Support Vector Machines
Sequence Kernels
Embeddings

13.6.2019

Agenda Today

- Support Vector Machines
- Sequence Kernels
- Learning Feature Representations: Embeddings
    - Autoencoders
    - Embedding Layers
    - NLP: Word2Vec, Doc2Vec, GloVe, ELMo, ULMFit, BERT, ELMo

Motivation

- Text-independent speaker identification (verification, recognition)
  - Large number of target speakers (hundreds of thousands)
  - Very little data per speaker (4-10 seconds)
  - Often just 1 sample for enrollment (training)
- Age estimation from speech **(today‘s exercise!)**
  - Regression problem
  - Sparse data

SVM

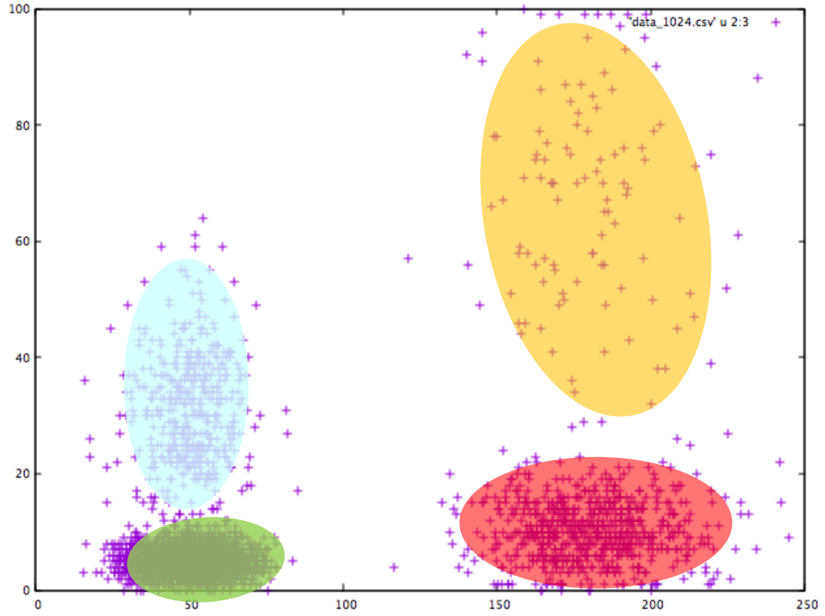
Slides by Martin Law

Sequence Kernels

Slides by Elmar Nöth (FAU-INF5)

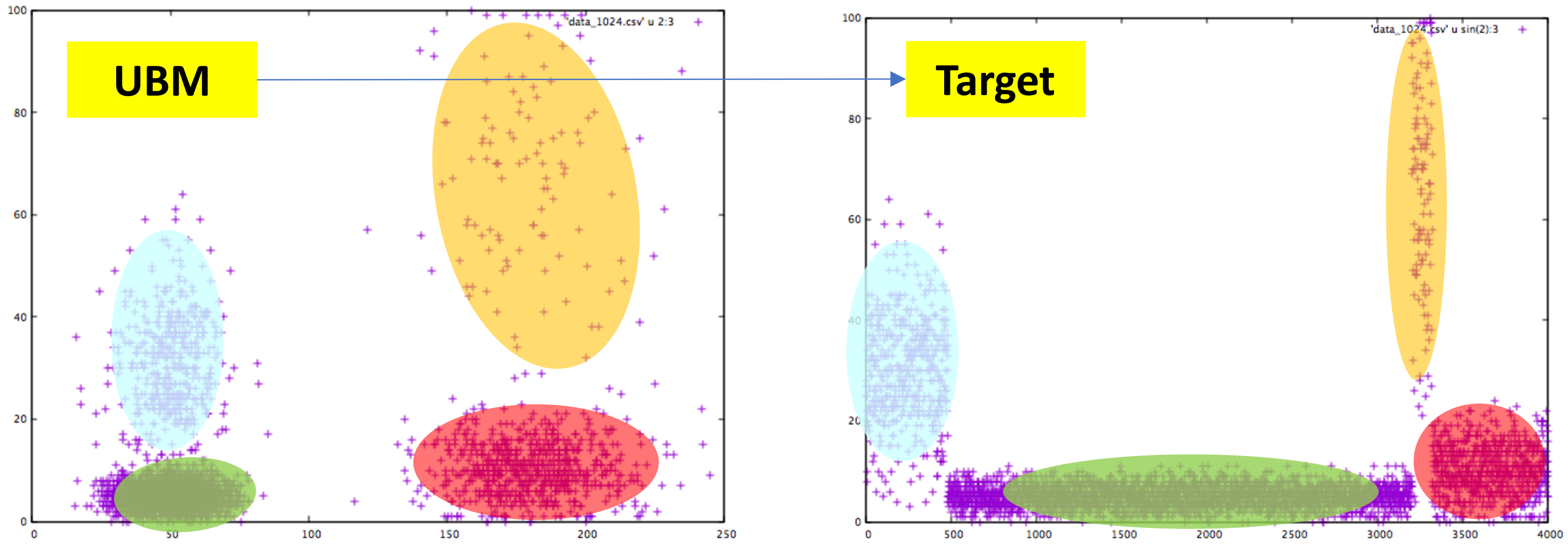
Modelling Speakers for SVM

- Sequences may differ in length
- Single decision for whole sequence (many-to-one)
- Use Gaussian mixture models to represent speakers



GMM Supervectors

- Train a „background“ model with many speakers – unsupervised!
- Adapt a „target“ model for every speaker (or class), eg. max-a-posteriori
- Use GMM parameters as features to SVM



