Waveform A1: A Transformer for Crypto Sentiment and Market Classification

Introduction

Waveform A1 is a transformer-based model engineered to analyze crypto-related messages with improved performance compared to larger, more expensive state-of-the-art language models.
In this work, we detail our end-to-end pipeline—from consensus-based labeling using multiple large language models (LLMs) to a semi-supervised training strategy that effectively leverages unlabeled data.

In the following sections, we provide a detailed technical description of each component, including the transformer architecture, training methodology and evaluation protocols.

Data Pipeline

We collected unlabeled text from multiple social media sources and labeled them with a consensus-labeling procedure across M large language models (GPT-4, o1, R1 DeepSeek, Claude, Grok, Llama, etc.). Each model independently predicted sentiment and market classification for each sample. Let

$$\ell^{(m)} \in \{\,\text{labels}\} \quad \text{for } m = 1, \ldots, M$$

denote the label predicted by model m. We then formed a majority vote label ℓ_{final} for each sample by selecting the category l such that

$$\sum_{m=1}^{M} \mathbf{1}\left[\ell^{(m)} = \ell\right] > \frac{M}{2}$$

If more than half of the models agreed on a single label, that label became ℓ_{final}.
In the “market classification” task, some disagreements involved closely related categories (e.g., “strong bullish” vs. “bullish”). In those cases, we ran a secondary consensus round, restricting each model to choose between the two remaining options.

If the vote was evenly split, we defaulted to the less extreme label (e.g., bullish rather than strong bullish).
We augmented this dataset by prompting additional language models to synthesize paraphrases or semantically similar messages. Ultimately, we randomly sampled a subset for manual inspection by human annotators. We observed a **62,18%** agreement between the final LLM consensus labels and the human annotations on that subset, this number is far from ideal and suggested a major flaw in our labeling pipeline.

Addressing labeling inconsistencies

Our initial consensus approach involved treating each large language model (LLM) as an equally reliable voter. This proved problematic due to our system introducing **unbalanced voting rounds**. Not all voting rounds included the same subset of LLMs, because of this, sub-performing models like Llama occasionally tipped ties in favor of incorrect labels. For example, if GPT-4 voted "bullish" and Llama voted "neutral," a tie would be resolved by defaulting to "neutral," thereby introducing systematic bias.

To mitigate these inconsistencies, we revisited the consensus system and assigned **weights** to each model in direct proportion to its agreement with human annotations. Formally, if model m accuracy in a controlled sample of labeled data, its vote in any subsequent consensus round was scaled by α_m. Thus, highly accurate models influenced decisions more strongly, while sub-performing models contributed less to the final label.

After introducing weighted voting, we observed a **94.87%** agreement between the resulting LLM consensus labels and human labels on a validation subset. Remaining discrepancies typically arose in messages truncated or referencing external media (images, embedded content), making full interpretive context unavailable to any LLM.

Implementation

Waveform A1 accepts sequences of integer IDs alongside optional attention masks, the model's input is simply a padded tensor $\mathbf{X} \in \mathbb{Z}^{B \times T}$ where B is the batch size and T is the (padded) sequence length. Each integer in X corresponds to a distinct lexical or symbolic unit within a vocabulary, which might be built manually or derived from a separate preprocessing pipeline. An optional attention mask $\mathbf{M} \in \{0,1\}^{B \times T}$ identifies valid (non-padded) positions. Inside the model, these integer tokens are mapped to continuous embeddings via:

$$\mathbf{E} = \mathrm{Embedding}(\mathbf{X})$$

The embedding layer maps each token to a d_{model} -dimensional vector, and a positional encoding is added to distinguish token positions within the sequence. A series of transformer encoder layers then refines these representations.

Within each encoder layer, the input undergoes multi-head self-attention. Specifically, the hidden state $\mathbf{X}$ is projected into query $(\mathbf{Q})$, key $(\mathbf{K})$, and value $(\mathbf{V})$ matrices of shape (B, T, d_{model}), where $(\mathbf{B})$ is the batch size and $(\mathbf{T})$ is the sequence length. These projections are split across $\mathrm{num\_heads}$ heads, each of dimension

$$d_{\text{head}} = \frac{d_{\text{model}}}{\mathrm{num\_heads}}.$$

The attention scores are computed as

$$\text{scores} = \frac{\mathbf{Q}\,\mathbf{K}^{\top}}{\sqrt{d_{\text{head}}}}\#\#$$

followed by a softmax over the last dimension. The resulting weights multiply $\mathbf{V}$ to produce a context vector that is concatenated across all heads and passed through a linear projection. A residual connection and layer normalization ensure stable training.

