TEXT MINING

L06. NEURAL NLP AND TRANSFER LEARNING

SUZAN VERBERNE 2019

Universiteit Leiden

QUESTIONS ABOUT WEEK 5

- What questions do you have about last week's lecture?
- What questions do you have about last week's literature (Finin et al. 2010 and McHugh, 2012)?

QUESTIONS ABOUT WEEK 5

1. What are the advantages and disadvantages of using benchmark data for training and evaluation?
2. Why should we have multiple human annotators if we create labelled data?
3. What is the purpose of chance agreement in the definition of Cohen's Kappa?
4. What is the interpretation of Kappa=0?

TODAY'S LECTURE

- Introduction to neural networks
- Neural language models
- Transfer learning with neural language models
- State of the art models: ULMFiT and BERT

AFTER THIS LECTURE…

- You can explain the basic architecture of feedforward neural networks
- You can explain the role of the hidden layer in feedforward neural networks
- You can explain how feedforward neural networks are used to train language models (word embeddings)
- You can explain how transfer learning from pre-trained language models is used for text mining tasks
- You can name two state-of-the-art methods for pre-trained language models and explain how they work on a conceptual level

INTRODUCTION TO NEURAL NETWORKS

Universiteit Leiden

Suzan Verberne 2019

NEURAL NETWORK COURSE

- https://studiegids.universiteitleiden.nl/courses/96375/deep-learning-and-neural-networks

Part One: Basics

- From statistical pattern recognition to Multi-layer Perceptron.
- Linear models: Perceptron, Logistic Regression, SVM, Cover's Theorem.
- Multi-layer Perceptron and Backpropagation.
- Alternative Loss functions, Regularization, Dropout, Batch Normalization.
- Introduction to TensorFlow.

Part Two: Deep Learning

- Convolutional Networks (LeNet, AlexNet, ResNet, UNet, ...).
- Recurrent Neural Networks (Vanilla, LSTM, GRU Networks).
- Generative Models (Variational Autoencoders, Generative Adversarial Networks).
- Deep Networks for Reinforcement Learning.

Additionally, several state-of-the-art applications of Deep Learning to image recognition, language modelling, game playing, anomaly detection, etc., will be discussed. The course consists of weekly lectures, three programming assignments (in Python/Tensorflow) and the final written exam.

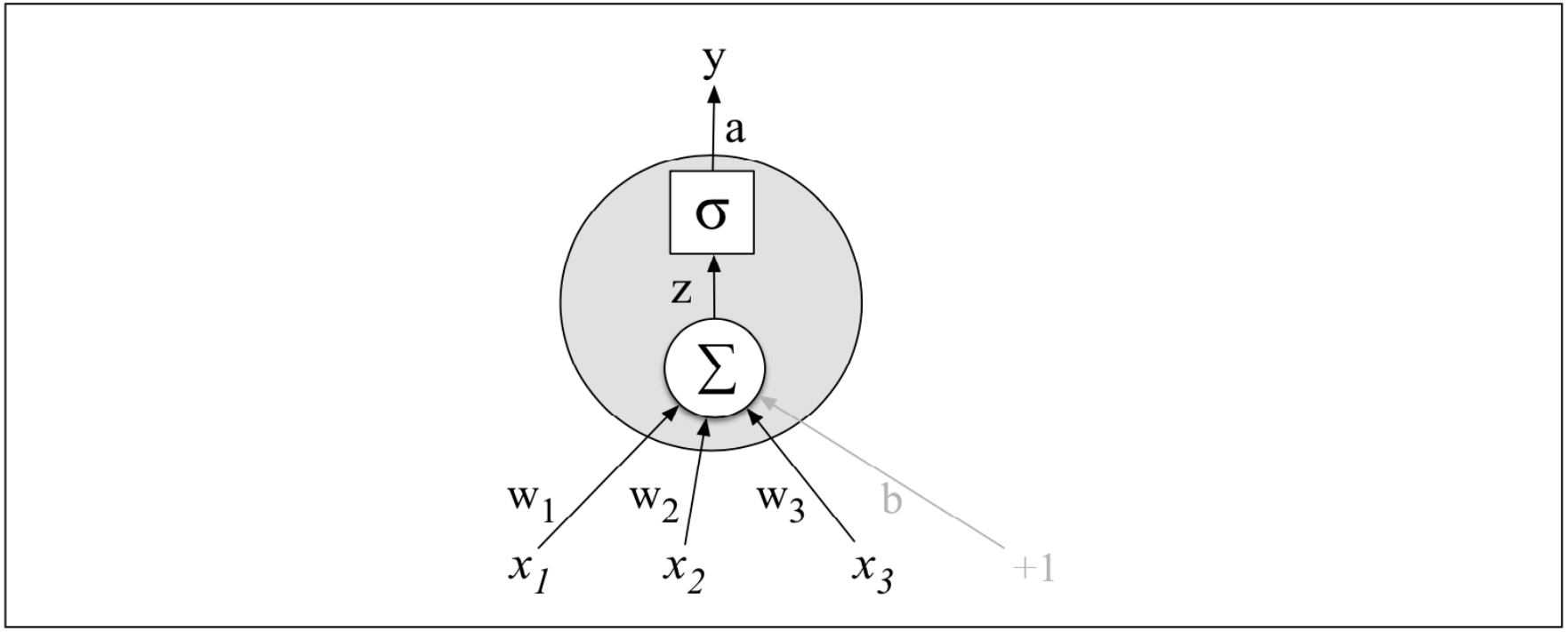
NEURAL NETWORKS

- Difference between neural network methods and traditional models:
  - Traditional classifiers use hand-derived features (although the bag of words model is already very low-level)
  - Neural networks take raw data as input and learn to induce features as part of the process of learning to classify
- Also see lecture 3: word embeddings (representation learning)