Bhattacharyya based Kernel

- You can use any kernel with the supervectors, however…
- Since all SV are derived from the same UBM, you can compute the „distance“ of the components based on Bhattacharyya distance

$$D(g_a \| g_b) = \int_{R^n} g_a(x) \log\left(\frac{g_a(x)}{g_b(x)}\right) dx$$

$$d(\mathbf{m}_a, \mathbf{m}_b) = \frac{1}{2} \sum_{i=1}^{N} \lambda_i (\mathbf{m}_i^a - \mathbf{m}_i^b) \boldsymbol{\Sigma}^{-1} (\mathbf{m}_i^a - \mathbf{m}_i^b).$$

*You et al. 2008: An SVM Kernel With GMM-Supervector Based on the Bhattacharyya Distance for Speaker Recognition (*https://ieeexplore.ieee.org/document/4734326*)*

Learning Feature Representations

- MFCC etc. are spectral/cepstral speech features, motivated by how the source signal is produced
- For text, we resorted to *one-hot* encoding
- It is often better to *learn* the feature representations instead
  - Convolutional layers learn to extract structural (temporal) information
  - Embedding layer learns a (typically compressed) representation of the one-hot input.
  - …randomly initialized, trained as part of the network (**supervised**!)

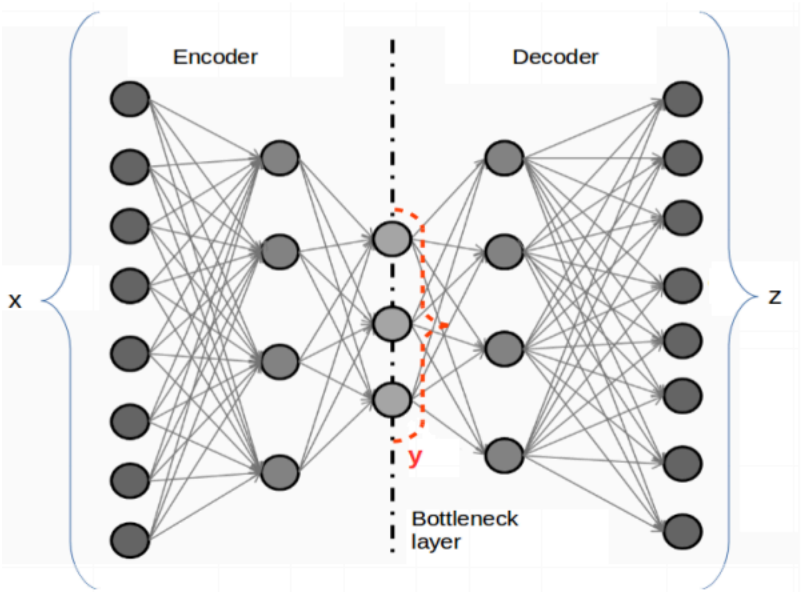
Learning Feature Representations cont‘d

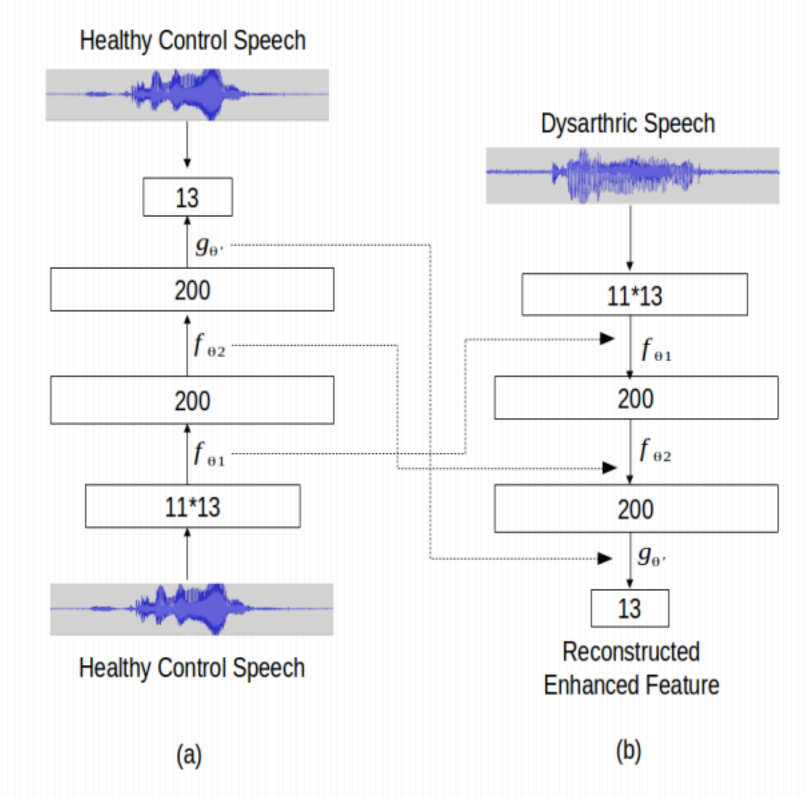
- Leverage unlabeled data to learn feature representations
- Learn how observations relate to their surroundings, hence the name „embedding“
- Most embeddings are some sort of auto-encoder (AE)
  - …they learn about the structure by by first encoding the data to an intermediate representation x -> h,
  - …then decoding/reconstructing it h -> r, where r should match x.
  - Undercomplete AE: dim(h) < dim(x)
  - Overcomplete AE: dim(h) > dim(x)

AE for Speech Tasks

Vachhani et al., 2017: Deep Autoencoder based Speech Features for Improved Dysarthric Speech Recognition

