Interpretable Text Classification Via Prototype Trajectories (ProtoryNet)

1. Settings

1.1 ProtoryNet Architecture Overview

ProtoryNet is designed for interpretable text classification by leveraging **prototype trajectories**, which capture the temporal patterns of prototypes across sentences in a text sequence. The architecture consists of the following components:

- **Sequence Encoder**: Maps each sentence in a text sequence to a fixed-length embedding using a Transformer-based encoder (e.g., Google Universal Encoder).
- **Prototype Layer**: Contains a set of trainable prototype vectors that represent typical sentences. Each sentence is mapped to its **one** closest prototype (active prototype) based on similarity scores.
- **LSTM Backbone**: Processes the sequence of prototype similarities to capture temporal patterns and produce the final classification.

The model is trained on a labeled dataset $D = \{(X^{(i)}, y^{(i)})\}_{i=1}^{N}$, where $X^{(i)}$ is a sequence of sentences, and $y^{(i)}$ is the corresponding label.

1.2 Sequence Encoder

The sequence encoder r transforms each sentence x_t in a sequence X into a fixed-length embedding e_t:

$$e_t = r(x_t), \quad e_t \in \mathbb{R}^J$$

- **Model Choice**: The encoder is implemented using a state-of-the-art Transformer encoder (e.g., Google Universal Encoder).
- **Design Assumption**: The encoder assumes that the embedding e_t captures the semantic meaning of the sentence, which is then compared to prototypes in the latent space.

1.3 Prototype Layer

The prototype layer contains K trainable prototype vectors $\{p_k\}_{k=1}^{K}$, where each prototype $p_k \in \mathbb{R}^J$. The layer operates as follows:

- **Distance Computation**: The similarity between a sentence embedding e_t and a prototype p_k is computed using a distance metric (e.g., Euclidean

distance):

$$s_{t,k} = \exp\left(-\frac{d(e_t, p_k)}{\psi^2}\right)$$

- **Active Prototype**: Each sentence is mapped to its closest prototype (active prototype) -> **prototypical encoding** to ensure interpretability.
- Use s

1.4 Sparsity Transformation

To enforce interpretability, ProtoryNet uses a **sparsity transformation** to ensure that each sentence is mapped to only one prototype (the most similar one). This is achieved by approximating the **argmax** operation using a **softmax** function with a large temperature parameter γ:

$$\Gamma = [\text{Softmax}(\gamma \cdot \tilde{s}_1), \ldots, \text{Softmax}(\gamma \cdot \tilde{s}_T)]$$

Here, $\tilde{s}_t$ is the similarity vector for the t-th sentence, and γ is a large constant (e.g., $\gamma \geq 10^6$). The sparsity transformation is then applied as:

$$S \approx \Gamma \odot \tilde{S}$$

where $\odot$ is the Hadamard product. This ensures that each sentence is mapped to only one prototype, simplifying the explanation.

1.5 Why Softmax Instead of Argmax?

The **argmax** operation is non-differentiable, which makes it unsuitable for gradient-based optimization in deep learning. To address this, ProtoryNet uses **softmax** with a large temperature parameter γ to approximate the **argmax** operation. This allows the model to:

1. **Maintain Differentiability**: The softmax function is differentiable, enabling gradient-based optimization.
2. **Enforce Sparsity**: By setting γ to a large value, the softmax function approximates a one-hot vector, effectively selecting the most similar prototype.
3. **Improve Interpretability**: Each sentence is mapped to a single prototype, making the model's decision process more transparent.

1.6 LSTM Backbone and Classification

The LSTM backbone processes the sequence of prototype similarities to capture temporal patterns and produce the final classification:

- **LSTM Layer**: The sequence of prototype similarities is fed into an LSTM to capture temporal dynamics.
- **Fully Connected Layer**: The LSTM output is passed through a fully connected layer to produce the final class probabilities.

---

2. Optimizing Objectives

2.1 Learning Objective

The training objectives of ProtoryNet include:

- **Accuracy Loss**: Promotes accurate predictions by minimizing the mean squared error between predicted and true labels:

$$\mathcal{L}_{\text{acc}} = \frac{1}{N} \sum_{i=1}^{N} \|y^{(i)} - \hat{y}^{(i)}\|^2$$

- **Diversity Loss**: Encourages prototypes to be distinct by enforcing a minimum distance δ between prototypes:

$$\mathcal{L}_{\text{div}} = \sigma(\eta(\delta - d_{\min}))$$

  where $d_{\min}$ is the minimum distance between any two prototypes.
- **Prototypicality Loss**: Ensures that each sentence is close to at least one prototype:

$$\mathcal{L}_{\text{proto}} = \frac{1}{M} \sum_{X \in D} \sum_{x_t \in X} \min_k d(r(x_t), p_k)$$

The final loss function combines these terms:

$$\mathcal{L} = \mathcal{L}_{\text{acc}} + \alpha \mathcal{L}_{\text{div}} + \beta \mathcal{L}_{\text{proto}}$$

2.2 Optimization Strategy

- **Prototype Initialization**: Prototypes are initialized using k-medoids clustering on sentence embeddings from the training data.
- **Prototype Projection**: During training, prototypes are periodically projected onto the closest sentence embedding every **10 epochs** to ensre interpretability.
- **Prototype Pruning**: After training, prototypes that are rarely used are pruned to reduce redundancy and improve interpretability.

