Advanced Methods in Natural Language Processing

Session 6: Few Shot Learning with BERT models

Arnault Gombert

May 2025

Barcelona School of Economics

Introduction

Introduction to Today's Lecture

Today, we dive into Zero/Few-Shot Learning (ZSL/FSL), understanding its implications in NLP and how BERT-like models facilitate these learning paradigms.

Session Overview:

- **Understanding ZSL/FSL:** Unveiling the concepts and their significance in overcoming data challenges.
- **BERT's Role:** Exploring why BERT-like models are apt for ZSL/FSL.
- **Methods and Approaches:** Delving into strategies like Cloze Tasks and Weak Learning Principles.
- **Practical Implementation:** Step-by-step guides on applying ZSL/FSL with BERT-like models.
- **Advanced Topics:** Exploring the future and addressing ethical considerations in ZSL/FSL.

Why few shot learning?

The Predicament of Labelled Data:

- **Scarce Resources:** Good quality labeled data is often a luxury, not a norm, in real-world applications.

Why few shot learning?

The Predicament of Labelled Data:

- **Scarce Resources:** Good quality labeled data is often a luxury, not a norm, in real-world applications.
- **Cost and Effort:** Acquiring, cleaning, and labeling data demands significant resources, time, and expert intervention.

Why few shot learning?

The Predicament of Labelled Data:

- **Scarce Resources:** Good quality labeled data is often a luxury, not a norm, in real-world applications.
- **Cost and Effort:** Acquiring, cleaning, and labeling data demands significant resources, time, and expert intervention.

Why few shot learning?

The Predicament of Labelled Data:

- **Scarce Resources:** Good quality labeled data is often a luxury, not a norm, in real-world applications.
- **Cost and Effort:** Acquiring, cleaning, and labeling data demands significant resources, time, and expert intervention.

Human Cognition:

- **Learning from Few Examples:** The human brain excels at making inferences from minimal information. For instance, children can understand new words or concepts from just a few examples.

Why few shot learning?

The Predicament of Labelled Data:

- **Scarce Resources:** Good quality labeled data is often a luxury, not a norm, in real-world applications.
- **Cost and Effort:** Acquiring, cleaning, and labeling data demands significant resources, time, and expert intervention.

Human Cognition:

- **Learning from Few Examples:** The human brain excels at making inferences from minimal information. For instance, children can understand new words or concepts from just a few examples.
- **Adaptability:** Our cognitive flexibility allows us to understand and adapt to new situations swiftly, a trait we aim to instill in AI models.

Why few shot learning?

BERT-like Models - A Beacon of Hope:

- BERT-like models, with their deep contextual understanding and transfer learning capabilities, offer a promising solution to the scarcity of labeled data.
- These models' ability to generalize and adapt with minimal examples aligns with the goal of achieving human-like flexibility in AI.

Today's session explores how BERT-like models leverage these principles to perform Zero/Few-Shot Learning, making significant strides in NLP.

Few Shot Learning

Overview of Zero/Few-Shot Learning

Zero/Few-Shot Learning - Learning from Few or No Examples:

- **Zero-Shot Learning (ZSL)**: The model makes predictions on cases that it has not seen during training.
- **Few-Shot Learning (FSL)**: The model learns to make accurate predictions from a very small number of examples, typically less than a few dozen.

Significance and Applications:

- **Data Scarcity**: ZSL and FSL are particularly valuable in scenarios where labeled data is scarce or expensive to obtain.
- **Rapid Adaptation**: These approaches enable models to quickly adapt to new tasks or domains with minimal training data.
- **Transfer Learning**: ZSL/FSL use models with general knowledge

Overview of Zero/Few-Shot Learning

Key References and Their Contributions:

1. Miller et al. (1976). "The influence of pattern similarity and transfer learning upon the training of a base perceptron B2."
    - *Summary:* Explores the impact of pattern similarity and transfer learning in training perceptron models, laying early groundwork for concepts used in few-shot learning.
2. Fei-Fei et al. (2006). "One-shot learning of object categories."
    - *Summary:* Introduces a Bayesian framework for one-shot learning, significantly advancing the field by demonstrating that sophisticated learning tasks can be achieved with very few examples.
3. Chang et al. (2008). "Importance of Semantic Representation: Dataless Classification".
    - *Summary:* Highlights the significance of semantic representation in the absence of labeled data, proposing a framework for dataless classification that leverages semantic similarities.

