

TEXT MINING

L07. NEURAL NLP AND TRANSFER LEARNING

SUZAN VERBERNE 2021

TODAY'S LECTURE

- Quiz about week 5
- Assignment 1
- Introduction to neural networks
- Neural architectures for sequential data: RNNs and Transformers
- State-of-the-art model: BERT
- Transfer learning with neural language models



QUIZ ABOUT WEEK 6

- What are the strengths of CRF compared to a HMM for sequence labelling?
 - a. It is multi-layered
 - b. It can use features to represent tokens
 - c. It optimizes the sequence as a whole
 - d. It takes a larger context

QUIZ ABOUT WEEK 6

- Why are part-of-speech tags informative for Named Entity Recognition?
 - a. Because only nouns can be entities
 - b. Because some word categories are more likely to be (part of an) entity
 - c. Because the occurrence of subsequent entities in a row

QUIZ ABOUT WEEK 6

- How would you approach the recognition of dates and times from emails?
 - a. With a regular expression
 - b. With labelled data and CRF
 - c. With labelled data and BERT

QUIZ ABOUT WEEK 6

- We have a text collection manually labelled with 1000 entities. Our BERT model identified 800 entities, 600 of which were also in the manual set. What is the recall?
- a. $800/1000$
 - b. $600/800$
 - c. $600/1000$

ASSIGNMENT 1 – TEXT CLASSIFICATION

- Grading (ongoing):

- 5 criteria; max 2 points per criterion

1. General: length correct (2-3 pages) and proper writing and formatting
2. Experiments on 20 newsgroups
3. Results table for 3 classifiers x 3 feature weights (counts, tf, and tf-idf)
4. Results for a. lowercase; b. stop_words; c. analyzer (in combination with ngram_range); d. max_features
5. Brief discussion on which classifier performs the best, with which features

ASSIGNMENT 1 – TEXT CLASSIFICATION

Some notes about your submissions:

- Please mention in your introduction what task you are addressing (text classification in 20 news categories)
- Please report what settings you compared and what the results were
- Please summarize the most important results in a neatly formatted table
 - It is good practice to highlight the best-in-class performances in boldface
- Grid search is a form of optimization should not be done on the test set (but on a separate development set)
- *Never* copy text from (web) sources or other students. This is considered plagiarism and will be reported to the Board of Examiners.

EXAMPLE REPORT

TM A1 - Text categorization

Job Mooij, Adéla Šterberová

1 Introduction

The main goal of this assignment is to perform a text categorization benchmark with scikit-learn on provided data. [3] Different classification methods and evaluations are compared against each other, to understand what different feature weights can be used for text classification and their advantages. The main features that are compared in this assignment are term count, term frequency (tf) and inverse document frequency (idf).

2 Background

To make a classification of text documents, we first need to preprocess them. It usually includes tokenization, removing stop words, lemmatization, or other processes where the result is each document's bag of words (terms). These terms are features for classification. We can use different methods to determine their weights. The most simple is term count $tc_{t,d}$ of term t in document d , which is the number of term occurrences in document d . Term frequency tf is also a number of occurrences concerning the length of the document given by:

$$tf_{t,d} = \begin{cases} 1 + \log_{10}(tc_{t,d}) & \text{if } tc_{t,d} > 0 \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

The document frequency df_t of term t expresses the informativeness of t . It is the number of documents where t occurs. And inverse document frequency idf_t can be expressed as

$$idf_t = \log_{10} \frac{N}{df_t}. \quad (2)$$

Where N refers to the total amount of documents. Finally, $tf-idf$ is defined as:

$$tf-idf = tf_{t,d} \cdot idf_t. \quad (3)$$

The Naive Bayes classifier (NB) is based on probabilistic and is given by the equation:

$$c^* = \arg \max_{c \in C} P(c|d) = \arg \max_{c \in C} \frac{P(d|c)P(c)}{P(d)} \quad (4)$$

This classifier iterates through all classes $c \in C$ and searches for the highest probability of document d belonging to that class. [4]

A Support Vector Machine (SVM) is a supervised learning model where the training data needs to be labeled to start. This labeled data is used to create a hyperplane in n -dimensions to classify the new datapoints the best way possible. This can be done by using the dot products which is very cheap to use in terms of calculations, which results to a very short runtime.

Logistic Regression (LR) is used to classify (multiple) labels, the sum of the probability of all labels is equal to 1. This algorithm uses the weighted combination of input features and passes them through a sigmoid function. This sigmoid function transforms any input number to a number between 0 and 1, this method is also very quick to use which results in short runtimes.

3 Experiments

The text classification was performed on the 20 Newsgroups dataset, which is a collection of around 18,000 newsgroup documents. [4] Documents are distributed evenly across 20 different newsgroups, each corresponding to a different topic [2]. The three classifiers were all executed with the default parameters to determine baseline results. Every classifier was run three times, to get the results of the three different features.

Table 1 shows the results of these 9 experiments. Based on the f1-score, the Logistic Regression classifier with the $tf-idf$ approach gives the best results. However, these results weren't final as experimenting with the parameters of the CountVectorizer function can yield better results on this dataset.

A grid search was performed on the parameters suitable for CountVectorizer() function. The following values for three parameters were examined: *lowercase*: {True, False}, *stop_words*: {None, 'english'}, *analyzer*: {'word', 'char', 'char_wb'} and *max_features*: {10000, 20000, 50000, 60000, 70000, 80000, 100000}. The parameters that yield the best results were *lowercase*: True, *stop_words*: 'english', *analyzer*: 'word' and *max_features*: 60000.

As this combination of parameters gave the best f1-score, they were used with rerunning the same 9 experiments which gave the results given in Table 2. Almost every experiment yields better results with these parameters, only the SVM with $tf-idf$ as a feature has better results with the default parameters. But other than that, the f1-score improves with all other classifiers. Just like in Table 1, the highest f1-score is given by the Logistic Regression when using $tf-idf$ as a feature.

Table 1: Results of the 9 experiments executed with default parameters. These results act as a baseline for the further analysis.

Classifier	Feature	Precision	Recall	f1-score
NB	Count	76.17	77.28	75.11
	TF	78.55	70.53	69.21
	TF-IDF	82.19	77.39	76.84
SVM	Count	76.35	75.21	75.24
	TF	77.85	76.98	76.30
	TF-IDF	82.80	82.49	81.91
LR	Count	77.71	77.28	77.29
	TF	78.26	78.32	78.18
	TF-IDF	83.62	83.54	83.54

Table 2: Results of the 9 experiments executed with the combination of parameters which gave the best results on a Logistic Regression, using $tf-idf$ as a feature.

