Few-Shot Learning for Text Classification

Nowcasting and Forecasting with Text as Data

Renato Vassallo

Institute of Economic Analysis (IAE-CSIC) - Barcelona School of Economics

May, 2025

Copyright © by Renato Vassallo, IAE-CSIC and BSE

Outline

Introduction

About Me

Academic Background

- B.A. in Economics - UDEP (2016)
- M.A. in Economics - PUCP (2021)
- M.Sc. in Data Science - BSE (2023)

Contact Information

- : renato.vassallo@bse.eu
- in : linkedin.com/in/renatovassallo
- : github.com/RenatoVassallo

Goal: share what I know and make it useful for you.

BANCO CENTRAL DE RESERVA DEL PERÚ

CF Consejo Fiscal

/EconAI

BSE Barcelona School of Economics

Session 1 - Structure

1. **Block 1**: Concepts and code walkthrough.
    - Github repo: here.
2. **Block 2**: Hands-on activity. ▸ Go to Activity
    - Select a topic and apply any method discussed in Block 1.
    - Work in groups (max 4 members).
    - Duration: 30-45 minutes.
    - Brief presentation: 5 minutes per group.

Important notice

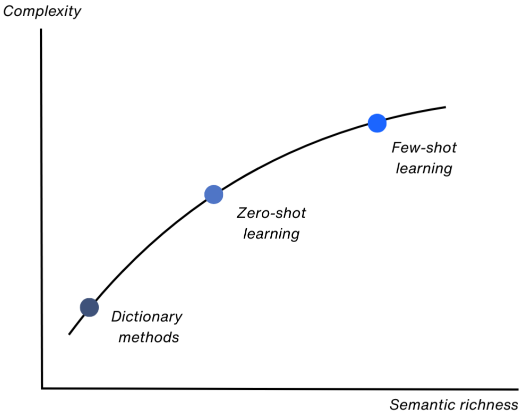
- The class materials, code, and methods are designed for teaching and illustrative purposes only.
- We will work with large Transformer models, so access to GPUs is highly recommended for optimal performance.
- You are encouraged to improve, extend, and optimize the routines — consider implementing parallelization where applicable.

Session 1 - Objectives

- Transforming text into numerical representations.
- Fine-tuning large language models (LLMs) for specific tasks.
- Leveraging state-of-the-art methods for text-based indicators.
- Real-world applications: Industry and academia.

How can we extract meaningful patterns from text data?

Semantic coverage

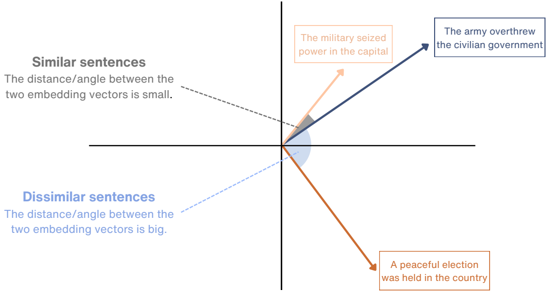


Embeddings and LLMs

What are embeddings?

When dealing with text data, how do we convert words and sentences into something that a model can understand?

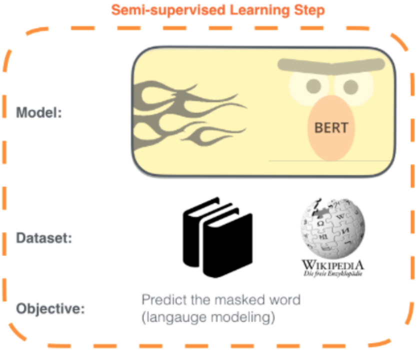
- Embeddings are dense vector representations of text.
- They capture semantic information in a continuous space.
- The closer two vectors are, the more similar the sentences are in meaning.



Source: Mayoral et al. (2025).

How embeddings are generated?

- BERT (Delvin et al., 2019) is pre-trained on large corpora.
- With pre-trained BERT we can directly obtain embeddings for any text.



Source: Alammar (2019).