- Input (x): 11 Frames of MFCC
- Output (z): center MFCC frame



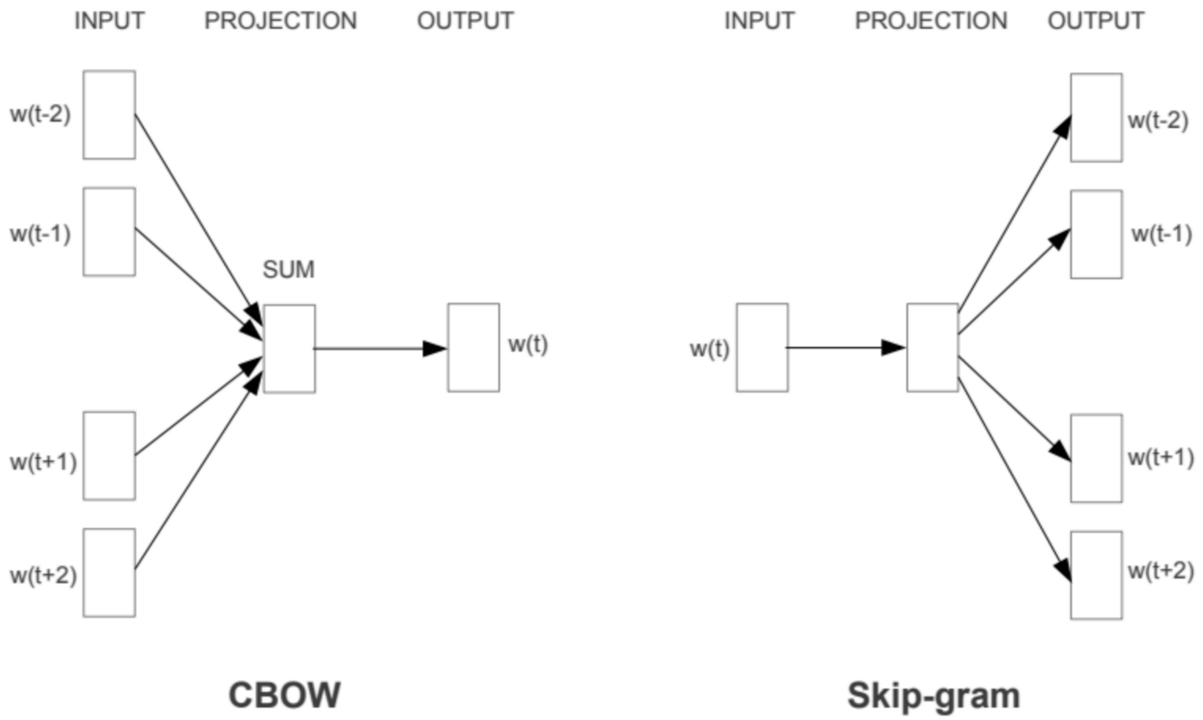


Embeddings for NLP
How to represent words in Context?

- Bag-of-words (BOW)
- Word2Vec: Continuous bag of words (CBOW) and skip-gram
- GloVe
- FastText
- CoVe
- Higher-level embeddings: ULMfit, ELMo, BERT

Word2Vec: CBOW and Skip-Gram

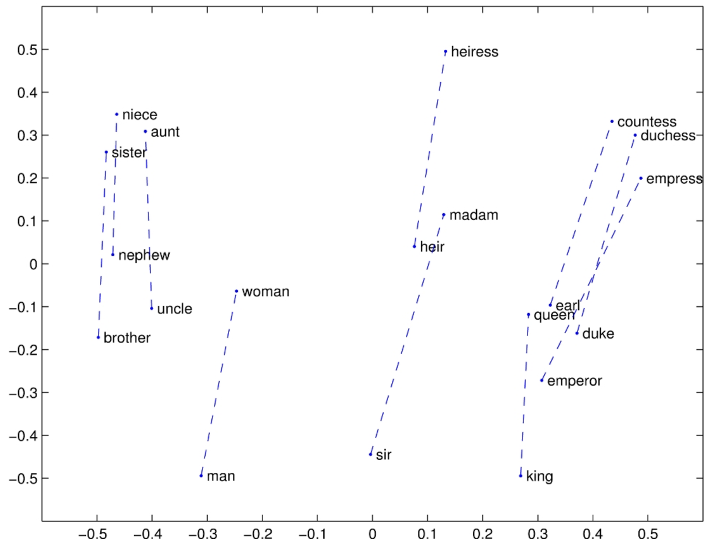
- Predictive model using trained feed-forward network

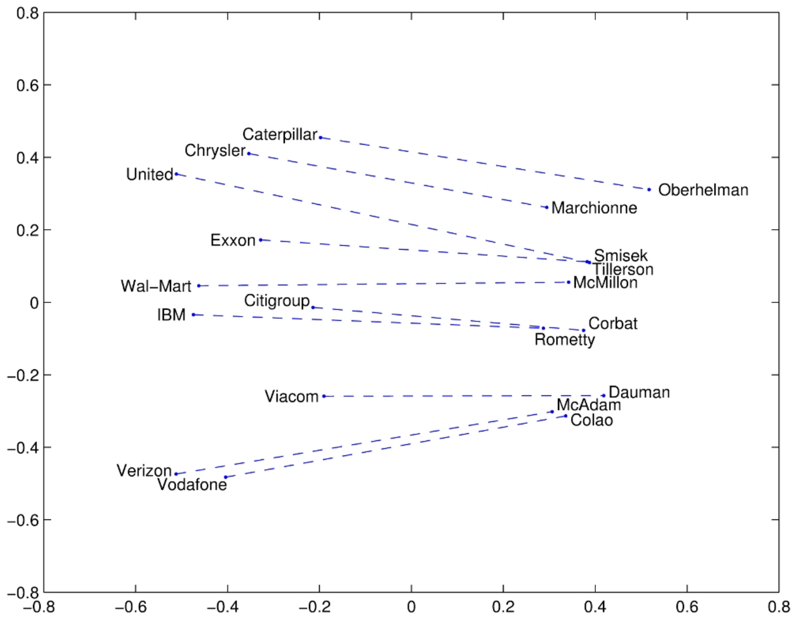


Mikolov et al., 2013: Efficient Estimation of Word Representations in Vector Space https://github.com/tmikolov/word2vec

