SJK006 - Master in Intelligent Systems

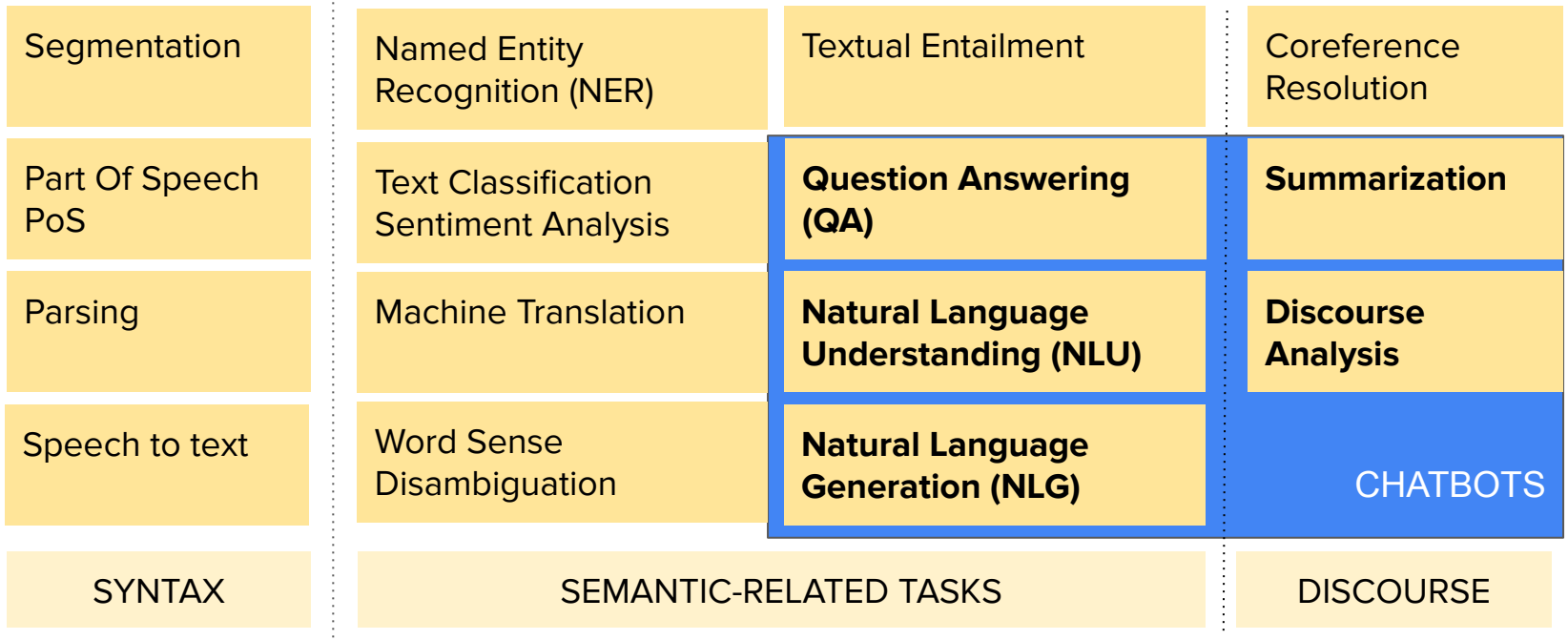
EMBEDDINGS & LLMs

How to process text & images in data science

Learning goals

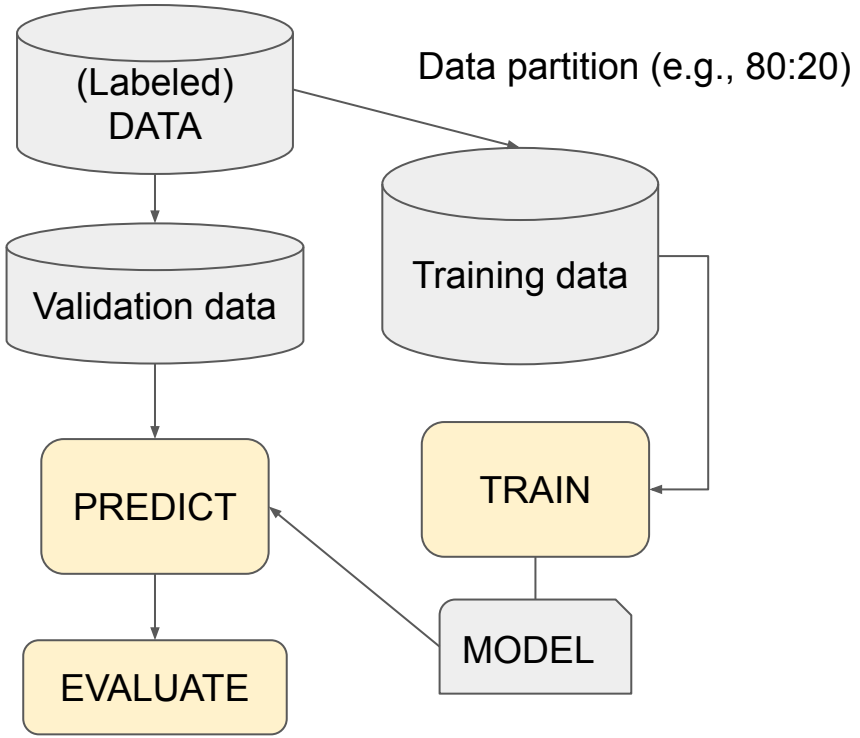
- Main issues in traditional text representation?
- Transformers: Encoders and Decoders
- BERT the base encoder
- Zero Shot Learning with LLMs
- Few Shot Learning with LLMs
- Using LLMs for automating tasks