NEURAL NETWORKS FROM THE INSIDE

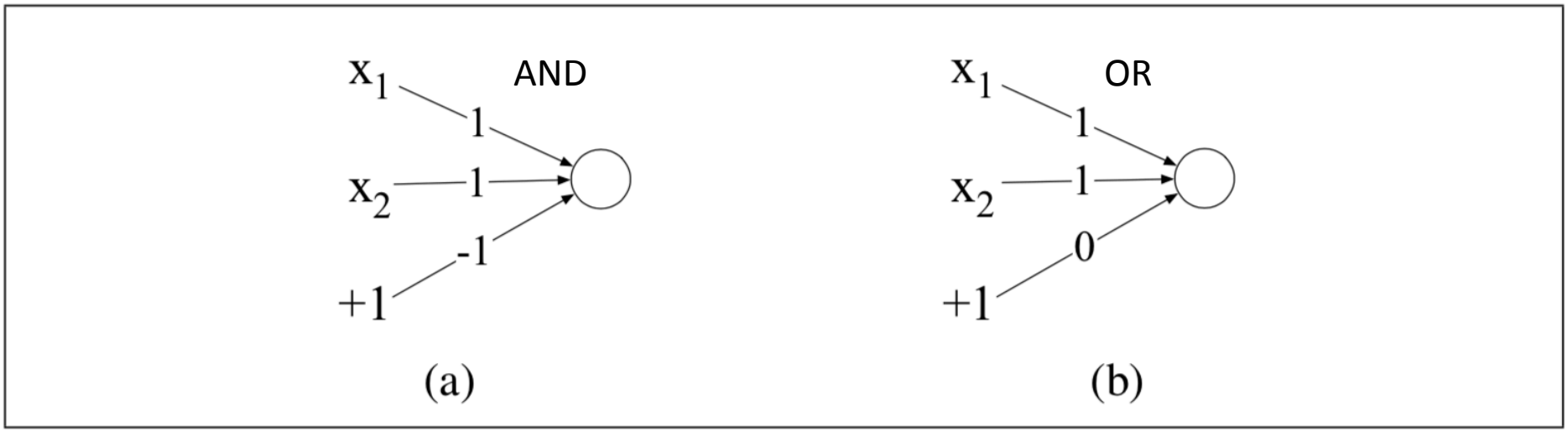
- Neural networks share much of the same mathematics as logistic regression
- But neural networks are a more powerful classifier than logistic regression:
  - multiple nodes = multiple functions = non-linearity
  - multiple layers = multiple abstractions over the input data
  - a minimal neural network can be shown to learn any function

NEURAL NETWORK UNITS (J&M 7.1)



- Figure of one single unit with one function for three input variables
- σ: non-linear activation function (sigmoid)
- Most used activation function : ReLU (Rectifier Linear Unit): y = max(x,0)

FUNCTIONS WITH ONE NEURON



- the numbers on the arrows are weights
- the input is (x1,x2)
- +1 is the bias (with weight -1 in (a) and weight 0 in (b)

THE XOR PROBLEM (J&M 7.2)

- Why one unit is not sufficient for non-linear functions ('exclusive OR')

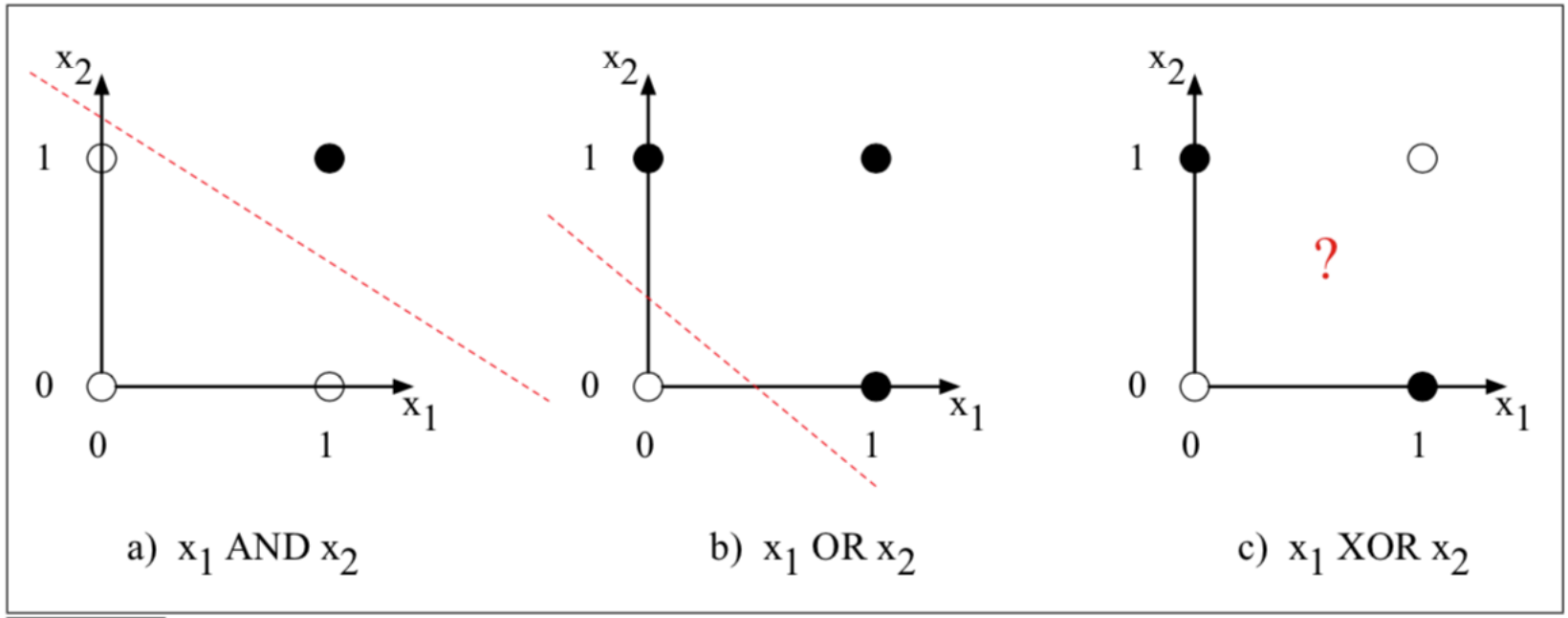
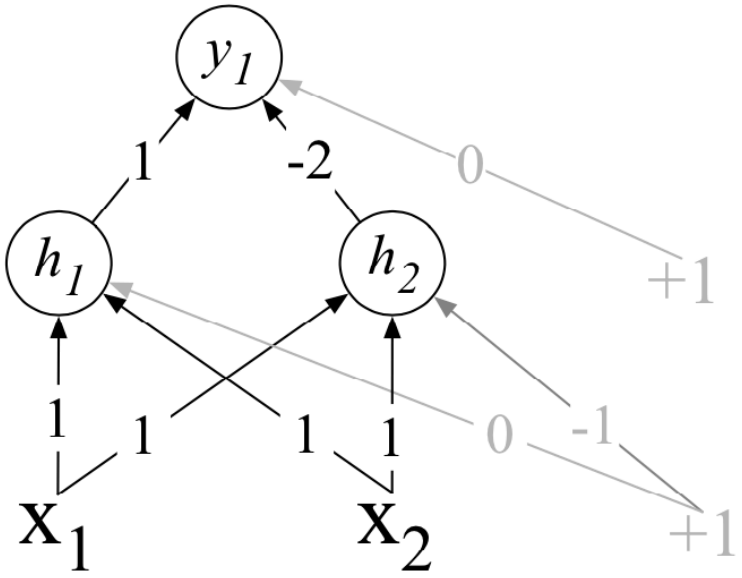


Figure 7.5 The functions AND, OR, and XOR, represented with input x_1 on the x-axis and input x_2 on the y axis. Filled circles represent perceptron outputs of 1, and white circles perceptron outputs of 0. There is no way to draw a line that correctly separates the two categories for XOR. Figure styled after Russell and Norvig (2002).