BERT embedding similarity

Let's see how we can use BERT to quickly assess the similarity between two sentences:

```
from transformers import AutoTokenizer, AutoModel
import torch

tok = AutoTokenizer.from_pretrained("bert-base-uncased")
model = AutoModel.from_pretrained("bert-base-uncased")

emb1 = model(**tok("Cats are cute", return_tensors="pt")).last_hidden_state[0, 0]
emb2 = model(**tok("Dogs are loyal", return_tensors="pt")).last_hidden_state[0, 0]

print(torch.cosine_similarity(emb1, emb2, dim=0).item())
```

Downstream tasks

- But many NLP tasks like sentiment analysis, NER, Q/A, or text classification require task-specific models.
- BERT embeddings are generic; we can fine-tune a model to adapt to specific tasks.
- Let's look at how to train our own simple LM for text classification!

Go to Training simple LM

Scaling up: efficient alternatives to LLMs

- Ideally, we could leverage state-of-the-art models like OpenAI GPT-4, Anthropic Claude, or Google PaLM 2, but unlimited access is costly and resource-intensive.
- On the other hand, training large models from scratch requires substantial computational power, time, and expertise.
- Fortunately, **pre-trained models** provide effective starting points, tailored for various NLP tasks:
  - BERT Variants - fine-tuned BERT for specific tasks like sentiment analysis or NER.
  - Hugging Face Models Hub - extensive library of models for NLI, NER, Q/A, and more.
  - Cohere Models - focused on semantic understanding and text generation.

Sentence transformers

- Sentence Transformers: efficient embeddings for semantic search, clustering, etc.
- Lightweight models can provide a balance between performance and efficiency.

Inference Time Comparison for 100 Texts

Model	**Time (seconds)**
BERT	0.5728
SBERT	0.4985
all-MiniLM-L6-v2	0.0900

- **BERT:** Larger and slower due to more layers and parameters. Layers: 12; Embedding size: 768.
- **SBERT:** Optimized for sentence-level embeddings but still relatively heavy.
- **all-MiniLM-L6-v2:** Much smaller and faster, designed for speed without substantial accuracy loss. Layers: 6; Embedding size: 384.

Zero-Shot vs. Few-Shot Learning

1. **ZSL:** make predictions without having seen specific labels during training.
   - **Example:** predict the probability of a news headline referring to a coup d'état.
2. **FSL:** adds a **supervised layer** with a small set of labeled examples in a specific domain.
   - **Example:** predict the probability of a news headline referring to a coup d'état, given a few labeled examples of similar headlines.

	Zero-Shot	**Few-Shot**
Training data	None for specific labels	Few labeled examples
Flexibility	Broad/generalized	More specific
Accuracy	Reasonable	Higher (for seen classes)
Efficiency	Faster	Slower (additional training)

Zero-Shot Learning

What is Zero-Shot Learning (ZSL)?

- **Definition:** Zero-shot learning enables models to classify data without having seen specific labels during training.
- **How it works:**
  - Leverages semantic information (e.g., embeddings) to link known and unknown classes.
  - Uses pre-trained models like BERT, GPT, or SBERT to generate rich contextual embeddings.
- **Example:** If trained on animals (cats, dogs) but asked to classify birds, the model uses textual similarity to infer the correct class.

References:

- *A comprehensive review on zero-shot-learning techniques* (Lazaros et al., 2024)
- Hugging Face Zero-Shot Pipeline: link

Zero-Shot learning methodology

1. **Pre-trained model as encoder**
   - BERT, SBERT, GPT, etc., are used to generate embeddings for input text.
2. **Label embedding generation**
   - Transform labels into descriptive sentences or keywords.
   - Example: instead of "Positive", use "This text expresses positive sentiment".
3. **Similarity scoring**
   - Compute cosine similarity between input text embedding and each label embedding.
   - Assign label based on the highest similarity score.

Formal Notation

Goal: map documents to labels using similarity in embedding space.

- Given a query q (e.g., a news article), predict a label ℓ from a set $\mathcal{L}$.
- Apply an encoder to both the query and each label to obtain embeddings.
- Assign the label that maximizes similarity with the query embedding in latent space.

