Algorithmic Feedback Optimization

Albert Pavlík

Class 6b
Kantonsschule Zimmerberg
Maturitätsarbeit

Supervised by: Mr. Martin Pfister
Korreferent: Mr. Gian Peter Ochsner

Au ZH
2024/2025

Contents

Abstract

With the rise of short-form-content-centered social media platforms such as TikTok, digital addiction has found itself conquering headlines the same way it has been conquering the free time of many. Evidently, it is dangerously easy and viciously seductive to keep on scrolling and consuming whatever the algorithm deems fit. On the other hand, public opinion leans towards labeling content production itself as difficult due to the dynamic nature of the internet, its users, and the ever-changing, seemingly arbitrary trends. But is it difficult? This paper investigates this question by means of algorithmic optimization and distance metrics. It goes on to show that by assuming certain fundamental characteristics about the target platform, a nigh 100% effective algorithm for the task can be constructed, proving that success on social media in modern times is a question of data-scientific prowess rather than creativity. Furthermore, the study highlights the significance of AI and deterministic programs in shaping everyday internet activity, and as such, this paper supports certain aspects of the dead internet theory. In addition to proposing a novel method for optimizing positive feedback algorithmically, this research raises critical ethical questions about the influence of AI on social media and its potential implications for society.

1 Introduction

With the rise of short-form-content-centered social media platforms such as TikTok, digital addiction has found itself conquering headlines the same way it has been conquering the free time of many. Evidently, it is dangerously easy and viciously seductive to keep on scrolling and consuming whatever the algorithm deems fit. On the other hand, public opinion leans towards labeling content production itself as difficult due to the dynamic nature of the internet, its users, and the ever-changing, seemingly arbitrary trends. But is it difficult? If so, how challenging is it to synthesize an "artificial content creation program" such that it gains popularity, requiring no work whatsoever once it exists? In other words: How easy is being an influencer really? And, is it possible to use today's technology, a bit of mathematics, and a few scripts of computer code to gain a stable or growing viewership? This paper shall explore and answer those questions by means of application creation and synthetic experimentation.

2 Theory overview

2.1 Relevant mathematical concepts

To gain a general understanding of how information, such as tags assigned to an internet post, is stored and processed in a mathematical environment, it is important to have at least a fundamental idea of the following key areas:

- **Discrete Analysis**: This topic elaborates on classic analysis in terms of discrete (non-continuous) data.
- **Stochastic and Statistic Methods**: This topic incorporates correlations, distance metrics, cosine similarity, mutual information, and denoising techniques.
- **Linear Algebra**: This topic includes concepts such as vectors, scalars, matrices, and their operations.
- **Algorithms**: A general understanding of algorithms is also considerably handy.

Even surface-level knowledge of the aforementioned topics facilitates comprehension of the methods described. Therefore, inspecting the relevant theory is key to understanding the essence of this project.

2.1.1 Discrete Analysis

As opposed to regular analysis, discrete analysis focuses on discrete data rather than continuous, meaning that the data being treated may be considered *finite* or *enumerable by integers*, whereas continuous functions may be regarded as *infinitely precise* on any given interval. To visualize this:

Continuous Function The following graph represents the continuous version of the function is $f(x) = x^2$, where x can take any real number. This function is smooth and defined over all real numbers.

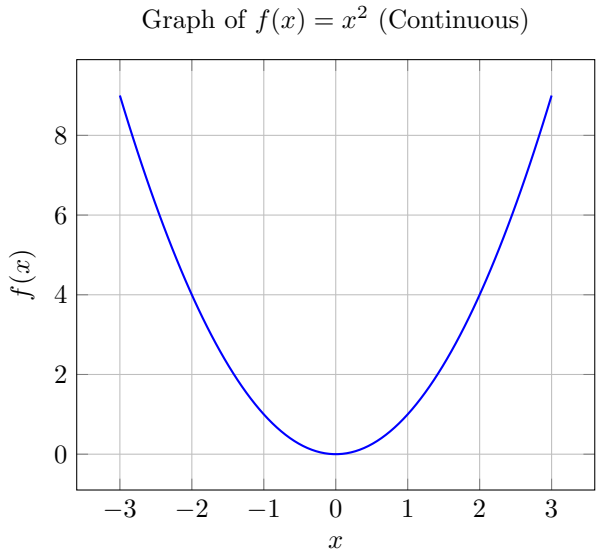


Figure 1: Graph of the continuous function $f(x) = x^2$.