GloVe

- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. [GloVe: Global Vectors for Word Representation](#).
- Co-occurrence count-based, analytically reduce dimensionality
- Linear substructures

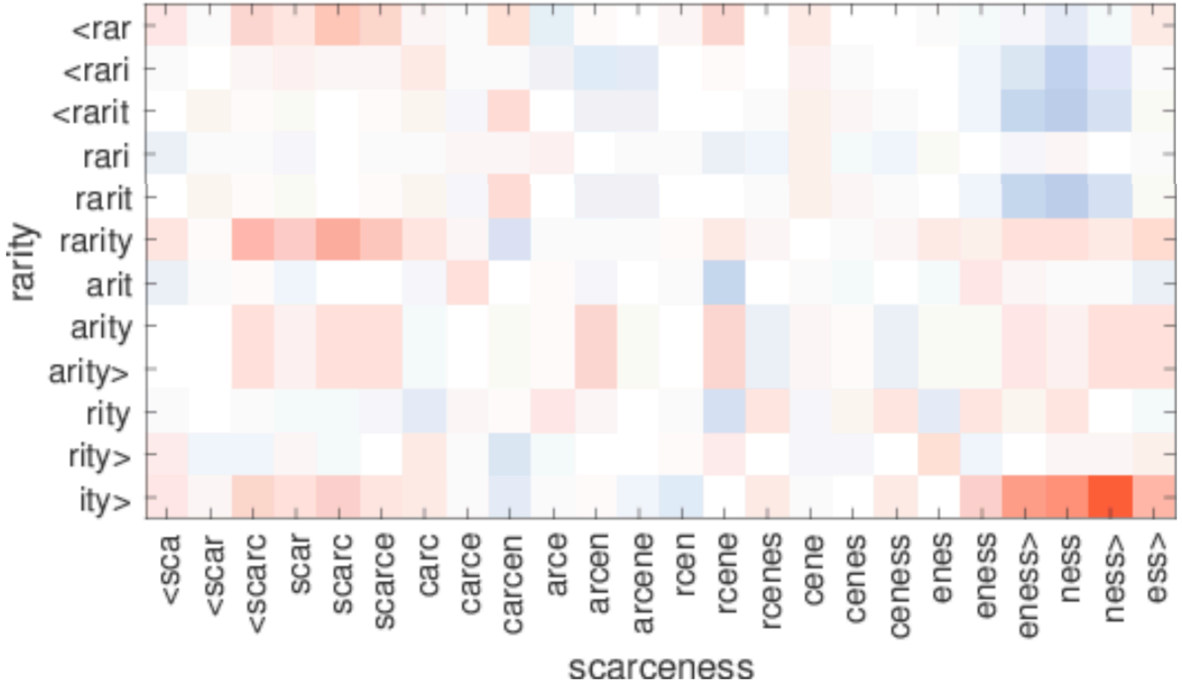




https://nlp.stanford.edu/projects/glove/

FastText (Facebook AI)

- Bojanowski et al., 2016: Enriching Word Vectors with Subword Information
- Use character n-grams instead of words to be able to predict representations for unseen words
- Example: high similarity between
  - scarce and rare
  - -ness and -ity



https://fasttext.cc/

CoVe

- McCann et al. 2017: Learned in Translation: Contextualized Word Vectors
- Use LSTM from (unrelated) machine translation task as embedding

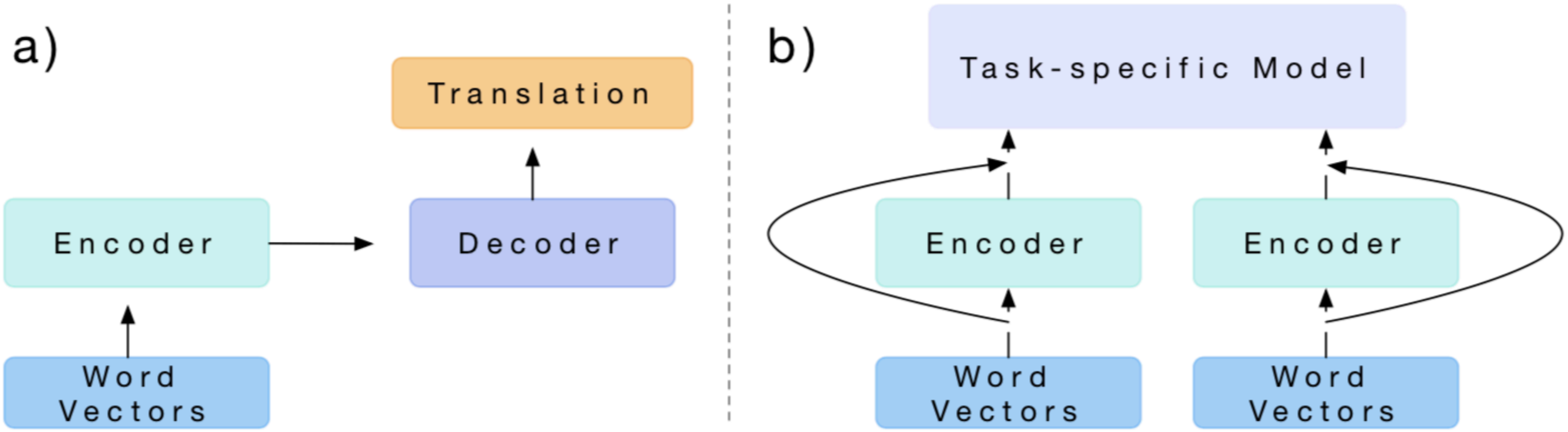
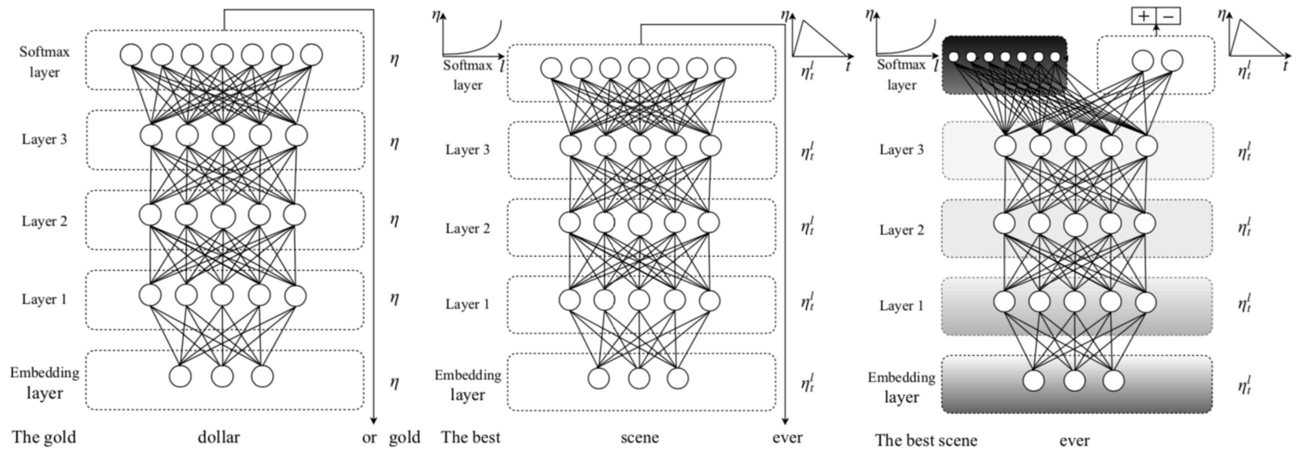