The predicted label $\hat{\ell}$ is defined as:

$$\hat{\ell}_{ZSL} = \arg\max_{\ell \in \mathcal{L}} \cos\left(\mathbf{E}(q), \mathbf{E}(\ell)\right)$$

Natural Language Inference (NLI)

- NLI considers two sentences: a *premise* and a *hypothesis*. The task is to determine whether the hypothesis is true (**entailment**) or false (**contradiction**) given the premise.

Premise	**Label**	**Hypothesis**
The cat is sleeping on the couch.	Contradiction	The cat is playing outside.
The company reported a rise in profits.	Neutral	The company launched a new product.
A group of people is protesting in the street.	Entailment	There is a protest happening.

Examples of NLI: Contradiction, Neutral, and Entailment

Go to NLI Architecture

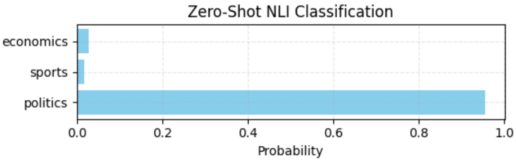
NLI in practice

- We will use pre-trained NLI models like `bart-large-mnli` and `roberta-large-mnli`.
  - These models are trained on large datasets like MNLI (Multi-Genre Natural Language Inference), containing over 430k pairs of sentences labeled as **entailment**, **neutral**, or **contradiction**.
- Methodology:
  - Take the input text as the **premise**.
  - Convert each candidate label into a natural language **hypothesis**.
  - The NLI model predicts whether the premise **entails**, is **neutral**, or **contradicts** each hypothesis.
- Interpretation:
  - If the model predicts **entailment**, we consider the label as likely true.
  - This method allows for classification without explicit training data for each label — a key strength of zero-shot learning.

Simple zero-shot classifier using NLI

```
text = "Opposition party gains ground ahead of national election."
labels = ["economics", "sports", "politics"]

ZeroShotNLI(tokenizer, model, text, labels, plot=True)
```



Output:

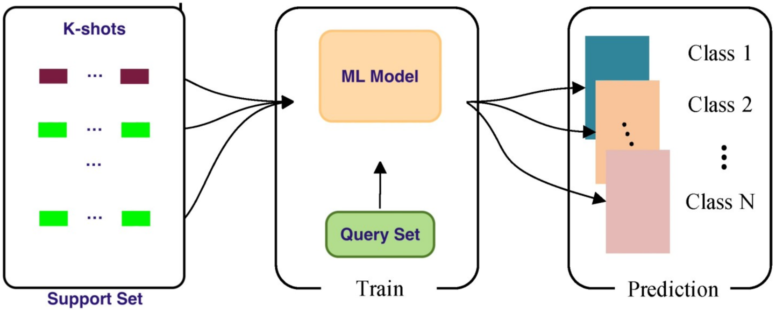
```
{'sequence': 'Opposition party gains ground ahead of national election.',
 'labels': ['economics', 'sports', 'politics'],
 'scores': [0.028, 0.017, 0.955]}
```

Few-Shot Learning

Why few-shot learning?

- There is a need to develop models that can generalise with limited data.
  - Medical diagnosis with few patient records.
  - Image classification with few labels (detecting deforestation, face recognition).
  - Rare event detection in news or official statements.
- Few-shot = Zero-shot + Supervised Layer.
  - Supervised layer: Fine-tunes with a small number of labeled examples.
- Be careful with **overfitting**!
  - Too few examples can lead to memorization rather than generalization.
  - Regularization techniques are crucial.

Few-shot flowchart



Source: Agrawal (2024), via LinkedIn.

A simple approach

Goal: learn a mapping between the documents and their labels.

