Sentiment analysis- Possible Models/Frameworks we can use

Since we'll be using mainly pretrained models let's have some context on them before we go to specific models we can use for zero/few-shot learning:

Architecture of Pretrained Language Models

Pretrained language models, such as GPT (Generative Pre-trained Transformer) and BERT (Bidirectional Encoder Representations from Transformers), serve as the foundation for zero- and few-shot learning in NLP. These models are typically based on the transformer architecture, which has revolutionized the field of NLP.

Transformer Architecture: The transformer architecture comprises multiple layers of self-attention mechanisms and feedforward neural networks. This architecture allows pretrained models to capture contextual information efficiently, making them highly effective for various NLP tasks.

Embeddings: Central to pretrained models are word embeddings, which represent words as dense vectors in a continuous space. These embeddings enable the model to understand the semantic relationships between words and phrases.

Transfer Learning in Pretrained Models

Transfer learning is the cornerstone of pretrained language models. It involves pretraining a model on a massive corpus of text and then fine-tuning it for specific downstream tasks. This process enables pretrained models to acquire a broad understanding of language and context, which they can apply to various tasks.

The mathematics of transfer learning in pretrained models can be summarized as follows:

Pretraining: During pretraining, a model is trained on a massive amount of unlabeled text. The primary objective is to predict the next word in a sentence, which is a language modeling task. This is typically done using maximum likelihood estimation. The likelihood of observing a sequence of words is maximized using the model's parameters.

Mathematically, this can be represented as maximizing the product of conditional probabilities of words given their preceding context:

$P(w_1, w_2, ..., w_n) = \Pi P(w_i \mid w_1, w_2, ..., w_{(i-1)})$

Fine-Tuning: After pretraining, the model is fine-tuned on specific downstream tasks. Fine-tuning involves training the model on labeled data for the target task, adapting its parameters to optimize task-specific objectives.

Mathematically, this can be represented as minimizing a task-specific loss function $L(\theta)$, where θ represents the model's parameters:

$\theta^* = \text{argmin } L(\theta)$

Transfer learning allows pretrained models to transfer knowledge acquired during pretraining to a wide range of tasks, effectively capturing the general language understanding that underlies many NLP problems.

Zero- and Few-Shot Learning Strategies

Zero- and few-shot learning in NLP rely on several strategies to enable models to generalize to new tasks or make predictions with minimal examples:

Prompt Engineering: For zero-shot learning, crafting informative prompts is crucial. A prompt is a textual description of the task or a query that instructs the model on what to do. The challenge is in designing prompts that effectively guide the model to perform the desired task based on its pretrained knowledge.

Mathematically, prompt engineering involves designing prompts P(prompt) that maximize the likelihood of generating the correct output:

P(output | prompt)

Few-Shot Learning with Examples: In few-shot learning, providing a small number of examples in the task description is essential. These examples serve as guidance for the model to generalize from. The model learns to find patterns and relationships between the examples and the task.

Mathematically, this can be represented as using examples to condition the model's output:

P(output | examples)

Task Descriptions: In some cases, a task description can be provided alongside examples or prompts. This description helps the model understand the task's context and requirements. The challenge lies in encoding task descriptions effectively.

Mathematically, incorporating task descriptions involves conditioning the model on the task description:

P(output | task description)

Open source models on github we can utilize:

BART: **for zero-shot learning (https://github.com/facebookresearch/fairseq/tree/main/examples/bart#bart-denoising-sequence-to-sequence-pre-training-for-natural-language-generation-translation-and-comprehension)**

At the core of BART lies the transformer architecture, which has become the de facto standard for many NLP tasks. Like BERT, BART consists of multiple transformer layers, but its distinguishing feature is the pre-training objective. While BERT employs a masked language model (MLM) objective, BART takes a different route. BART's objective is to predict missing or corrupted portions of a text, effectively transforming it into a denoising autoencoder.

Pre-training Objective

BART's pre-training consists of two main steps: the masked language model (MLM) step and the conditional sequence-to-sequence denoising step.

Masked Language Model (MLM): The MLM step is reminiscent of BERT. In this step, BART randomly masks out words or tokens in a given input sequence and tasks the model with predicting these masked tokens. The aim is to make the model learn contextual information by understanding the relationships between words in a sentence. This step ensures that BART, like BERT, is bidirectional in its understanding of the text.