Figure 1: We a) train a two-layer, bidirectional LSTM as the encoder of an attentional sequence-to-sequence model for machine translation and b) use it to provide context for other NLP models.

ULMFiT

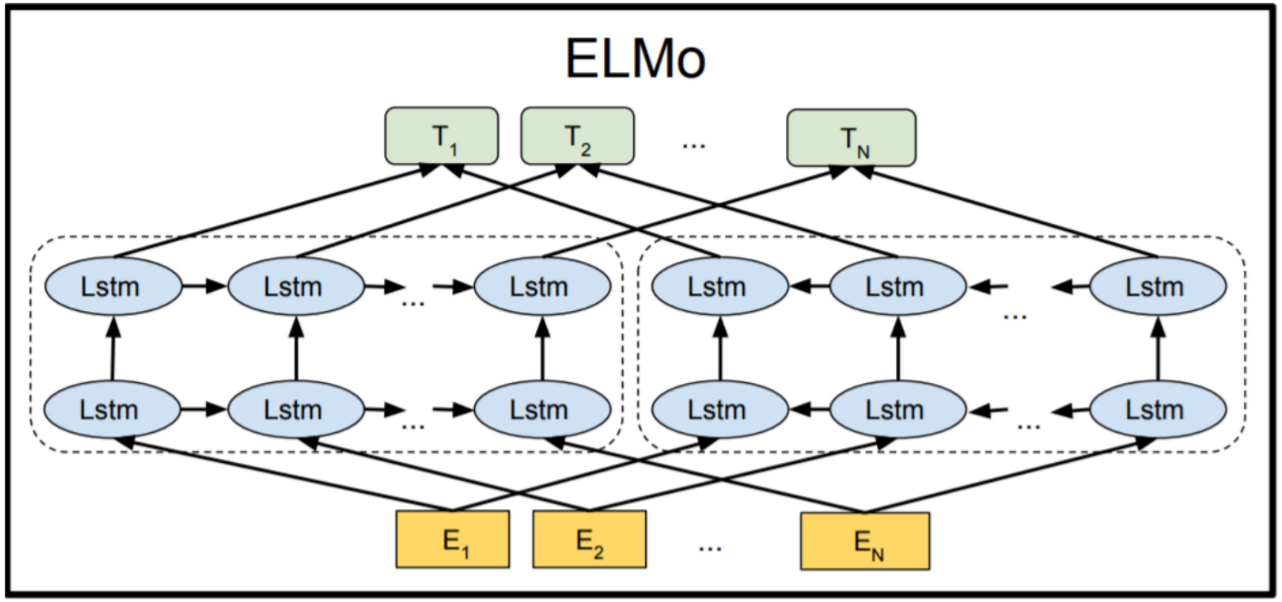
- Howard and Ruder, 2018: Universal Language Model Fine-tuning for Text Classification
- Multi-purpose: sentiment, question classification, topic classification
- 3 stages for Training
  - General domain LM training
  - Fine-tuning on target domain
  - Train actual task with gradual unfreezing