Natural Language Processing Tasks

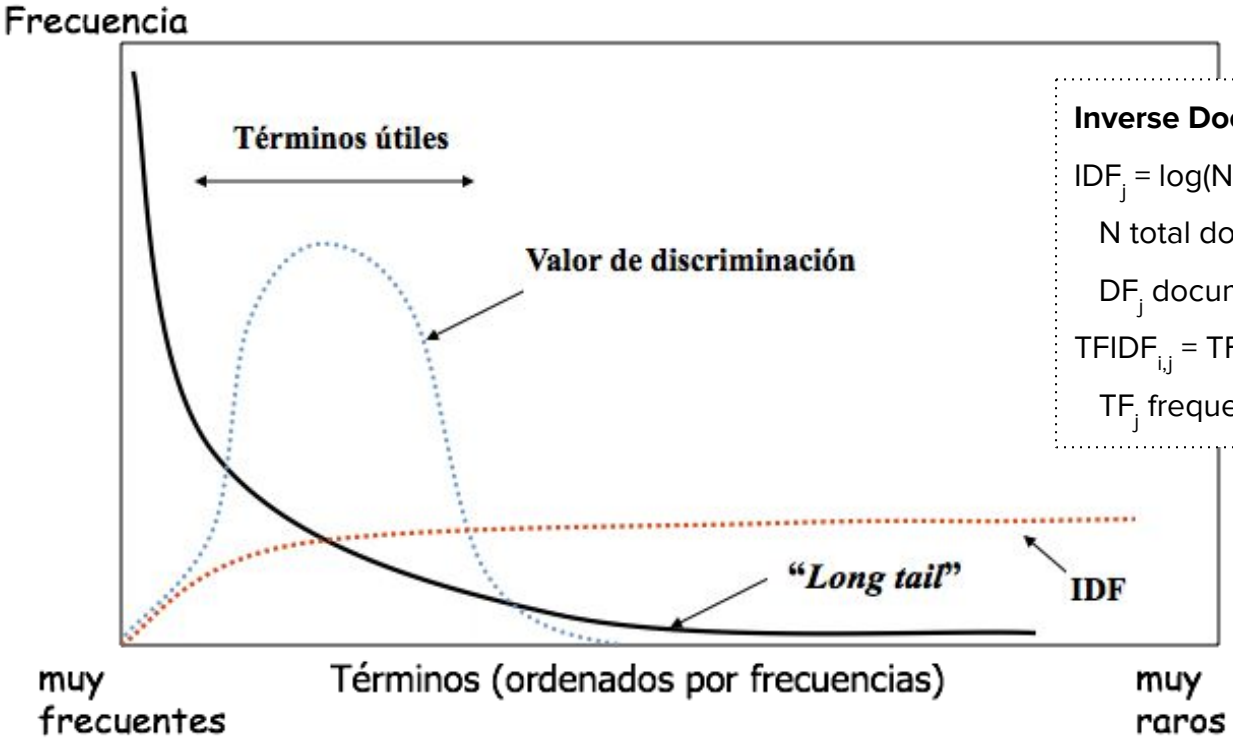


Traditional machine learning applied to texts

- **Bag of Words**: Text is split in words and words are weighted by some indexing scheme (TF, TF-IDF, …)
- Feature selection (vocab reduction)
- Each task has its own labeled data:
  - Sentiment analysis (+/-)
  - Topic classification
  - Regression for rankings
  - etc.
- Each task adopts the most suited model for predictive analysis
  - Logistic regression
  - XG-Boost
  - Decision trees (Random Forests)
  - Naïve Bayes
  - etc.



Word frequency distribution: Zipf law



Inverse Document Frequency/Term Frequency

$IDF_j = \log(N/DF_j)$

N total documents in the collection

DF_j documents containing word *j*

$TFIDF_{i,j} = TF_{i,j} * IDF_j$

TF_j frequency of word *j* in document. *i*

Som reflections

- The “Long tail” issue has several implications:
  - We can always find a new word not seen before (OOV - Out of Vocabulary)
  - We only have enough statistics for a small set of words
  - Overfitting in traditional machine learning methods
- Semantics are difficult to capture with traditional models
  - Synonymy, antonymy, etc.
  - Frequent terms are often ambiguous
  - They are used in many different senses in different contexts.
  - Combinations of frequent terms can also be ambiguous
- Any collection will be heavily biased towards certain topics
  - Large latent class imbalance
  - Few very big topics, many very small topics

Neural-based word embeddings

- Main Methods:
  - **Word2Vec** [seer DEMO Projector TF]
  - GloVe
  - FastText (embeddings sub-word level)
- Addresses some main problems of traditional text representation:
  - **Long tail**: semantics come from the context regardless the word frequencies
  - **Dense vectors** instead of sparse vectors (suited for linear algebra ML methods).
- But other issues still present:
  - Averaging word embeddings do not account for word ordering
  - An ambiguous word (e.g. bank) is only associated to one vector
  - Words out of vocabulary have no vector (**OOV**)

Deep Learning architectures

Convolutional Networks (CNN 2D)

Recurrent Networks

LSTM/GRU	bi-LSTM	Seq2Seq

Transformers

BERT	GPT-x	XLNet

Basic Principle: *encoder-decoder*

- Encode the input (*embeddings*)
- Decode the output (task)

The encoder can be shared in many different tasks:

- **Transfer Learning** (pre-trained models)
- **Fine-tuning** (re-training the model)