- Weights, ($\mathbf{W}$), are learned through minimizing the loss function, as expressed below:

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} \left(\|\mathbf{X}^\top \mathbf{W} - \mathbf{Y}\|^2 + \lambda \|\mathbf{W} - \mathbb{I}\|^2 \right)$$

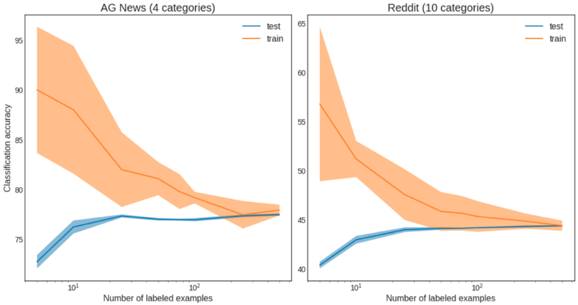
- *1st term*: tells $\mathbf{W}$ how to map X to Y. *2nd term*: the elements of the weight matrix are pushed towards the identity matrix.
- If few examples: $\mathbf{W}$ will likely be quite close to the identity matrix (FSL $\sim$ ZSL).
- If many examples: $\mathbf{W}$ will be pushed further away from the identity matrix.

The predicted label ($\hat{\ell}$) is obtained by maximizing the *similarity* between the query embedding and each label embedding, as follows:

$$\hat{\ell}_{FSL} = \arg\max_{\ell \in \mathcal{L}} \cos\left(\mathbf{E}(q)\mathbf{W}^*,\ \mathbf{E}(\ell)\mathbf{W}^*\right)$$

How many labels are enough?

- **AG News**: subset of 120k for training and 7.6k for testing. 4 categories.
- **Reddit**: subset of 540k for training and 60k for testing. 16 categories.



Source: Cloudera (2020).

Applications

Applications

Training a simple LM

Training Process Overview

- We use a simple dataset of sentences with categories: Learning, Pets, Coding.
  - (“`I love machine learning`”, `[1, 0, 0]`)
  - (“`Cats are cute`”, `[0, 1, 0]`)
  - (“`Python is great for programming`”, `[0, 0, 1]`)
- Model Architecture:
  - Pre-trained BERT model as the encoder.
  - A linear layer to project the CLS token to a 3D space.
  - MSE Loss to align embeddings with target vectors.

Training Process

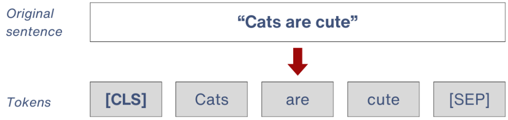
Original sentence

"Cats are cute"

- **Input Text**: We start with a simple sentence. This is just a string of words, but the model can't understand text directly.
- We need to convert it to numbers.

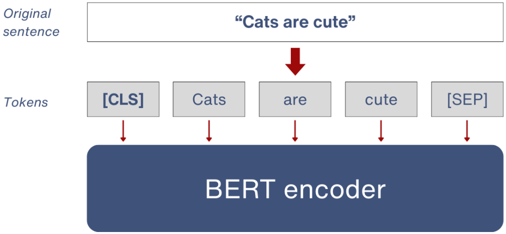
Training Process

- **Tokenization**: breaks down the sentence into individual words or subwords and assigns each a unique number (ID).
- These numbers are packed into a fixed-size array. This ensures that every input has the same length.



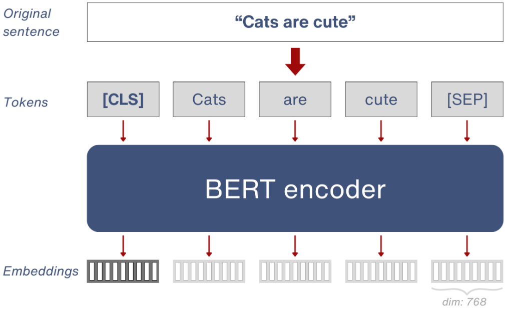
Training Process

- **BERT**: Think of BERT as a huge library of knowledge about language.
- It has been pre-trained on massive amounts of text to understand grammar, meaning, and context.



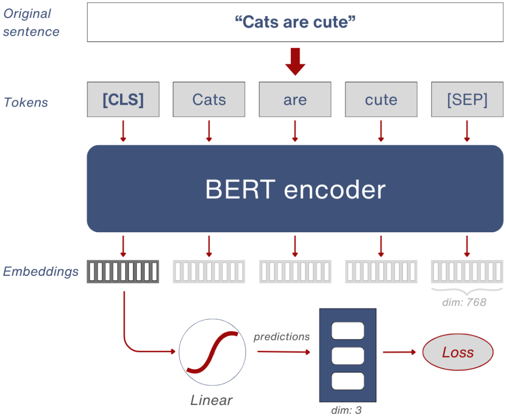
Training Process

- Takes in the tokenized numbers and converts them into meaningful vectors that capture the context of the words.
- **Example**: The word *Python* could mean a programming language or a snake.
- BERT can understand the context and represent it as a vector that reflects the intended meaning.