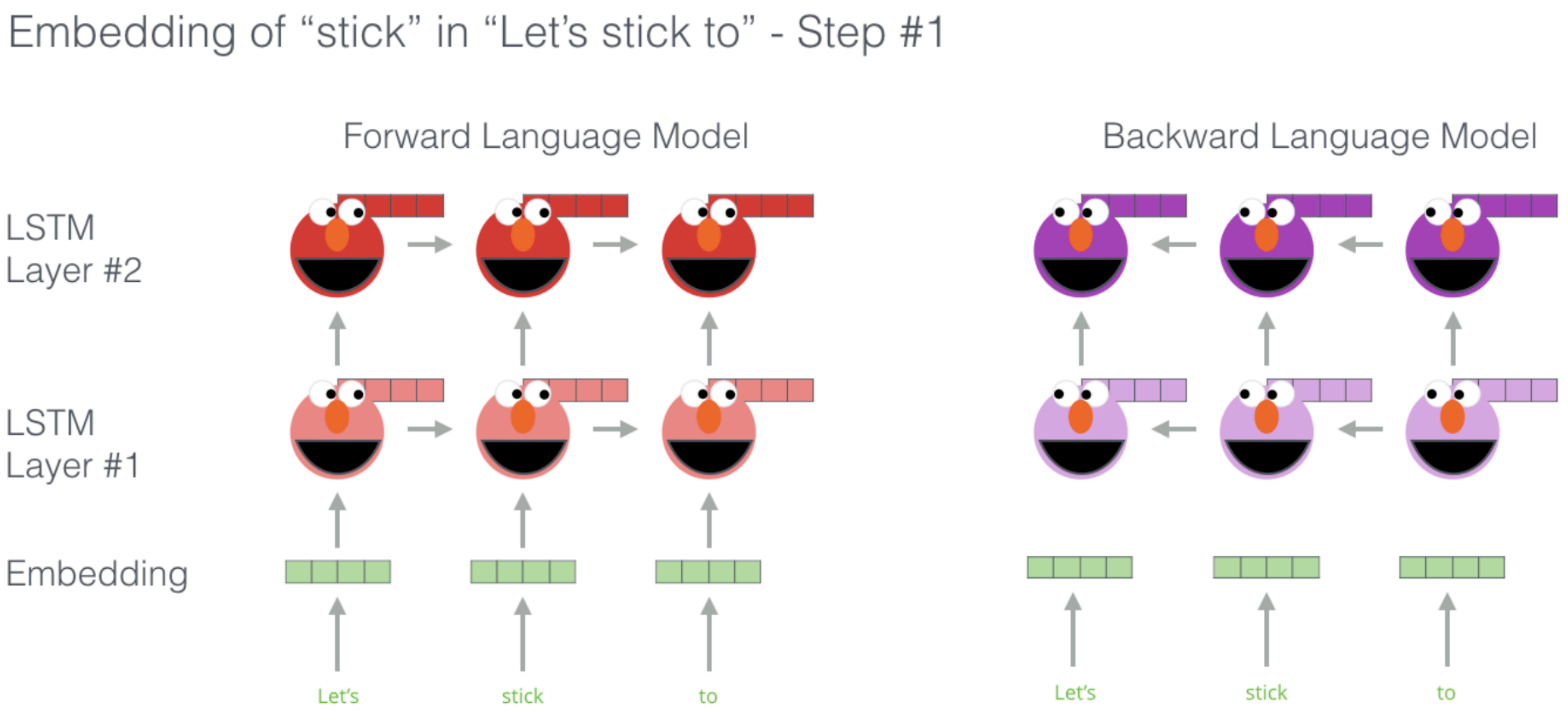
(a) LM pre-training (b) LM fine-tuning (c) Classifier fine-tuning

ELMo

- Peters et al. 2018: Deep contextualized word representations
- Train forward and backward LM to learn surroundings
- Embeddings are context-dependent (non-static)