Discrete Function The following graph represents the discrete version of the function is $g(n) = n^2$, where n is restricted to integer values (regularly spaced). This function is only defined at integer points, and its graph shows distinct points rather than a smooth curve. However, the similarities between the discrete and continuous versions are evident.

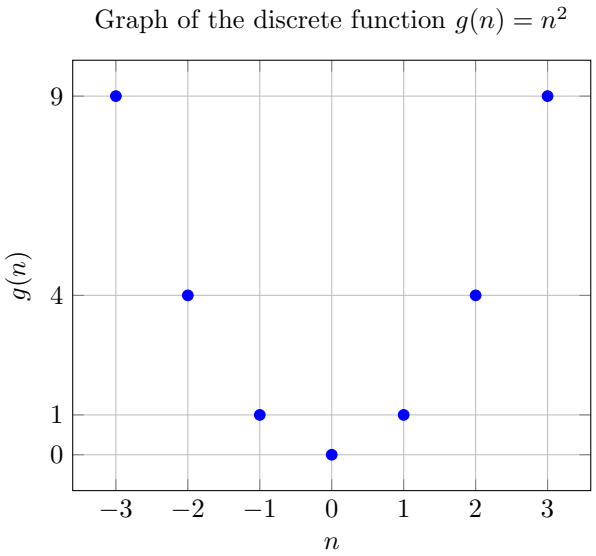


Figure 2: Graph of the discrete function $g[n] = n^2$.

Context The reason discrete mathematics is important in everyday life is that no process in the real world is perfectly infinitesimally precise. Additionally, no past, current, or future computer is able to hold or measure perfectly continuous mathematical functions, as they are inherently infinite in the data they hold, which forces data science to fundamentally operate on discrete data, obtained by regularly spaced or event-based measurements or, logically so, by computer generation. Due to this project's main focus on operations using well-curated data, this chapter may be confined to only the techniques associated with regularly enumerable functions in the form of n-dimensional arrays. [1] [2]

Essential discrete practices Let f = [1,2,3,4,5,6] be a discrete function defined on the integer interval [0;5]

- **Differentiation:** f can be differentiated by simply taking the difference between each entry and it is predecessor (if possible), yielding f' = [1,1,1,1,1]. As evident, the derivative is, as per all expectations, constant at 1; however, it inevitably loses information (the array is now 1 entry shorter). Note that this analogous to the difference quotient for analytic

functions[1]:

$$f'[x] = \lim_{\lceil h \rceil \to 0} \frac{f[x + \lceil h \rceil] - f[x]}{\lceil h \rceil}$$

- **Integration:** f can be integrated on the whole interval by simply adding up all of its contained values, yielding 21, or the anti-derivative may be computed to F = [C, C+1, C+3, C+6, C+10, C+15, C+21], where C is some constant.

- **Normalization:** To normalize f, its values are adjusted to fit within a specific range, typically [0,1]. This may be performed by dividing each value in f by its maximum value:

$$f_{\text{norm}} = \left[\frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}, \frac{5}{6}, \frac{6}{6}\right] = \left[\frac{1}{6}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{5}{6}, 1\right]$$

  Note that sometimes, the minimum value is subtracted from the numerator to force an initial value of 0. Usually, that is done for nothing more than aesthetic purposes.

- **Resampling and Interpolation:** The definition of those practices should be familiar enough, but they need to be mentioned here, as they play a key roll in working with varyingly sampled data.

Gradients: When dealing with multi-dimensional functions particularly in terms of optimization, one of the most essential tools is considered to be the gradient. defined as follows:

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right),$$

The gradient of f provides the direction of the steepest ascent of the function at any given point x_n in the form of a vector of $n-1$ dimensions. In the discrete realm, differentiation is implemented as mentioned above.

