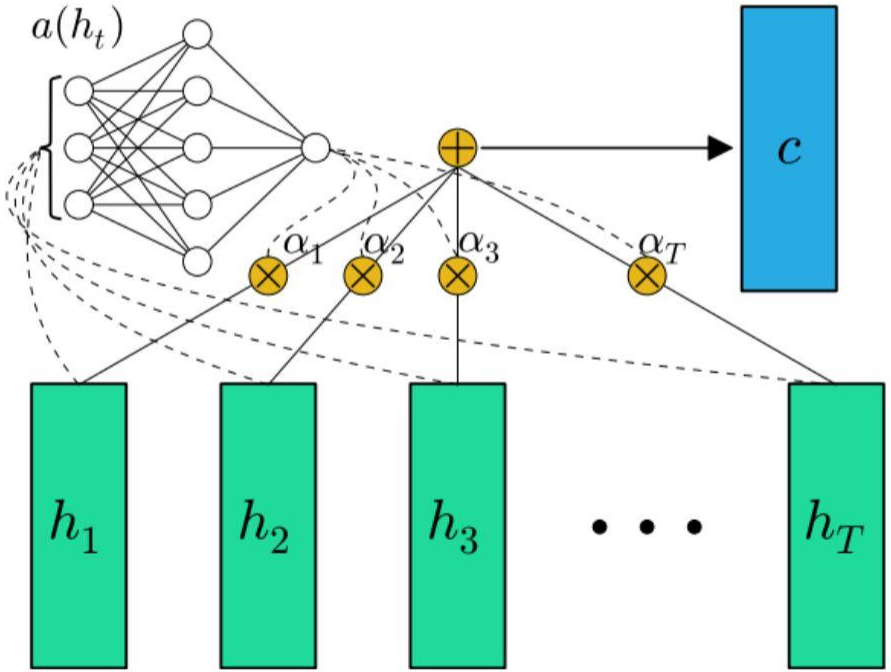
# Attention please!




- Instead of using the output from the last LSTM cell, send the entire sequence to a global attention layer
- Vectors from the hidden sequence $h_t$ are fed into the learnable function $a(h_t)$
- Produces a probability vector $\alpha_t$
- The final context vector c is a weighted average given by $\alpha_t.h_t$

Feed-Forward Networks with Attention Can Solve Some Long-Term Memory Problems, Raffel et al.

# Attention please!




$$e_t = a(h_t) = \tanh(Wh_t + b)$$

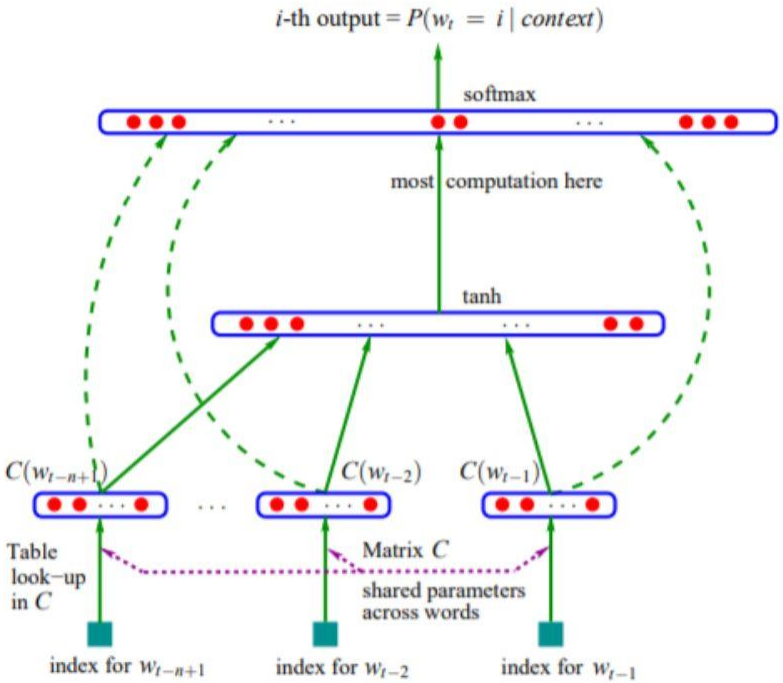
$$\alpha_t = \text{softmax}(e_t) = \frac{\exp(e_t)}{\sum_{k=1}^{T} \exp(e_k)}$$

$$c = \sum_{t=1}^{T} \alpha_t h_t$$

$T$ is the total number of time steps in the input sequence


Universal Embeddings

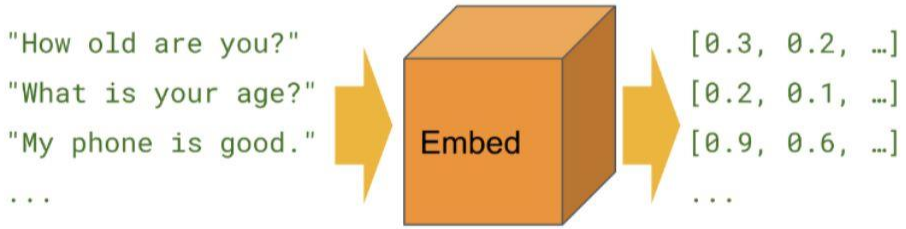
# Neural Network Language Model (NNLM)




- NNLM based sentence encoder also gives fixed length vectors for any document
- Pre-trained model is a 3 hidden layer neural network trained based on next word prediction tasks
- Embedding layer in pre-training is shared across all words

A Neural Probabilistic Language Model, Bengio et al.