Conditional Sequence-to-Sequence Denoising: The second pre-training step is where BART truly shines. Unlike BERT, which only focuses on individual tokens, BART operates at the sequence level. It takes a sequence of text and corrupts it by removing spans of text, shuffling sentences, or even inverting them. The model is then tasked with the challenging job of reconstructing the original sequence, taking into account both the context and the removed or shuffled portions. This auto-regressive denoising task pushes BART to understand not just the content of the text but also its structure and flow.

Mathematical Insights

Let's delve into the mathematical underpinnings of BART's pre-training objectives:

In the MLM step, if we denote the input sequence as X and the masked tokens as M, the objective is to maximize the conditional probability of predicting the masked tokens given the unmasked context. Mathematically, this can be expressed as:

$P(M \mid X)$

The conditional sequence-to-sequence denoising step introduces a more complex objective. Given an input sequence X' and its corrupted version Xc, the goal is to maximize the likelihood of reconstructing the original sequence X' from Xc, which can be represented as:

P(X' | Xc)

The key insight here is that BART's pre-training not only focuses on understanding individual tokens but also captures the dependencies between them in the context of the entire sequence.

Fine-tuning for NLP Tasks

After the pre-training phase, BART can be fine-tuned for specific NLP tasks. Fine-tuning involves adapting the model to a particular task by providing task-specific inputs and training the model to generate appropriate outputs. BART's flexibility shines through in this phase, as it can be tailored to a wide range of NLP applications.

For instance, in text summarization tasks, the input might be a lengthy document, and the output is a concise summary. BART can be fine-tuned to take the document as input and generate a coherent and informative summary. Similarly, in machine translation tasks, the input is a sentence in one language, and the output is the translation in another language. BART can be fine-tuned to produce accurate translations by learning the alignment and context between the source and target languages.

Results and Achievements

The effectiveness of BART is evident in its performance on various NLP benchmarks. The model consistently achieves state-of-the-art results, showcasing its capabilities across different tasks. Let's examine some of the notable achievements:

Text Summarization: BART excels in abstractive summarization tasks, where it generates concise and coherent summaries of long documents. Its ability to capture the essence of the content while maintaining readability sets it apart.

Machine Translation: BART's prowess extends to translation tasks, where it competes with specialized translation models. Its versatility allows it to handle multiple languages and translation directions effectively.

Text Comprehension: In question-answering and text understanding tasks, BART demonstrates its understanding of the context and generates accurate and contextually relevant answers.

These results show just how BART can be used to our advantage while trying to returning

LFPT5: **for few-shot learning**
(https://github.com/qcwthu/Lifelong-Fewshot-Language-Learning)

Lifelong Few-Shot Learning in NLP

Lifelong few-shot learning builds upon the concepts of few-shot and zero-shot learning, extending them into a lifelong learning setting. The goal is to enable models to continually acquire new skills, adapt to emerging tasks, and improve their performance over time.

Few-Shot Learning: In few-shot learning, a model is provided with a small number of examples or a prompt to generalize and perform a task with minimal data. For instance, with a few examples, a few-shot learning model can classify images into multiple categories.

Lifelong Learning: Lifelong learning takes the concept of few-shot learning further by allowing models to accumulate knowledge and adapt to new tasks continually. This is particularly valuable in dynamic environments where new language tasks emerge regularly.

Architecture of LFPT5

The LFPT5 framework builds on the robust T5 architecture, a text-to-text transformer that has demonstrated exceptional performance in various NLP tasks. T5 casts all NLP problems into a unified text-to-text format, where both inputs and outputs are represented as text. LFPT5 leverages this architecture and introduces prompt tuning as a mechanism for incremental task adaptation.

T5 Architecture: T5 comprises a stack of transformer layers, incorporating both encoder and decoder components. The encoder processes input text, while the decoder generates the desired output. This architecture has shown remarkable versatility across NLP tasks.

Prompt Tuning: The innovative aspect of LFPT5 lies in prompt tuning. Prompt tuning involves refining the model's behavior by designing specific prompts. The prompts guide the model's understanding of the task and the expected output. LFPT5 utilizes prompt tuning to enable continual learning of new tasks.

Mathematical Foundations of Prompt Tuning

Prompt tuning in LFPT5 is underpinned by mathematical principles that ensure the model's adaptability to new tasks. Let's explore the mathematical details:

Prompt Representation: A prompt is represented as a text string, denoted as P, that encapsulates the instructions and context for the task. Mathematically, a prompt P can be considered as a sequence of tokens: $P = [p_1, p_2, ..., p_n]$.