ADDING A HIDDEN LAYER

Small neural network for solving the XOR problem

Inputvector (x1,x2)
x1 ∈ {0,1}
x2 ∈ {0,1}



What is the output for each of these vectors:
[0,0]
[0,1]
[1,0]
[1,1]

[0, 0] -> [0*0 +1*0 + 0*1, 1*0 + 1*0 -1*1] = [0, -1] ~>ReLU: [0, 0] -> 1*0 − 2*0 +0*1 = 0-0+0 = 0
[0, 1] -> [1*0 +1*1 + 0*1, 1*0 + 1*1 -1*1] = [1, 0] ~>ReLU: [1, 0] -> 1*1 − 2*0 +0*1 = 1-0+0 = 1
[1, 0] -> [1*1 +1*0 + 0*1, 1*1 + 1*0 -1*1] = [1, 0] ~>ReLU: [1, 0] -> 1*1 − 2*0 +0*1 = 1-0+0 = 1
[1, 1] -> [1*1 + 1*1 + 0*1, 1*1 + 1*1 -1*1] = [2, 1] ~>ReLU: [2, 1] -> 1*2 -2*1 + 0*1 = 2-2+0 = 0

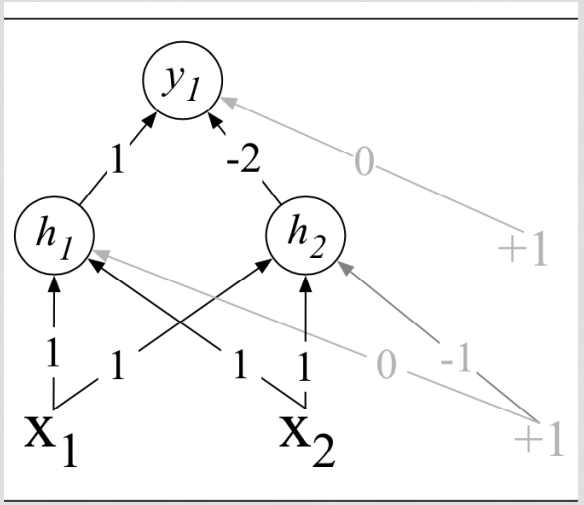
- The hidden layer is a representation of the unit, mapping [0,1] and [1,0] to the same point

LEARNING THE WEIGHTS

- In this example we manually added the weights
- For real examples the weights for neural networks are learned automatically (supervised learning using the true labels)
- By learning the weights to optimize the output classification the hidden layers will learn to form useful representations
  - Think about XOR:
    input = [x1,x2]
    output = 1 if x1 = 1 XOR x2=1; else output is 0
    learn weights so that the output is correct for all [0,0] [0,1] [1,0] [1,1]

FEEDFORWARD NETWORKS (J&M 7.3)

- A feedforward network is a multilayer network in which the units are connected with no cycles:
  - the outputs from units in each layer are passed to units in the next layer, and no outputs are passed back to lower layers
- Simple feedforward networks have three kinds of nodes: input units, hidden units, and output units
- In the standard architecture, each layer is fully-connected
  - each unit in each layer takes as input the outputs from all the units in the previous layer. Thus each hidden unit sums over all the input units



FEEDFORWARD NETWORK AS CLASSIFIER

- Binary classification (e.g. yes/no): single output node
  - *y* is the probability of postive output (yes)
- Multi-class classification: one output node for each category,
  - y is the probability of that category
- The output layer thus gives a probability distribution
- The output of the classifier z (real-valued numbers) is transformed to a probability distribution using the softmax function

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{d} e^{z_j}} \quad 1 \leq i \leq d$$

d=dimensionality
z = [0.5,3.6,-1.2]
softmax(z) = [0.043,0.949,0.0078]

EQUATIONS FOR A 2-LAYER NET

Thus we can re-represent our 2-layer net from Eq. 7.10 as follows:

$$\begin{aligned} z^{[1]} &= W^{[1]} a^{[0]} + b^{[1]} \\ a^{[1]} &= g^{[1]}(z^{[1]}) \\ z^{[2]} &= W^{[2]} a^{[1]} + b^{[2]} \\ a^{[2]} &= g^{[2]}(z^{[2]}) \\ \hat{y} &= a^{[2]} \end{aligned} \tag{7.11}$$

Note that with this notation, the equations for the computation done at each layer are the same. The algorithm for computing the forward step in an n-layer feedforward network, given the input vector $a^{[0]}$ is thus simply:

for i **in** 1..n
  $z^{[i]} = W^{[i]} a^{[i-1]} + b^{[i]}$
  $a^{[i]} = g^{[i]}(z^{[i]})$
$\hat{y} = a^{[n]}$

The activation functions $g(\cdot)$ are generally different at the final layer. Thus $g^{[2]}$ might be softmax for multinomial classification or sigmoid for binary classification, while ReLU or tanh might be the activation function $g(\cdot)$ at the internal layers.

TRAINING NEURAL NETS (J&M 7.4)

- Feedforward neural net: supervised learning with input x and correct output y
- The system produces $\hat{y}$, an estimation of the true *y*
- The goal of the training procedure is to learn parameters *W*[*i*] and *b*[*i*] for each layer *i* that make $\hat{y}$ for each training observation as close as possible to the true *y*. Needed:
  1. Loss function that models the distance between ŷ and y: cross-entropy loss
  2. Find the parameters that minimize this loss function: gradient descent optimization algorithm

$$L_{CE}(\hat{y}, y) = -\sum_{i=1}^{C} y_i \log \hat{y}_i$$

TRAINING NEURAL NETS

- Challenge: optimizing the weight parameters in all layers of the network, even though the loss is computed only at the very end of the network
- The solution is the error backpropagation algorithm
  - iterative, recursive and efficient method for computing the weights updates to improve the network
- Dropout to prevent overfitting: randomly dropping some units and their connections from the network during training

HYPERPARAMETERS

- The parameters of a neural network are the weights *W* and biases *b*; those are learned by gradient descent
- Hyperparameters of the architecture and learning set-up:
  - E.g. learning rate η, the mini-batch size, the number of layers, the number of hidden nodes per layer, the choice of activation functions, dropout rate
  - The hyperparameters are chosen by the algorithm designer; optimal values are tuned on a development set rather than by gradient descent learning on the training set

NEURAL LANGUAGE MODELS

J&M SECTION 7.5

Universiteit Leiden

Suzan Verberne 2019

RECAP OF LECTURE 3: WORD2VEC

- Word2vec learns word embeddings from raw text
- Intuition:
  - Train a classifier on a binary prediction task (on a text without labels!): "Is word *w* likely to show up near the word *bicycle*?"
  - We don't actually care about this prediction task; instead we'll take the learned classifier *weights* as the word embeddings
- The classifier is a neural network with one hidden layer
- Logistic sigmoid functions are used as activation functions in the hidden layer. The regression weights are the embeddings

