Modelex: A Platform For Integrating Blockchain With Large-Language Models for On-Chain Sentiment

V0.12 DRAFT

Jamiel Sheikh
jamiel@scifn.com
https://linkedin.com/in/jamiel
https://modelex.ai

Modelex is a suite of tools that integrate the emerging world of large-language models with blockchain in powerful and useful ways that drive immediate value.

Modelex is an ecosystem of services consisting of:

(a) an LLM-based oracle with an initial set of parameterized prompts geared towards sentiment with terse responses, useful for trading and prediction markets, referred to as ***oracle*** or *oracles*

(b) tokenized metering of LLM usage ex-blockchain which we refer to as ***leasing***

(c) catalog of nodes that wrap LLM services, offering distributed and decentralized inference

(d) the **AI token** used as the staking and payment token for oracle and node services

~~(e) an AMM to allow swaps between AI and stablecoin pairs, providing robust liquidity for consumers of services to acquire tokens~~,

(f) both a centralized and a decentralized **prompt library** allowing parameterized prompts to be registered, indexed, referenced and verified, referred to as the *prompt registry* and

(g) centralized repository of canonical units of compiled datasets useful for fine-tuning LLMs hosted by a node, for example, if the IMF wished to compile 10,000 research papers into a single tradable block of data, referred to as **data blocks**, a term unaffiliated with blockchain blocks.

The focus of the current version of this paper is the LLM-based oracle and AI token.

LLM-Based Oracle

At the most fundamental level, Modelex allows the response of a prompt to be made available to a smart contract, a service paid for by AI tokens. Because responses can be text that is quite lengthy, prompt templates that force the LLM to narrow its response to a few words, a single numerical score or grade can be possible. In this narrow case, use cases like sentiment scores, credit ratings or intent may be possible. The type of questions an LLM may answer and a smart contract may find useful include:

1. Given the news of the past 90 days for <country>, what is consumer confidence in that <country>?
2. Given the statements made by the Federal reserve, what is the probability of a rate hike?
3. Given the headlines of a stock <symbol>, is sentiment positive or negative?
4. Given the press releases by <company>, does the CEO intend to resign?

It should be noted that LLMs are lexical in nature and not quantitative. They cannot compute probabilities but can evaluate the mood of text and evaluate context. As LLMs continue to become more and more sophisticated, it will become harder to differentiate between the lexical analysis an LLM can perform versus a proprietary quantitative model.

Example: Sentiment Oracle Service

A large number of financial applications rely on sentiment. For example, premiums or spreads on insurance (such as credit default swaps) adjust as the risk of the underlying deteriorates. Often, a measure of risk includes market sentiment towards an entity -- sentiment that is often driven by news, filings and other textual sources of information and could be early indicators of increase of default risk. As sentiments shift, markets shift, but this currently is not the case in decentralized finance ("DeFi") where sentiment is not used in pricing and there is a reliance entirely on market dynamics to arrive at prices. Traders and investors may use sentiment to make decisions but the protocols and algorithms themselves, like a lending protocol's pricing formula, do not adjust if sentiment changes.

While large-language models are probabilistic in nature and have the tendency to hallucinate, they also perform exceptionally well with sentiment analysis. There is active research in using sentiment scores generated by large-language models in trading equities[1] and the results are promising.

For DeFi applications, retrieval of sentiment scores is impossible to obtain except via an oracle and most sentiment engines are proprietary technologies. Currently, oracles that provide sentiment scores do not exist. Open-source large-language models like Llama2, Mistral and others change this allowing a whole host of text-based analysis to come on chain. Modelex AI is a sentiment oracle that leverages open-source large-language models to analyze large amounts of text and generate actionable sentiment scores that can be injected into smart contracts.

[1] https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4412788

For example, in the below prompt, we request ChatGPT 4 evaluate only a news headline[2] and ask it to return a terse binary response[3]:

> **[Prompt]** In between triple backticks below, a headline is provided. In regards to GM, which the article is about, state 1 if the news is optimistic for GM, -1 if the news is pessimistic for GM. Provide only 1 or -1 in your response and nothing else.
>
> Headline: ```*GM to resume sales of Chevy Blazer EV after software issues prompted a stop sale late last year*```
>
> **[Response]** 1

In the above prompt and response, a real-world news headline is supplied to ChatGPT 4.0. The prompt requested a single measurement of the sentiment of the headline as it pertained to General Motors. The prompt can be parameterized and stored as a hash on-chain, building an on-chain catalog of parameterized prompts (we'll soon show how that would be useful). The response of 1 would be injected into a calling smart contract for use in pricing, premium or other settings.

