Introduction to Blockchain Technology: Meme Economy Blockchain

Rahul Agrawal, Akshay Katyal, Mehmed Mustafa
Anant Sujatanagarjuna, Steffen Tunkel, Chris Warin

July 25, 2020

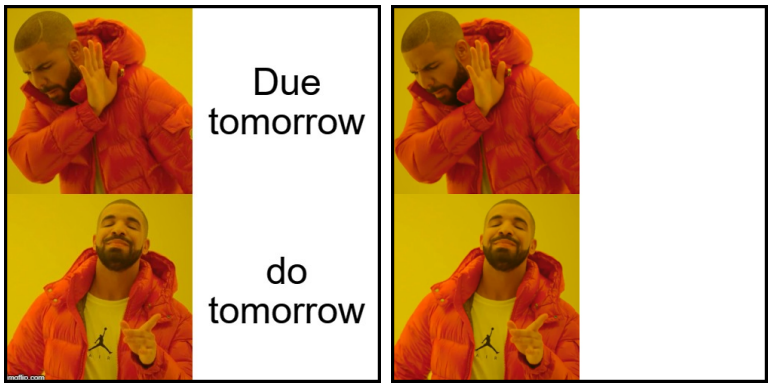
Abstract

The meme culture is spread widely over online communities. This project aims to create a novel decentralized platform for the sharing of memes. The *Meme Economy Blockchain* is build up like a virtual marketplace, based on the principles of market economy. Through this users are rewarded for creating new and innovative memes and also for their intuition about the potential popularity of memes. The current state of the project contains all core functionalities, which are required for the application.

Contents

1 Introduction

A meme is *"an amusing or interesting item (such as a captioned picture or video) or genre of items that is spread widely online especially through social media"* according to Merriam-Webster dictionary [**?**]. An example for such an item is shown in Figure **??** a. This particular joke about procrastination is combined with the pictures of the musician Drake to highlight the message. This pictures, as shown in Figure **??** b, are a genre of memes. These are templates which are used over and over again to create new memes. In this work, templates for memes are named 'meme format', while 'meme' refers to the captioned pictures as in Figure **??** a.



(a) Meme (b) Meme Format

Figure 1: 'Drake Hotline Bling' as an example for a meme and meme format [**?**].

Memes are not only spread over classic social media platforms as Facebook or Twitter, but also on more dedicated platforms as 9GAG or Reddit. One of Reddit's sub-forums is called MemeEconomy [**?**]. It is a virtual market place for memes, which is implemented with bots on the Reddit server. It served as inspiration for us to create a more free, decentralized version of a economy system for memes, which can therefore be run by the community itself. The application, which is implemented on a blockchain technology, can be thought of as a virtual stock market for memes. That means, users are able to buy and sell shares of meme formats and speculate on their value. The value is defined by the estimated lifetime of the meme format and its circulation. This is based on the concept that the shareholders of a format get a reward every time it is used for a meme. The basic principles of financial markets are also applied in the approach. Users are motivated to buy a share when its value is low and has a higher potential to increase, and vice versa, to sell when its value is high and expected to go down in the future. Moreover, users have the chance to invest in particular memes with their upvotes. The revenue a user can gain with an upvote is proportional to the amount of following upvotes by other users. Therefore, a good intuition and early support of a meme is rewarded. The credits used in the economy system can be gained through smart investments, the creation of new and potentially successful content or the support of the blockchain with the mining of new blocks. The credits do not have any money value. Therefore, they can be interpreted as the user's score. The motivation of the project is to offer users a fresh and interesting platform for meme sharing by enhancing the concept of the MemeEconomy

subreddit. Additionally, the platform encourages the creation of new memes and even new meme genres. By recognizing memes as a form of art this factor gets even more important. Connected to this it is also possible to track down the origin of popular meme formats over the blockchain, which means the credit of the work goes where it belongs.

This report aims to give a rather technical introduction to the approach. Section **??** presents the different technologies used for the implementation. Afterwards, Section **??** gives an introduction to the MemeEconomy from Reddit as a related work. The main part is the technical description of our approach, outlining the different functional parts of the blockchain, in Section **??**. Finally, Section **??** gives an outlook on further possibilities with the approach before Section **??** concludes the report.

2 Foundations

This section gives information regarding the software and interfaces used during the development of our Blockchain application.