Classifier	Feature	Precision	Recall	f1-score
NB	Count	80.62	80.58	78.68
	TF	81.20	79.25	78.17
	TF-IDF	83.41	81.88	81.20
SVM	Count	77.90	77.62	77.49
	TF	79.40	79.08	78.33
	TF-IDF	82.55	82.25	81.67
LR	Count	80.17	79.49	79.65
	TF	81.06	80.75	80.77
	TF-IDF	83.91	83.70	83.68

4 Discussion and conclusions

From the conducted experiments, we can conclude that the most efficient classifier was the Logistic Regression with the use of $tf-idf$ as a feature. It is probably because $tf-idf$ holds more information about the word in the document with respect to other documents than only tf or the counts. Therefore using its weight can contribute to better classification. According to the literature [5], Logistic Regression can give the best results even on very small datasets. This might be the cause for why this method consequently gives the best results.

EXAMPLE RESULTS TABLES

	MultiNB			SGD			LinearSVM		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
Counts	0.76	0.77	0.75	0.77	0.76	0.76	0.79	0.79	0.78
TF	0.79	0.71	0.69	0.81	0.81	0.80	0.83	0.83	0.82
TF-IDF	0.82	0.77	0.77	0.85	0.85	0.85	0.85	0.85	0.85

	Naive Bayes			SGD			SVC		
	count	tf	tf-idf	count	tf	tf-idf	count	tf	tf-idf
Precision	0.7622	0.7924	0.8255	0.7487	0.7752	0.8265	0.7318	0.7592	0.8360
Recall	0.7636	0.6822	0.7565	0.7247	0.7574	0.8122	0.7272	0.7492	0.8271
F1	0.7451	0.6728	0.7558	0.7239	0.7541	0.8103	0.7271	0.7498	0.8287

sklearn.metrics.classification_report has a parameter 'digits' for the number of decimals. General: use 3 digits: 0.763 or 76.3%

INTRODUCTION TO NEURAL NETWORKS

J&M CHAPTER 7

NEURAL NETWORK COURSES

- Introduction to Deep Learning (now)
<https://studiegids.universiteitleiden.nl/courses/105069/introduction-to-deep-learning>

Part Two: DeepLearning

- Convolutional Networks; key architectures and applications
- Recurrent Neural Networks; LSTM and GRU Networks; Word Embeddings
- Autoencoders
- GAN Networks
- Deep Learning for Reinforcement Learning

- Advances in Deep Learning (spring)
<https://studiegids.universiteitleiden.nl/courses/105089/advances-in-deep-learning>

INTRODUCTION

Two options:

- If you are familiar with feedforward neural networks and the role of the hidden layer, I have an [exercise](#) for you (the XOR problem)
- If your knowledge is a bit rusty, you can listen to my [explanation](#)

EXERCISE: THE XOR PROBLEM

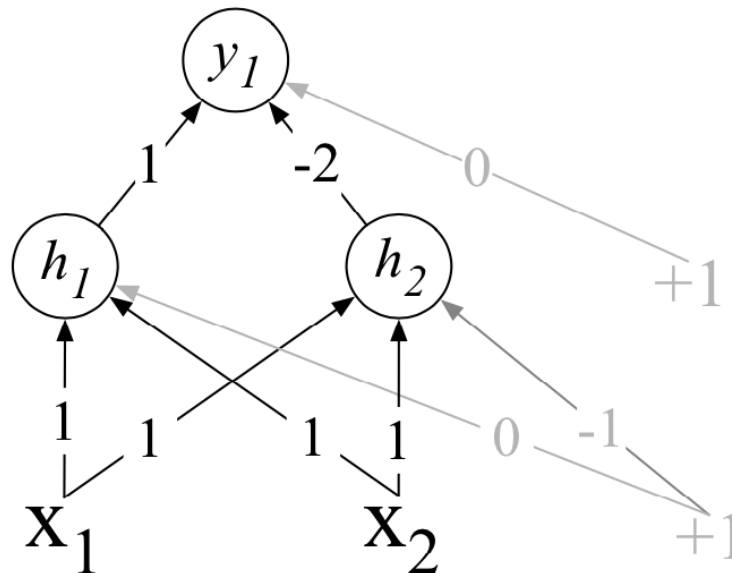
- The XOR (exclusive OR) function is a non-linear function
- Unlike AND and OR it cannot be solved with one neuron
- A feedforward network with one hidden layer:

Small neural network
that solves the XOR
problem

Inputvector (x1,x2)

$x_1 \in \{0,1\}$

$x_2 \in \{0,1\}$



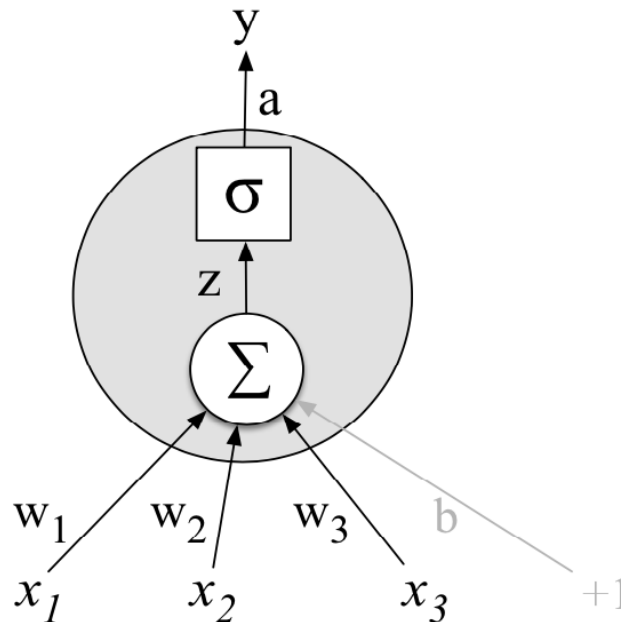
What is the
output for each
of these vectors:
[0,0]
[0,1]
[1,0]
[1,1]

INTRODUCTION

- 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

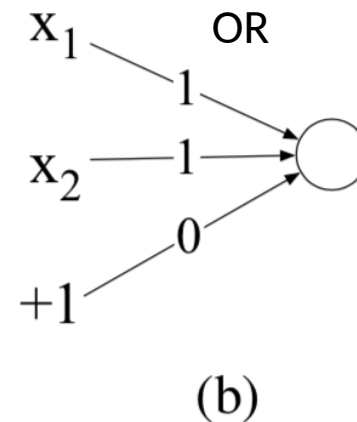
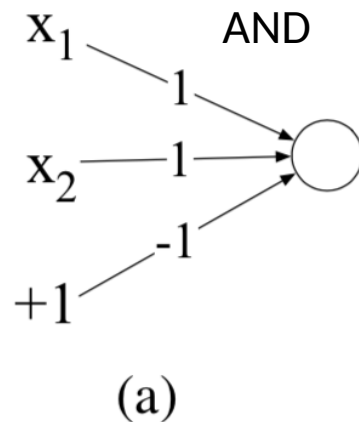
* If you are not familiar with Logistic Regression you can read J&M chapter 5

NEURAL NETWORK UNITS (J&M 7.1)



- Figure of one single unit with one function for three input variables (x_1, x_2, x_3)
- σ : non-linear activation function
- Commonly used activation function : ReLU (Rectifier Linear Unit): $y = \max(x, 0)$