BERT Models, pefect fit for FSL?

BERT's benefits:

- **BERT's Pre-training:** Extensive pre-training on large text corpora allows BERT models to develop a nuanced understanding of language, akin to a rich, multi-faceted learning experience.
- **Word Representations:** BERT's deep contextual embeddings capture subtle semantic and syntactic nuances, offering a robust foundation for generalization from few examples.
- **Fine-tuning:** The fine-tuning step is much less resources intensive, enables the model to adapt efficiently to downstream tasks with limited amount of data.

Language Models as Few-Shot Learners

LLMs are few shot learners, Brown et al. (2020):

- **Groundbreaking Results:** GPT-3 can perform a variety of tasks with few or no task-specific training examples.
- **Performance:** GPT-3 achieves strong performance across a broad spectrum of NLP tasks, including translation, question-answering, and cloze tasks, with just a few examples provided in the prompt.

Language Models as Few-Shot Learners

Setting	PIQA	ARC (Easy)	ARC (Challenge)	OpenBookQA
Fine-tuned SOTA	79.4	**92.0**[KKS+20]	**78.5**[KKS+20]	**87.2**[KKS+20]
GPT-3 Zero-Shot	**80.5***	68.8	51.4	57.6
GPT-3 One-Shot	**80.5***	71.2	53.2	58.8
GPT-3 Few-Shot	**82.8***	70.1	51.5	65.4

Table 3.6: GPT-3 results on three commonsense reasoning tasks, PIQA, ARC, and OpenBookQA. GPT-3 Few-Shot PIQA result is evaluated on the test server. See Section 4 for details on potential contamination issues on the PIQA test set.

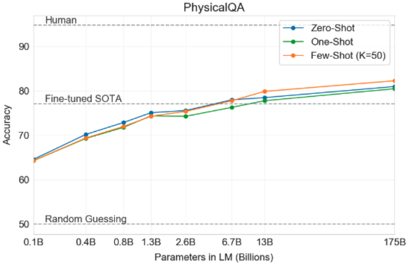


Figure 3.6: GPT-3 results on PIQA in the zero-shot, one-shot, and few-shot settings. The largest model achieves a score on the development set in all three conditions that exceeds the best recorded score on the task.

GPT-3 Few-Shot Learning Performance

Credit: Brown et al. (2020)

Language Models as Few-Shot Learners

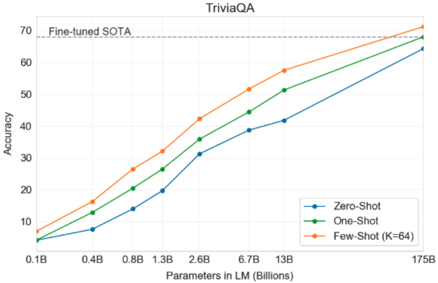


Figure 3.3: On TriviaQA GPT3's performance grows smoothly with model size, suggesting that language models continue to absorb knowledge as their capacity increases. One-shot and few-shot performance make significant gains over zero-shot behavior, matching and exceeding the performance of the SOTA fine-tuned open-domain model, RAG [LPP+20]

GPT-3 Few-Shot Learning Performance

Credit: Brown et al. (2020)

Conclusion: The paper marks a significant milestone in NLP, showcasing LLMs' aptitude for Zero/Few-Shot Learning by their ability to generalize and adapt to new tasks with minimal task-specific data.

Few Shot Learning Methods

Exploring Few-Shot Learning Methods

Diverse Approaches in Few-Shot Learning:

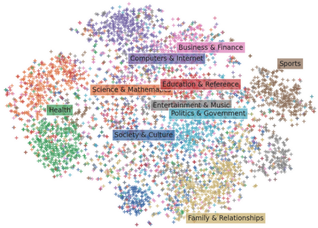
- **Latent Embedding Approach:** Employs embedding spaces that capture semantic relationships.
- **Natural Language Inference (NLI):** Leverages linguistic relationships between sentences to infer information.
- **Classification as Cloze Task:** Adapts the cloze test concept, where models predict missing information in a sentence.
- **Dataset Generation:** Generate from scratch datasets features and labels to train models, capitalizing on larger models.
- **Contrastive Learning:** Employs pair-wise comparisons to learn: distinguishing between similar and dissimilar instance.