2.1 RESTful Applications

A RESTful application is an **API!** (**API!**) that uses HTTP requests to access the resources. An API for an application is code that allows two software programs to communicate with each other. A RESTful application can also be said as a web service based on REST, abbreviated as **Re**presentational **S**tate **T**ransfer. The application breaks down user requests into a small series of modules, where each module addresses the part of the request. The Restful application takes JSON strings as the input given as: *{"name": "example"}*. As discussed earlier a RESTful application uses HTTP methodologies to access the resources (object, file, or block) given as follows:

- GET: Request to retrieve a resource
- PUT: Request to update a resource
- POST: Request to create a new resource
- DELETE: Request to remove a resource

Figure 2: RESTful Application(Source)

2.2 Flask Framework

Flask is a Python-based web-development framework, also called as**WSGI!** (**WSGI!**). Used as a simple wrapper around Werkzeug and Jinja and now has become the most popular Python web application framework. The Flask framework is used to start a local development server on the user system, this is done by importing **flask run** in the application by the use of the FLASK_APP environment variable.

The flask uses the *route()* decorator to determine which function will execute in the program code. Flask is used to handle the request and response between the local server nodes in our blokchain application.

Sample Flask example:

```
1 from flask import Flask
2 app = Flask(__name__)
3 @app.route('/example/')
4 def example():
5     return {'hello': 'world'}
6
7 python ./example.py
8  * Running on http://127.0.0.1:5000/
```

Figure 3: Flask(Source)

2.3 Postman

Postman is a platform for **API!** development. The Postman interface has a bunch of features that help in the building of an **API!** given as:

- API Client: Helps to send requests within postman
- Automated Testing: Helps in the testing of the API
- Monitor: Helps to check the performance of the API
- Documentation: Helps to create documentation for our API

The **API!** Client allows the user to Send Requests and View response generated by the execution of the REST queries within the Postman. It also allows the user to visualize data by the use of Postman Visualizer, in our application the Postman helps to visualize the

Memes and the Meme Formats. The platform also allows the user to create a test suite for their application by the use of the Collections feature, where each collection consists of multiple HTTP requests.

Figure 4: Postman(Source)

3 Related Work

Reddit /r/memeEconomy, community portrays memes as a potential investment where the members can buy, sell, share, and invest in them based on their potential to become a popular meme on the website. Basically, memes and economics are combined in perfect proportion. The official currency of the subreddit is **MC!** (**MC!**).

When user create an account, they are given 10,000 **MC!** and also a debt of 10,000M to pay back. The return on the meme investment is determined by the demand of that meme and the incentive is calculated by higher the meme charts on the subreddit, more meme coins the user earns. The Meme Economy limits the amount of **MC!** in the economy. It follows a risk-return trade-off pattern which is similar to crypto currencies.

Members who create a post on /r/MemeEconomy receives **MC!** in the form of royalties totaling 5% of all investments on the post. They are paid once in 24 hours after the post is released and investments after this 24 hour period will not count towards the 5% bonus. This is a way for rewarding the creators for their work.

3.1 Block Design

Each block inside the chain has the following fields:

- index - the index of the block.
- minerID - the id of the miner. This identification is used when distributing mining rewards.
- previous_hash - the hash of the previous block inside the chain
- proof_of_work - a nonce value (magic number) which makes the hash of the block to match the difficulty pattern

- timestamp - an instance of time showing the creation time of the block
- transaction_counter - the number of transactions inside the block
- transactions - a list which holds all transactions inside the block

The genesis block has "0" value on all fields and an empty list for the transactions. The hash value of the genesis block does not match any hash patterns and when validating the chain the genesis block is skipped. It should be noted that even if the block is not skipped, since all values inside the genesis block are predefined, the hash value will always be the same value and could be hard coded.

All consecutive blocks, after the genesis block, have their proper field values. The minerID field is a special field because it's value is different for each node. Each node assigns its ID to the minerID field and tries to find a nonce value with this assigned specific minerID. Also each time a new transaction is added to the memory pool of transactions, the finding nonce value process is restarted, since the transactions list is updated.

3.2 Peer to Peer Network

In a blockchain peer-to-peer connection, any node connects to any other node in the network collection of nodes. The nodes in the network are connected in the form of mesh network as shown in the **??**, where each node is connected to all of its neighboring nodes and each node contains a copy of the complete blockchain. As a new node is connected to a blockchain network then all the nodes in the network are notified about the newly connected node in the network. Moreover, the newly connected node receives a copy of the complete blockchain and also receives the list of pending transactions in the blockchain.