An identical response is received from Llama2 and Mistral, respectively, using Ollama[4]:

Llama2

>>> In between triple backticks below, a headline is provided. In regards to GM, which the article is about, state 1 if
... the news is optimistic for GM, -1 if the news is pessimistic for GM. Provide only 1 or -1 in your response and nothi
... ng else.
GM's Stock Price Soars After Earnings Report
```
1
```

Mistral:

>>> In between triple backticks below, a headline is provided. In regards to GM, which the article is about, state 1 if
... the news is optimistic for GM, -1 if the news is pessimistic for GM. Provide only 1 or -1 in your response and nothi
... ng else. Headline: ```GM to resume sales of Chevy Blazer EV after software issues prompted a stop sale late last
yea
... r```
Based on the provided headline, I would assess the news as optimistic for General Motors (1). The article indicates that GM is resuming sales of its Chevy Blazer EV model following software-related issues that caused a stop sale late last year. This news suggests that the problem has been addressed and resolved, which is generally viewed positively by investors and consumers alike. It's important to note that this assessment is based solely on the information provided in the headline and additional context may be needed for a more comprehensive analysis of
the situation.

---

[2] https://finance.yahoo.com/news/gm-resume-sales-chevy-blazer-181838654.html
[3] https://chat.openai.com/share/4e68a7b7-162e-486e-b32f-f7457e726289
[4] https://ollama.com

The parameterized version of the prompt could look like so:

> **[Prompt]** In between triple backticks below, a headline is provided. In regards to ***<subject>***, which the article is about, state 1 if the news is optimistic for ***<subject>***, -1 if the news is pessimistic for ***<subject>***. Provide only 1 or -1 in your response and nothing else.
>
> **[Headline]** ```***<headline>***```

An event from a blockchain can request a parameterized prompt with parameters accompanied, via a prompt id hash and a set of key value pairs. Alternatively, a hash of the parameters and a URL to the location of the parameters can also be supplied. An oracle that listens for and traps the event would then invoke one or more large-language models and return the response vector or aggregate (i.e. average) the responses and return a single evaluation to the initiating smart contract.

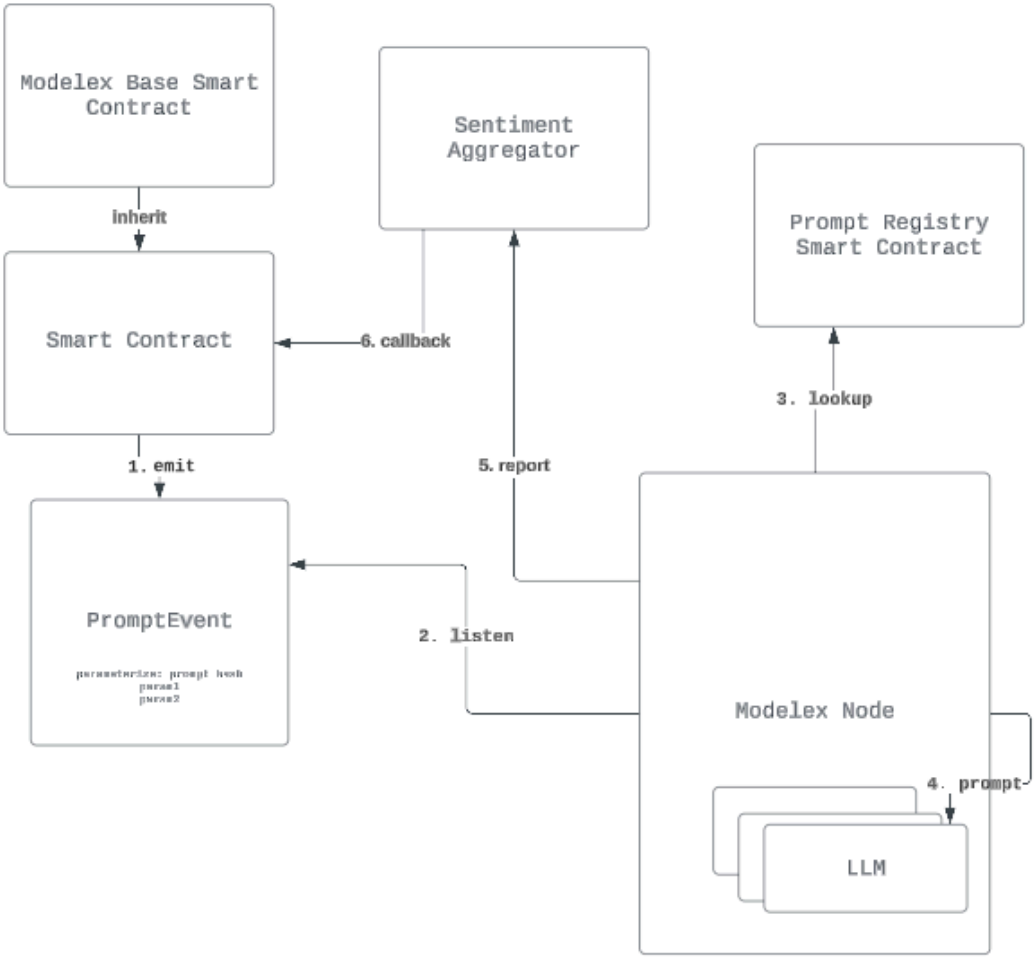
In a prediction market, the results of events may be useful and soliciting humans to provide responses may be less reliable than an LLM (especially if it employs retrieval augmented generation or RAG). For example, in determining if the Yankees won the 2023 World Series, the following parameterized prompt[5] could be used.

