# Classification of Unstructured Documents

*Transfer Learning with BERT*

*06 December 2022*

Ma. Adelle Gia Arbo, *m.arbo@students.hertie-school.org*

Janine De Vera, *j.devera@students.hertie-school.org*

Lorenzo Gini, *l.gini@students.hertie-school.org*

Lukas Warode, *l.warode@students.hertie-school.org*

Hertie School

# Presentation Outline

1. **Motivation**
   Leveraging information from unstructured PDFs
2. **Text Classification Pipeline**
   Building text and document classifiers
3. **Models**
   Comparing machine learning and deep learning algorithms
4. **BERT**
   Using BERT for text classification tasks
5. **Application and Initial Results**
   Illustrating the text classification pipeline using DMA consultation documents
6. **Policy Implications**
   Integrating results with EU policy process

# Motivation

*Leveraging information from unstructured PDFs*

**80%** of all information is **unstructured**

- The most **common type of unstructured data is text** – from blogs, social media, official documents, etc.
- Text can be hard to analyze and organize because it comes in a **format that a computer cannot readily understand**

Only **18%** of companies are able to take advantage of such data

- For most organizations, decisions are made on the basis of only **10-20% of data available**
- Tools and know-how needed to successfully leverage unstructured text data is concentrated on select industries (e.g. tech, retail, finance)

Use of text data is beneficial to public institutions like the EC

- The EU Commission conducts an average of **100 public consultations** annually, with the number of responses ranging from **10,000 to 4 million**
- Impossible to review and extract information manually! Streamlined review of documents lead to **more efficient use of scarce manhours**

# Text Classification Pipeline

*Building text and document classifiers*

## Data preparation

**Extracting, cleaning, and processing text** from raw inputs into tidy text format

1. *Parsing PDFs*

   **Extracting raw text** from input data by removing non-textual information and converting text to **required encoding format**

2. *Cleaning text*

   Standard **data wrangling** steps – unicode normalization, spelling correction, lowercasing

3. *Pre-processing*

   Processing to **keep information relevant to feature extraction** – stop words removal, stemming, lemmatization, language detection, coreference resolution

## Text representation

**Converting raw text into numerical form** that can be understood by ML/DL algorithms

1. *Basic vectorization*

   Mapping each word of the corpus vocabulary (V) to a unique ID, and **representing each document as a V-dimensional vector**

   Common methods: bag of words, bag of n-grams, TF-IDF

2. *Distributed representation*

   **Compact and dense vector representations** that capture distributional similarities and context

   Common methods: pre-trained embeddings (Word2vec, GloVe), training own embeddings (CBOW, SkipGram)

## Model training

Using text representation to learn associations (**feature-category mapping**)

1. *Classical NLP/ML-based*

   Using text features (can be handcrafted) to make predictions

   Common classification models: Naïve Bayes classifier, logistic regression, support vector machine, gradient boosting

2. *Deep Learning-based NLP*

   Learning features from the data through multilayered neural architecture

   Common DL models: CNN, RNN (LSTM), transformer-based models (BERT, GPT-3)

## Model evaluation

**Measuring how good a model is** by assessing performance on unseen data

1. *Intrinsic evaluation*

   Measuring model performance using system metrics

   Common metrics: Precision, Recall, F1-score, ROC-AUC

2. *Extrinsic evaluation*

   Measuring model performance on the **final objective**

   Used with business metrics to see if model achieved desired improvements (e.g. time saved after implementing classifier)

# Text Classification Pipeline

*Working with the PDF format*

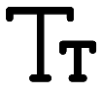
## Importance of the PDF parsing step

- PDFs were **not designed as a data input** format. They consist of **a set of instructions** describing how to draw on a page.
- **Text is not stored as words** but rather as characters/objects in a specific location.
- PDFs cannot be processed directly by ML/NLP frameworks – they must be **converted to text.**

## Using Python to work with PDF formats

- Python has text analytics libraries and frameworks that deal with PDF format.
  *Note: these libraries may yield different results.*


Find location of text objects


Find other info e.g. fonts, lines


Extract text data

Convert to other readable formats

- For **scanned PDFs, Optical Character Recognition (OCR)** libraries are used to pull text out of images.