# Universal Sentence Encoder (USE)




- **USE encodes text of any length into fixed high dimensional vectors that can be used for various downstream tasks**
- **Pre-trained model has two architecture variants**
  - Encoding sub-graph in Transformers
  - Deep Averaging Networks

Universal Sentence Encoder, Cer et al.

# Universal Sentence Encoder (USE) Methodology

## 1 Transformer Sub-graphs (Encoder stack)

- Uses the encoding sub-graph of the transformer architecture (Vaswani et al., 2017)
- This sub-graph uses attention to compute context aware representations of words in a sentence that take into account both the ordering and identity of all the other words.
- The context aware word representations are converted to a fixed length sentence encoding vector by computing the element-wise sum of the representations at each word position
- The encoder takes as input a lowercased (Penn TreeBank) PTB tokenized string and outputs a 512 dimensional vector as the sentence embedding

## 2 Deep Averaging Networks (DAN)

- The input embeddings for words and bi-grams are first averaged together and then passed through a feedforward deep neural network (DNN) to produce sentence embeddings
- Similar to the Transformer encoder, the DAN encoder takes as input a lowercased PTB tokenized string and outputs a 512 dimensional sentence embedding.

## 3 Training Methodology

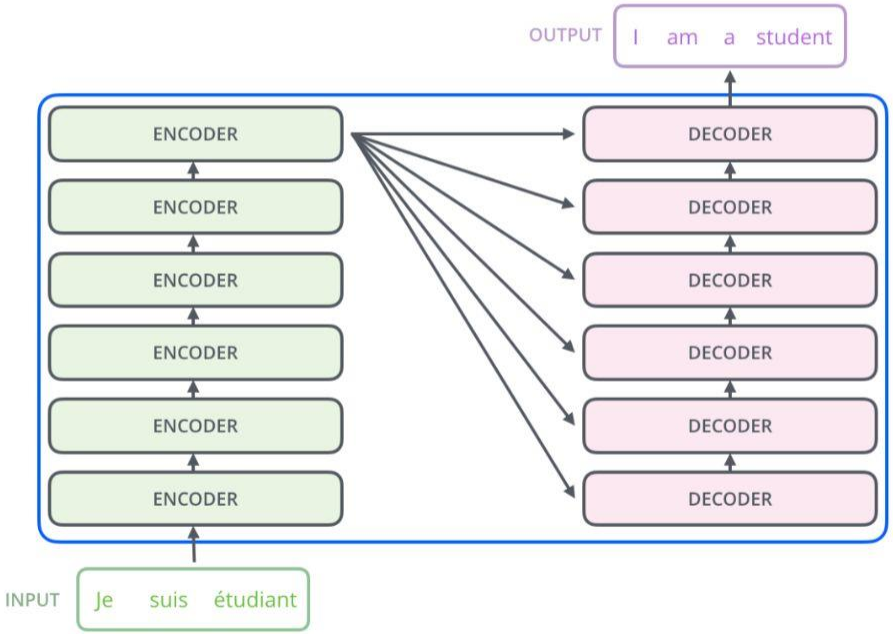
- Multi-task learning using a single encoding model is used to feed multiple downstream tasks
- Unsupervised learning is augmented with training on supervised data from the Stanford Natural Language Inference (SNLI) corpus


Transformers

# Transformer Model - Architecture

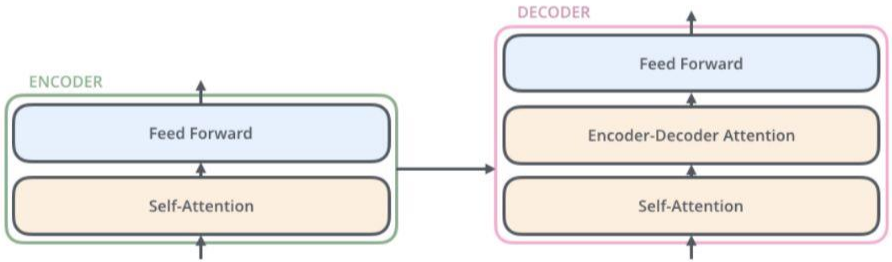

- Layered & Stacked Encoder - Decoder Model
- Relies on Multi-headed Self-Attention & Encoder-Decoder Attention
- No sequential RNN based training (completely parallelized)
- Achieved state-of-the-art performance on several NLP tasks