> **[Prompt]** In the baseball event known as the World Series, did the Yankees win in 2022? Reply 1 if yes, and -1 otherwise. Do not provide any other response.
>
> **[Response]** -1
>
> **[Prompt]** In the baseball event known as the ***<event>***, did the ***<player>*** ***<win | lose>*** in ***<date>***? Reply 1 if yes, and -1 otherwise. Do not provide any other response.

If the parameters are lengthy, they would be stored in an external repository and a hash would be the parameters emitted by the event. The oracle node would then retrieve the parameters, match hashes and respond accordingly. An example use case is the prompt requiring 50 headlines instead of one.

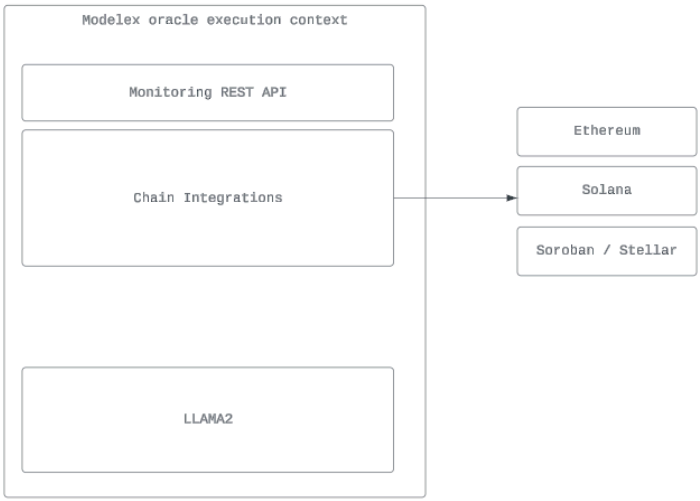
Investors, funds, and traders have been using sentiment analysis tools for decades. In the world of decentralized finance, sentiment analysis can be used for automated trading strategies, prediction markets, dynamic fee and premium adjustments, DAO governance and more.

[5] https://chat.openai.com/share/e35ea13e-1045-43b4-acc4-121bfd4382e4


Modelex Base Smart Contract
inherit
Smart Contract
1. emit
PromptEvent
parameters: prompt hash
param1
param2
Sentiment Aggregator
6. callback
5. report
Prompt Registry Smart Contract
3. lookup
2. listen
Modelex Node
4. prompt
LLM

Node architecture

A Modelex node is written in Rust and runs an LLM within the same execution context, albeit on different threads. Included in the same process is a simple REST engine for queries. An elliptic curve (ECDSA) keypair is generated when a node is first deployed and all LLM responses are digitally signed. Initially, LLAMA2 will be used with more performant models substituted when available. A GPU is required to run the node, however a slower pure CPU mode is allowed.



EVM Integration

A number of smart contracts will play a role in allowing Modelex nodes to interact on-chain and off-chain. This includes a registry smart contract that allows nodes or oracles to register their existence.