LANGUAGE MODELLING

- The word prediction task is called language modelling
  - Traditional n-gram model: given the previous n words, predict the next word
  - Neural language model: feedforward neural network that takes as input a representation of the previous n words and outputs a probability distribution over possible next words
  - Neural language models can handle much longer histories, and they can generalize over contexts of similar words
- The resulting embeddings matrix is referred to as language model
- The language model can be used as representation in other tasks

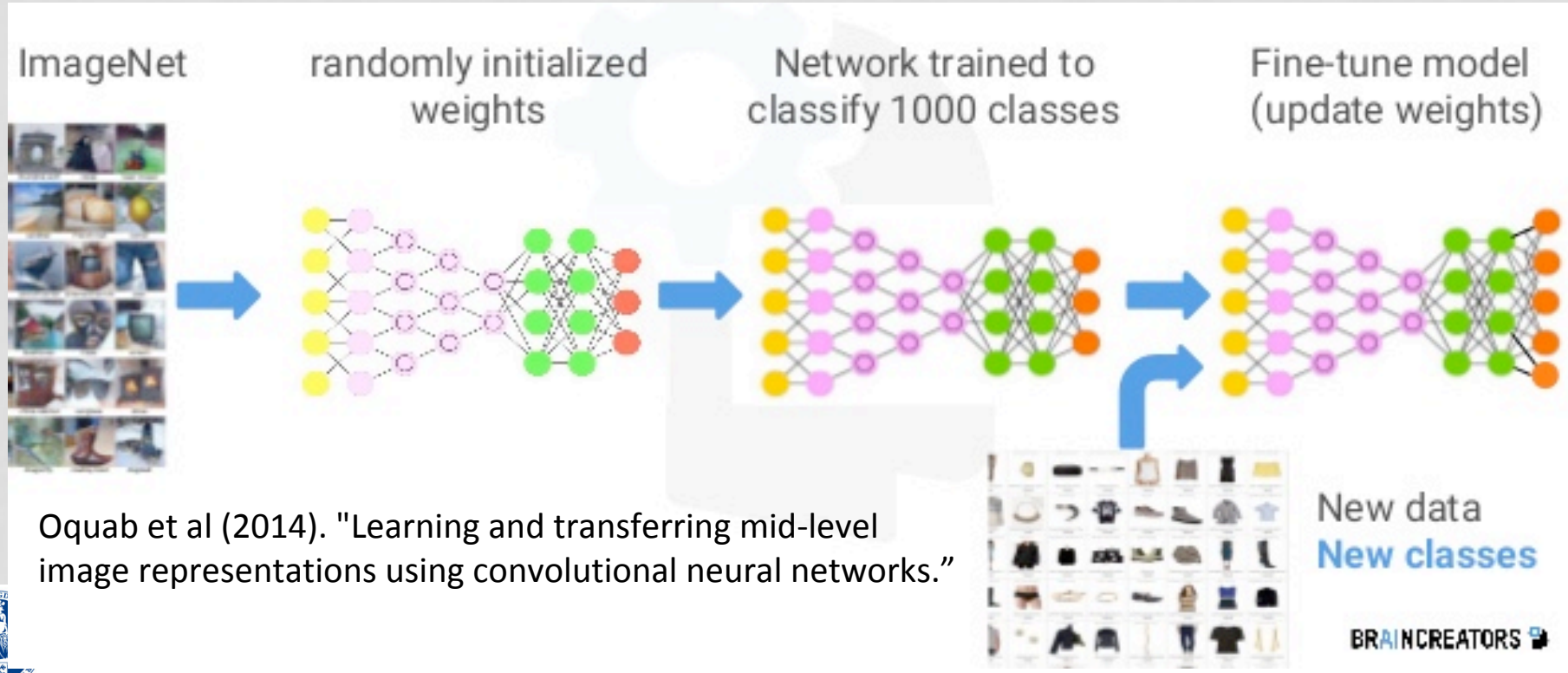
TRANSFER LEARNING

Universiteit Leiden

Suzan Verberne 2019

TRANSFER LEARNING FOR IMAGES

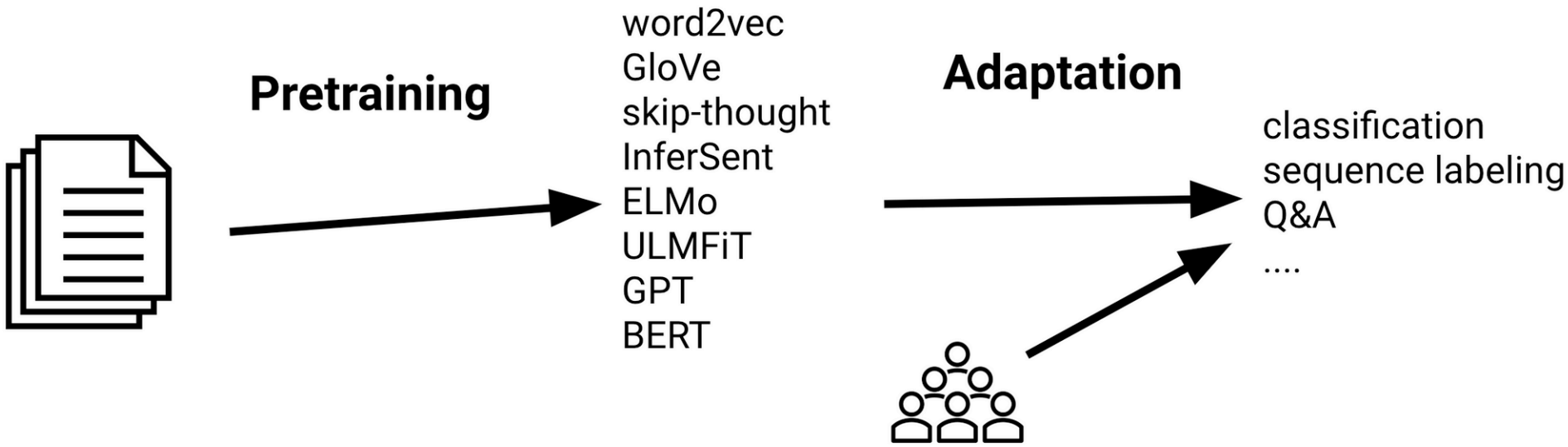
- Learn feature representations on a large labelled image set
- Then use these representations to learn to classify other classes in a much smaller set