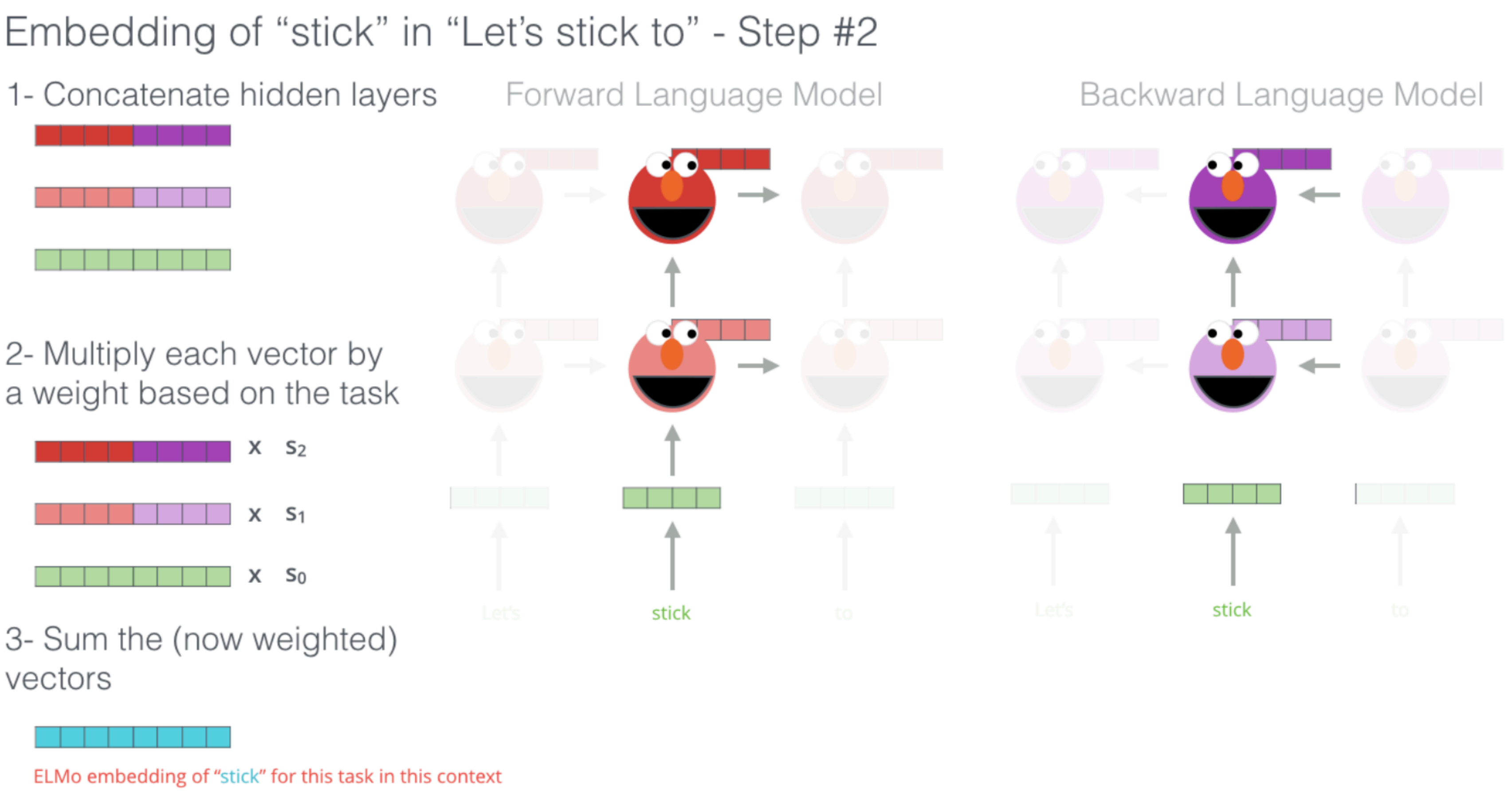


ELMo



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

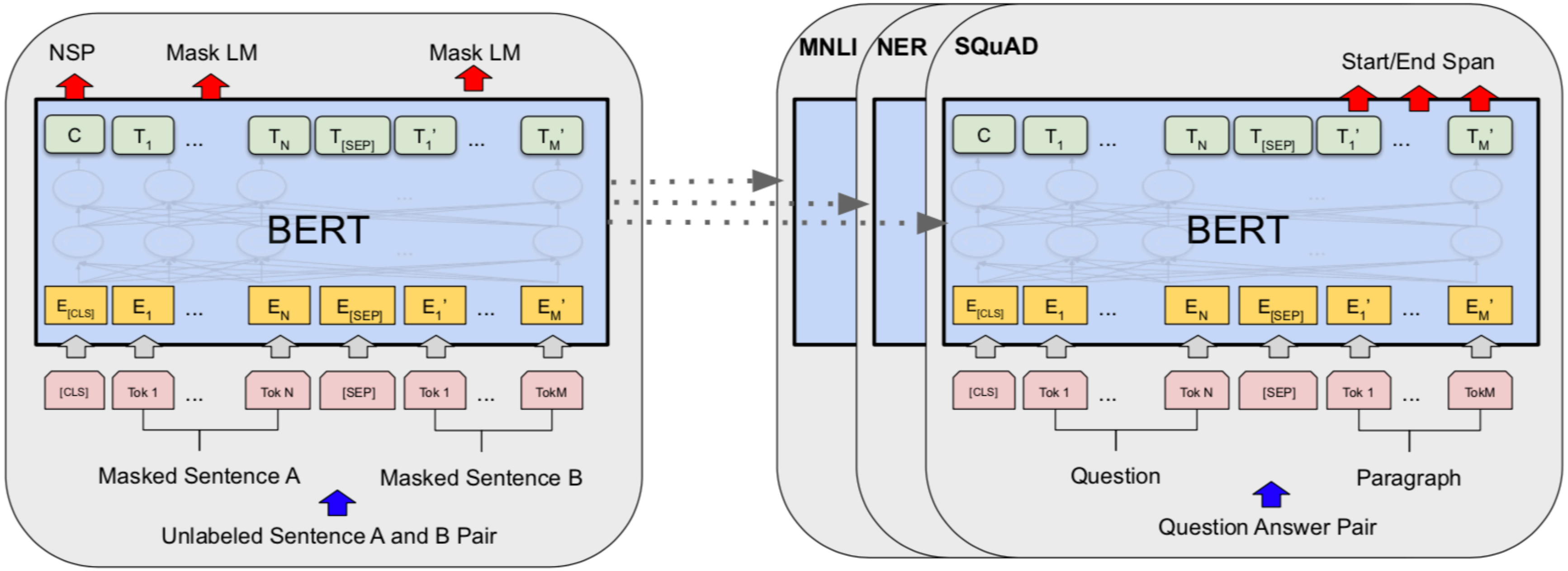
ELMo



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

BERT

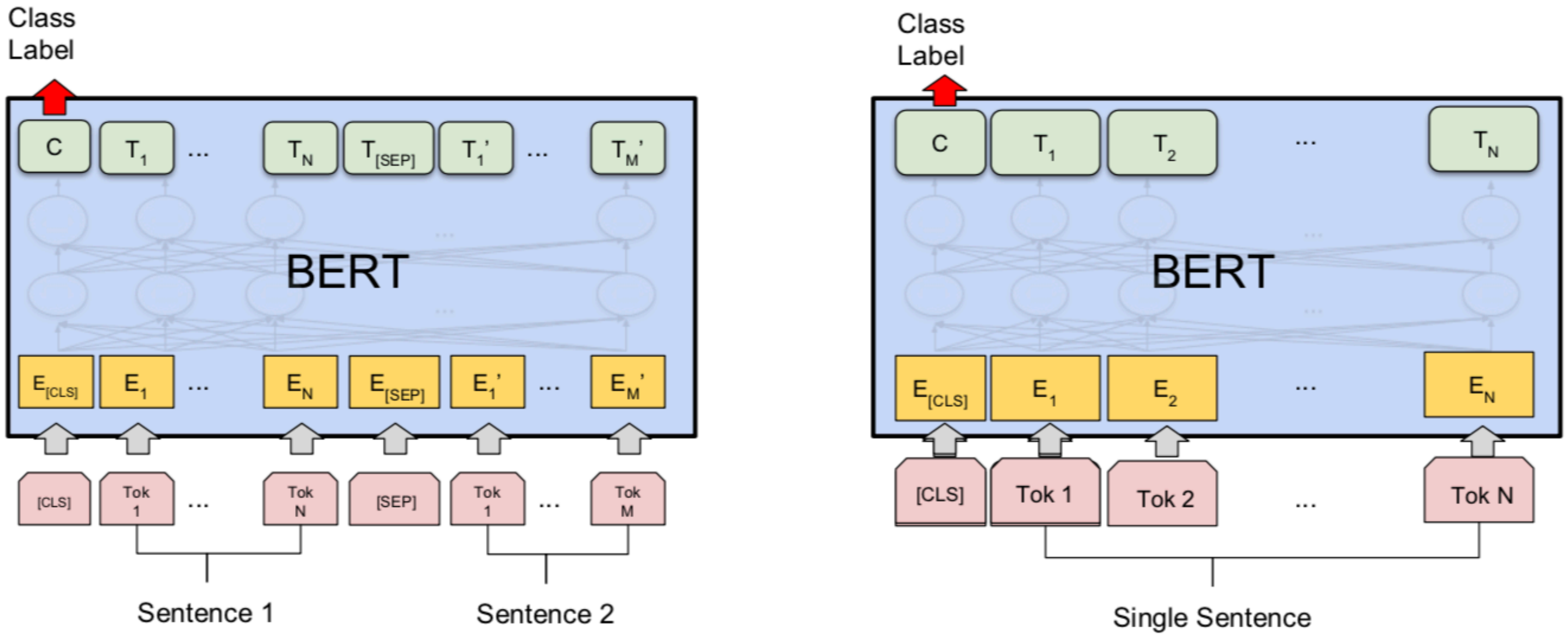
- Devlin et al. 2018: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
- Bidirectional Transformer, using masking during training



Pre-training

Fine-Tuning

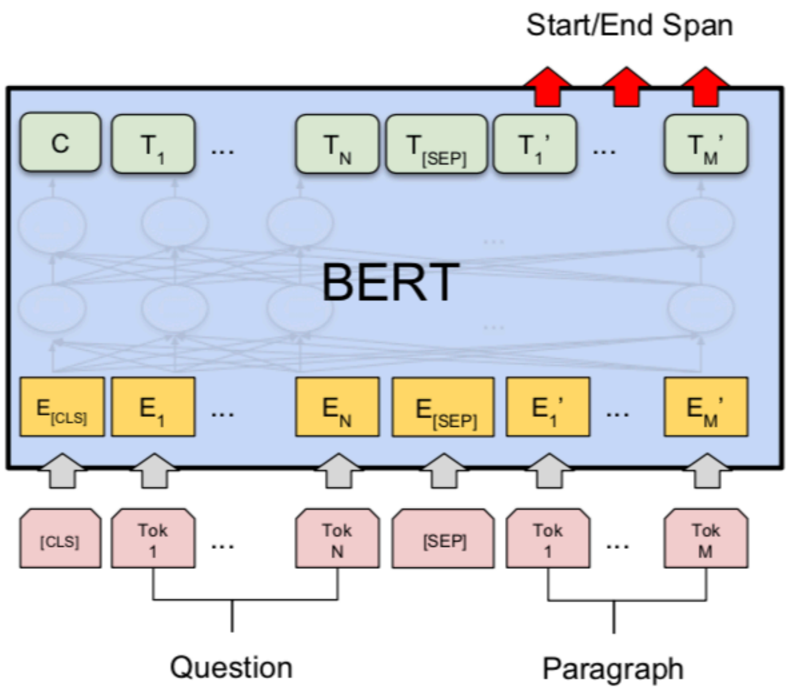