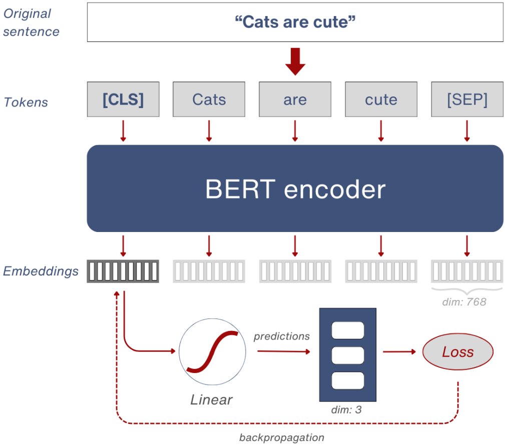
Training Process

- BERT outputs a vector of 768 dimensions for **each token**.
- The **linear layer** acts as a projector, reducing those 768 dimensions down to just **3 dimensions**.
- For *Cats are cute*, the target vector is $[0, 1, 0]$.
- The loss function defined in the model is Mean Squared Error (MSE).



Training Process ▸ Return

- The computed loss is then **backpropagated** through the model.
- The optimizer updates the parameters of the linear layer and potentially some of the BERT parameters.

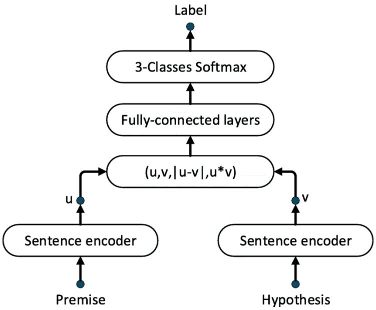


Applications

Natural Language Inference

NLI Model typical architecture ▸ Return

- NLI datasets are typically modeled via *sequence-pair classification*.
- The input premise and hypothesis are encoded (e.g. using BERT). The obtained vectors u and v are then concatenated along with their element-wise product and absolute difference.
- This representation captures information from both inputs. This vector is then passed to a 3-class classifier consisting of multiple fully-connected layers.



Source: Sadeghi et al. (2022).

Applications

Hands-on activity

Mini Project 1: political violence analysis

Data: ACLED, daily data.

- **Use case:** analyze the increase in riots and protests in Sudan. Characterize these events as peaceful or violent.
  - Visualizations: time series plots, heatmaps, geographical maps.
- **Predictive task:** divide data into train (until Dec 2021) and test (2022 onwards). Train a classifier to predict event types (e.g., riots/protests vs battles, explosions).
  - Products: ROC-AUC, PR-AUC, Confusion Matrices.
- **Open exploration:** identify other countries or event types. Use embeddings to extract patterns, but keep in mind computational limits.
- **Support:** if you need specific data for a country, ask and I can retrieve it.

Mini Project 2: Banking77 dataset analysis

Data: Banking77, intent classification dataset.

- **Use case:** analyze customer support requests to identify common intents such as fraud reporting, balance inquiry, or card blocking.
  - Visualizations: distribution of intents, time-based trends, sentiment analysis.
- **Predictive task:** train a few-shot learner to classify intents based on support requests.
  - Products: confusion matrices, precision-recall curves.
- **Open exploration:** use embeddings to cluster similar intents or identify misclassifications.
- **Support:** you can subset data to focus on specific intents to reduce training time.

Mini Project 3: extending econland analysis ▸ Return

Goal: extend the application of Lab 2 and apply more advanced techniques for economic analysis.

- Use few-shot learning and compare its performance against zero-shot predictions.
- Replace the AR(1) model with your favorite ML regressor (RF, CatBoost, etc).
- Apply a rolling forecast with TimeSeriesSplit to prevent data leakage.
- Visualize predictions and compute error metrics.

Thank You

Questions?