Next, a position wise feed-forward network (two linear layers separated by a ReLU) transforms each position embedding. This feed-forward step is also wrapped with residual connections and layer normalization. After passing through multiple identical encoder layers, the sequence dimension is averaged globally, yielding a single vector representation per message.

Classification occurs via two separate heads, the first predicts sentiment categories while the second predicts market labels. Each head consists of a linear down-projection to $\frac{d_{\text{model}}}{2}$ a nonlinearity, and a final linear layer to produce logits for each classification task.
We tested the model's performance both without using the optional attention mask and without employing a tokenizer.

On smaller datasets, these implementations yielded inconclusive results. However, with a larger pool of training data and a more comprehensive ground truth, it is reasonable to believe that discarding the tokenizer could lead to a faster and cheaper inference pipeline that wouldn't sacrifice on performance. For all reported benchmarks and performance metrics herein, we use DeBERTaV3 small together with an attention mask generated by its tokenizer.

Training Pipeline

We construct a training set of $\left|\mathcal{D}_{\text{train}}\right|$ crypto-related messages, each labeled with sentiment and market categories. These labels are integers drawn from discrete sets $\{0, \ldots, K_s - 1\}$ and $\{0, \ldots, K_m - 1\}$ for the sentiment and market tasks, respectively (with K_s and K_m the number of classes in each classification problem). A DataLoader partitions $\mathcal{D}_{\text{train}}$ into minibatches of size b. Within each training epoch $e = 1, \ldots, E$, we iterate over all minibatches:

- Load a batch of input IDs $\mathbf{X} \in \mathbb{Z}^{b \times T}$ and an attention mask $\mathbf{M} \in \{0, 1\}^{b \times T}$
- Forward the batch through the model. Let $\hat{\mathbf{y}},\ \hat{\mathbf{z}} = \text{model}(\mathbf{X}, \mathbf{M})$ where $\hat{\mathbf{y}} \in \mathbb{R}^{b \times K_s}$ are the sentiment logits and $\hat{\mathbf{z}} \in \mathbb{R}^{b \times K_m}$ the market logits.
- Let $\mathbf{y} \in \{0, \ldots, K_s - 1\}^b$ and $\mathbf{z} \in \{0, \ldots, K_m - 1\}^b$ be the true sentiment and market labels for the batch. We compute two cross-entropy losses:

$$L_{\text{mkt}} = \frac{1}{b} \sum_{i=1}^{b} \Big[-\log\big(\sigma(\hat{\mathbf{z}}_i)\big)_{\mathbf{z}_i} \Big] \quad L_{\text{sent}} = \frac{1}{b} \sum_{i=1}^{b} \Big[-\log\big(\sigma(\hat{\mathbf{y}}_i)\big)_{\mathbf{y}_i} \Big]$$

  where $\sigma(\cdot)$ denotes the softmax function, and $(\cdot)_j$ accesses the logit for class j.
- The total loss is

$$L_{\text{total}} = L_{\text{sent}} + L_{\text{mkt}}$$

- We perform backpropagation with respect to L_{total} and update the model parameters via AdamW, constrained by gradient clipping if necessary: $\|\nabla_\theta L_{\text{total}}\| \leq \tau$ for some clip threshold τ.

Training Evaluation

To evaluate model performance, we use a validation DataLoader partition $\mathcal{D}_{\text{val}}$. For each batch in the validation set, we:

1. Compute $\hat{\mathbf{y}}, \hat{\mathbf{z}} = \text{model}(\mathbf{X}, \mathbf{M})$
2. Compute the same cross-entropy losses as in training, summing them across the validation set to produce an average validation loss.
3. Determine predicted classes $\hat{y}_i$ and $\hat{z}_i$ via:

$$\hat{y}_i = \arg\max_k \hat{\mathbf{y}}_{i,k}, \quad \hat{z}_i = \arg\max_k \hat{\mathbf{z}}_{i,k}.$$

4. Compare $\hat{y}_i$ and $\hat{z}_i$ against true labels y_i and z_i to calculate metrics such as accuracy for both sentiment and market classification.
5. If the mean validation loss drops below the previous best value, we update our checkpointed model.