Input: text is tokenized, where tokens are numbers

Since texts have non-fixed length,

- there is a maximum length
- “padding” is used for completing sequences

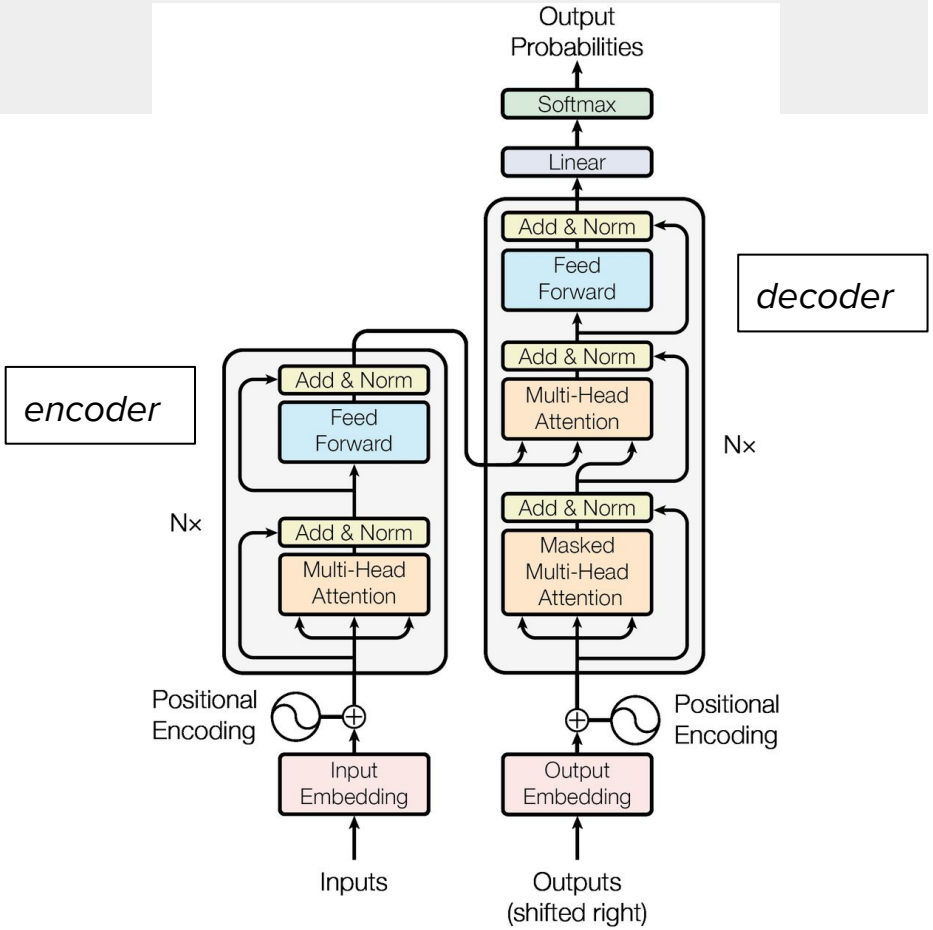
Transformers

2017: All you need is attention

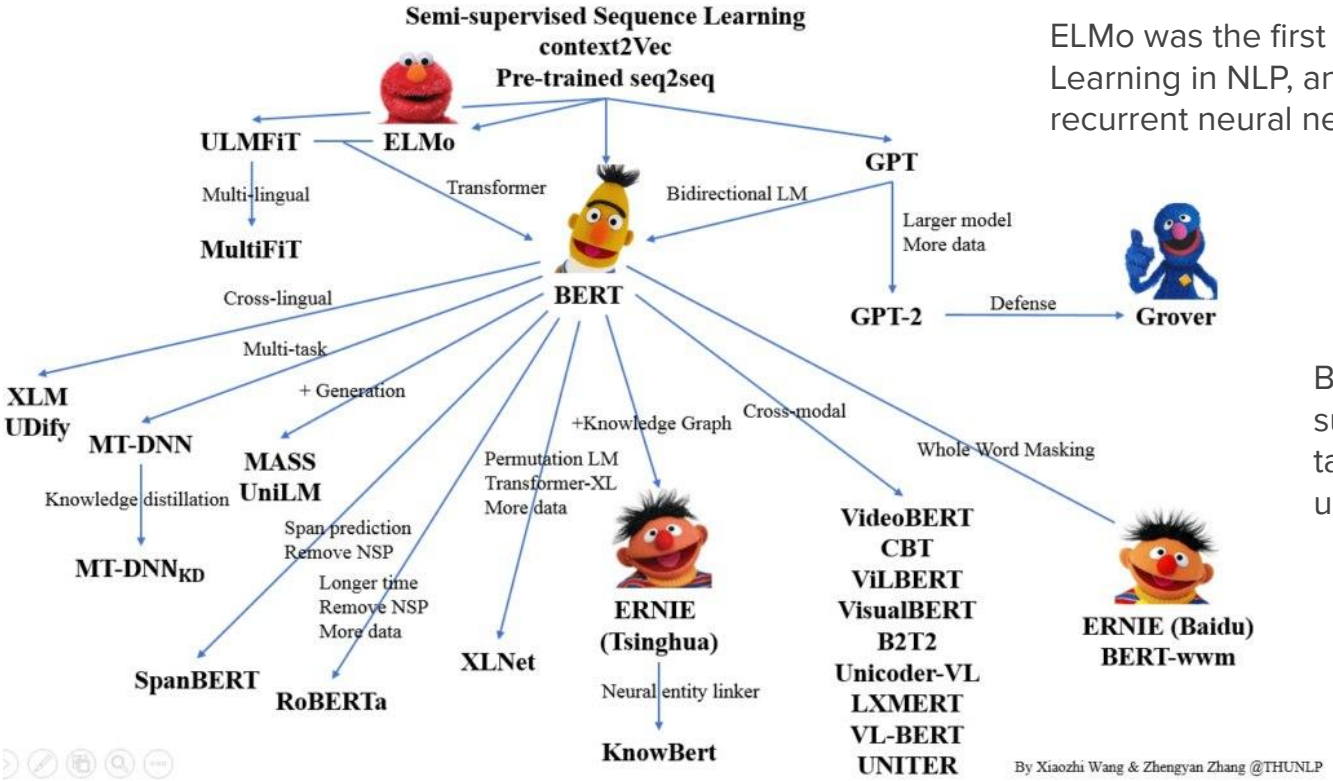
KEY POINTS:

- Divide words into tokens (tokenizer)
- Self-supervised + Fine tuning
- Token position embeddings
- Self-attention layers
- Predict with all → encoder
- Predict with mask → decoder

VISUALISATION OF SELF-ATTN



Encoders and Decoders



ELMo was the first approach to Transfer Learning in NLP, and was based on recurrent neural networks (Bi-LSTM).