Scanned PDF converted to PIL


Mark regions where info is


OCR on identified regions


Extract text data

# Models

*Comparing machine learning and deep learning algorithms*

Traditional ML and DL models differ in their treatment of feature representations

## Traditional ML classifiers

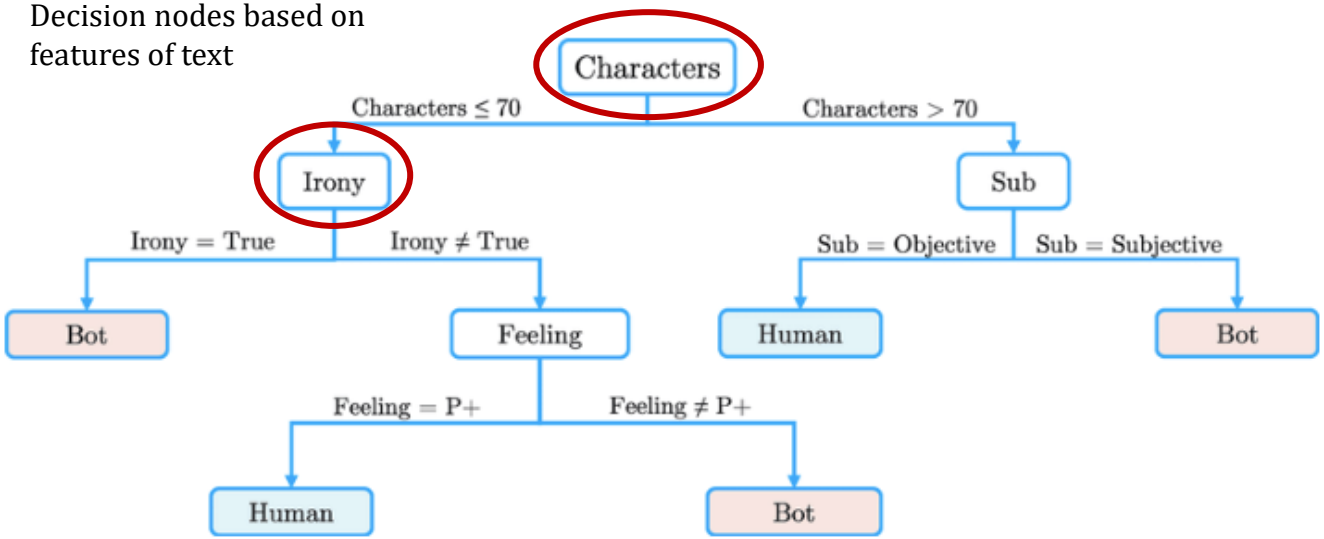
- Using word vectors (one-hot encoding, TF-IDF) as features that can be "handcrafted" and tailored to domain knowledge
- Using embedding vector as the feature vector that represents the entire text

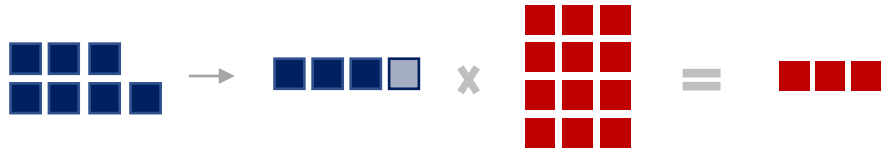
## DL classifiers

- Further processing of input vectors fed into neural network architecture

1. Creating word-index vectors by tokenizing
2. Padding sequence so vectors are same length
3. Mapping word-index vector to embedding vector