# Transformer Model - Architecture



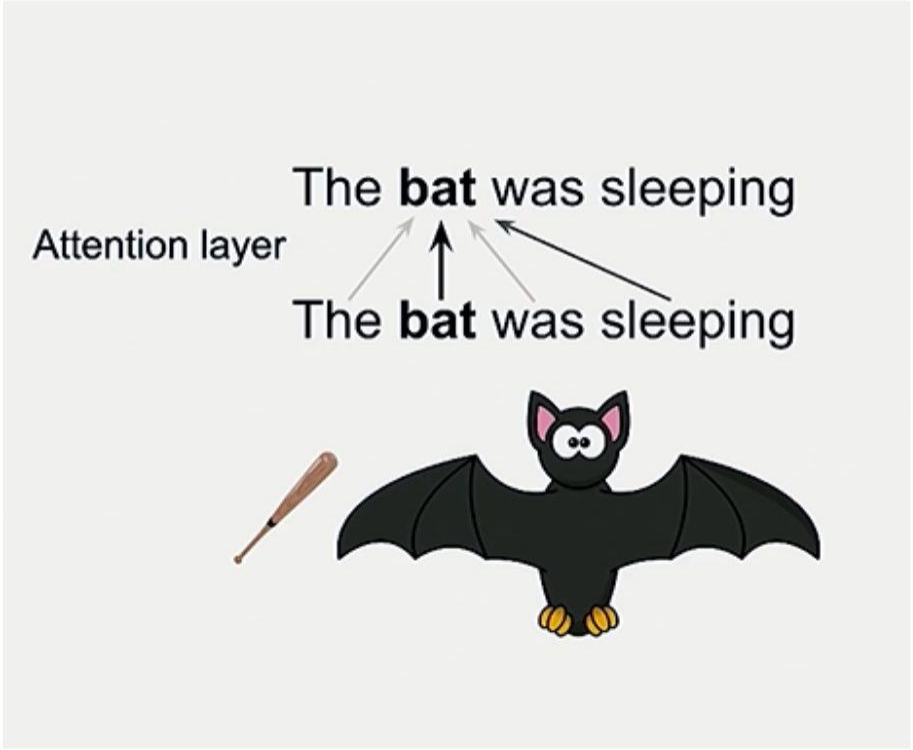

- At-heart a transformer model is a stacked encoder-decoder model
- Has several stacked encoder and decoder blocks usually based on standard architectures
- Based on the type of transformer model (only encoder \ decoder or both are used)

# Transformer Model - Architecture



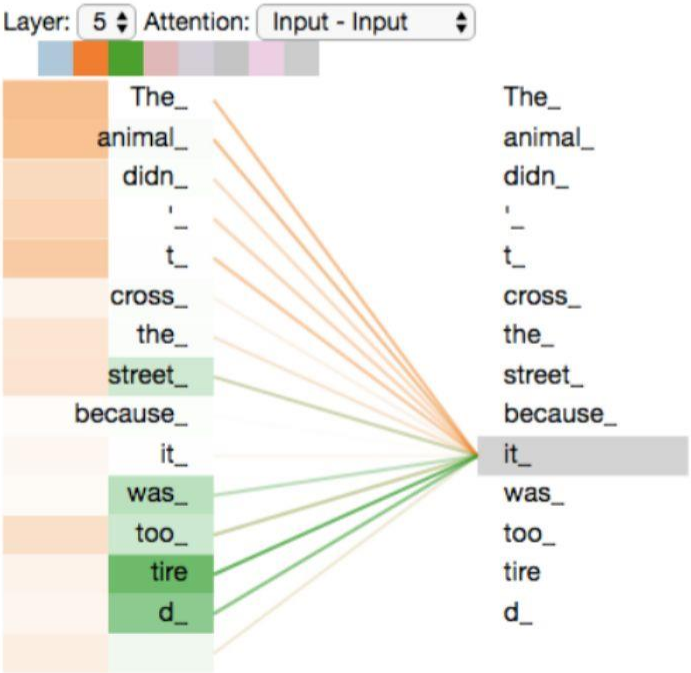

- **The encoder's inputs flow through a self-attention layer**
  - Each word attends (looks at) every other word in this process
- **The outputs of the self-attention layer are fed to a feed-forward neural network**
- **The decoder has similar layers**
- **Between them is an attention layer that helps the decoder focus on relevant parts of the input sentence**
  - This is the encoder-decoder attention layer

# Self-Attention: A Simplistic Overview




- What does “bat” in this sentence refer to? Is it referring to a baseball bat \ cricket bat or the animal?
- When the model is processing the word “bat”, self-attention allows it to associate ”bat” with “sleeping” strongly
  - This gives it  the indication that it could be an animal
- Self attention allows the model to look at other positions in the input sequence for clues that can help lead to a better encoding for each word

# Multi-Headed Attention - Encode different aspects




As we encode the word "it", one attention head is focusing most on "the animal", while another is focusing on "tired" -- in a sense, the model's representation of the word "it" bakes in some of the representation of both "animal" and "tired".