Model's Behavior: The model's behavior is controlled by the prompt, which influences both the encoding of the input and the decoding of the output. Given an input text sequence X, the model's behavior can be represented as:

$Y = T5(X, P)$

Here, Y represents the model's output, which is generated based on the input X and the prompt P.

Prompt Tuning Objective: Prompt tuning aims to optimize the prompt such that it improves the model's performance on a specific task. This is achieved by maximizing a task-specific objective function L_task, which measures the model's performance on the task:

max L_task(P)

In practice, this involves iteratively fine-tuning the prompt to maximize the likelihood of generating the desired outputs for the given task.

Continual Adaptation in Lifelong Few-Shot Learning

The key innovation in LFPT5 is its ability to continually adapt to new tasks. This involves a process where the model encounters a new task, receives a few examples or a prompt for guidance, and fine-tunes its prompt to excel in the task. The mathematical foundations of this adaptation can be outlined as follows:

Initial Prompt: When a new task is introduced, LFPT5 starts with an initial prompt P_0, which may be generic or task-agnostic.

Task-Specific Examples: To adapt to the new task, the model is provided with a small number of task-specific examples, denoted as D_task. These examples include input sequences X and their corresponding target sequences Y, where Y represents the desired output.

Prompt Tuning for Task Adaptation: LFPT5 fine-tunes the initial prompt P_0 using the task-specific examples D_task. The objective is to update the prompt to maximize the likelihood of generating the correct outputs Y for the given inputs X:

P_task* = argmax L_task(P_task | D_task)

Continual Learning: Over time, as LFPT5 encounters more tasks, it accumulates a repertoire of task-specific prompts, allowing it to perform well on a wide range of tasks. The model can then adapt to new tasks by selecting or fine-tuning prompts from its repertoire

STS Model: (https://aclanthology.org/2020.clinicalnlp-1.25.pdf)

Semantic Textual Similarity in Clinical NLP

Semantic Textual Similarity (STS) involves quantifying the degree of semantic equivalence or similarity between two texts. In clinical NLP, STS can be particularly useful for our usecase.

Challenges in Clinical STS

Lack of Labelled Data: Annotating clinical data for STS is expensive and time-consuming. Therefore, labelled clinical STS datasets are often limited in size.

Clinical Language Complexity: Clinical texts are highly specialized, often containing medical jargon and complex terminology.

Variability: Clinical texts can exhibit significant variability, making it challenging to capture their nuances accurately.

Learning from Unlabelled Data

To address the limitations of labelled data in clinical STS, the proposed approach focuses on learning from unlabelled data. This method leverages large, unannotated clinical text corpora to enhance the performance of STS models.

Technical Details

The technical details of learning from unlabelled data for clinical STS can be broken down into several components:

Embedding Representation: Clinical text data, including medical records and reports, are transformed into numerical representations using word embeddings or subword embeddings like Word2Vec, FastText, or BERT.

Let's denote the embedding function as E, where E(text) represents the embedding of a clinical text.
STS Model: An STS model is used to compute semantic similarity scores between pairs of clinical texts. The model may be based on Siamese networks, transformers, or other architectures designed for STS tasks.

The STS model can be denoted as S, where S(E(text1), E(text2)) yields the semantic similarity score between two clinical texts, text1 and text2.
Learning from Unlabelled Data:

a. Contrastive Learning: Unlabelled clinical text pairs are used to train the STS model. Contrastive loss functions, such as the triplet margin loss, encourage the model to embed similar clinical texts closer together in the latent space while pushing dissimilar texts apart.

The contrastive loss can be mathematically represented as:

$L_{contrastive} = \sum[S(E(text1), E(text2)) - S(E(text1), E(text3)) + margin]_+$

Here, [x]_+ denotes the positive part of x, and the margin is a hyperparameter that defines the minimum separation required for dissimilar text pairs.

b. Self-Supervised Learning: Self-supervised learning is another technique for learning from unlabelled data. It involves creating pseudo-labels for unlabelled clinical texts by creating artificial training tasks. These tasks may include predicting missing words in a sentence (masked language modeling) or predicting the order of sentences in a document.

The self-supervised loss can be defined as:

$L_{self\_supervised} = \sum \ell(X, Y)$

Where ℓ is the loss function associated with the self-supervised task, X represents the input text, and Y represents the pseudo-label.