Note: This list is not exhaustive but serves to highlight a variety of approaches in Few-Shot Learning, each offering unique insights and methodologies to tackle the challenges of learning from limited data.

Latent Embedding Approach for Zero-Shot Learning

Leveraging BERT's deep understanding of language:

- **Sentence Embedder:** Transform sentences into contextual embeddings.
- **Class Embeddings:** Embeds a list of predefined classes, enabling comparison in the embedding space.
- **Similarity Computation:** Measures similarity between sentence and class embeddings and pick the most relevant class for a given sentence.
- **Alignment:** Aligns word and sentence embeddings, through projection techniques to ensure compatibility.



t-SNE visualization of embeddings with SBERT to Wordvec projection. This extra projection step results in labels which appear much closer to their corresponding data clusters compared to the previous visual.

Credit: Davison (2020)

F1 of 46.9 on Yahoo dataset

Implementing Zero-Shot Learning with Hugging Face

Python Code Example

```
from transformers import pipeline

# Load zero-shot classification pipeline
classifier = pipeline("zero-shot-classification")

# Define the sequence to classify
sequence_to_classify = "The discovery of exoplanets has expanded our knowledge."

# Define the candidate labels
candidate_labels = ["education", "politics", "science"]

# Perform zero-shot classification
results = classifier(sequence_to_classify, candidate_labels)

# Print the classification results
print(results)
```

Classification via Natural Language Inference

NLI involves classifying the relationship between a premise and a hypothesis into categories like entailment, contradiction, or neutral:

- **Premise:** A statement/assertion (e.g., "BERT model is made of transformer blocks").
- **Hypothesis:** A proposition/laim (e.g., "This text talks about science").
- **Inference:** Determining if the hypothesis is true (entailment), false (contradiction), or undetermined (neutral) based on the premise.

Yin et al. (2019) leveraged this approach for classifications using a pre-trained MNLI model.

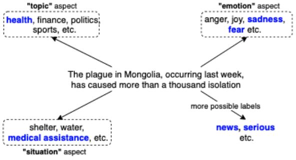


Figure 1: A piece of text can be assigned labels which describe the different aspects of the text. Positive labels are in blue.

Credit: Yin et al. (2019)

F1 of 37.9 on Yahoo dataset

Implementing Zero-Shot Learning with Hugging Face

Python Code Example

```
from transformers import pipeline

# Load zero-shot classification pipeline
nli_classifier = pipeline("zero-shot-classification")

# Define the premise and hypothesis
premise = "BERT model is made of transformer blocks."
hypotheses = ["This text talks about science.",
              "This text is about cooking.",
              "This text discusses technology."]

# Perform zero-shot classification
results = nli_classifier(premise, hypotheses)

# Print the classification results
print(results)
```

Classification Leveraging the Cloze Task Approach

The Cloze task approach involves creating a fill-in-the-blank style question, where the model predicts the missing word or phrase:

- **Prompting as a Cloze Task:** Leverages BERT's MLM capabilities.
- **Example:** Given a sentence "Best pizza ever. It was [MASK].", BERT predicts the masked word.
- **Focused Prediction:** The model can be guided to predict from a specific set of tokens, aligning the task with classification objectives.

This approach transforms classification tasks into a language modeling problem.

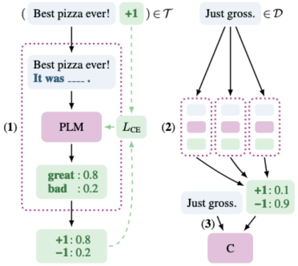


Figure 1: PET for sentiment classification. **(1)** A number of patterns encoding some form of task description are created to convert training examples to cloze questions; for each pattern, a pretrained language model is finetuned. **(2)** The ensemble of trained models annotates unlabeled data. **(3)** A classifier is trained on the resulting soft-labeled dataset.