FUNCTIONS WITH ONE NEURON



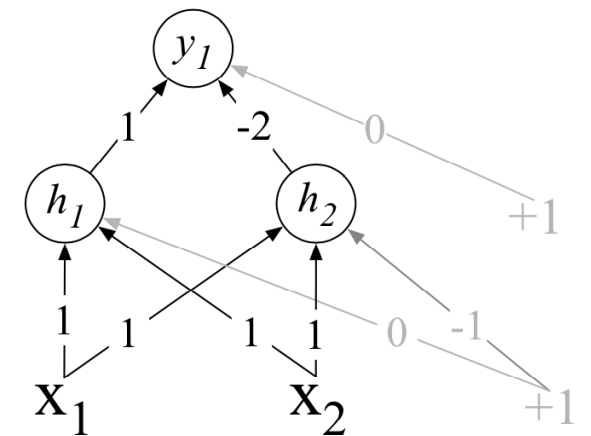
- the numbers on the arrows are **weights**
- the **input vector** is (x_1, x_2)
- $+1$ is the **bias** (with weight -1 in (a) and weight 0 in (b))

LEARNING THE WEIGHTS

- In this example we manually added the weights
- When trained on data, 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

FEEDFORWARD NETWORKS (J&M 7.3)

- A feedforward network is a multilayer network in which the units are connected with **no cycles**
- Simple feedforward networks have three kinds of nodes: input units, hidden units, and output units
- In the standard architecture, each layer is **fully-connected**



FEEDFORWARD NETWORK AS CLASSIFIER

- Binary classification (e.g. yes/no): single output node
 - y is the probability of positive 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

FEEDFORWARD NETWORK AS CLASSIFIER

- To get the probability distribution:
- The output of the classifier z (real-valued numbers) is transformed to a probability distribution using the **softmax function**

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

Example:

$z = [0.5, 3.6, -1.2]$

$\rightarrow \text{softmax}(z) = [0.043, 0.949, 0.0078]$

TRAINING NEURAL NETS

- 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 weights $W[i]$ and bias $b[i]$ for each layer i that make \hat{y} for each training observation as close as possible to the true y
- Find the parameters that minimize this loss function: **gradient descent optimization algorithm**

(see Introduction to Deep Learning, or J&M section 7.6)

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

THE XOR PROBLEM (J&M 7.2)

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

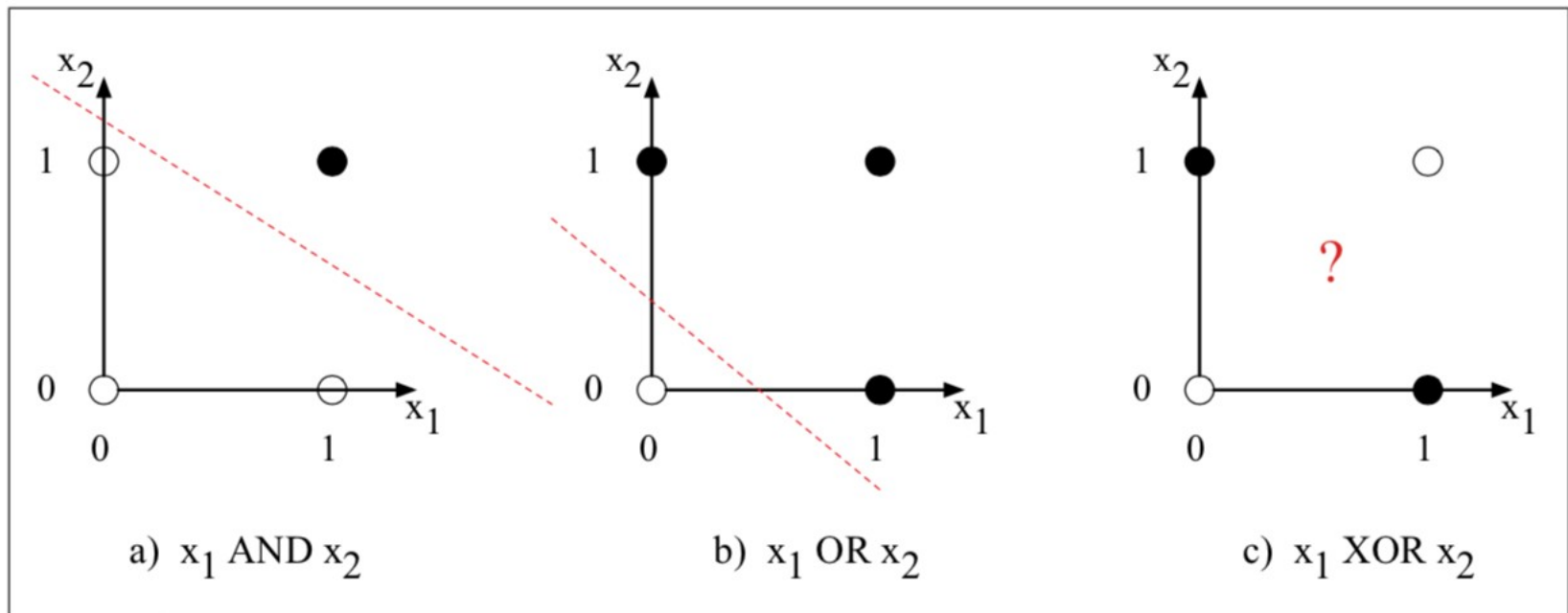


Figure 7.5 The functions AND, OR, and XOR, represented with input x_1 on the x-axis and input x_2 on the y axis. Filled circles represent perceptron outputs of 1, and white circles perceptron outputs of 0. There is no way to draw a line that correctly separates the two categories for XOR. Figure styled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