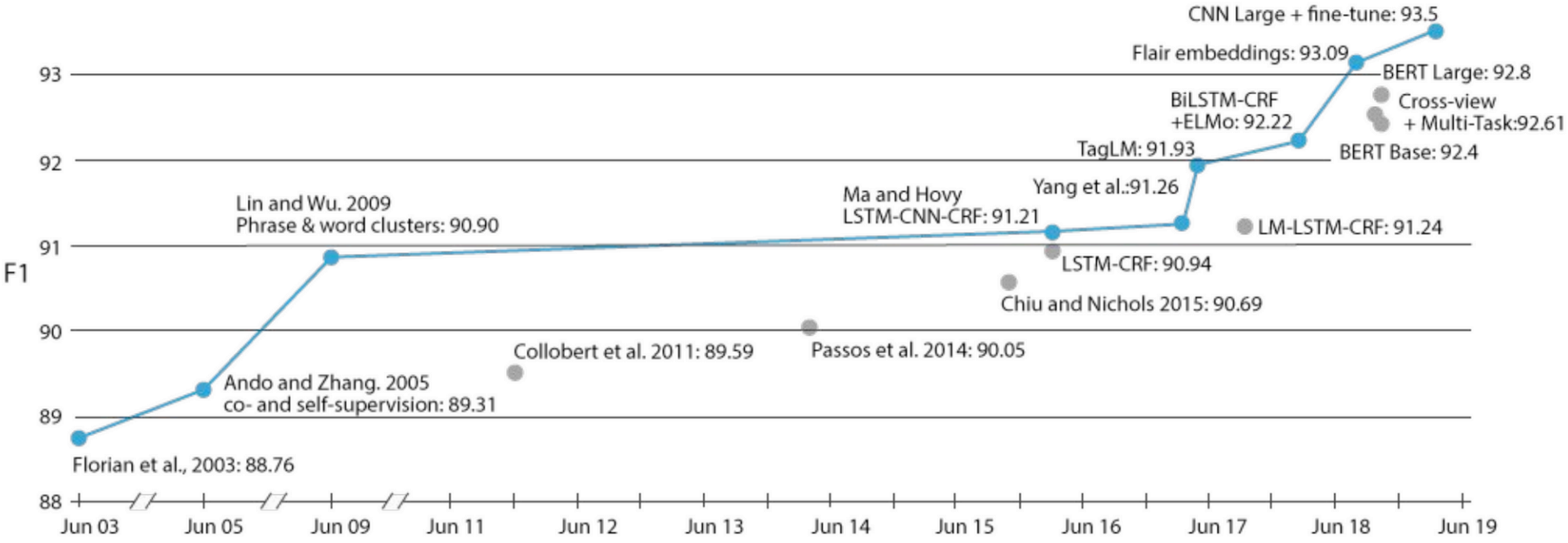
Oquab et al (2014). "Learning and transferring mid-level image representations using convolutional neural networks."

TRANSFER LEARNING FOR TEXT

- Learn feature representations (word meanings) on a large text collection (e.g. Wikipedia)
- Then use these representations to learn to classify or label text with a much smaller labelled training set



THE SUCCESS OF TRANSFER LEARNING



- Performance on Named Entity Recognition on CoNLL-2003 (English) over time

http://ruder.io/state-of-transfer-learning-in-nlp/

TRANSFER LEARNING WITH NEURAL LANGUAGE MODELS

- Inductive transfer learning: transfer the knowledge from pre-trained language models to any text mining task
- Why does this work?
  - In order to perform the language modelling task, the model is required to learn about syntax, semantics, as well as certain facts about the world
  - Given enough data, a large number of parameters, and enough compute, a model can do a reasonable job

HOW TO TRANSFER TO THE TARGET TASK

Two options when using a pre-trained model:

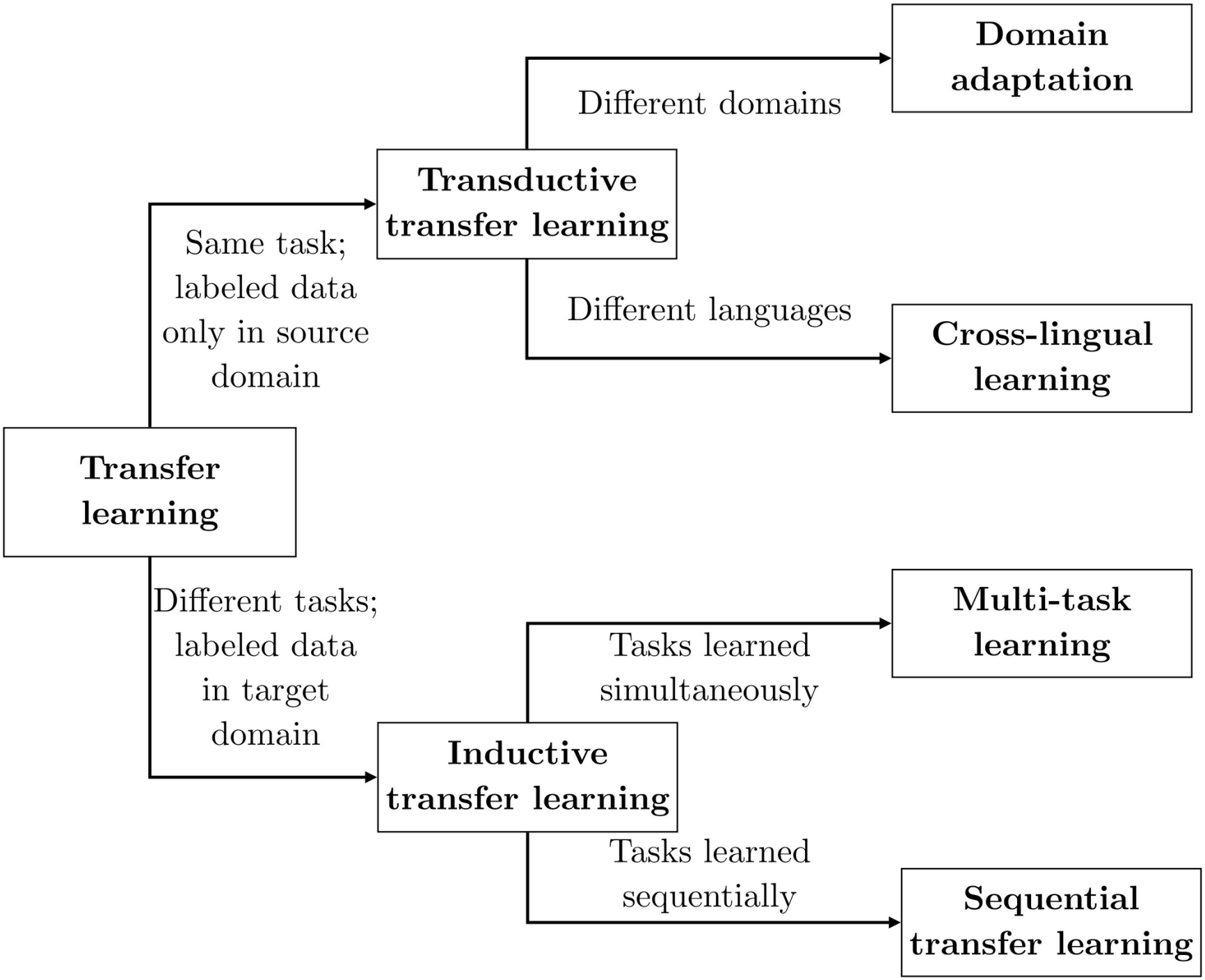
1. Direct use (feature extraction): do not change the pretrained weights
   - A linear classifier is trained on top of the pretrained representations
2. Fine-tuning: change the pretrained weights
   - The pretrained weights are used as initialization for parameters of the downstream model. The whole pretrained architecture is then trained during the adaptation phase