Stellar / Soroban Integration

Solana Integration

Large-language Model Selection

A large number of open-source large-language models exist beyond the proprietary ones like GPT3, GPT4 and Gemini. Although OpenAI maintains a current lead in advanced large-language models, the latest and greatest is not necessary for sentiment analysis and in some cases could even be counterproductive due to the amount of computation power required to run them. In addition, there is

emerging evidence to suggest that open-source large-language models can outperform[6] closed-source models, at least on certain tasks. Models like FinLlama[7] that are fine-tuned on top of Llama can be expected to perform better than GPT4 on sentiment analysis within the context of finance.

Hugging Face maintains a leaderboard[8] of such models and their number available continues to grow with each model becoming more and more sophisticated and complex, based on the number of tokens the models are trained on and the parameters they contain. A Modelex node is agnostic to the large-language model it employs but will come packaged with the option of running LLAMA2, FinLLM, FinGPT, FinBERT or Mistral. LLMs will be swappable and a node will eventually be able to run more than one LLM at a time.

While LLMs typically require GPUs for fast processing, nodes will be able to run purely on a CPU, albeit dramatically slower.

Score Aggregation

In order to maintain a trustless environment and reduce the probability of hallucinations outweighing a response, a number of nodes need to be prompted and scores returned can be aggregated for further weighting. For example, for the GM sentiment, if five nodes return 1 and one node returns -1, what score should the calling smart contract receive? At the most fundamental level, the scores can be averaged, or weighted by how credible a node's rating[9] is.

Centralized Services And Token Buy Backs

While Modelex is inherently decentralized, there arises revenue-generating opportunities to provide centralized SaaS options, specifically for enterprise customers. The centralized services utilize decentralized Modelex services but make it convenient for enterprise customers to leverage those services. For example, while the $AI token is required to inference a model, the token can be held in custody by Modelex on behalf of an enterprise, providing them a more familiar web2 experience when buying tokens.

As the centralized services revenues grow, so too do the $AI tokens held in custody for the enterprise, placing additional demand on the $AI token and reducing token availability in the broader market. For the SaaS services, a fee can be charged and a portion of the fee can be used to purchase tokens to be held in reserve or potentially be burned.

Token Demand

Assuming $AI is traded at $1 and the below market rate for GPT4 inference, as of Q1 2024, as a benchmark, a 20%/80% split between prompt and completion language tokens and a ratio of .7 words

[6] https://arxiv.org/pdf/2311.16989
[7] https://arxiv.org/abs/2403.12285
[8] https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard
[9] We have not covered node ratings yet

per token, 128k context window a total supply of 1 billion tokens imply 6.4 trillion words can be exchanged, without any turnover, if all tokens were used once. 6.4 trillion words is about 14 billion pages or about 458 million instances of IBM's 2022 10-K filing. If a large cap 10-K averages 14K characters per filing, as a reference point, implies about ~2.2 million inferences can occur before token supply is exhausted.

In other words, an $AI market capitalization of $1 billion is sufficient to fuel the inference of 200 10-Ks approximately 2.2 million times. Given that the universe of text content used for sentiment or intent, including filings, news and more across a large number of asset classes, suggest that eventually the demand for the $AI token may exceed available supply.

Given the bespoke nature of the LLMs services wrapped by Modelex, i.e. financial LLMs, the price per language token is expected to be higher than what OpenAI charges for its GPT4, placing further strain on token supply.

GPT4 Pricing				
	Per 1K		**Per 1M**	
Context Length	*Prompt Tokens*	*Completion Tokens*	*Prompt Tokens*	*Completion Tokens*
128k	$0.01	$0.03	$10.00	$30.00
8k	$0.03	$0.06	$30.00	$60.00
32k	$0.06	$0.12	$60.00	$120.00

Table 1: GPT4 Pricing from OpenAI as of Q1 2024

$AI Price	$1
$AI Supply	1000000000
Market Cap	$1,000,000,000
Prompt/Completion Split	0.2
Language Tokens	9,259,259.26
Words to Token	0.7
Words	6,481,481.48
Per 1m	6,481,481,481,481.48
IBM 2022 10-K Word Count	14,151
Number of 10-Ks	200.00
200 10-Ks	2830200
Number of 200 10-Ks Processed	2290114.296

Table 2: What-if Scenario of Token Exhaustion

Token Split

In order to account for reduced token availability, which may price out certain Modelex nodes, the ability to split and double the token supply and balances will be available in the smart contract. To perform a split, a 2-step process is required. The first is the pausing of all transactions for a certain number of blocks or time, then a snapshot is taken of balances and all balances are doubled with the token supply also doubled. The smart contract is then unpaused. This may leave a minor and immaterial attack vector open for token transactions that are sitting in the mempool and are flushed out after the split, but a deeper study in this area will be required.