Decision nodes based on features of text



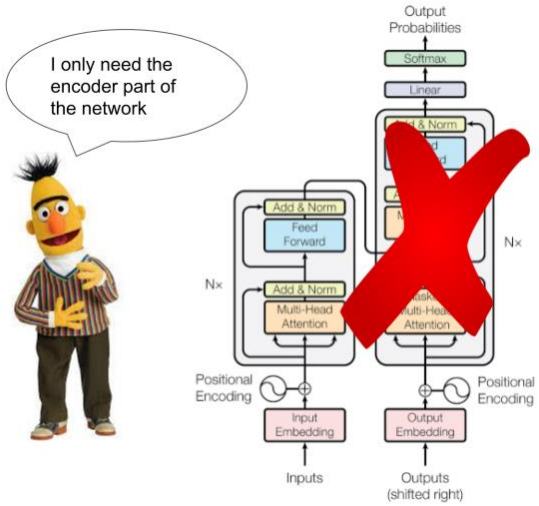
Word-index vectors per document | Padding | Embedding matrix | Embedding vector

Hertie School

# BERT

*Using BERT for text classification tasks*

## BERT uses only the encoder part of the Transformer architecture

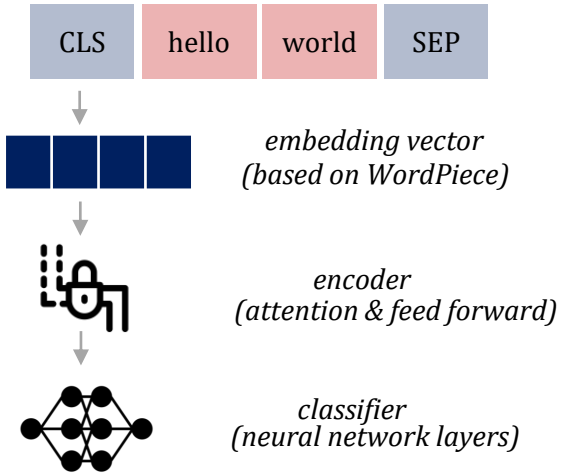



BERT is pretrained using Wikipedia + BookCorpus and with two pre-training objectives:

**Masked language modeling (MLM):**
Randomly masking 15% of the words in the input and predicting the masked words

**Next sentence prediction (NSP):**
Concatenating two masked sentences as inputs and predicting if sentences were following each other or not

## BERT has 3 embedding layers for vector representations




**Token Embeddings**: transforms words into a vector representation of fixed dimension. Each word is represented as a 768-dimensional vector.

**Segment Embeddings**: distinguishes tokens from semantically similar input pairs.

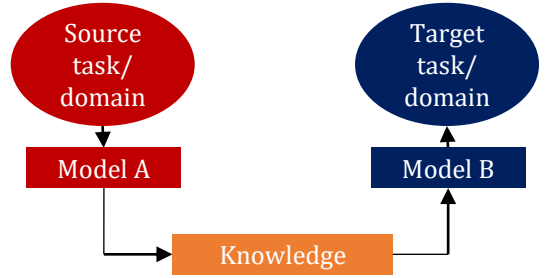
**Position Embeddings**: learns temporal properties (sequential nature) of inputs.

**Input Representation**: element-wise sum of 3 embeddings fed into the encoder layer

## Advantages of pre-trained models and how to fine-tune them

**Dataset size:** Limited resources to train models with big neural architecture with a high number of parameters

**Fine-tuning:** Pre-trained models could be used "off-the-shelf", i.e. without any changes




**Fine-tuning techniques:**

- **Train the entire architecture:** train the entire pre-trained model on a new dataset and feed the output to a softmax layer
- **Train some layers while freezing others:** keep the weights of initial layers of the model frozen while retrain only the higher layers
- **Freeze the entire architecture:** freeze all the layers of the model and attach a few neural network layers

# Application and Initial Results

*Research motivation and objectives*

## Classifying EU Digital Markets Act consultation documents

### Competition in EU digital markets

The **Digital Markets Act (DMA)** is a regulation in the European Union enacted in 2022. It aims to promote fair competition within the digital market by defining rules for "gatekeepers" or large online platforms.