HOW TO TRANSFER TO THE TARGET TASK

- We want to avoid overwriting useful pretrained information ('catastrophic forgetting') and maximize positive transfer
- The more parameters you need to train from scratch the slower your training will be
- Fine-tuning often requires a more extensive hyper-parameter search, but it can give better results for large fine-tuning sets

TYPES OF TRANSFER LEARNING

- The target task is often a low-resource task
- Sequential transfer learning: If related tasks are available, we can fine-tune our model first on a related task with more data before fine-tuning it on the target task
- Multi-task fine-tuning: Alternatively, we can also fine-tune the model jointly on related tasks together with the target task. The related task can also be an unsupervised auxiliary task.


Transfer learning
Same task; labeled data only in source domain
Transductive transfer learning
Different domains
Domain adaptation
Different languages
Cross-lingual learning
Different tasks; labeled data in target domain
Inductive transfer learning
Tasks learned simultaneously
Multi-task learning
Tasks learned sequentially
Sequential transfer learning

STATE OF THE ART MODELS

NLP PROGRESS – STATE OF THE ART

http://nlpprogress.com/english/text_classification.html

http://nlpprogress.com/english/named_entity_recognition.html

AG News

The AG News corpus consists of news articles from the AG's corpus of news articles on the web pertaining to the 4 largest classes. The dataset contains 30,000 training and 1,900 testing examples for each class. Models are evaluated based on error rate (lower is better).

Model	Error	Paper / Source	Code
XLNet (Yang et al., 2019)	4.49	XLNet: Generalized Autoregressive Pretraining for Language Understanding	Official
ULMFiT (Howard and Ruder, 2018)	5.01	Universal Language Model Fine-tuning for Text Classification	Official
CNN (Johnson and Zhang, 2016) *	6.57	Supervised and Semi-Supervised Text Categorization using LSTM for Region Embeddings	Official
DPCNN (Johnson and Zhang, 2017)	6.87	Deep Pyramid Convolutional Neural Networks for Text Categorization	Official
VDCN (Alexis et al., 2016)	8.67	Very Deep Convolutional Networks for Text Classification	Non Official
Char-level CNN (Zhang et al., 2015)	9.51	Character-level Convolutional Networks for Text Classification	Non Official

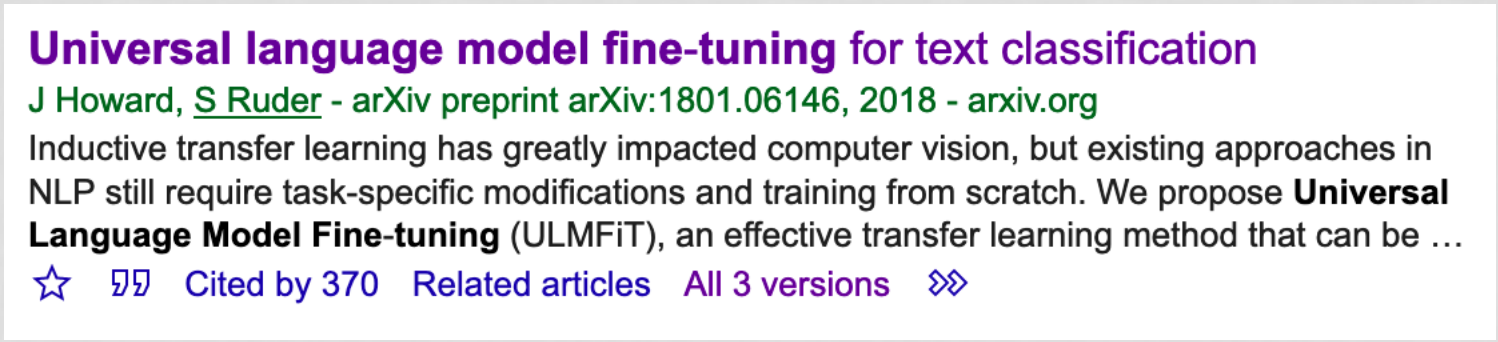
CoNLL 2003 (English)

The CoNLL 2003 NER task consists of newswire text from the Reuters RCV1 corpus tagged with four different entity types (PER, LOC, ORG, MISC). Models are evaluated based on span-based F1 on the test set. ♦ used both the train and development splits for training.