Cloze Task Example

Implementing Cloze Tasks with Hugging Face for ZSL

Python Code Example

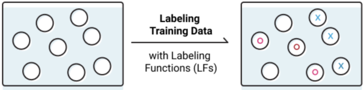
```
from transformers import pipeline

# Load fill-mask pipeline
fill_mask = pipeline("fill-mask")

# Define the cloze-style prompt
prompt = "Best pizza ever. It was [MASK]."

# Perform prediction
results = fill_mask(prompt)

# Print top 5 predictions for the masked token
for result in results:
    print(f"Token: {result['token_str']}, Score: {result['score']:.4f}")
```

Weak Learning in NLP

Generating noisy labels for training data.

Snorkel is a prominent framework worth using:

- **Labeling Functions:** Domain-specific heuristics for programmatically labeling data, reducing manual effort.
- **Modeling Label Noise:** Snorkel assesses and correlates the accuracy of labeling functions, refining their collective output.
- **NLP Applications:** Employs weak supervision in tasks like text classification and entity recognition, where acquiring labeled data is challenging.



Snorkel Framework

This approach allows for quick development, to benefit from large amounts of weakly-labeled data.

Cloze Questions for FSL: Schick & Schutze (2020)

Schick & Schutze (2020) propose an innovative approach for FSL by exploiting cloze questions:

- **Multiple Patterns:** Employs 6 cloze question templates to generate labels.
- **Iterative Refinement:** Utilizes an iterative process where the model's predictions refine subsequent LLM.
- **Leveraging FSL:** Significant improvements when fine-tuning the LLM with few examples at first.

Why It Excels: Compared to other few-shot learning methods, this approach integrates multiple perspectives through diverse cloze patterns and iteratively hones the model's understanding, making it particularly powerfull. F1 on Yahoo: 70% !

Cloze Questions for FSL: Schick & Schutze (2020)

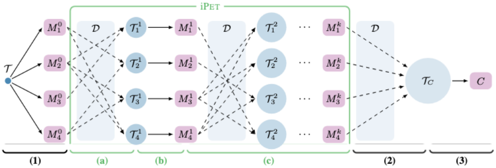


Figure 2: Schematic representation of PET (1-3) and iPET (a-c). **(1)** The initial training set is used to finetune an ensemble of PLMs. **(a)** For each model, a random subset of other models generates a new training set by labeling examples from $\mathcal{D}$. **(b)** A new set of PET models is trained using the larger, model-specific datasets. **(c)** The previous two steps are repeated k times, each time increasing the size of the generated training sets by a factor of d. **(2)** The final set of models is used to create a soft-labeled dataset $\mathcal{T}_C$. **(3)** A classifier C is trained on this dataset.

iPET Schema

Line	Examples	Method	Yelp	AG's	Yahoo	MNLI (m/mm)
1		unsupervised (avg)	33.8 $\pm$9.6	69.5 $\pm$7.2	44.0 $\pm$9.1	39.1 $\pm$4.3 / 39.8 $\pm$5.1
2	$\lvert\mathcal{T}\rvert = 0$	unsupervised (max)	40.8 $\pm$0.0	79.4 $\pm$0.0	56.4 $\pm$0.0	43.8 $\pm$0.0 / 45.0 $\pm$0.0
3		iPET	**56.7** $\pm$0.2	**87.5** $\pm$0.1	**70.7** $\pm$0.1	**53.6** $\pm$0.1 / **54.2** $\pm$0.1
4		supervised	21.1 $\pm$1.6	25.0 $\pm$0.1	10.1 $\pm$0.1	34.2 $\pm$2.1 / 34.1 $\pm$2.0
5	$\lvert\mathcal{T}\rvert = 10$	PET	52.9 $\pm$0.1	87.5 $\pm$0.0	63.8 $\pm$0.2	41.8 $\pm$0.1 / 41.5 $\pm$0.2
6		iPET	**57.6** $\pm$0.0	**89.3** $\pm$0.1	**70.7** $\pm$0.1	**43.2** $\pm$0.0 / **45.7** $\pm$0.1
7		supervised	44.8 $\pm$2.7	82.1 $\pm$2.5	52.5 $\pm$3.1	45.6 $\pm$1.8 / 47.6 $\pm$2.4
8	$\lvert\mathcal{T}\rvert = 50$	PET	60.0 $\pm$0.1	86.3 $\pm$0.0	66.2 $\pm$0.1	63.9 $\pm$0.0 / 64.2 $\pm$0.0
9		iPET	**60.7** $\pm$0.1	**88.4** $\pm$0.1	**69.7** $\pm$0.0	**67.4** $\pm$0.3 / **68.3** $\pm$0.3
10		supervised	53.0 $\pm$3.1	86.0 $\pm$0.7	62.9 $\pm$0.9	47.9 $\pm$2.8 / 51.2 $\pm$2.6
11	$\lvert\mathcal{T}\rvert = 100$	PET	61.9 $\pm$0.0	88.3 $\pm$0.1	69.2 $\pm$0.0	74.7 $\pm$0.3 / 75.9 $\pm$0.4
12		iPET	**62.9** $\pm$0.0	**89.6** $\pm$0.1	**71.2** $\pm$0.1	**78.4** $\pm$0.7 / **78.6** $\pm$0.5
13	$\lvert\mathcal{T}\rvert = 1000$	supervised	63.0 $\pm$0.5	**86.9** $\pm$0.4	70.5 $\pm$0.3	73.1 $\pm$0.2 / 74.8 $\pm$0.3
14		PET	**64.8** $\pm$0.1	**86.9** $\pm$0.2	**72.7** $\pm$0.0	**85.3** $\pm$0.2 / **85.5** $\pm$0.4