When new legislation is proposed by the European Commission, **public consultations** are opened.

Stakeholders submit **position papers and reports** (in the form of PDFs) where they detail their views on the proposed law.

### Research objectives

- **General objective:**

  **Analyze** the importance of a new proposed legislation based on unstructured text data from public consultation documents

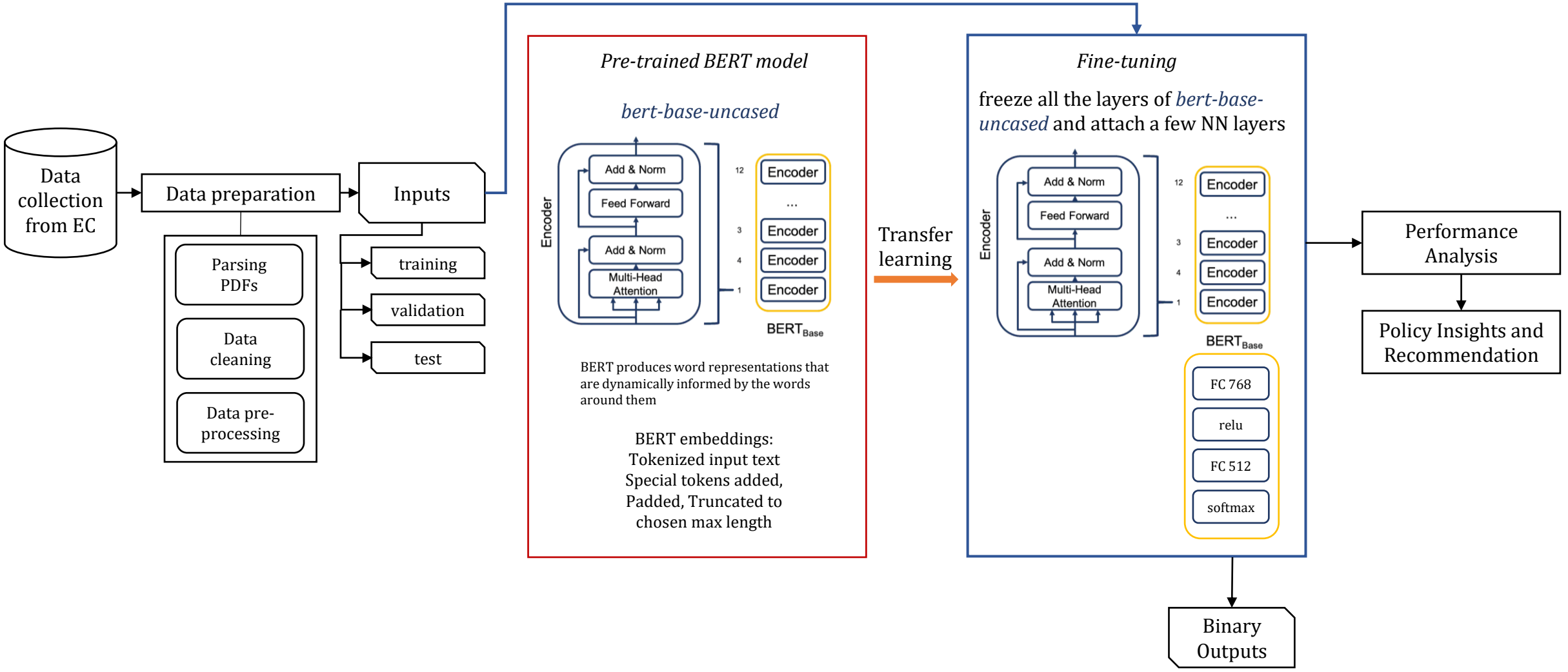
- **Specific objectives:**

  **Classify public consultation documents** according to whether stakeholder agrees or disagrees with the DMA proposal.

  **Understand** how different stakeholders in digital markets **respond** to the DMA proposal

# Application and Initial Results

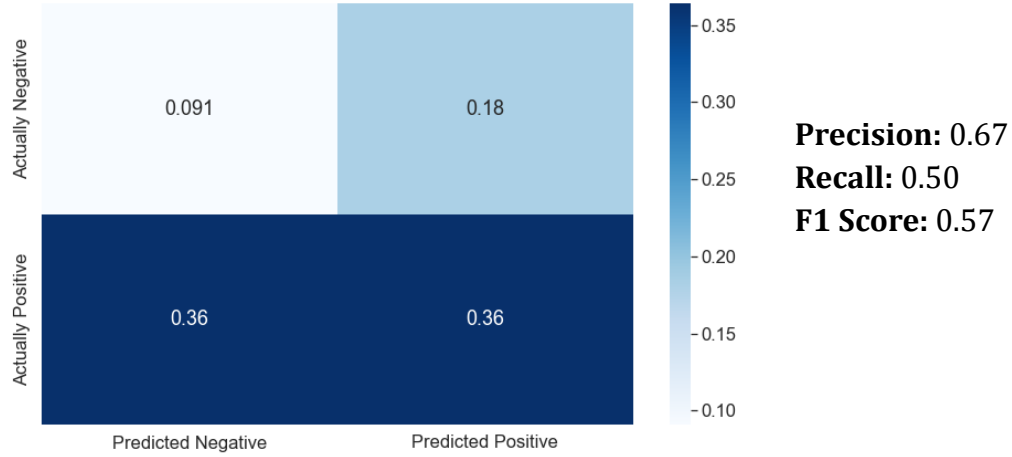
*Transfer learning with BERT using DMA consultation documents*



# Application and Initial Results

*Transfer learning with BERT using DMA consultation documents*

| | Documents that **predicted disagreement** with legislation | Documents that **predicted agreement** with legislation | |
|---|---|---|---|
| | TN | FP | Documents that **actually disagree** with legislation |
| | FN | TP | Documents that **actually agree** with legislation |




**Precision:** 0.67
**Recall:** 0.50
**F1 Score:** 0.57

**Precision:** out of all documents that were classified by the model as agreeing with the legislation**, how many actually agree?**

$$\frac{TP}{TP + FP}$$

**Recall:** out of all documents that actually agree with the legislation, **how many were classified correctly?**

$$\frac{TP}{TP + FN}$$

## Next steps

- Further **fine tuning of base BERT** model
- Explore **BERT variants** trained on a specific type of corpus (e.g. academic or legal documents)
- **Comparison of results** with machine learning classifiers (e.g. gradient boosting using TF-IDF) and other deep learning classifiers (e.g. RNN, CNN)

- Different stakeholders and policy actors might be interested in **optimising different metrics**
- Being aware of **policy context**
- **Add communication component for tutorial** given non-technical background of policy audience