In our application meme economy, the peer to peer concept is demonstrated by the use of the Postman as:

- The core node must run on port 5000 in order to initialize the network, the port 5000 is selected specifically to prevent creation of multiple networks at the same time.
- Add other nodes on different port addresses.
- Now to show the peer to peer concept, we send a POST request using the **connect_to_node** method, if the request is successful we get a response showing Connection Successful on the console with status code 200.

As the connection is successful, the newly connected node receives a copy of the blockchain and list transitions and pending transactions in the blockchain network.

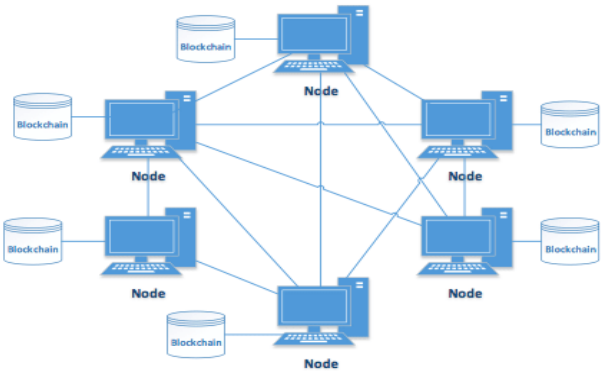


Figure 5: Peer-to-Peer(Source)

3.3 Transactions

In the Block Design section it was discussed that each block has a list of transactions. There are 5 different kind of possible transactions:

- add_memeFormat - Used to add a new meme format to the network
- add_meme - Used to add a new meme to the network
- add_upvote - Used to upvote a meme inside the network
- sell_ownership - Used to sell the ownership of a meme format
- purchase_ownership - Used to purchase the ownership of a meme format

3.3.1 Raw-JSON body input formats of transactions

- add_memeFormat - {"imagePath" : "imagePathValue",
  "name" : "nameValue"}
- add_meme - {"imagePath" : "imagePathValue",
  "name" : "nameValue",
  "memeFormat" : "memeFormatID"}
- add_upvote - {"imageVoteId":"memeID",
  "upvoteID" : "upvoteID"}
- sell_ownership - {"ownershipSaleOfferID" : "ownershipSaleOfferID",
  "memeFormat" : "memeFormatID",
  "saleAmount" : "saleAmount"}
- purchase_ownership - {"ownershipPurchaseID" : "ownershipPurchaseID",
  "ownershipSaleOfferID" : "ownershipSaleOfferID"}

3.3.2 Transaction requirements and validation

This subsection discusses the requirements of each transaction. Once all of the requirements for a specific transaction are met, then the transaction is validated and ready to be mined in the next block.

- add_memeFormat - There are not any requirements for this transaction.
- add_meme - Each meme to be uploaded must belong to a specific meme format. This meme format must be already mined inside a block and appended to the chain.
- add_upvote - The meme to be up voted must be already mined inside a block and appended to the chain. The node which up votes must have enough credits available in its wallet.
- sell_ownership - The meme format to be sold must be already mined inside a block and appended to the chain. The node which creates this transaction must have the ownership of the meme format to be sold.
- purchase_ownership - The meme format to be purchased must be already offered for a sale, mined and appended to the chain. The purchasing node must have enough credits in order to buy the meme format. The worth of the meme format is specified inside the sell_ownership transaction.

3.4 Mining and Consensus

3.5 Wallet and Rewards

This subsection discusses the role Wallets and Rewards play in the operation of the MemeEconomy Blockchain.

Every node has a unique and implicit wallet associated with it. Once a node mines a block or issues a transaction that is added to a block, all nodes start to internally, and independently track the state of the wallet associated with the node. Every node starts out with an initial wallet amount of 5 credits. The reason for this will be explained after a detailed explanation of rewards and credit distribution.