Jay Alammar, 2018


Transformer Model Architectures

# Transformer Model Types

## 1 Autoregressive Models

- Correspond to the decoder of the original transformer model
- Mask is used on top of the full sentence so that the attention heads can only see previous words
- Pretrained on the classic language modeling task: guess the next token
- Example - GPT family

## 2 Autoencoding Models

- They correspond to the encoder of the original transformer model
- They get access to the full inputs without any mask
- Pretrained by corrupting the input tokens in some way and trying to reconstruct the original sentence (Masked Language Modeling)
- Example - BERT family

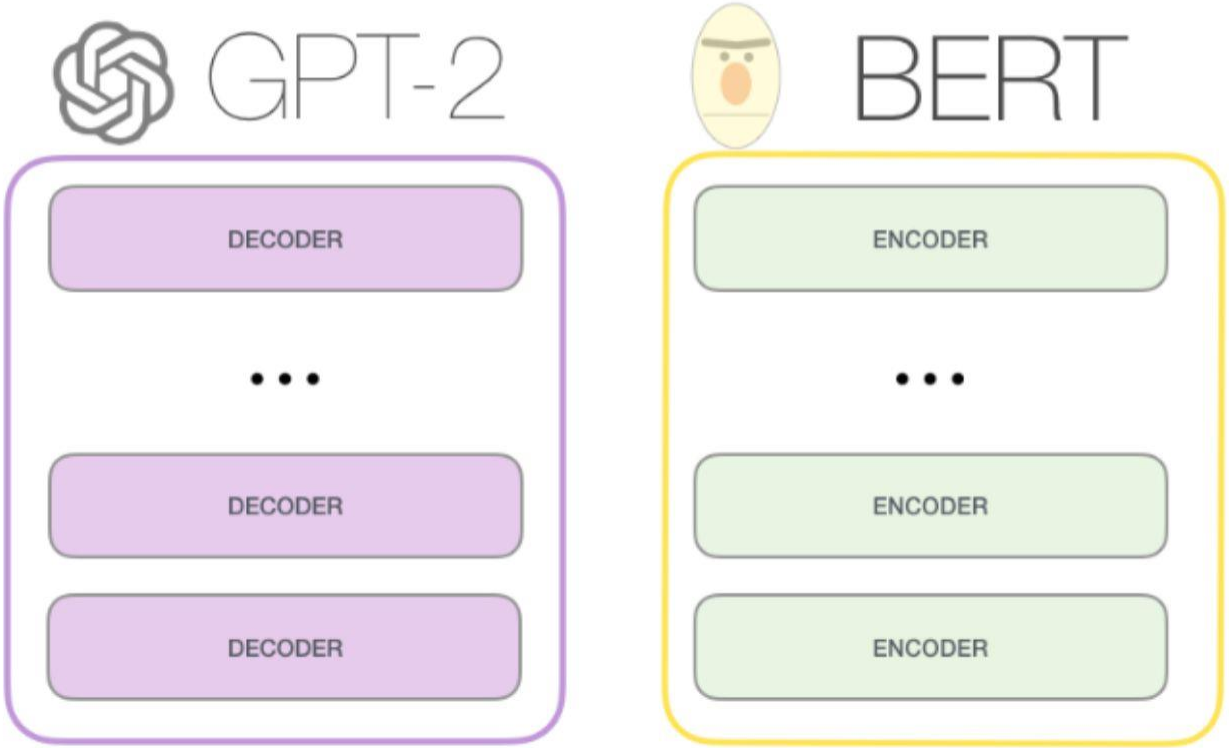
## 3 Sequence to Sequence Models

- Uses both the encoder and the decoder of the original transformer
- Most popular applications are translation, summarization and question answering
- Example - BART, T5