# Policy Implications

*Integrating results with EU policy process*

Model **results** and technical **implications**

**Structural integration**

Adaption to **institutional** and **policy** context

**Disentangling complexity** of documents

**Identification**

**Concise**, **implementable** and **objective** results

**Immediate** model **results**

**Temporality**

Timely **information** for **stakeholders**

**1. Introducing** solution to **stakeholders**

**2. Classification** of documents

**Policy cycle**

**1. Setting agenda**

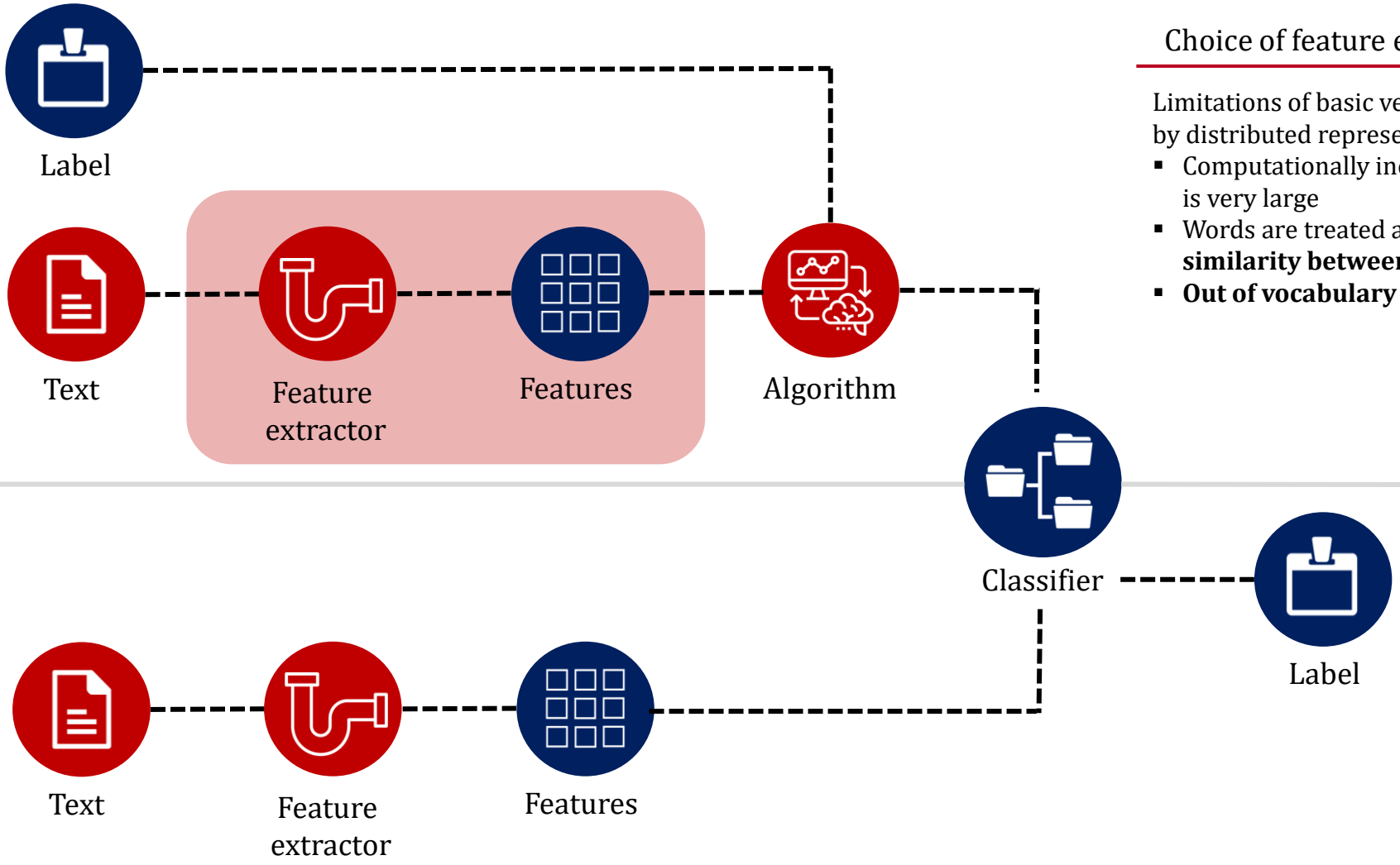
**2. Implement** and **evaluate** policies

Hertie School

# Appendix

# Models

*Comparing machine learning and deep learning algorithms*




Choice of feature extraction method is important

Limitations of basic vectorization approaches addressed by distributed representations:

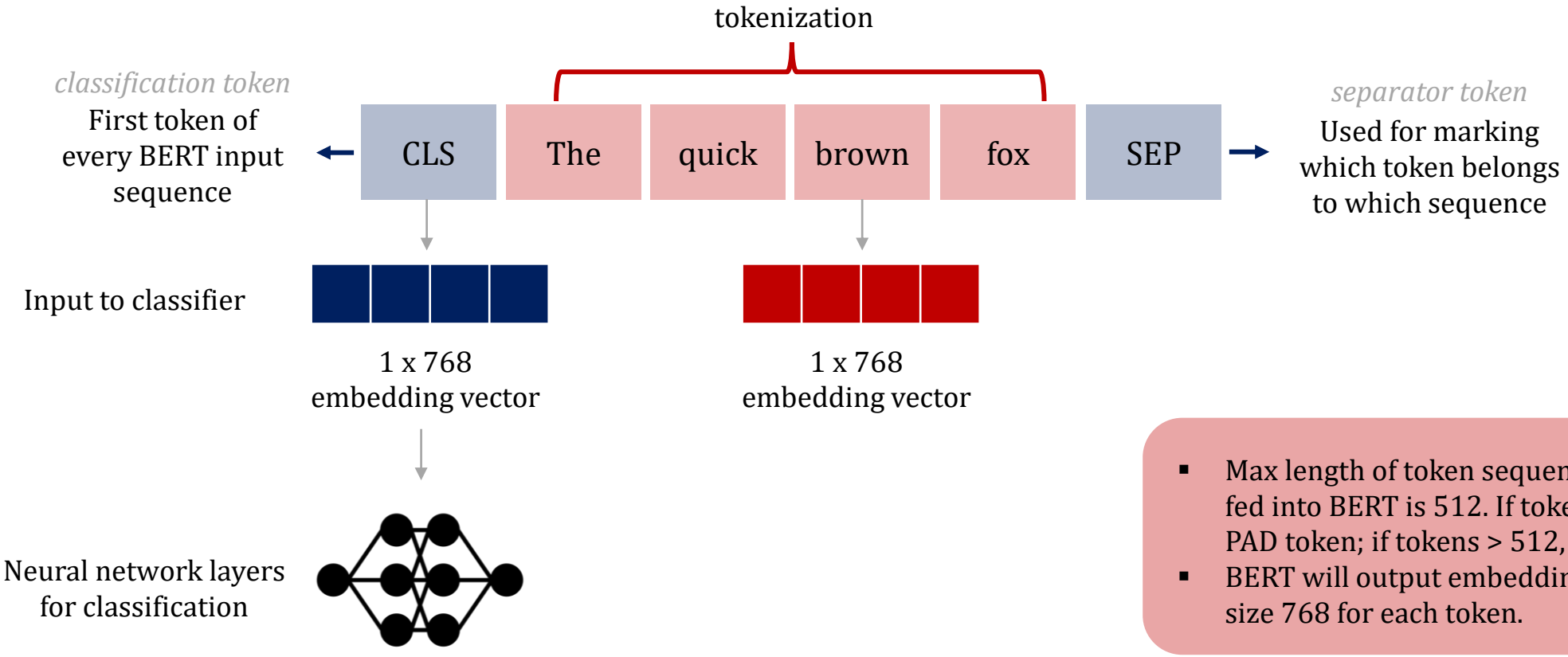
- Computationally inefficient to store and train if dataset is very large
- Words are treated as atomic units, with no notion of **similarity between words**
- **Out of vocabulary (OOV)** problem

# BERT

*Using BERT for text classification tasks*

## Bidirectional Encoder Representations from Transformers

- BERT consists of several **transformer encoders** stacked together
- Each transformer encoder consists of two sub-layers: **self-attention layer and feed-forward layer**




- Max length of token sequence that can be fed into BERT is 512. If tokens < 512, use PAD token; if tokens > 512, truncate.
- BERT will output embedding vector of size 768 for each token.