Fine-Tuning with Labelled Data: After pretraining the STS model on unlabelled data, it is fine-tuned on the available labelled clinical STS dataset. This fine-tuning process adapts the model to the specific clinical domain and further refines its STS performance.

The fine-tuning loss can be represented as:

$L_{fine\_tuning} = \sum [S(E(text1), E(text2)) - gold\_score]^2$

Where gold_score represents the human-annotated similarity score for the labelled clinical text pair.

Mathematical Foundations

To dive deeper into the mathematical foundations, let's consider the contrastive loss in more detail. Given three clinical texts (text1, text2, text3), the contrastive loss encourages the model to pull similar texts closer and push dissimilar texts apart.

Define the similarity score between text1 and text2 as S(E(text1), E(text2)) and the similarity score between text1 and text3 as S(E(text1), E(text3)).

The loss for a triplet of texts is calculated as the positive part of the difference between the similarity scores for similar and dissimilar text pairs, with a margin:

$L_{contrastive} = \sum [S(E(text1), E(text2)) - S(E(text1), E(text3)) + margin]_+$

The margin is a hyperparameter that determines how much separation is required between similar and dissimilar text pairs. It prevents the model from pushing similar texts too far apart.

Some words that can be further added into our classification table split by topics:

Park Reviews:
Excellent, Beautiful, Scenic, Tranquil, Clean, Crowded, Peaceful, Charming, Impressions, Favorite

Park Activities:
Hiking, Biking, Picnicking, Birdwatching, Camping, Fishing, Boating, Running, Playground, Sports

Nature and Wildlife:
Wildlife, Birds, Trees, Flowers, Insects, Ecosystem, Flora, Fauna, Habitats, Scenery

Events and Festivals:
Concerts, Festivals, Exhibitions, Workshops, Performances, Celebrations, Entertainment, Cultural, Music, Art

Environmental Conservation:
Sustainability, Conservation, Preservation, Biodiversity, Green initiatives, Emission reduction, Habitat protection, Recycling, Climate change, Eco-friendly

Photography:
Photos, Cameras, Shots, Landscapes, Wildlife shots, Photography tips, Camera gear, Filters, Selfies, Sunsets

Park History:
Heritage, Landmarks, Monuments, Historical sites, Centuries-old, Evolution, Historical facts, Past, Stories, Restoration

Park Safety:
Safety tips, Caution, Rules, Emergency, First aid, Awareness, Preparedness, Responsible, Security, Hazards

Family and Recreation:
Family outing, Recreation, Fun, Bonding, Playtime, Quality time, Games, Relaxation, Leisure, Adventure

Local Park News:
Updates, Construction, Improvements, Closures, Announcements, Renovations, Upcoming changes, Maintenance, Renovation, Notices

Trail and Route Recommendations:
Trails, Routes, Favorites, Recommendations, Best paths, Exploring, Must-see spots, Walking routes, Hiking trails, Route tips

Park Access and Amenities:
Accessible, Amenities, Restrooms, Parking, Facilities, Services, Convenience, Accessibility, Amenities list, Amenities map

Weather and Seasonal Changes:
Weather updates, Seasons, Temperature, Rainfall, Climate, Seasonal changes, Weather conditions, Forecast, Seasonal activities, Climate variations

Community Engagement:
Community events, Volunteer, Initiatives, Engagement, Participation, Collaboration, Involvement, Local support, Community projects, Civic engagement

Park Advocacy and Policy:
Advocacy, Policy change, Funding, Preservation, Conservation efforts, Park policy, Advocacy groups, Nonprofits, Environmental policy, Campaigns

Travel and Tourism:
Traveling, Tourism, Destination, Exploring, Adventure, Sightseeing, Tourist attractions, Lodging, Travel tips, Vacation

Dog-Friendly Parks:
Dogs, Pet-friendly, Canine, Pets, Dog owners, Dog parks, Leash rules, Responsible pet ownership, Dog-friendly amenities, Dog walking

Art and Creativity:
Artistic, Creativity, Inspiration, Art in nature, Art installations, Creative expression, Nature-inspired art, Artistic photography, Poetry, Creative projects

Local Food and Dining:
Local cuisine, Food vendors, Dining options, Picnic food, Restaurant, Snacks, Outdoor dining, Local flavors, Food trucks, Culinary experiences

Community Events:
Events, Gatherings, Community celebrations, Social events, Local happenings, Community festivals, Get-togethers, Community meetups, Neighborhood events, Park-sponsored events