## 4 Multi-modal Models

- Mixes data of different modalities e.g text & images

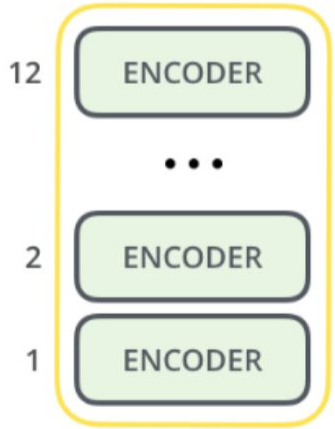
# Popular Transformer Models



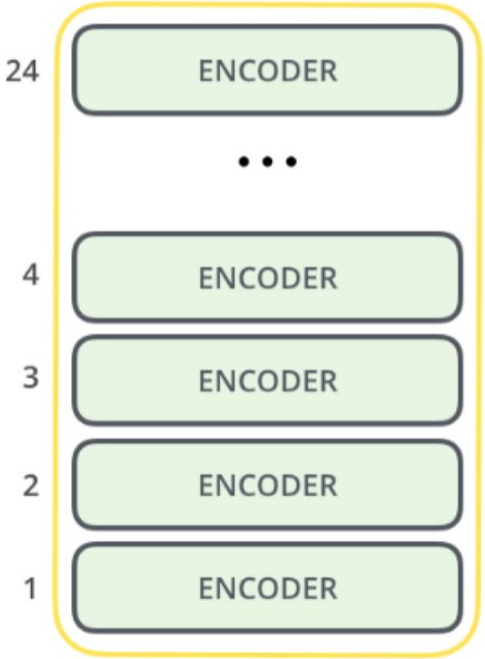

JAY ALAMMAR, 2019


Understanding BERT & DistilBERT

# BERT - Trained Transformer Encoder Stack



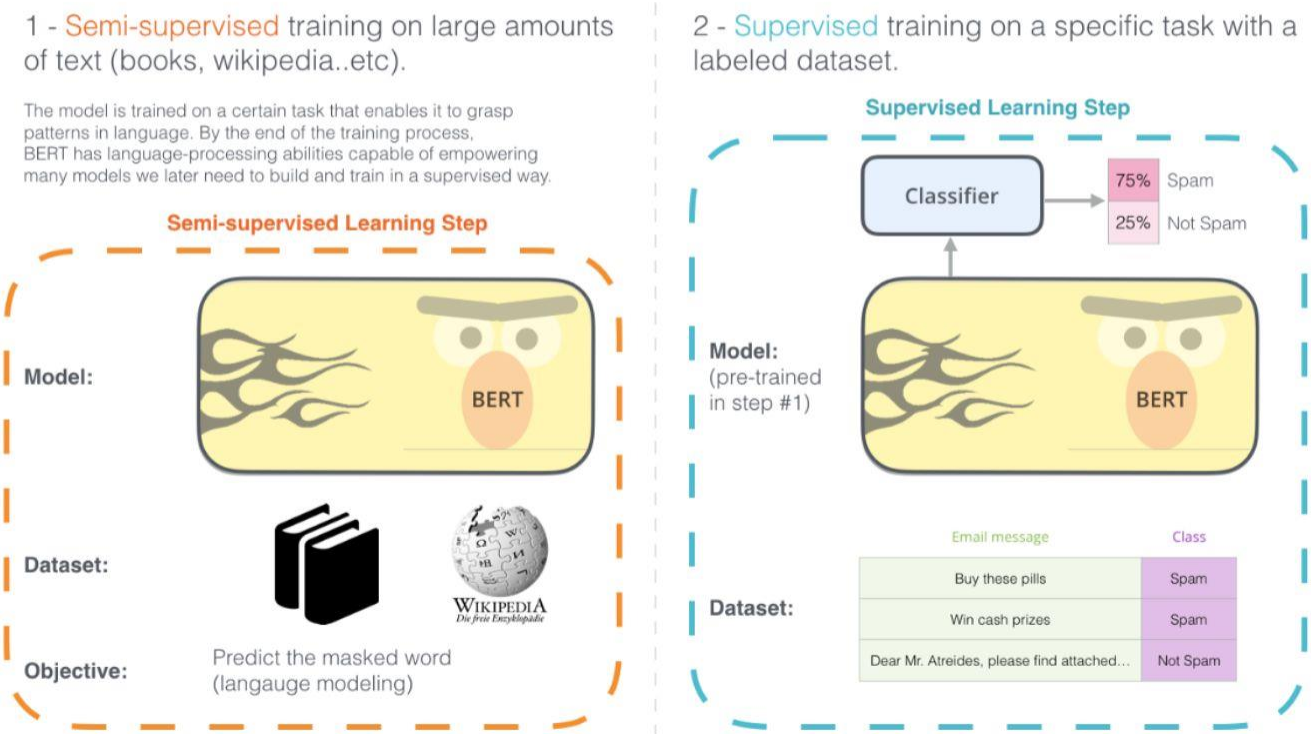

$BERT_{BASE}$




$BERT_{LARGE}$

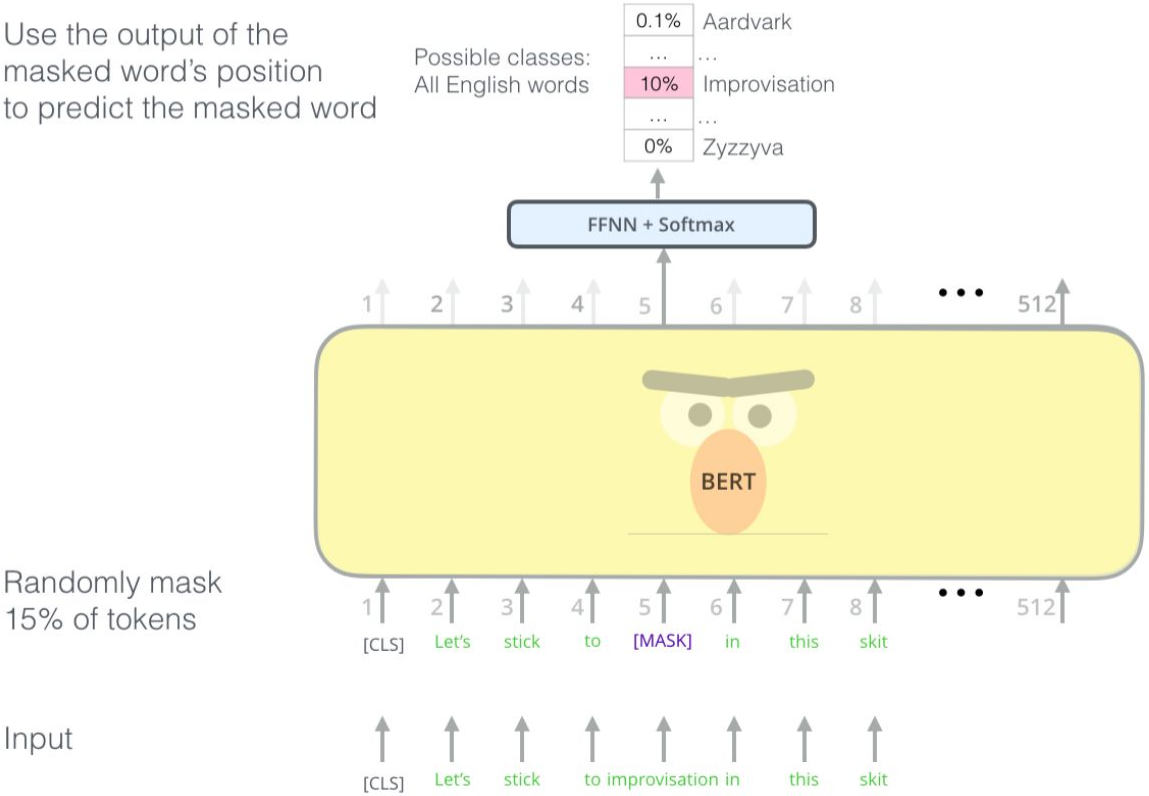
JAY ALAMMAR, 2018

# BERT Training Workflow



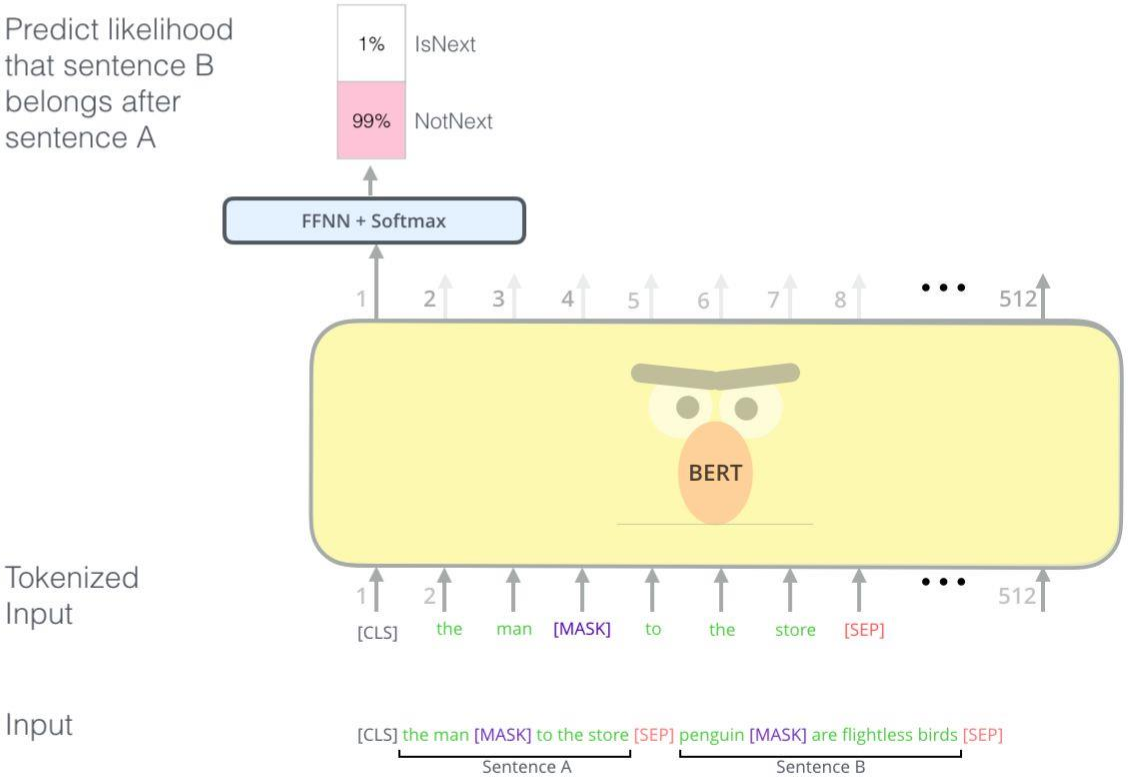