Table 1: Average accuracy and standard deviation for RoBERTa (large) on Yelp, AG's News, Yahoo and MNLI (m:matched/mm:mismatched) for five training set sizes $\lvert\mathcal{T}\rvert$.

Credit:Schick & Schutze (2020)

Synthesizing Datasets with Language Models

Why not synthesizing datasets from scratch ?

- **Leveraging LLMs:** Utilize the extensive knowledge of models like GPT-3.
- **Style Replication:** Harness the models' ability to mimic various writing styles.
- **Directed Generation:** Guide content generation to get desired categories.

Schick & Schutze (2021) demonstrated it:

- **Instructive Prompts:** Generate movie titles and reviews.
- **Synthesizing Data:** Creating a balanced dataset of reviews, for sentiment analysis tasks, outperforming GPT3 with a RoBERTa model!

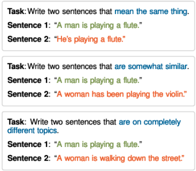


Figure 1: Continuations generated by GPT2-XL with DINO for three different task descriptions. We investigate two different unsupervised approaches to generating sentence-similarity datasets: (i) The input sentence is given and only the continuation is generated. This requires that an (unlabeled) set of sentences is available. (ii) Both input sentence and continuation are generated. This does not rely on the availability of any resources.

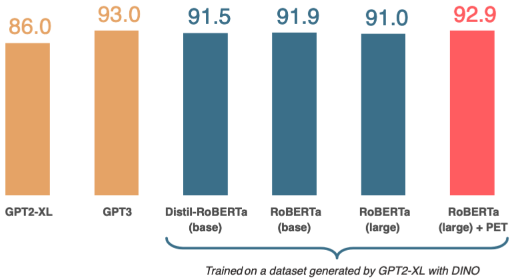
Credit: Schick & Schutze (2021)

Roberta + PET equals GPT-3

DINO Results on Synthetic Datasets

Training smaller models on synthetic datasets generated by GPT2-XL:

- **Model Variants:** Distil-RoBERTa (base), RoBERTa (base), and RoBERTa (large) were fine-tuned.
- **Enhanced Performance:** All models trained on GPT2-XL generated datasets notably outperformed zero-shot GPT2-XL.
- **Prompting with PET:** RoBERTa (large) with prompting approached GPT3's performance.



Contrastive Learning in NLP

- **Learning by Comparison:** Models learn to embed similar items closely in vector space while pushing dissimilar items apart.
- **Positive/Negative Samples:** Involves distinguishing between 'positive' (similar) and 'negative' (dissimilar) sample pairs.
- **Triplet Loss Function:** Introduced by Schroff et al. (2015), this loss function is pivotal in contrastive learning. Defined as:

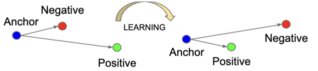


Figure 3. The **Triplet Loss** minimizes the distance between an *anchor* and a *positive*, both of which have the same identity, and maximizes the distance between the *anchor* and a *negative* of a different identity.