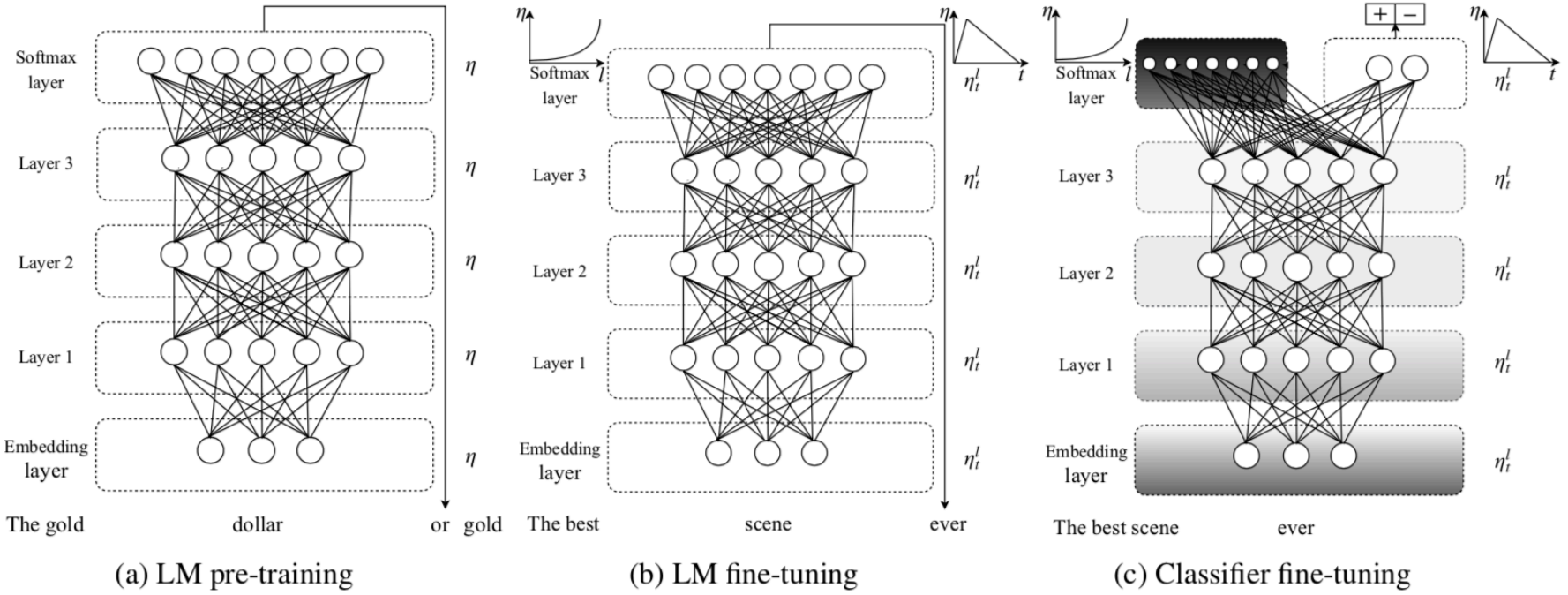
Model	F1	Paper / Source	Code
CNN Large + fine-tune (Baevski et al., 2019)	93.5	Cloze-driven Pretraining of Self-attention Networks	
Flair embeddings (Akbik et al., 2018)♦	93.09	Contextual String Embeddings for Sequence Labeling	Flair framework
BERT Large (Devlin et al., 2018)	92.8	BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding	
CVT + Multi-Task (Clark et al., 2018)	92.61	Semi-Supervised Sequence Modeling with Cross-View Training	Official
BERT Base (Devlin et al., 2018)	92.4	BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding	
BiLSTM-CRF+ELMo (Peters et al., 2018)	92.22	Deep contextualized word representations	AllenNLP Project AllenNLP GitHub

ULMFIT

- Howard, Jeremy, and Sebastian Ruder. "Universal Language Model Fine-tuning for Text Classification." In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 328-339. 2018.



ULMFIT ARCHITECTURE



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

Figure 1: ULMFiT consists of three stages: a) The LM is trained on a general-domain corpus to capture general features of the language in different layers. b) The full LM is fine-tuned on target task data using discriminative fine-tuning ('*Discr*') and slanted triangular learning rates (STLR) to learn task-specific features. c) The classifier is fine-tuned on the target task using gradual unfreezing, '*Discr*', and STLR to preserve low-level representations and adapt high-level ones (shaded: unfreezing stages; black: frozen).

SUCCESS OF ULMFIT

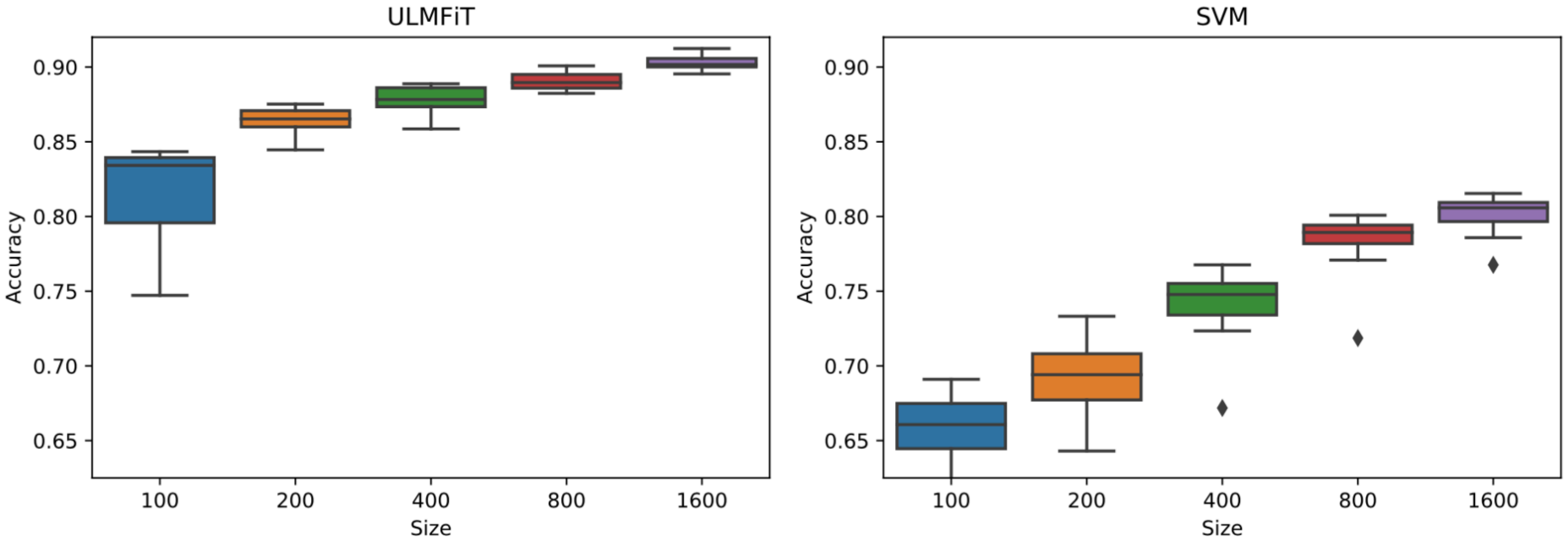


Figure 1: Results for ULMFiT (a) and SVM (b) in terms of accuracy on the test set with varying training set sizes. The boxes represent the deviation among the random subsample per training set size.

BERT

- Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4171-4186. 2019.