# BERT

*Three embedding layers*

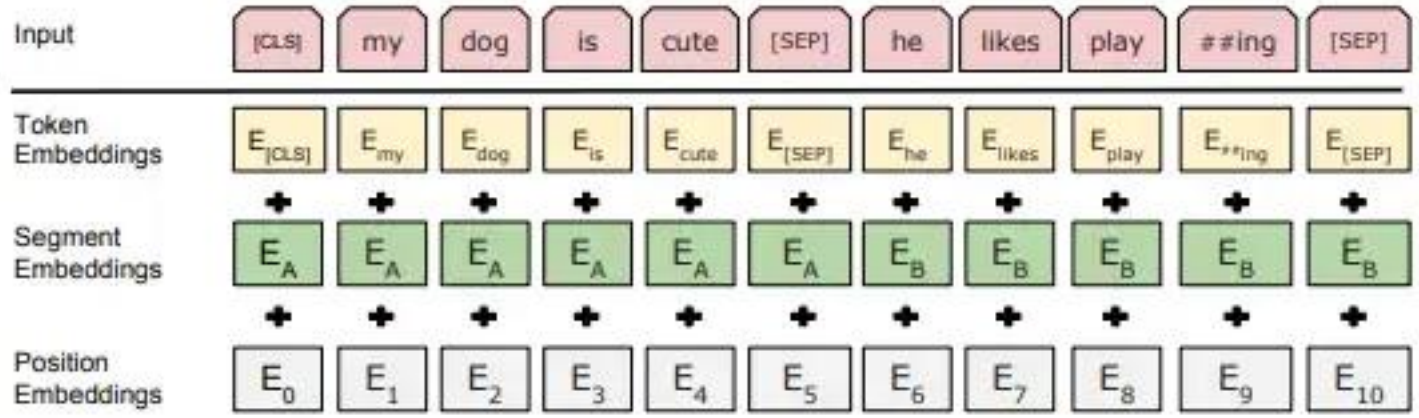



Figure 2: BERT input representation. The input embeddings is the sum of the token embeddings, the segmentation embeddings and the position embeddings.

# BERT

*Other variants*

## BERT variants

| Comparison | BERT<br>October 11, 2018 | RoBERTa<br>July 26, 2019 | DistilBERT<br>October 2, 2019 | ALBERT<br>September 26, 2019 |
|---|---|---|---|---|
| **Parameters** | **Base:** 110M<br>**Large:** 340M | **Base:** 125<br>**Large:** 355 | **Base:** 66 | **Base:** 12M<br>**Large:** 18M |
| **Layers / Hidden Dimensions / Self-Attention Heads** | **Base:** 12 / 768 / 12<br>**Large:** 24 / 1024 / 16 | **Base:** 12 / 768 / 12<br>**Large:** 24 / 1024 / 16 | **Base:** 6 / 768 / 12 | **Base:** 12 / 768 / 12<br>**Large:** 24 / 1024 / 16 |
| **Training Time** | **Base:** 8 x V100 x 12d<br>**Large:** 280 x V100 x 1d | 1024 x V100 x 1 day<br>(4-5x more than BERT) | **Base:** 8 x V100 x 3.5d<br>(4 times less than BERT) | [not given]<br>**Large:** 1.7x faster |
| **Performance** | Outperforming SOTA in Oct 2018 | 88.5 on GLUE | 97% of BERT-base's performance on GLUE | 89.4 on GLUE |
| **Pre-Training Data** | BooksCorpus + English Wikipedia = 16 GB | BERT + CCNews + OpenWebText + Stories = 160 GB | BooksCorpus + English Wikipedia = 16 GB | BooksCorpus + English Wikipedia = 16 GB |
| **Method** | Bidirectional Transformer, MLM & NSP | BERT without NSP, Using Dynamic Masking | BERT Distillation | BERT with reduced parameters & SOP (not NSP) |

## BERT base and large variants

| Shortcut name | Model details |
|---|---|
| bert-base-uncased | 12-layer, 768-hidden, 12-heads, 110M parameters. Trained on lower-cased English text |
| bert-large-uncased | 24-layer, 1024-hidden, 16-heads, 3364M parameters. Trained on lower-cased English text |
| bert-base-cased | 12-layer, 768-hidden, 12-heads, 110 parameters. Trained on cased English text |
| bert-large-cased | 24-layer, 1024-hidden, 16-heads, 336M parameters. Trained on lower-cased English text |