BERT was the first system to successfully address all NLP tasks posed. It is still widely used today.

Tokenizers

- They address the **OOV** problem by split words into known tokens
- Simplest tokenizers just detect frequent prefixes and suffixes
- Byte Pair Encoding (BPE) is intended to learn a tokenizer according to the frequency of consecutive letters in a corpus

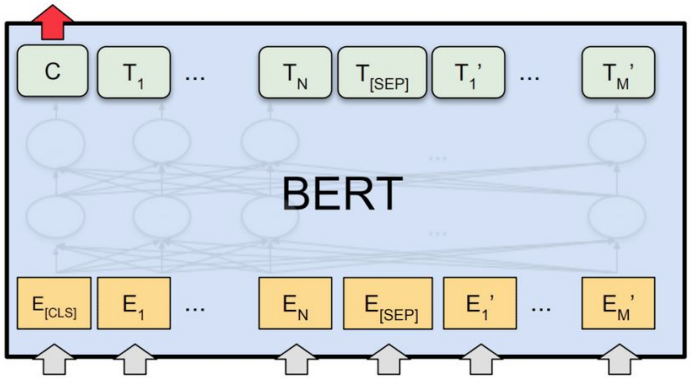
More details of tokenizers can be found at Huggingface

BERT: an encoder transformer for multiple tasks

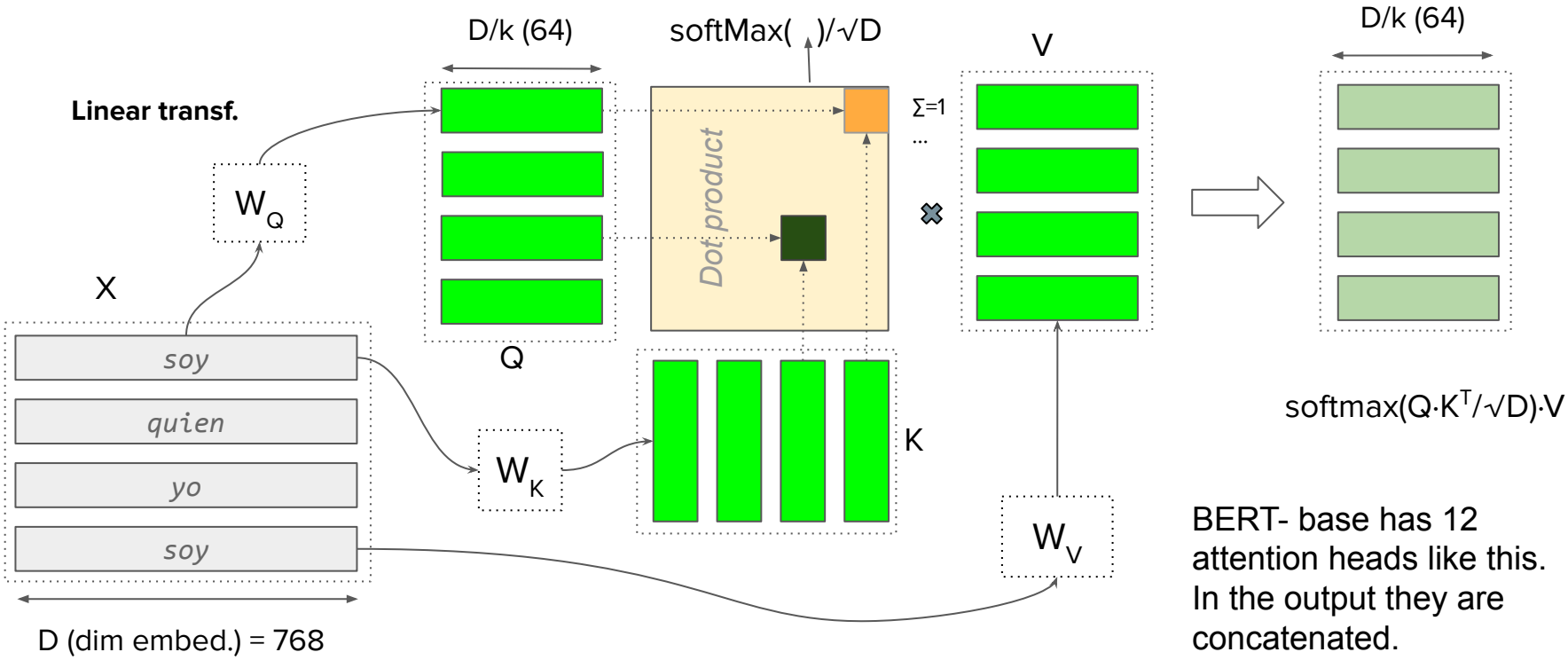
Convolucionales (CNN 2D)		
Recurrentes		
LSTM/GRU	bi-LSTM	Seq2Seq
Transformers		
BERT	GPT-2	XLNet

BERT only uses the *encoder* part of a transformer. Two self-supervised tasks are used for training BERT with large collections of texts:

- Mask filling (MASK)
- Next sequence prediction (CLS, SEP)