BERT – multi-task!



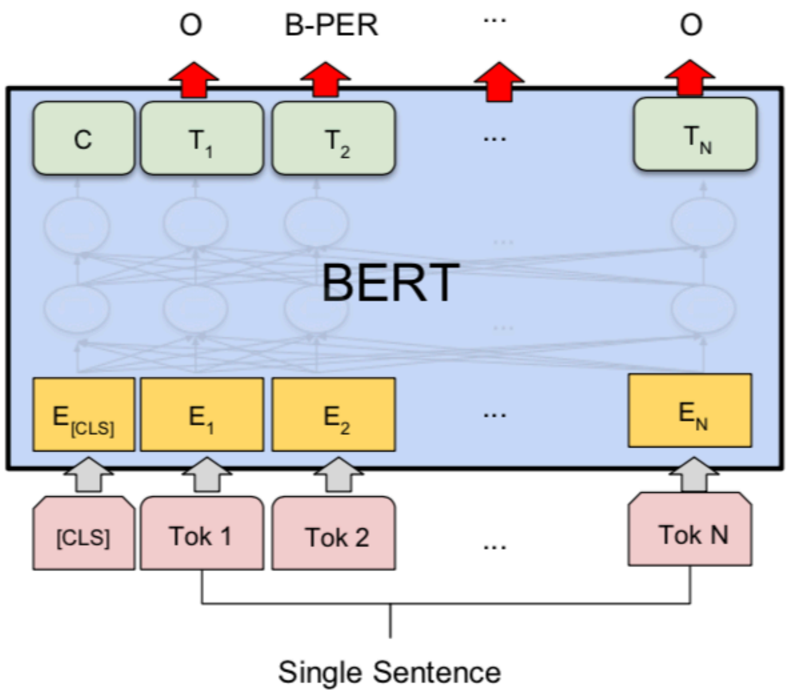
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

BERT



(c) Question Answering Tasks:
SQuAD v1.1



(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

BERT

- https://colab.research.google.com/github/tensorflow/tpu/blob/master/tools/colab/bert_finetuning_with_cloud_tpus.ipynb
- https://github.com/google-research/bert
- Use embeddings from intermediate layers
- Fine-tune to specific task in few minutes