JAY ALAMMAR, 2018

# BERT Training Workflow - Masked Language Modeling



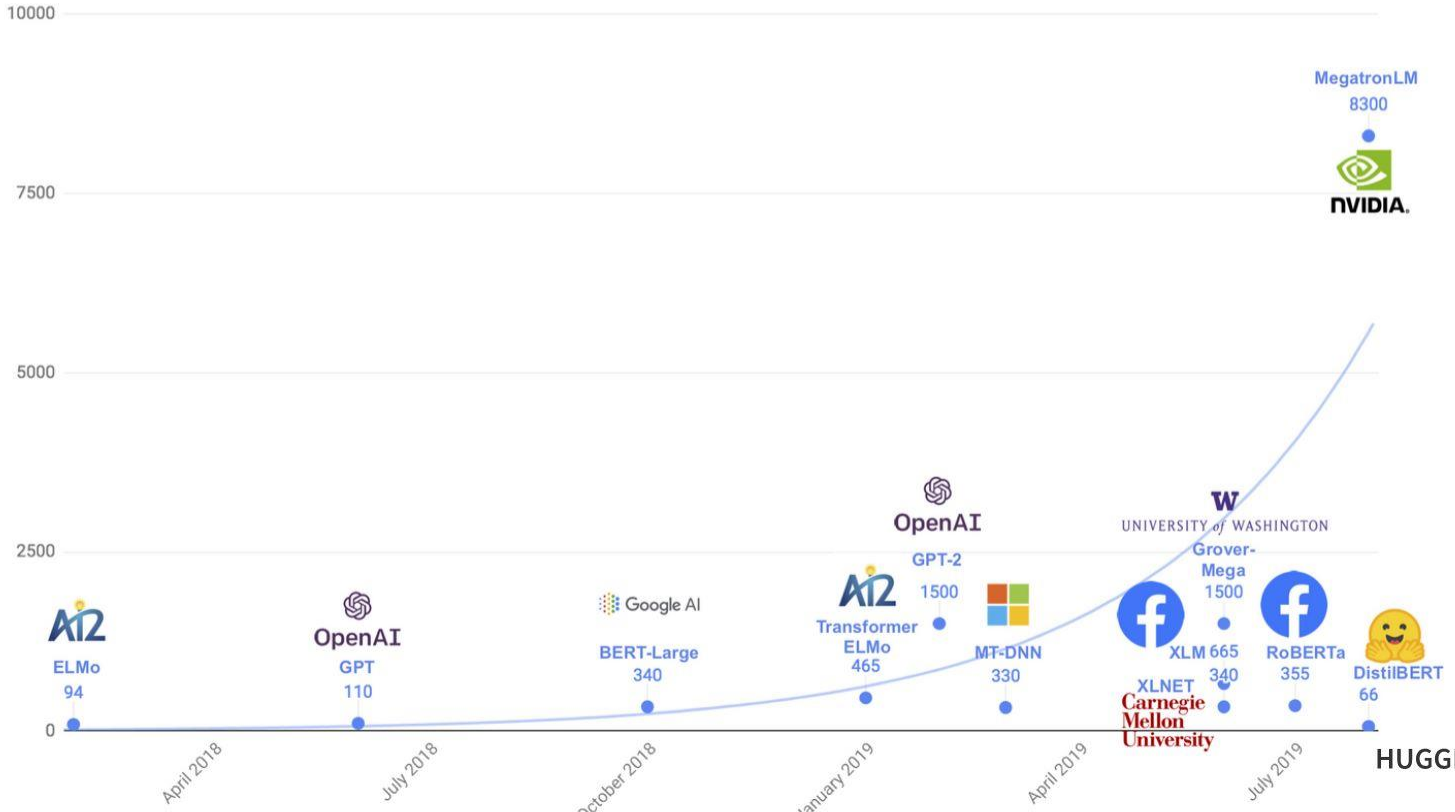

JAY ALAMMAR, 2018

# BERT Training Workflow - Next Sentence Prediction




JAY ALAMMAR, 2018

# Challenges with Massive Size of Pre-trained Models




HUGGINGFACE, 2019

# DistilBERT - Compressing BERT

- **Compression technique known as knowledge distillation is used**
- **Teacher-Student Training is used - training student to mimic output distribution of teacher model (BERT)**

  Trained with cross-entropy on soft targets (probabilities from teacher model)
- **HuggingFace uses KL-divergence loss to train DistilBERT**
- **Has almost halved the number of parameters from BERT and retains 95% of performance**

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


Hands-on Tutorials

# Hands-on Tutorials

## 1 Text Classification - Less Data Availability Problem

- Logistic Regression Baseline
- Pre-trained Embeddings + DL Models
- Universal Embeddings
- Transformers