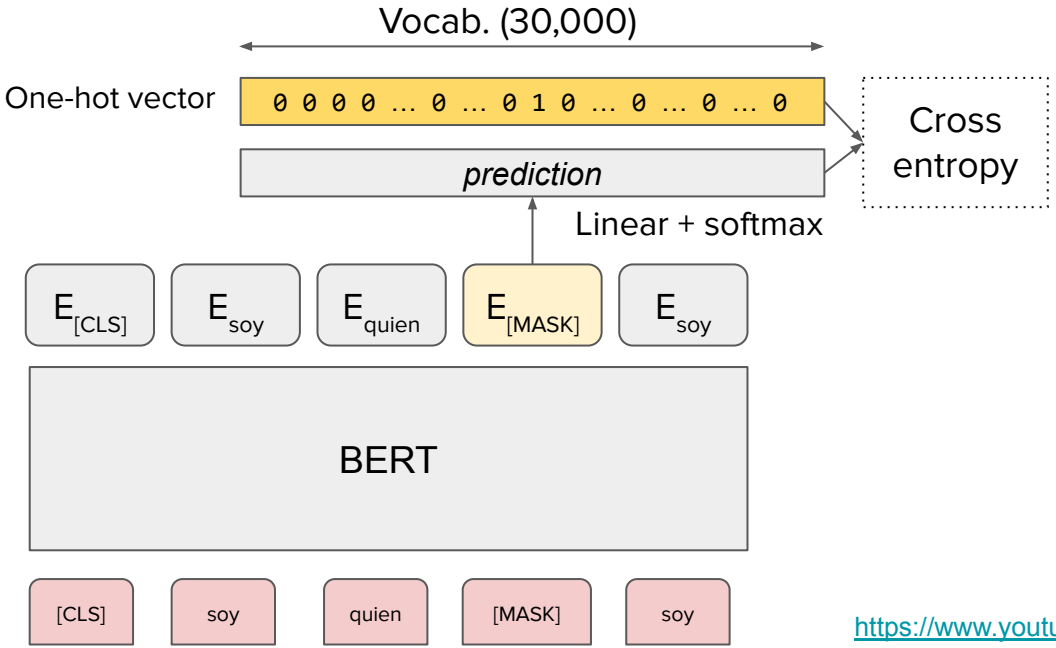


Self-attention mechanism (BERT base)



BERT- base has 12 attention heads like this. In the output they are concatenated.

MASK task



- Each token has an initial pre-defined *embedding* (GloVe)
- ~15% of the tokens are masked at each train sentence
- Loss: cross entropy

The result is a conditional distribution of tokens at each mask:

$$p(w_i \mid \ldots, w_{i-1}, w_{i+1}, \ldots)$$

https://www.youtube.com/watch?v=eMlx5fFNoYc

https://www.youtube.com/watch?v=9-Jl0dxWQs8

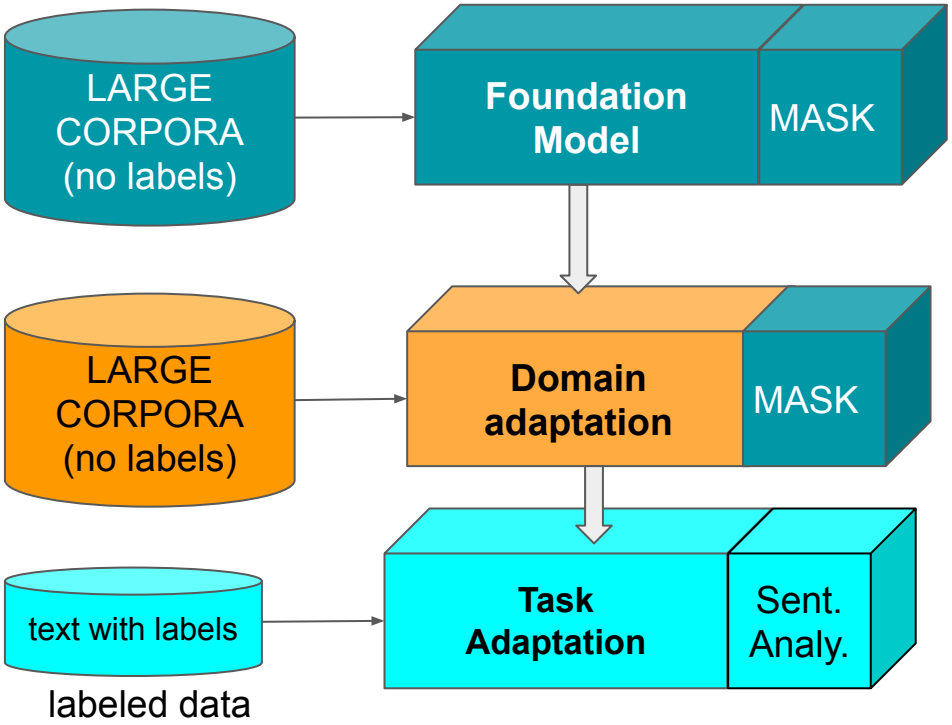
Pre-train & fine tune models

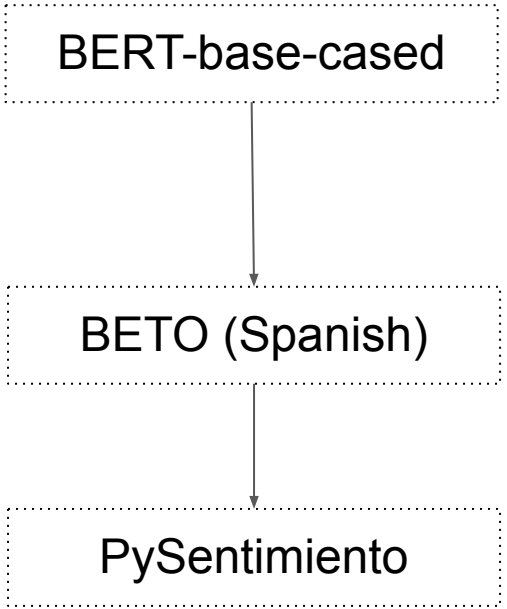
Applying Transformers

Pre-trained models (Foundation models) can be used to

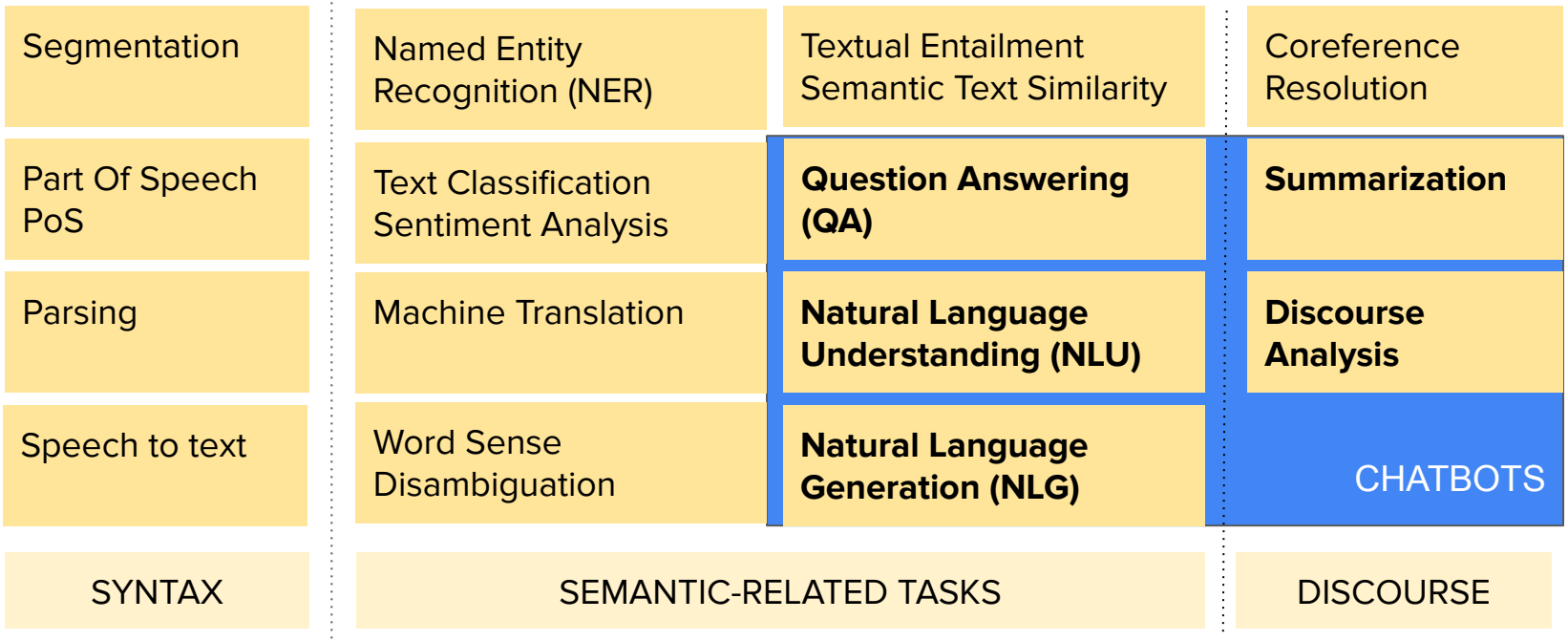
- generate word/sentence embeddings
- fine-tune to other languages, tasks, etc.

Transfer Learning - Fine Tuning

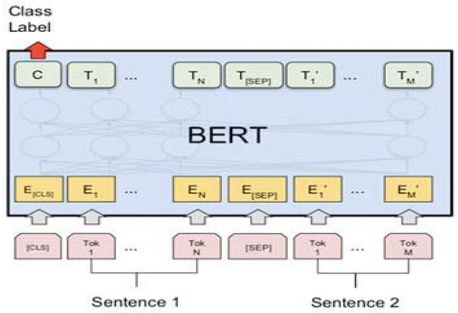




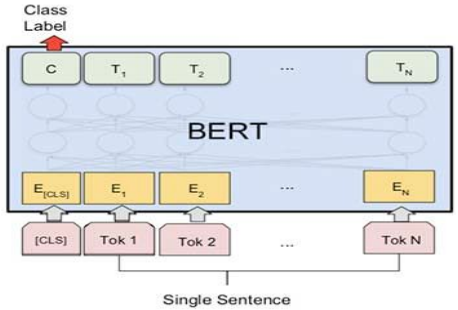
NLP “Downstream” Tasks



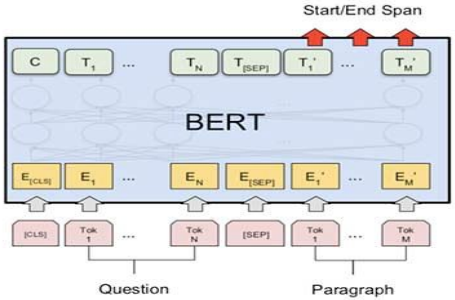
BERT: change the head and re-train the model



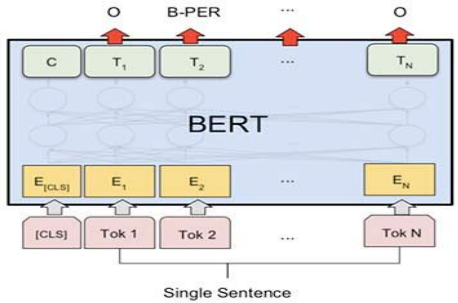
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