There are two situations in which a node is rewarded credits, either from another node, or from the system. Following are these situations:

- Upvote - When a meme is upvoted, a credit is deducted from the node that issued the upvote. This credit is then distributed in portions to:
  - the Meme Format owner - 30%
  - the Meme Poster - 60%
  - the Meme miner - 10%

  In addition to this, 10% of the upvote credits (which amounts to 0.1 credits) are created independently and rewarded to:

  - the Meme Format miner
  - the Upvote miner
  - every previous Upvoter from previous blocks who upvoted the same Meme

- Ownership Purchase - When a node issues a transaction to purchase ownership, 10% of the sale amount is taken in excess to the sale amount from this node, and is shared evenly between:

  - Sell transaction miner
  - Purchase transaction miner

  In effect, the Sell transaction miner and Purchase transaction miner each recieve 5% of the sell amount.

It must be mentioned that all rewards are implicitly credited to the respective node's wallet.

The reason for rewarding upvoters, is to incentivize upvoters who find a meme interesting or funny. Over time (as more blocks are mined), an upvoter is rewarded proportional to how much they contributed to the 'virality' of the meme.

There is no mechanism for a node to cancel a sell transaction. The node is free to issue a purchase transaction for their own sell transaction, if they see fit. In this scenario, they would lose 10% of the amount that they posted the ownership for sale for.

As mentioned earlier, every node is given 5 credits. This is a necessity, since if not, no nodes have any credits to upvote, nor buy ownership.

3.6 Visualization

4 Outlook

5 Conclusion

6 Template Section

This template should give us a first version we can start of with. This last section should support us in writing a more coherent paper together. Therefore I put some guidelines for the writing in section **??**. Also there are some example for the use of functionalities in Section **??**. You can copy them and adapt them the way you need them. Just leave this section here for now.

6.1 Guidelines

- **pushing to the repo**
  When you push the newest version to the repo, please leave out the files created by the compiler (besides the pdf). The report on the repo just needs the tex-file, the bib-file

(a) Meme Format (b) Meme

Figure 6: Example for 2 subfigures.

and maybe the most current pdf-file. Of course push the changes to the sub folders if you added images or sources.

- **references**
  For every section, subsection, figure, or table you include give it a label. Therefore everyone can refer to it later. It can be done by *\label{marker}*. The marker should declare the type of the object and a short (one word in the best case) name for the object. The types are "sec:" for a section, "fig:" for a figure and "tab:" for a table. You can refer to them then by *\ref{fig:example}*. Also you should use a $\sim$ symbol before instead of normal space to avoid line breaks there.

- **citations**
  For cititions the BibTex code for the source needs to be in the 'literature_list.bib' file. You can usually get them pretty easily from *Google Scholar*. Please safe all papers/sources you used in the "sources" folder in addition. The citation can then be made by *\cite{antonopoulos2017mastering}* for example. Please use the $\sim$ here too. The result then looks like this [**?**].

- **abbreviations**
  Please use the \ac{...} command to handle abbreviations. You can define them at the end of the document. Here is one example... When used the first time it automatically defines the abbreviation: **AI!** (**AI!**). For all furhter times it just prints: **AI!**. Also the plural is possible **AI!**s

6.2 Examples

```
1      # Computes the hash of the Block
2      def compute_hash(self):
3          # self.__dict__ -> all variables inside the Block class
4          encoded_block = json.dumps(self.__dict__, sort_keys=True).encode()
5          return hashlib.sha256(encoded_block).hexdigest()
```

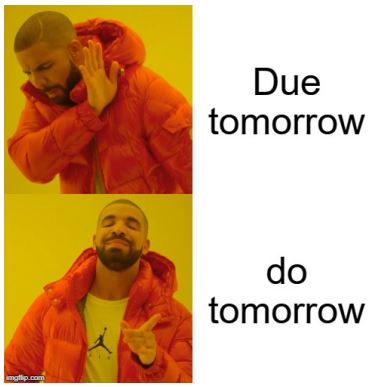


Figure 7: Example for a single figure.

	SVM	**Neural Network**
MR-1	Permutation of training & test features	Permutation of input channels (RGB channels) for training & test data
MR-2	Permutation of order of training instances	Permutation of the convolution operation order for training & test data
MR-3	Shifting of training & test features by a constant (only for RBF kernel)	Normalizing the test data
MR-4	Linear scaling of the test features (only for linear kernel)	Scaling the test data by a constant

Table 1: Content totally out of context, literally just as an example for a table.

Abbreviations and Acronyms

AI *Artificial Intelligence*

ML *Machine Learning*

API *Application Programming Interface*

WSGI *Web Server Gateway Interface*

MC *Meme Coin*

List of Tables

List of Figures