## 2 Multi-task NLP with Transformer Pipelines

- Sentiment Analysis
- NER
- Question & Answering
- Text Generation - Mask Filling
- Summarization
- Translation
- Generation
- Feature Extraction

# References

- **Research Papers**
  - https://github.com/dipanjanS/live-manning-nlpconf20/tree/master/papers
- **Visuals & Content**
  - https://jalammar.github.io/illustrated-transformer/
  - NLP Using Transformer Architectures - A. Géron
  - Medium \ Towards Data Science
  - https://ruder.io/state-of-transfer-learning-in-nlp/
  - https://colah.github.io/posts/2015-08-Understanding-LSTMs/
  - https://medium.com/huggingface/distilbert-8cf3380435b5

# Stay in Touch!

LinkedIn

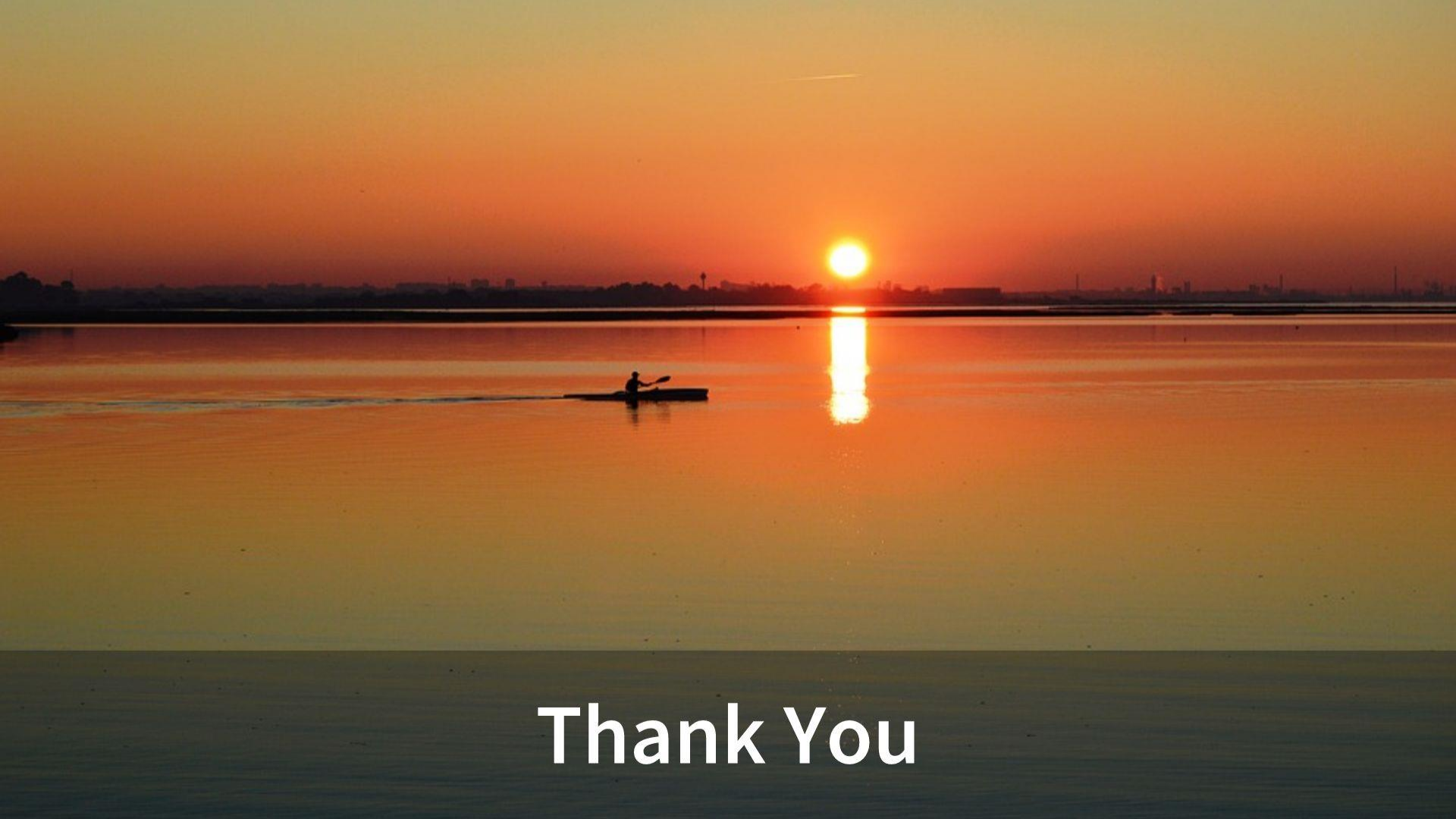
https://www.linkedin.com/in/dipanzan

GitHub

https://github.com/dipanjanS

Medium

https://medium.com/@dipanzan.sarkar


Thank You