ADDING A HIDDEN LAYER

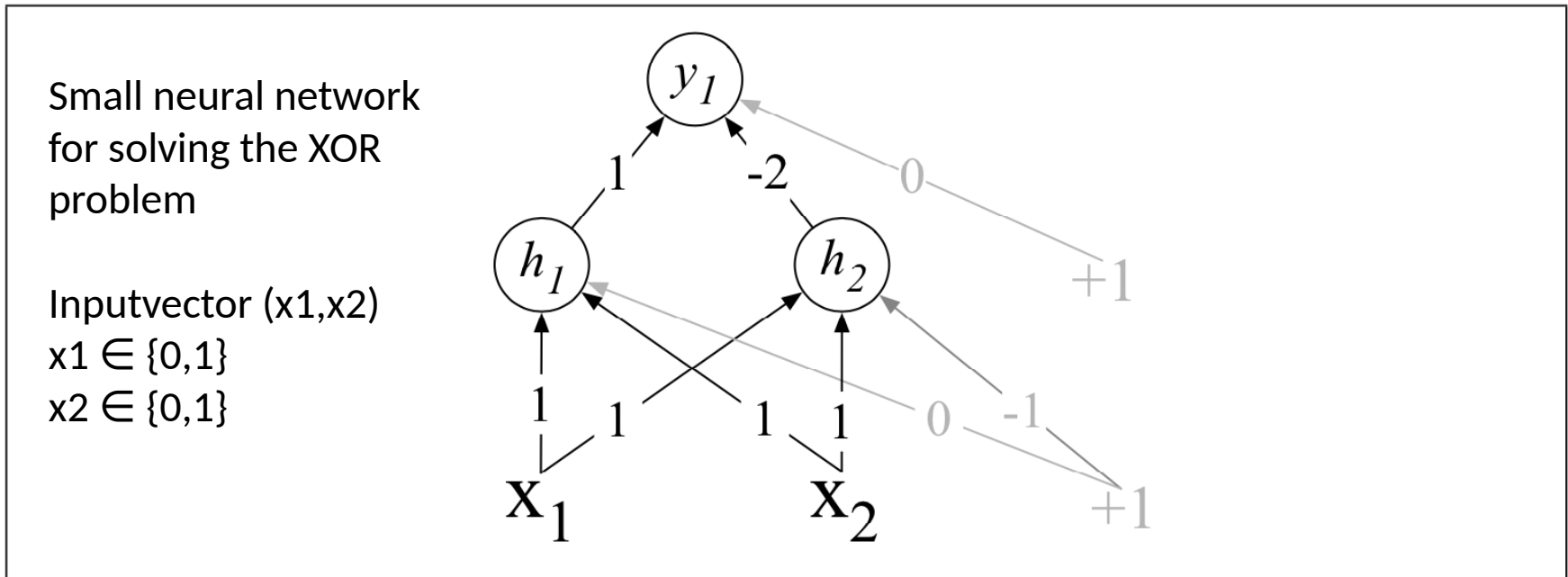
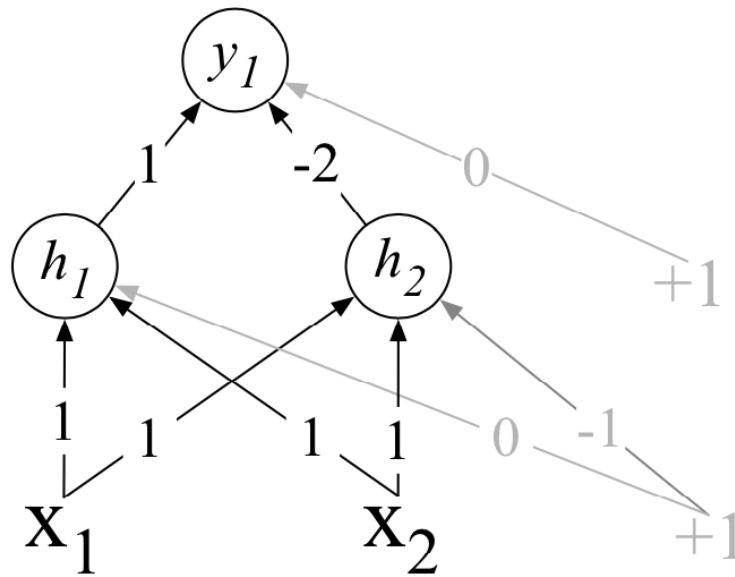


Figure 7.6 XOR solution after [Goodfellow et al. \(2016\)](#). There are three ReLU units, in two layers; we've called them h_1 , h_2 (h for “hidden layer”) and y_1 . As before, the numbers on the arrows represent the weights w for each unit, and we represent the bias b as a weight on a unit clamped to +1, with the bias weights/units in gray.

ADDING A HIDDEN LAYER

Small neural network
for solving the XOR
problem

Inputvector (x1,x2)
 $x_1 \in \{0,1\}$
 $x_2 \in \{0,1\}$

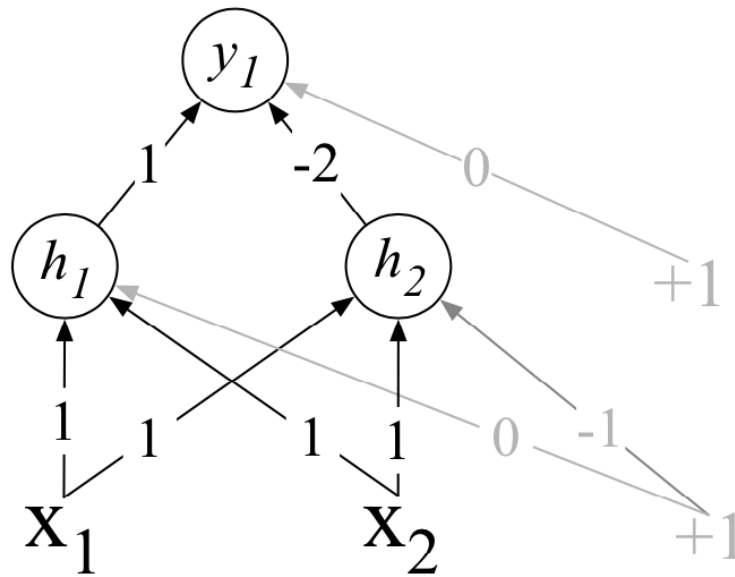


What is the
output for each
of these vectors:
[0,0]
[0,1]
[1,0]
[1,1]

ADDING A HIDDEN LAYER

Small neural network that solves the XOR problem

Inputvector (x1,x2)
 $x_1 \in \{0,1\}$
 $x_2 \in \{0,1\}$



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

$[0, 0] \rightarrow [1*0 + 1*0 + 0*1, 1*0 + 1*0 - 1*1] = [0, -1] \sim \text{ReLU}: [0, 0] \rightarrow 1*0 - 2*0 + 0*1 = 0 - 0 + 0 = 0$
 $[0, 1] \rightarrow [1*0 + 1*1 + 0*1, 1*0 + 1*1 - 1*1] = [1, 0] \sim \text{ReLU}: [1, 0] \rightarrow 1*1 - 2*0 + 0*1 = 1 - 0 + 0 = 1$
 $[1, 0] \rightarrow [1*1 + 1*0 + 0*1, 1*1 + 1*0 - 1*1] = [1, 0] \sim \text{ReLU}: [1, 0] \rightarrow 1*1 - 2*0 + 0*1 = 1 - 0 + 0 = 1$
 $[1, 1] \rightarrow [1*1 + 1*1 + 0*1, 1*1 + 1*1 - 1*1] = [2, 1] \sim \text{ReLU}: [2, 1] \rightarrow 1*2 - 2*1 + 0*1 = 2 - 2 + 0 = 0$