Ensuring Honesty

A fundamental problem exists with large-language models and generative AI in general: how can an AI model prove that it is in fact the source of the output it creates?

In the case of a Modelex oracle, how can it be proven that a node is running a correct version of the node software and the large-language model(s) it claims to host? The use of watermarks[10], where text is injected in the response that proves where it comes, is being considered by the industry[11] but poses a number of technical challenges and problems.

Hashes of the open-source code base are futile because they do not ensure it is generated on the code that is executing. A number of ways to alleviate, but not fully solve this problem include: a) Enterprise host their own oracles for their own permissioned blockchain applications, b) the oracle runs in a trusted execution environment, and/or c) auditors certify nodes, albeit for a single moment in time

Rust & Python Impedance

A large majority of current artificial intelligence models, including large-language models, are generated by toolkits that rely on Python. Toolkits like PyTorch and TensorFlow are now the mainstay of every artificial intelligence project, and each toolkit produces models in file formats that may not be standardized or readable by other toolkits or languages.

Modelex nodes will be written in Rust and will deploy and inference models built in Python. This impedance may pose issues in the future and workarounds currently exist in the market. Currently, a Rust-based Modelex node binds to a Python subprocess and we have found a solution to have Rust interoperate with a Python process hosting an LLM.

Leasing

[10] https://arxiv.org/abs/2301.10226
[11] https://www.lawfaremedia.org/article/digital-watermarks-are-not-ready-for-large-language-models

Currently, there is no standard way to charge for the usage of a large-language model, especially if it is open source. OpenAI charges for GPT4 API access via the number of tokens (fractions of words) sent and returned from the API and is able to do so because the GPT4 model is closed source or available for use via Azure only.

For other large-language models, infrastructure does not exist to deploy and meter their usage. Hugging Face houses models and provides the ability to deploy on cloud infrastructure but is typically used by highly technical persons.

A foundational (base) large-language fine-tuned on domain specific data could be leased out for use to other enterprises. Modelex is an end-to-end solution that allows an LLM to be stored, deployed and then leased, allowing creators of LLMs and canonical sets of data (see Data Library) to monetize.

Decentralized Model Hub

Currently, Hugging Face is the global leader in housing AI models, it is the GitHub for AI models, raising over $400M in their latest Series C. Modelex proposes an alternative method of storing AI models in a decentralized manner, with the catalog stored in smart contracts and deployment and execution of the AI models done by nodes that are willing to offer up the CPU cycles in return for AXUs.

Prompt Library

Understanding how to prompt and query generative AI models is a mix of art and science and knowledge of how to build them effectively means the efficiency and efficacy of use of these models can be maximized. Currently, most commercial large-language models allow prompts and responses to be shared via a URL but there is no repository for these prompts. Enterprises that want to maintain internal knowledge and share prompts have few to no options. Modelex provides the ability to prompt one or more LLMs and store, share, annotate prompt / response for teams, divisions and others.

In addition, a decentralized storage of parameterized prompts on, for example, IPFS, means the prompts can be stored outside of centralized systems, even if the prompts are encrypted by the generating enterprise.

When a smart contract requests services from a Modelex oracle, it can reference a parameterized prompt via a hash and a set of parameters. The oracle then looks up the prompt and injects in parameters.

Data Blocks

Currently, a large number of canonical data sets exist for internet data. For example, the C4[12] (colossal, cleaned, common crawl) ~400GB dataset containing internet text data that can be used atomically to train a large-language model.

[12] https://commoncrawl.org

While a number of such datasets exist for public data, other public and private datasets are not available in single unit canonical format ready to be used to train foundational LLMs.

With data blocks, an enterprise can compile a large number of datasets into a single unit and make it available for lease for other enterprises to train their LLms.

The AI Utility Token

The AI token will be an ERC-20 Ethereum token with ERC-677 `transferAndCall()` functionality, allowing the invocation of Modelex oracle to be accompanied with a payment in AI tokens for invocation of Modelex services.

Note: This allocation table is subject to change and is currently being optimized.

Symbol	AI			
Max Supply	1,000,000,000			
	Allocation	**Amount**	**Price**	**Value**
Team				
Accredited Investors & Institutions	25%	250,000,000	0.20	$50,000,000
Founders, Team & Advisors	15%	150,000,000	3.00	$450,000,000
Network Users	60%	600,000,000	0.05	$30,000,000
	100%	1,000,000,000		