Semi-Supervised Learning

We studied the possibility of extending the model with a semi-supervised learning component. After training a model on our labeled dataset, we use it as a teacher model to generate pseudo-labels on an unlabeled corpus.

The approach begins with a labeled dataset $\mathcal{D}_{\text{labeled}} = \{(x_i, y_i)\}$, where x_i is a textual sample and y_i is its sentiment and market classification.
Waveform A1 f_θ is first trained in a fully supervised manner using $\mathcal{D}_{\text{labeled}}$. This generates our initial parameter configuration $\theta^{(0)}$ that generalizes reasonably well but may be limited by the size of the labeled set.
We also have a larger pool of unlabeled examples that didn't pass the consensus phase or were excluded from it, $\mathcal{D}_{\text{unlabeled}} = \{u_j\}$.
To leverage this data, the model $f_{\theta^{(0)}}$ generates pseudo-labels for each unlabeled example u_j.
Specifically, the model computes two sets of logits:

$$\hat{\mathbf{y}}_j, \quad \hat{\mathbf{z}}_j \;=\; f_{\theta^{(0)}}\big(u_j\big)$$

where $\hat{\mathbf{y}}_j \in \mathbb{R}^{K_s}$ are the sentiment logits and $\hat{\mathbf{z}}_j \in \mathbb{R}^{K_m}$ the market logits, for K_s and K_m classes respectively. Softmax then transforms these logits into probability distributions:

$$p_m(\hat{z}_{j,k}) = \frac{\exp(\hat{z}_{j,k})}{\sum_{k'=1}^{K_m} \exp(\hat{z}_{j,k'})} \, p_s(\hat{y}_{j,k}) = \frac{\exp(\hat{y}_{j,k})}{\sum_{k'=1}^{K_s} \exp(\hat{y}_{j,k'})}$$

A pseudo-label is only assigned if both predicted probabilities exceed a confidence threshold $\tau \in (0, 1)$, that is:

$$\max_k p_s(\hat{y}_{j,k}) \text{ and } \max_k p_m(\hat{z}_{j,k})$$

If these conditions hold, the corresponding sentiment and market class indices become the pseudo-labels. Formally, let

$$\hat{y}_j = \arg\max_k \hat{\mathbf{y}}_{j,k} \text{ and } \hat{z}_j = \arg\max_k \hat{\mathbf{z}}_{j,k}.$$

The model discards all unlabeled examples that do not meet the confidence requirement. The remaining pairs $\left(u_j, (\hat{y}_j, \hat{z}_j)\right)$ form a pseudo-labeled dataset $\mathcal{D}_{\text{pseudo}}$.
The labeled and pseudo-labeled data are then concatenated into a single training set:

$$\mathcal{D}_{\text{combined}} = \mathcal{D}_{\text{labeled}} \cup \mathcal{D}_{\text{pseudo}}$$

With the new dataset $\mathcal{D}_{\text{combined}}$, we start a new training session that incorporates information from unlabeled samples that are confidently categorized.
In practice, we again perform minibatch stochastic gradient descent on the cross-entropy losses for both sentiment and market classification. The result is a final model $f_{\theta^{(\text{final})}}$ that generalizes better than one trained solely on labeled data.

Results

We evaluated Waveform A1 both in terms of classification accuracy and computational efficiency. While previous labeling using larger SOTA LLMs (e.g., GPT-4, Grok, Claude, Llama) produced high-quality annotations, the average cost was approximately $0.002 per message. In addition, these systems often introduced throughput limits and resource constraints (e.g., GPUs, spending-based constraints, hard platform limits, etc).

In contrast, Waveform A1 delivers comparable performance at a significantly lower cost. Running inference on an ARM-based CPU with 8vCPUs and 4 GB of RAM (with no GPU needed), each classification task (sentiment and market) costs around $0.00002665. This reduction in cost—on the order of tens of thousands of messages for each dollar—enables large-scale real-time deployments on a volatile market that is associated with Crypto Currencies. Consequently, Waveform A1 addresses scalability and budget constraints far more efficiently than standard LLM-based pipelines.