THE REPRESENTATION ON THE HIDDEN LAYER

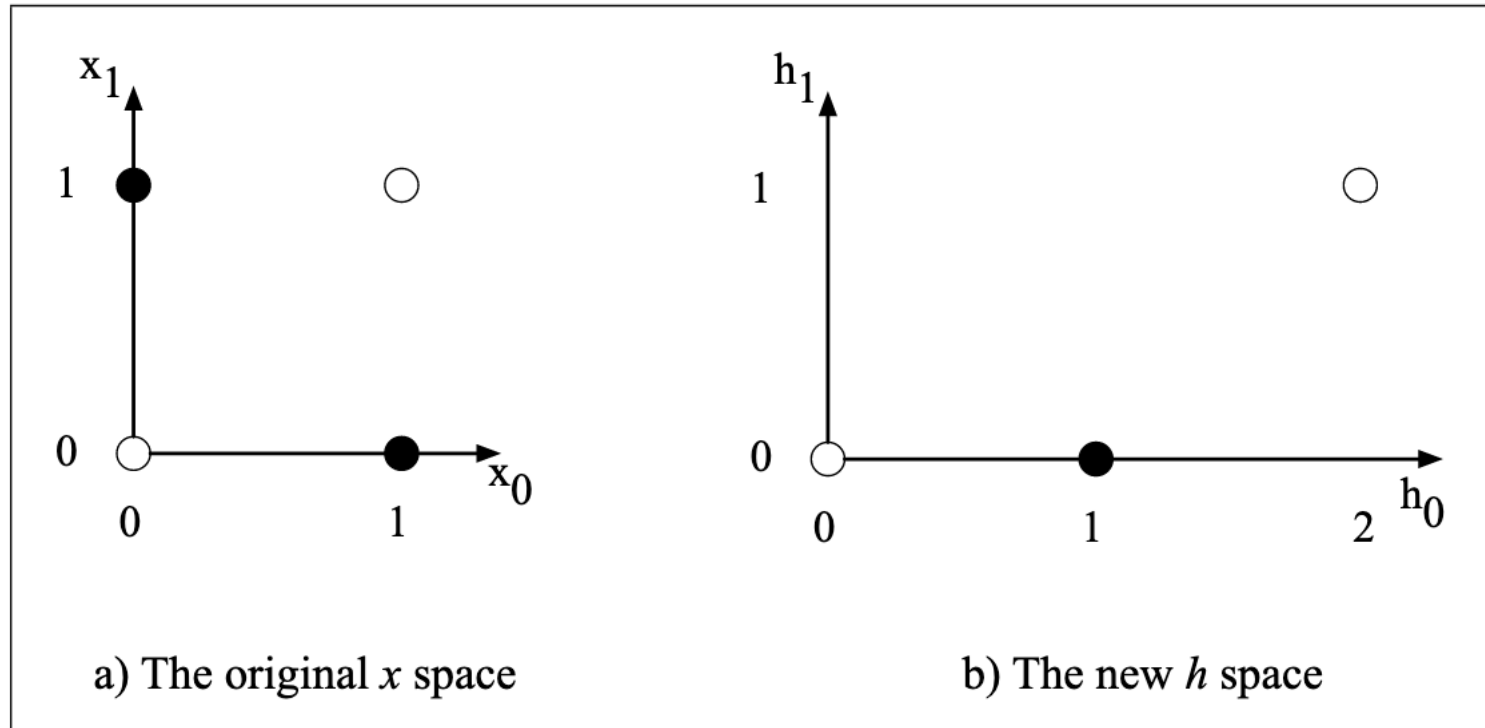


Figure 7.7 The hidden layer forming a new representation of the input. Here is the representation of the hidden layer, h , compared to the original input representation x . Notice that the input point $[0\ 1]$ has been collapsed with the input point $[1\ 0]$, making it possible to linearly separate the positive and negative cases of XOR. After [Goodfellow et al. \(2016\)](#).

NEURAL MODELS FOR SEQUENTIAL DATA

J&M CHAPTER 9

RECURRENT NEURAL NETWORKS

- RNNs have **connections between the hidden layers** of subsequent 'time steps' (words in a text)
- RNNs have an internal state that is updated in every time step. In the simplest case this state consists of a single *hidden* vector h

➤ J&M 9.2

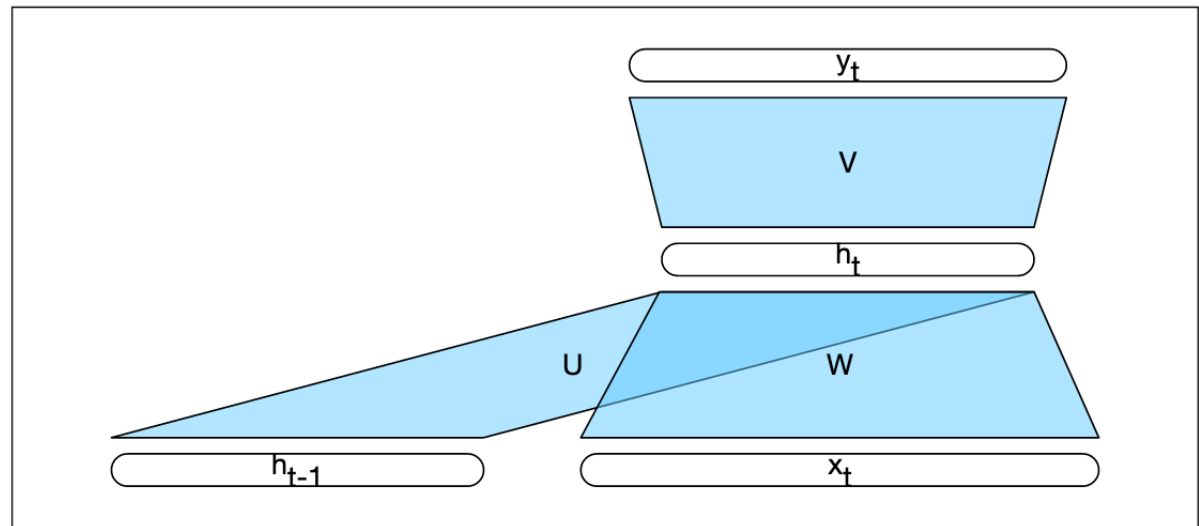


Figure 9.3 Simple recurrent neural network illustrated as a feedforward network.