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



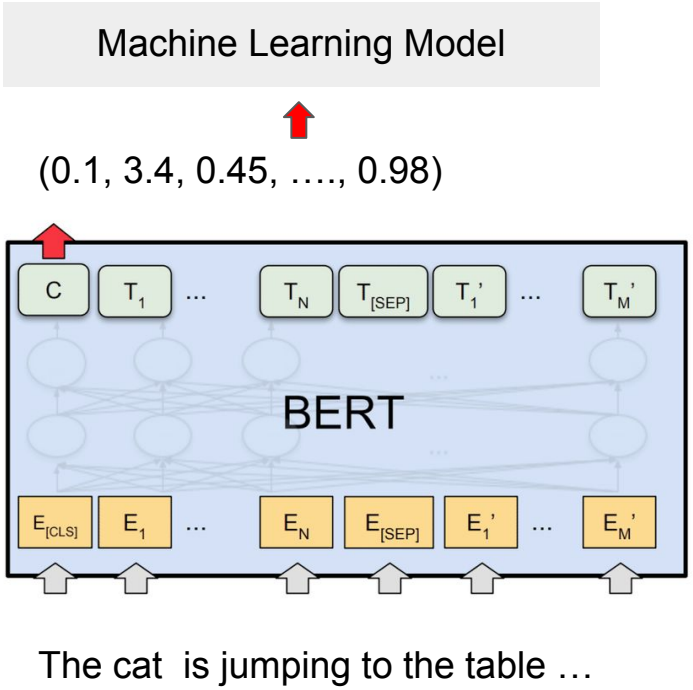
(c) Question Answering Tasks: SQuAD v1.1



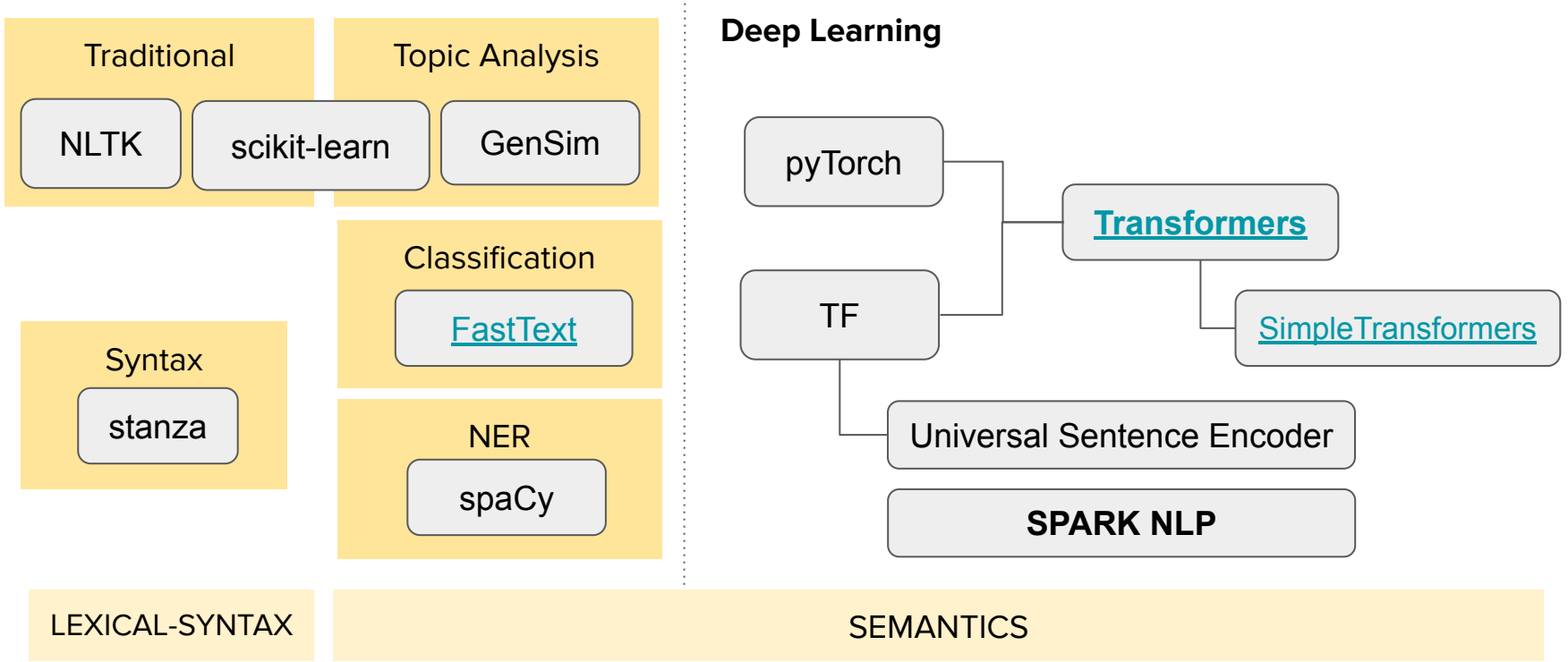
(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Encoders: text to vectors (embeddings)

- The text is compressed into a dense vector of size 768 real numbers.
- Related words are reinforced by the self-attention mechanism.
- Distance of two encoded texts are somehow related to their the semantic similarity.
- APIs like Cohere provide multilingual vectors as a service.
- Visual Transformers (ViT) allow us to vectorize images/texts (SBERT)



NLP and Python (libraries)



HOW TO WORK

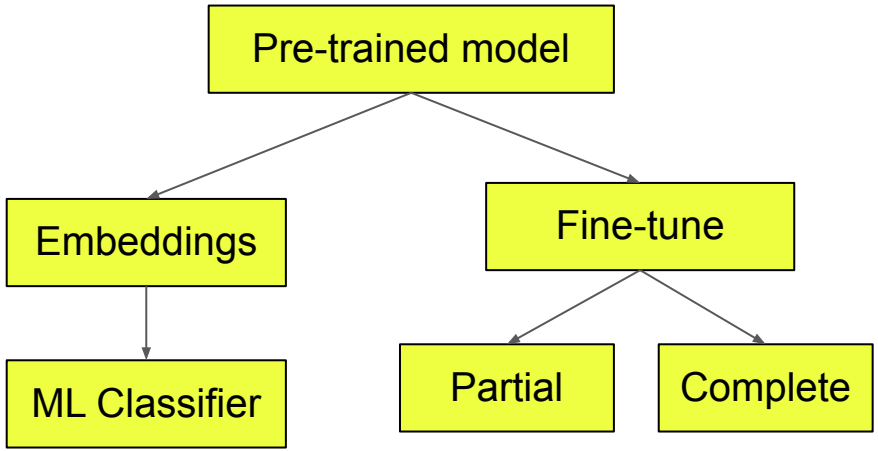
Hubs vs. APIs

Hubs provide models to be downloaded in our local programming environment

- TF Hub
- Huggingface transformers
- SPARK NLP

SAS/APIs provides embeddings and functionalities by means an API REST

- Huggingface API
- Cohere
- Open AI



Which model?