Comparison with Baseline Models

The table below summarizes our experimental results on sentiment and classification tasks across multiple models, including Waveform A1 in both its fully supervised and semi-supervised configurations. Classification Accuracy refers to the market label task, while Sentiment Accuracy pertains to the positive/neutral/negative sentiment categorization. All percentages represent the accuracy achieved on a held-out validation set of 60,000 text samples.

Model	Classification Accuracy	Sentiment Accuracy
o1-mini-2024-09-12	90.95%	95.84%
llama-3.3-70b	58.89%	84.53%
claude-3.5-sonnet	88.51%	84.64%
grok	81.07%	85.78%
Waveform A1 (**Initial dataset + Supervised**)	64.32%	70.74%
Waveform A1 (**Initial dataset + Semi Supervised**)	65.26%	71.92%
Waveform A1 (**Revised dataset + Supervised**)	76.11%	81.92%
Waveform A1 (**Revised dataset + Semi Supervised**)	79.30%	81.03%

Although models like “o1-mini” demonstrate high accuracy on this particular dataset, they typically incur higher inference costs and resource demands. In contrast, Waveform A1 trades off performance for a drastically lower deployment cost (see Cost Analysis below) while still attaining competitive classification and sentiment accuracy.

Robustness Against Adversarial Attacks

Large language models (LLMs) are known to exhibit weakness when subjected to adversarial prompts that override system instructions or otherwise manipulate model output.

Even sophisticated architectures and carefully designed system prompts have shown susceptibility to such “prompt-injection/jailbreak” attacks, wherein a user-crafted query can cause the LLM to produce compromised responses. While techniques like strategic system prompts and fine tuning may help, they add operational overhead and often inflate inference costs—especially at scale, while not guaranteeing a solution to the underlying issue.

In contrast, A1 depends on patterns embedded in the training corpus, such as the frequency with which certain keywords co-occur in bullish, bearish, or neutral contexts. An attacker would need to identify these underlying signals—often by analyzing large numbers of model inputs and outputs—and craft inputs that systematically trigger incorrect classifications. In practice, this requires building or collecting an adversarial dataset that maps specific token sequences to desired (mis)classifications.

This reliance (or different weakness) on task- and domain-specific triggers is qualitatively different from the general “jailbreak” attacks seen with standard LLMs.

While adversarial robustness can never be fully guaranteed, having a specialized model reduces the types of user-driven prompt-based vulnerabilities observed in real-world scenarios such as social media platforms, forums, etc.

Improving A1 results

By incorporating the hidden-state representations of small language models, we can feed each input sequence into a LLM to extract its final hidden-layer states..

We then feed those hidden states into our Waveform A1 transformer in place of the tokenized inputs. This hybrid approach allows us to preserve A1's architecture while benefiting from a pretrained LLM. We ran this test using DeBERTaV3-small as base LLM.

Model	Classification Accuracy	Sentiment Accuracy
o1-mini-2024-09-12	90.95%	95.84%
llama-3.3-70b	58.89%	84.53%
claude-3.5-sonnet	88.51%	84.64%
grok	81.07%	85.78%
Waveform A1 (**Revised dataset + Supervised**)	76.11%	81.92%
Waveform A1 (**Revised dataset + Semi Supervised**)	79.30%	81.03%
Waveform A1 + DeBERTaV3 Small (**Supervised**)	89.18%	92.30%

Waveform A1 combined with DeBERTaV3 small attains a near-LLM-SOTA level of accuracy while preserving a relatively light deployment footprint. This synergy illustrates the potential for smaller, specialized transformers to achieve state-of-the-art results when enhanced by compact pretrained models.

Overall, Waveform A1 has demonstrated its potential as an efficient, cost-effective alternative for sentiment and market classification tasks in the crypto domain. The implementation of a semi-supervised training pipeline achieves near-SOTA accuracy while maintaining a modest computational footprint.

However, this is still an early iteration of the concept, and there remains significant room for improvement in both accuracy and resource usage. In future versions, we plan on investigating more domain-specific tokenization strategies at scale, further refining pseudo-labeling techniques for more robust class separation, and conducting comprehensive evaluations of real-time throughput under production conditions.

We invite other researchers to collaborate on these next steps. We are open to sharing our labeled dataset, training source code, and related artifacts under appropriate research agreements.