RECURRENT NEURAL NETWORKS

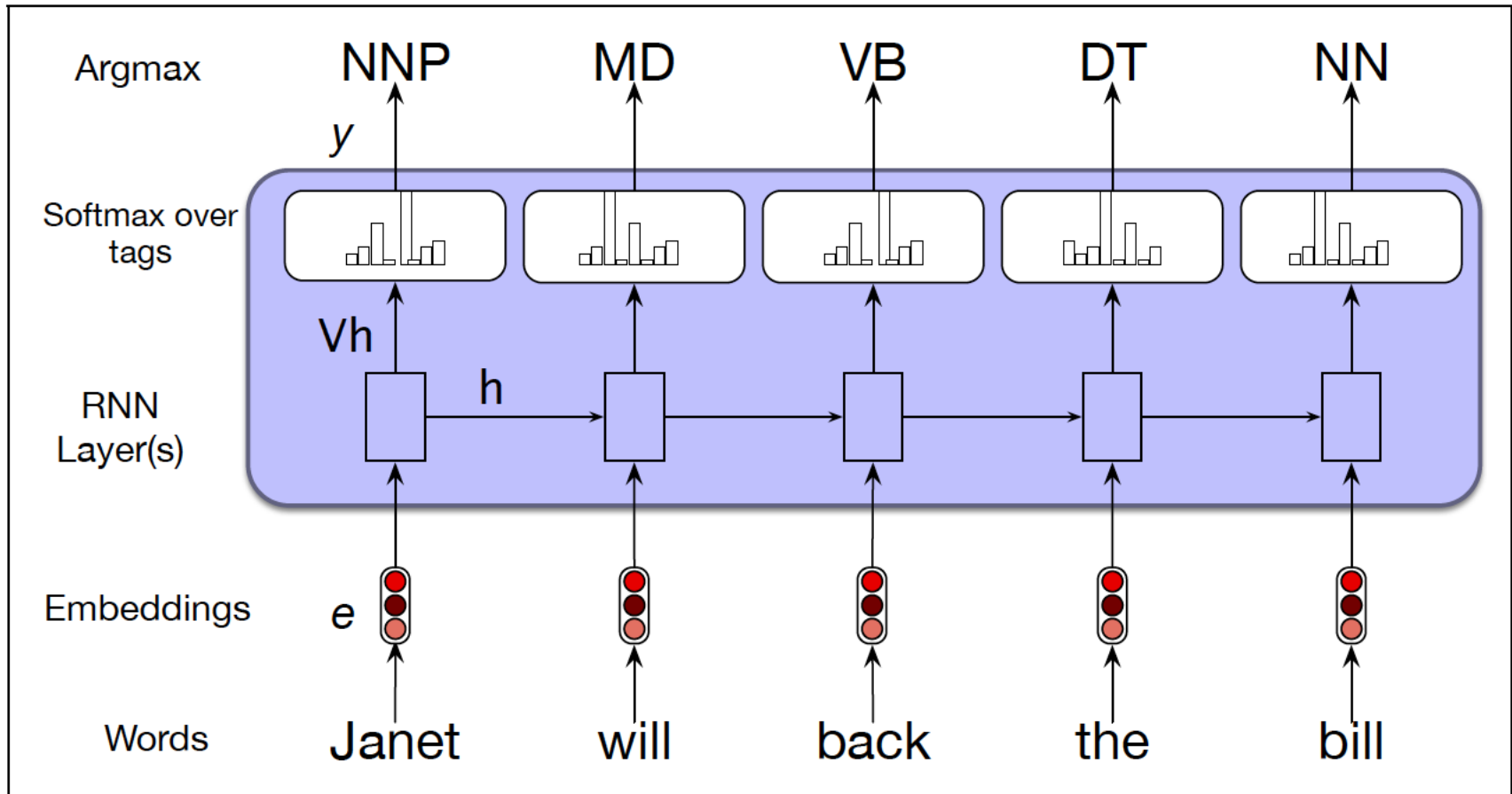


Figure 9.7 Part-of-speech tagging as sequence labeling with a simple RNN. Pre-trained word embeddings serve as inputs and a softmax layer provides a probability distribution over the part-of-speech tags as output at each time step.

THE LSTM

- LSTMs are more powerful (and more complex) RNNs that take longer contexts into account (see also week 6)
 - Longer context:
*"The **lectures** that I teach on Wednesday morning **are** about Text Mining"*
- RNNs only have one type of nodes on the hidden layer
- LSTMs "remove information no longer needed from the context, and adding information likely to be needed for later decision making" (J&M 9.6)

TRANSFORMER MODELS

- LSTMs are inefficient to train
- Breakthrough in neural sequence architectures: the **Transformer**

[PDF] **Attention is all you need**

[A Vaswani, N Shazeer, N Parmar...](#) - Advances in neural ..., 2017 - papers.nips.cc

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks in an encoder and decoder configuration. The best performing such models also connect the encoder and decoder through an attention mechanism.

☆ 🔖 Cited by 29922 Related articles All 28 versions >>

<http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf>

Suzan Verberne 2021



TRANSFORMER MODELS

- The Transformer architecture is a sequence-to-sequence architecture
 - Uses **self-attention** (computes multiple relations between pairs of input words/embeddings)
 - Much more efficient than BiLSTMs and other RNNs because input is processed in **parallel**
 - Can model **longer-term dependencies** because the complete input is processed at once
 - (but if too long, it is too heavy because of quadratic complexity)

TRANSFORMER MODELS

➤ If you are interested:

YouTube NL

Search

The diagram illustrates the Transformer architecture, consisting of an encoder and a decoder. The encoder processes the input sequence, while the decoder generates the output sequence. Both stacks are composed of multiple layers (N x) of self-attention and feed-forward sub-layers, each followed by a residual connection and layer normalization.

Encoder Stack (Left):

- Input Embedding
- Positional Encoding
- Multi-Head Attention
- Add & Norm
- Feed Forward
- Add & Norm

Decoder Stack (Right):

- Output Embedding
- Positional Encoding
- Masked Multi-Head Attention
- Add & Norm
- Multi-Head Attention
- Add & Norm
- Feed Forward
- Add & Norm

Inputs

Outputs (shifted right)

10:44 / 27:06

CC Settings Full Screen

Attention Is All You Need

162,618 views • Nov 28, 2017

3.8K 104 SHARE SAVE ...

Yannic Kilcher
44K subscribers

SUBSCRIBE

BERT

PRE-TRAINING OF DEEP BIDIRECTIONAL TRANSFORMERS FOR LANGUAGE
UNDERSTANDING

BERT

- The Transformer led to a breakthrough in NLP:

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 ...

☆ 📄 Cited by 28621 Related articles All 30 versions 🔗

<https://www.aclweb.org/anthology/N19-1423.pdf>

BERT INTUITION

Pre-training of Deep Bidirectional Transformers for Language Understanding

1. Pre-training: language modelling
2. Bidirectional: Predicting randomly masked words in context
3. Transformers: efficient neural architectures with self-attention

LANGUAGE MODELLING

- Language modelling tasks:
 1. Predicting randomly masked words in context
 - To capture the meaning of words
 2. Next-sentence classification
 - To capture the relationship between sentences
- Both are trained in parallel

BERT PRE-TRAINING TASKS

Input = [CLS] the man went to [MASK] store [SEP]
he bought a gallon [MASK] milk [SEP]

Label = IsNext

Input = [CLS] the man [MASK] to the store [SEP]
penguin [MASK] are flight ##less birds [SEP]