Triplet Loss

Credit: Schroff et al. (2015)

$$L = \max(d(a, p) - d(a, n) + margin, 0)$$

Where $d(a, p)$ and $d(a, n)$ are the distances between the anchor-positive and anchor-negative pairs, respectively, and *margin* is a user-defined separation margin.

SetFit: FSL with Contrastive Learning

SetFit, Tunstall et al. (2022), leverages Constrative Learning for FSL:

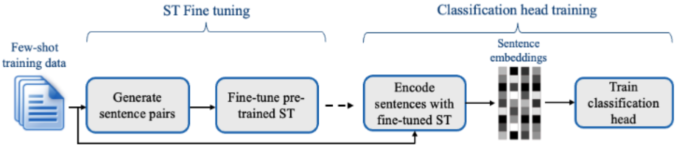


Figure 2: SETFIT 's fine-tuning and training block diagram.

Credit: Tunstall et al. (2022)

- **Contrastive Learning Approach:** Fine-tune Sentence Transformer via constrastive learning.
- **Fine-Tuning a Classifier**: Training a classifier head on the embeddings generated from the fine-tuned Sentence Transformer
- **Minimizing Data Requirements:** Demonstrates remarkable effectiveness with 8 to 16 examples

SetFit: FSL with Contrastive Learning

Rank	Method	Accuracy	Model Size
2	T-Few	75.8	11B
4	Human Baseline	73.5	N/A
6	SetFit (Roberta Large)	71.3	355M
9	PET	69.6	235M
11	SetFit (MP-Net)	66.9	110M
12	GPT-3	62.7	175 B

Credit: Tunstall et al. (2022)

- **RAFT Benchmark:** Outperforms PET and GPT-3. Nearly matches human performance and T-few model.
- **Speed and Cost Efficiency:** Remarkably fast training times (30s on an NVIDIA V100) and low costs.
- **GPU and CPU Compatibility:** Compatible with single GPUs and even CPUs, making it accessible for a wide range of users.

Implementing SetFit with Hugging Face in Python

Python Code: SetFit Training

```
from datasets import load_dataset
from sentence_transformers.losses import CosineSimilarityLoss
from setfit import SetFitModel, SetFitTrainer

dataset = load_dataset("SetFit/SentEval-CR")
# Select N examples per class (8 in this case)
train_ds = dataset["train"].shuffle(seed=42).select(range(8 * 2))
test_ds = dataset["test"]

# Load SetFit model from Hub
model = SetFitModel.from_pretrained("sentence-transformers/paraphrase-mpnet-base-v2
# Create trainer
trainer = SetFitTrainer(
    model=model, train_dataset=train_ds, eval_dataset=test_ds,
    loss_class=CosineSimilarityLoss, batch_size=16,
    num_iterations=20, # Number of text pairs to generate for contrastive learning
    num_epochs=1 # Number of epochs to use for contrastive learning
)
```

Implementing SetFit with Hugging Face in Python

Python Code: SetFit Inference

```
trainer.train()
trainer.push_to_hub("setfit_finetuned")

# Load the fine-tuned model
model = SetFitModel.from_pretrained("setfit_finetuned")

# Create a pipeline for text classification
classifier = pipeline("text-classification", model=model)

# Run inference
predictions = classifier("Example text for classification")
```

QA and Takeaways

QA

Open Discussion

- Feel free to ask questions or share your thoughts about today's topics.
- Any insights, experiences, or perspectives you'd like to discuss are welcome.

Summary of Key Takeaways

- **Introduction to Few-Shot Learning:** We discussed the significance of few-shot learning in NLP, addressing the challenge of limited labeled data and its alignment with human learning efficiency.
- **BERT's Adaptability:** Explored how BERT and similar large language models (LLMs) are exceptionally suited for few-shot learning, leveraging their extensive pre-trained knowledge.
- **Innovative Few-Shot Methods:** Examined various few-shot learning approaches including NLI, latent embedding, cloze tasks, and contrastive learning, highlighting their unique contributions to the field.
- **SetFit's Breakthrough:** Unveiled SetFit's remarkable performance, combining contrastive learning with few-shot principles, achieving high accuracy with minimal training data and at a lower computational cost.