---

3. Interpretability

3.1 Prototype Trajectories for Interpretability

- **Sentence-Level Prototypes**: Each sentence in a text is mapped to a prototype—a representative sentence from the training data. This breaks down documents into smaller, interpretable units, ideal for long or complex texts.
- **Prototype Trajectories**: The sequence of prototypes forms a trajectory, capturing temporal dynamics (e.g., sentiment shifts). This trajectory is processed by the LSTM, providing a human-understandable representation of the text. Important when we trying to understand in a more fine-grained manner to the prediction of model.

3.2 Explain LSTM

- AdaAX -> explain RNNs
- Idea? Build a deterministic finite automata(DFA) that mimics the transitions and states of a RNN
  - Input symbols -> words in a sentence
  - States -> a group of hidden states(core sets) that share the same transitional symbol to get to the accepting states
- Mapping to prototype trajectories:
  - Input symbols -> associated prototype definition for each sentence in a document

3.3 LSTM Component and Its Role

- **Temporal Pattern Capture**: The LSTM processes the sequence of prototypes, learning how sentiments or topics evolve over time. This is critical for tasks like sentiment analysis, where overall sentiment depends on sentence progression.
- **Interpretable Features**: Unlike traditional LSTMs that use raw text or embeddings, ProtoryNet's LSTM operates on prototype encodings, which are inherently interpretable since each prototype represents a meaningful sentence.
- **DFA for LSTM Explanation**: A Deterministic Finite Automaton (DFA) is used to summarize the LSTM's decision-making. For example, a pattern like "$7 \rightarrow 4$" means the model predicts positive sentiment if the first sentence maps to prototype 7 and the second to prototype 4. This provides a high-level, interpretable summary of the LSTM's behavior.

3.4 Prototype Pruning for Enhanced Interpretability

- **Reducing Prototype Count**: ProtoryNet prunes rarely used prototypes, leaving only the most relevant ones (around 20 in experiments). This

simplifies the model and improves interpretability.

- **Impact on LSTM**: Pruning reduces the complexity of prototype trajectories fed into the LSTM, making the temporal patterns easier to interpret. Fewer prototypes mean the LSTM's behavior is more transparent and focused on key patterns.

---

4. Experiments

4.1 Prediction Accuracy

ProtoryNet achieves competitive performance compared to state-of-the-art models like DistilBERT and outperforms interpretable baselines like ProSeNet. Key findings include:

- **Fine-Tuning vs Non-Fine-Tuning**: Fine-tuning the sentence encoder improves performance but increases computational cost.
- **Prototype Pruning**: Pruning reduces the number of prototypes without compromising accuracy, significantly improving interpretability.

4.2 Prototype Trajectories

ProtoryNet provides fine-grained interpretability by mapping each sentence in a text sequence to a prototype, creating a trajectory of prototypes and sentiments. This allows users to understand the dynamics of sentiment within a text.

---

5. Conclusion

ProtoryNet introduces a novel approach to interpretable text classification using prototype trajectories. The model achieves high accuracy while providing intuitive explanations for its predictions. Key contributions include:

- **Prototype Trajectories**: Captures temporal patterns of prototypes across sentences.
- **Prototype Pruning**: Reduces the number of prototypes without losing accuracy, improving interpretability.
- **Human Evaluation**: Users find ProtoryNet more intuitive and easier to understand compared to other prototype-based methods.

6. Limitations and Future Work

- **Embedding Quality**: Interpretability depends on the quality of sentence embeddings. Future work could use advanced techniques (e.g., LLMs like GPT-4) to improve prototype mappings.

- **Similarity Score Rationalization**: The similarity score between sentences and prototypes is based on embeddings, which may not always be intuitive. Future research could focus on making these scores more interpretable, possibly by incorporating linguistic features or LLM-based explanations.

References

- [1] Ming et al. (2019). ProSeNet: Interpretable and Steerable Sequence Learning via Prototypes.
- [2] Chen et al. (2019). This Looks Like That: Deep Learning for Interpretable Image Recognition.
- [3] Arik and Pfister (2020). ProtoAttend: Attention-Based Prototypical Learning.