Label = NotNext

BERT uses subtokens for words that are not in the vocabulary

BERT ARCHITECTURE

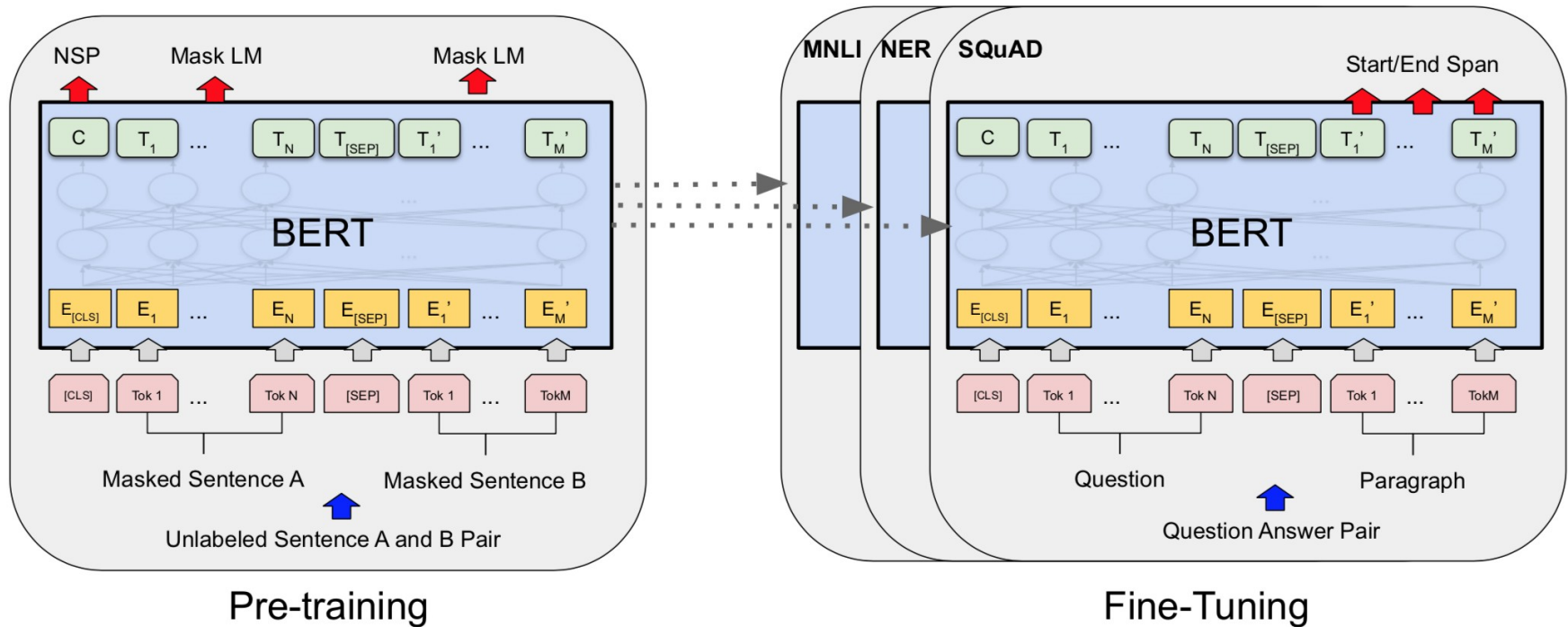


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are 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).

BERT ARCHITECTURE

- Two models presented in the BERT paper
 - BERT_{BASE} (L=12, H=768, A=12, Total Parameters= 110M)
 - BERT_{LARGE} (L=24, H=1024, A=16, Total Parameters=340M)
- (L: number of layers; H: dimensionality of hidden layers; A: the number of attention heads)
- “Compared to pre-training, fine-tuning is relatively inexpensive. All of the results in the paper can be replicated in at most 1 hour on a single Cloud TPU, or a few hours on a GPU, starting from the exact same pre-trained model”

<https://cloud.google.com/tpu/docs/colabs>

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 (even archeoBERTje)
- <https://huggingface.co/> is a very popular package for training and applying Transformer models

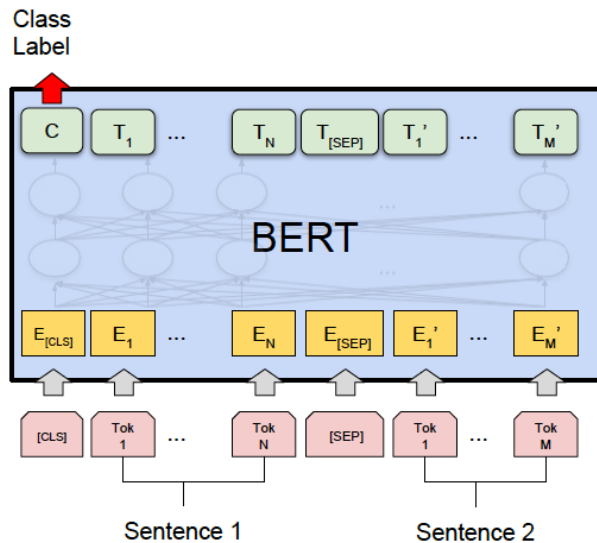
TRANSFER LEARNING



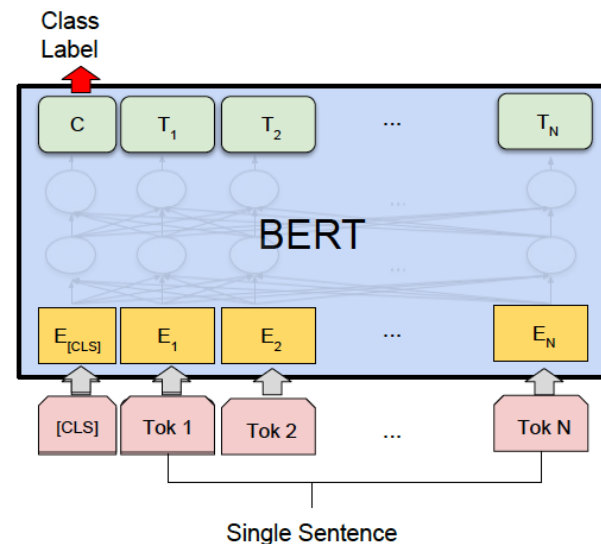
TRANSFER LEARNING WITH NEURAL LANGUAGE MODELS

- Inductive transfer learning: transfer the knowledge from pre-trained language models to any NLP task
 1. During pre-training, the model is trained on unlabeled data (self-supervision) over different pre-training tasks.
 2. For finetuning, the BERT model is first initialized with the pre-trained parameters,
 3. All of the parameters are fine-tuned using labeled data from the downstream tasks.
- Each downstream task has separate fine-tuned models, even though they are initialized with the same pre-trained parameters.