We must take into account:

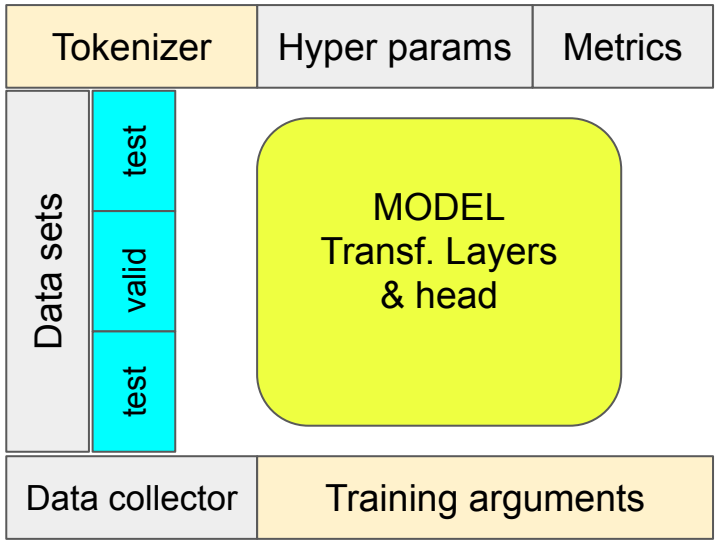
- Input max. size (BERT ~ 512 tkns, LongFormer ~ 4000 tkns)
- Model size (base vs. large)
- Language (English, monolingual, multilingual models)
- Task to be performed: text classification, NER, sentiment analysis, summarization, etc.
- Reduced model versions: distilled (encoders) and quantized (LLMs)

Example: longformer-base-4096-Spanish (huggingface)

RoBERTa Spanish (BERTIN) → reshaped as LongFormer

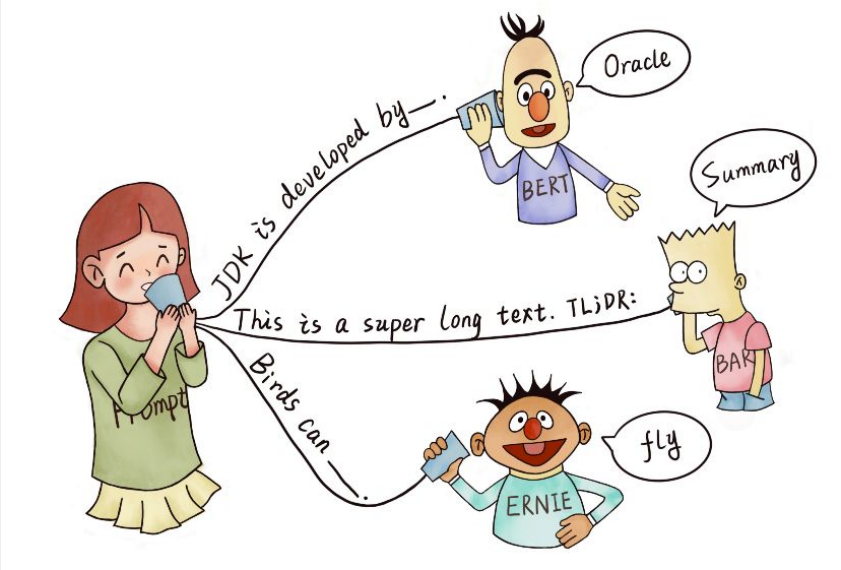
Transformer models

Trainer



Pre-train, Prompt & Predict

A new paradigm for NLP



Survey:
https://arxiv.org/abs/2107.13586

What is wrong with fine-tuning?

- A model for each different *downstream* task
  - In complex applications we need to handle a lot of LLMs, one for each task
  - New tasks require new labelled datasets and fine-tune another LLM
  - Inspect for example Huggingface Hub, where the number of models increase exponentially
- Catastrophic forgetting
  - Fine-tuning a model with new samples can lead the model to forget knowledge of the original dataset
  - To avoid it, it is necessary to include samples of the original model to maintain that knowledge in the LLM
- Dynamic decision making will require few- or zero-shot approaches
  - Use no labelled data to make the inference of the intended labels with the LLM
  - Use few examples to guide the LLM to solve the inference problem

Prompts

- A prompt is a chunk of text followed by a *slot* that must be filled with the desired inference.
- We assume two parts: an input (x) and the intended output (y)
- Both parts are expressed in natural language
- The output *y* will include the *slot* to be filled in
- Examples:
  - I missed the bus today. I felt so _____
  - English: I missed the bus today. French: _____
  - I love this movie. Overall, it was a ___ movie.

Templates

- We must also define the permissible values in the slot (Z)
  - These values are verbalizations of our intended classes:
    - positive $\in$ Y $\rightarrow$ {fantastic, great, …} $\in$ Z
    - negative $\in$ Y $\rightarrow$ {poor, bad, …} $\in$ Z
- Templates allow to abstract the tasks:
  - [X]. I felt so [Z]
  - [X]. Overall, it was a [Z] movie.
  - [X1]. [X2] is a [Z] entity. (Mike went to Paris. Paris is a location entity)
- The goal is to find the most likely predicted $z \in$ Z whose class $y \in$Y minimize the loss function.

Some approaches and examples

Zero-shot learning at Huggingface:

- https://huggingface.co/tasks/zero-shot-classification
- GPT-3 completion task

Few-shot learning:

- PET, ADAPET (prompting approaches)
- SetFit: https://huggingface.co/blog/setfit (without prompt)