Bert: Pre-training of deep bidirectional transformers for language understanding

J **Devlin**, MW Chang, K Lee, K Toutanova - arXiv preprint arXiv …, 2018 - arxiv.org

We introduce a new language representation model called BERT, which stands for Bidirectional Encoder Representations from Transformers. Unlike recent language representation models, BERT is designed to pre-train deep bidirectional representations by …

☆ 99 Cited by 1717 Related articles All 7 versions »

BERT INTUITION

- Language modelling tasks:
  - Predicting randomly masked words in context (bi-directional)
    - To capture the meaning of words
  - Next-sentence classification
    - To capture the relationship between sentences

BERT ARCHITECTURE

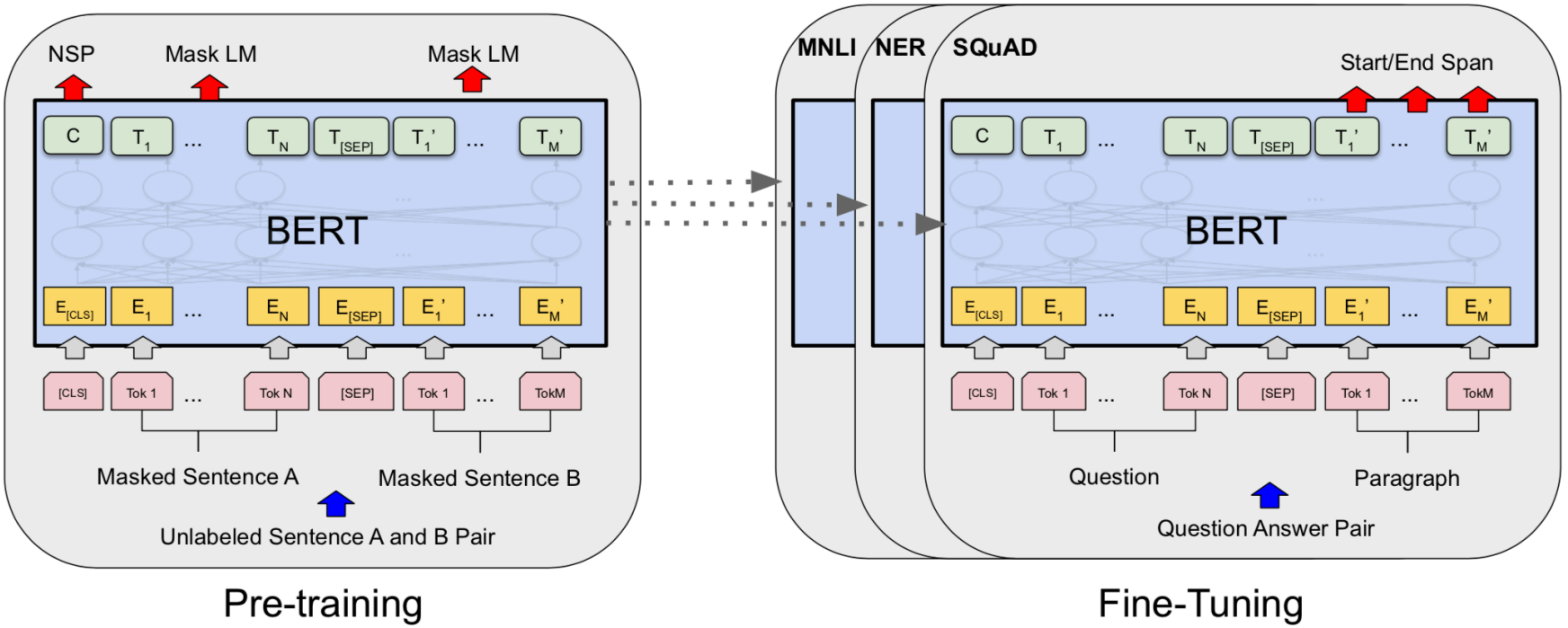
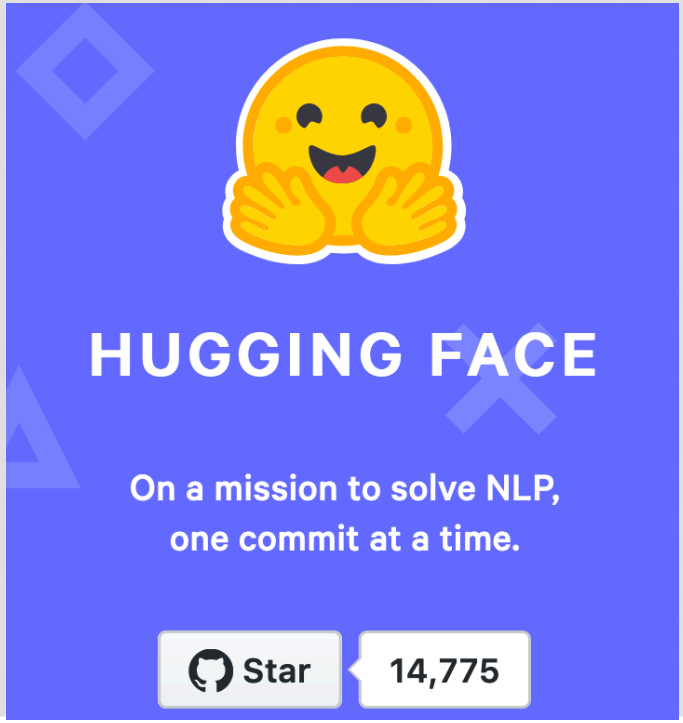


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. `[CLS]` is a special symbol added in front of every input example, and `[SEP]` is a special separator token (e.g. separating questions/answers).

SUCCESS OF BERT

- Achieves state-of-the-art results on a large range of tasks and even in a large range of domains
- Pre-trained models can easily be fine-tuned
- Domain-specific pre-trained BERT models: bioBERT, sciBERT, clinicalBERT
- There exist multiple implementations of the BERT paradigm
- Have a look at https://huggingface.co/, also including DistilBERT ("Smaller, faster, cheaper, lighter")



CHALLENGES OF STATE-OF-THE-ART METHODS

- Heavy pre-training: takes time and computing power needed
- Risk of overfitting (dropout or other forms of regularization needed)
- Parameter freezing/updating/forgetting
- Hyperparameter tuning:
  - optimization on development set (takes time)
  - adoption of hyperparameters from pre-training task
- Interpretation/explainability: additional effort (attention layer)

CHALLENGES OF STATE-OF-THE-ART METHODS

- Therefore, traditional models are also still commonly used in classification tasks
- Sklearn can efficiently process text data into a term-document matrix and use a range of classifiers for the learning task
  - LinearSVC and NaïveBayes are still suitable classifiers for a lot of tasks
  - They can run on one CPU in a short amount of time
- The research community has completely moved to neural models, but in applied contexts you will still find the traditional models

CONCLUSIONS

SUZAN VERBERNE 2019

Universiteit Leiden

HOMEWORK

- Read:
  - Jurafsky & Martin chapter 7. Neural Networks and Neural Language Models (you can skip 7.4.3 and 7.4.4)
  - Note that it refers to chapter 5 (Logistic Regression) often. If you are not familiar with Logistic Regression, be referred to that chapter as well
- Optional exercise:
  - Tutorial: Finetuning with BERT. Instructions can be found on Blackboard (Course documents → Week 6)
- Work on the first hand-in assignment (slides last week & Blackboard) on text categorization. Deadline: October 20

AFTER THIS LECTURE…

- You can explain the basic architecture of feedforward neural networks
- You can explain the role of the hidden layer in feedforward neural networks
- You can explain how feedforward neural networks are used to train language models (word embeddings)
- You can explain how transfer learning from pre-trained language models is used for text mining tasks
- You can name two state-of-the-art methods for pre-trained language models and explain how they work on a conceptual level