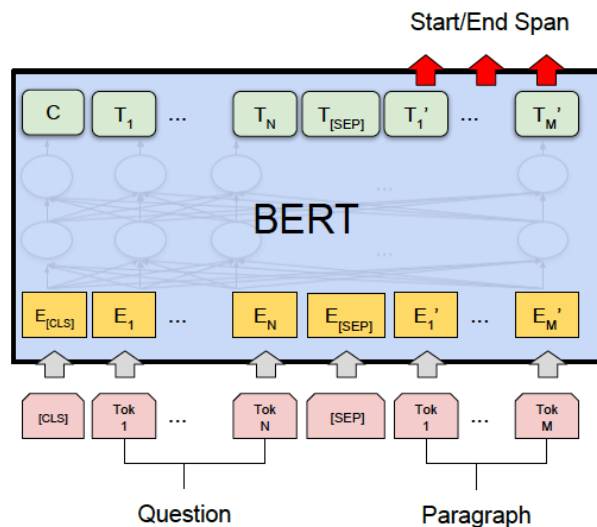
BERT FINE-TUNING ON DIFFERENT TASKS



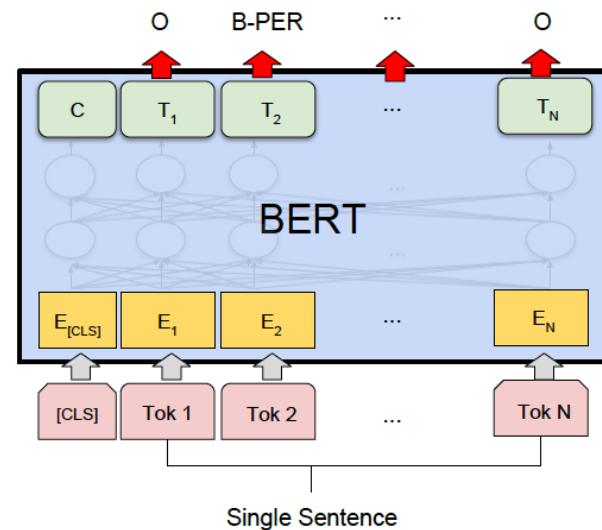
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG



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

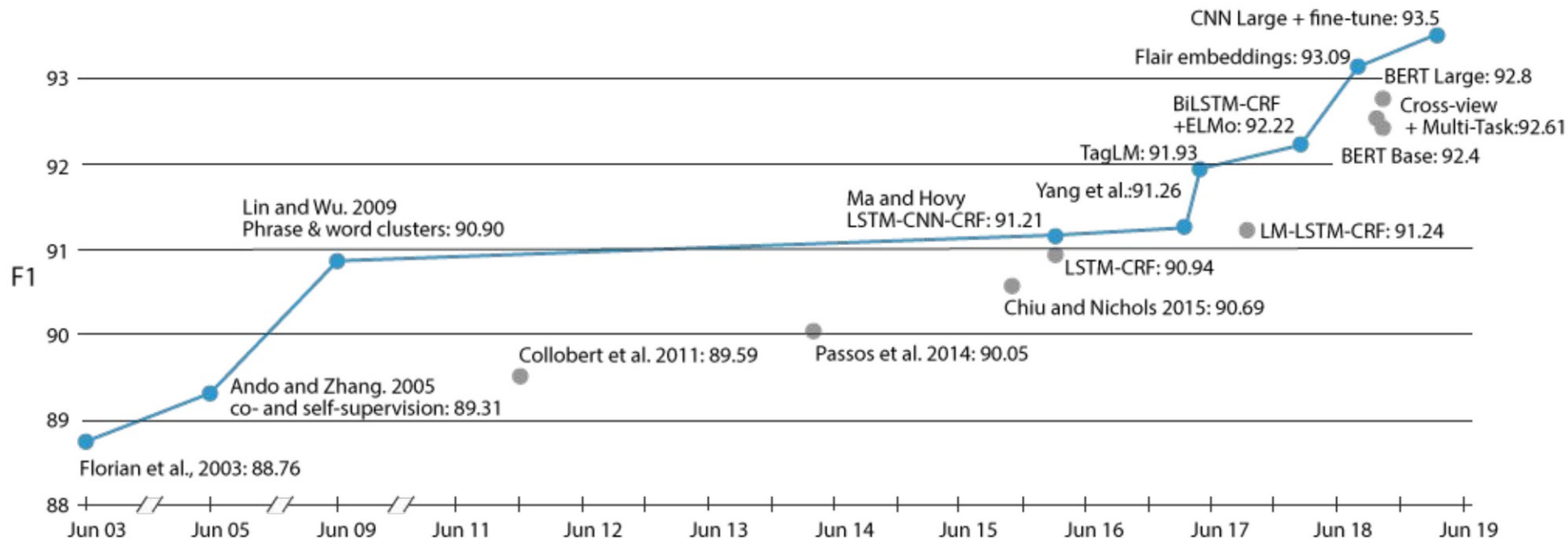


(c) Question Answering Tasks:
SQuAD v1.1



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




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/>

PRACTICAL USE

 alexbrandsen / **ArcheoBERTje-NER**   like 1

 Token Classification  PyTorch  JAX Transformers bert AutoNLP Compatible

Model card Files and versions

ArcheoBERTje-NER

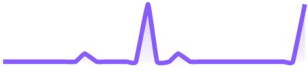
A Dutch BERT model for Named Entity Recognition in the Archaeology domain


This is the ArcheoBERTje model finetuned for NER, targeting the following entities:

- Time periods
- Places
- Artefacts
- Contexts
- Materials
- Species

NEW Select AutoNLP in the “Train” menu to fine-tune this model automatically.

Downloads last month
16




 **Hosted inference API** ⓘ

Token Classification Exam... ▼

In de buurt van Den Haag zijn middeleeuwse speerpunten gevonden **Compute**

Computation time on cpu: 0.072 s

In de buurt van Den Haag **LOC** zijn middeleeuws **PER** e speerpunt **ART** en gevonden

</> JSON Output  Maximize



CHALLENGES OF STATE-OF-THE-ART METHODS

- **Heavy** pre-training: takes time and computing power needed
- Finetuning and inference need GPU power
- **Hyperparameter tuning:**
 - select the optimal settings
 - optimization on development set (takes time)
 - adoption of hyperparameters from pre-training task
- Interpretation/explainability: additional effort

CHALLENGES OF STATE-OF-THE-ART METHODS

- Traditional models are also still commonly used in **classification** tasks
 - Typically, BERT is better on the larger categories and SVM is better on the smaller categories
- Sklearn can efficiently process text data into a term-document matrix and use a range of classifiers for the learning task
- The research community has almost completely moved to neural models, but in **applied contexts** you will still find the traditional models

CONCLUSIONS

SUZAN VERBERNE 2021

HOMework

➤ Read:

- J&M chapter 7. Neural Networks and Neural Language Models (you can skip 7.6.2, 7.6.3, 7.6.4)
- J&M chapter 9. Deep Learning Architectures for Sequence Processing
 - Note that the chapters refer to chapter 5 (Logistic Regression), sometimes with broken references: ??
 - If you are not familiar with Logistic Regression, you might need to read that chapter as well

➤ Optional exercise:

- Tutorial: Finetuning for sentence classification with BERT in Pytorch.
<https://mccormickml.com/2019/07/22/BERT-fine-tuning/>

HOMEWORK

- Assignment 2
 - Task: “Emerging and Rare [entity recognition](#)” from the Workshop on Noisy User-generated Text (W-NUT)
 - Sequence labelling with CRFsuite
 - Make sure you complete the [exercise of week 6](#)
 - **Deadline assignment 2: November 15**

AFTER THIS LECTURE...

- You can explain the role of the **hidden layer** in feedforward neural networks
- You can explain **Recurrent Neural Networks** on a conceptual level
- You can explain BERT on a conceptual level
- You can explain how **transfer learning** from pre-trained language models is used for text mining tasks