[1]Note that $\lceil h \rceil$ will always be 1, leaving the difference of the $n+1-$th element and the $n-$th element in the array

To visualize this: for $f(n, m) = n^2 + m^2$, the gradient

$$\nabla f(n, m) = \begin{bmatrix} 2n \\ 2m \end{bmatrix}$$

is pointing in the direction where the function increases most rapidly, the proof of which is left to the reader as an exercise for intuition purposes. In this use case, gradients are particularly useful when trying to maximize a dynamically changing process, or, on the other hand, when attempting to discern local minima in a loss function for example.

2.1.2 Stochastic and Statistic Methods

Virtually any analysis in regard to reverse-engineering a process with multiple components, the mutual effects on each other are not known, is reliant on stochastic and statistic methods such as correlations or mutual information to unveil and establish a solid metric for those exact mutual effects. Of those methods, the most essential for this application are the following ones: [3] [4] [5] [6]

- **Pearson correlation coefficient:** Usually denoted as r, the Pearson correlation coefficient is a metric for the linear correlation between two arrays of data X *&* Y by dividing their covariance by the product of their respective standard deviations:

  $$r = \frac{\mathrm{Cov}(X, Y)}{\sigma_X \sigma_Y}$$

  where $\mathrm{Cov}(X, Y)$ is the covariance between X and Y, and σ_X and σ_Y are the standard deviations of X and Y, respectively. The coefficient r ranges from -1 to 1, where 1 indicates a perfect positive linear relationship, -1 indicates a perfect negative linear relationship, and 0 indicates no linear relationship whatsoever. Note that, at times, it is sensible to use $|r|$ instead of r

- **Correl-factor:** To understand the functionality of the cross-correlation relevance factor, correl-factor for short, it is apparent that one needs to have a light understanding of what a cross-correlation is:

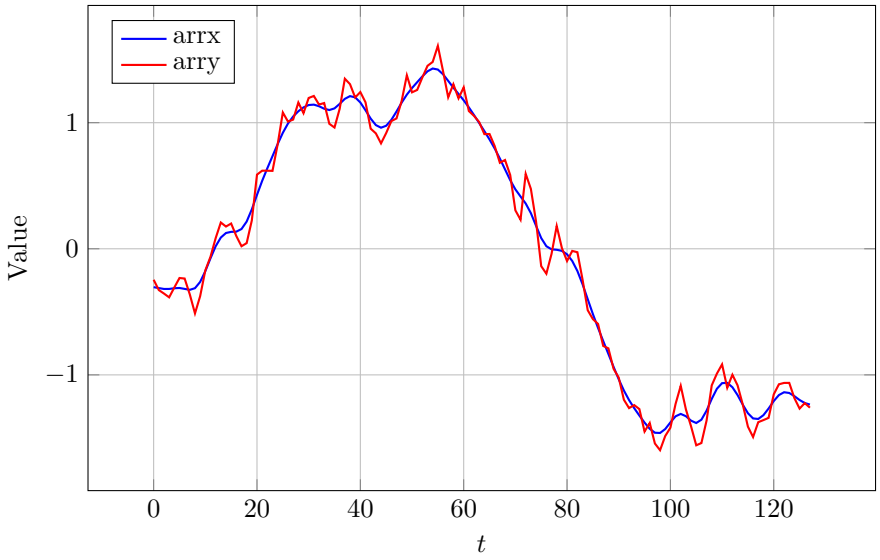
Cross-correlation The cross-correlation function is an element from **signal processing** which shows the development of the correlation vector over time. For the sake of completeness, this is how one may compute the cross-correlation function for discrete signals, which is what is being dealt with in this scenario:

$$(f \star g)[n] = \sum_{m=-\infty}^{\infty} f[m] \cdot g[m+n]$$

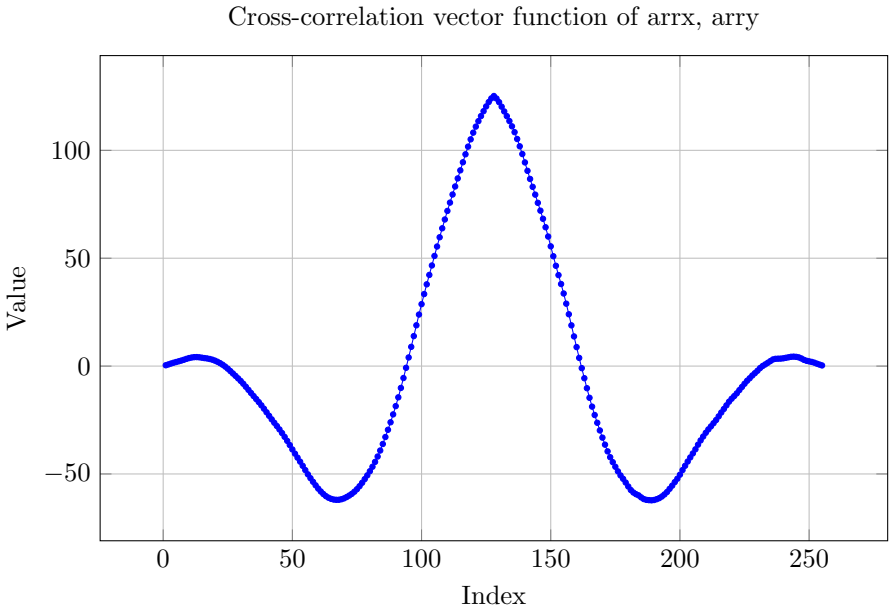
where $\star$ is considered the cross-correlation operator. Note that the bounds are set at ∞ and $-\infty$ in this case, which should be substituted for the desired range in a realistic case.

Visualization Rather than understanding the math, it is of an exponentially higher import to get an intuition for how such a cross-correlation function might behave. For this:

let *arrx, arry* be two differently filtered Brownian Noise Functions

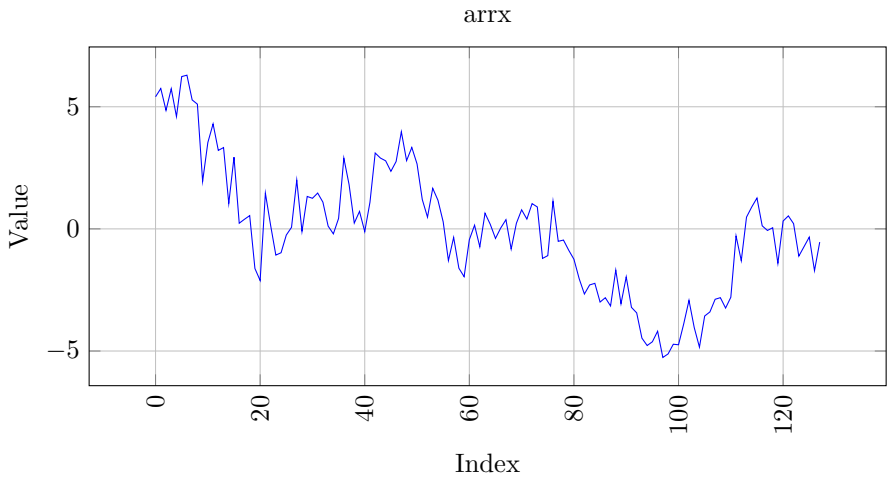


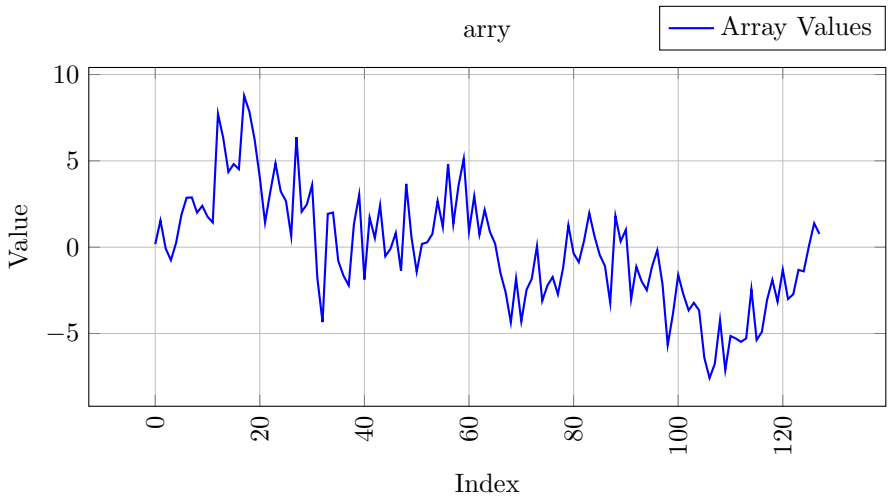
Obviously, those functions are extremely similar, as they are but the same original function filtered in a different manner. Therefore, one would expect to be able to deduce that from the cross-correlation vector function:



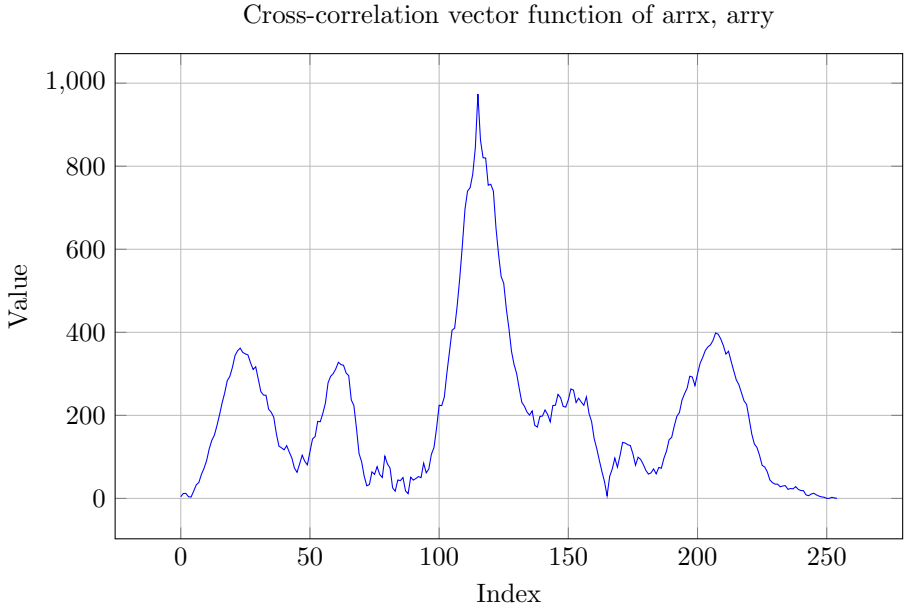
As expected, the function is showing a high peak at the point of the highest correlation, which is, logically, the middle, signifying a difference in τ (time lag) of 0. This makes sense, as the functions haven't been transposed in time to create a τ. If they were, the peak would have shifted either to the right or to the left, depending on the shift's $sgn()$.

Of course, if two arrays similar to such a degree are provided, computing a correlation is nigh nonsensical. Therefore, consider the following more realistic example:





Consider the following two arrays carefully. Both of them have a component in common, and, additionally, one has been slightly transposed in time. Regard the patterns. To verify that there, in fact, are similarities to be discerned, one may inspect the cross-correlation function:



Note that the absolute value of the function is being represented here. This is conventionally done, as a high deviation from 0 signals correlation regardless of the $sgn()$. Also, notice how the tallest peak has been shifted left by about 12 units. One may have inferred this attribute of the function even before viewing the graph by observing that the events in $arrx$ precede their corresponding counterparts in $arry$ by this exact number of units.

The factor: Now, that it is clear how cross-correlation itself works, the gate to computing the factor has finally been opened. The computation, in its essence, is fairly simple:

1. let f be a correlation-vector function
2. let $p(f)$ be a boolean function, returning f(t) if f(t) is a peak of the function f, else 0
3. let P be the reversed sorted array of the output of p
4. let H be the first element of P and remove it from P, as that is the tallest peak's magnitude
5. let h_{ave} be the average of the first half of P's elements [2]
6. let r equal the squared ratio between h_{ave} and H:

$$r_{\text{correl}} = (\frac{h_{\text{ave}}}{H})^2$$

Note that, in most cases, the functions are differentiated and smoothed prior to computing the correlation function, after which said correlation function is smoothed again to remove noise prior to computing r.

[2]Note here that the fraction of the elements used for computation is highly dependent on the size of the array used. For arrays on a small scale as in this use case, $\frac{1}{2}$ is reasonable; however, as the order of magnitude rises, the fraction of relevance has to be scaled down accordingly.

- **Cosine similarity:** Given two vectors $\mathbf{A}$ and $\mathbf{B}$, the cosine similarity between them is given by:

$$\|\mathbf{A}\|\|\mathbf{B}\| \cos\theta = \mathbf{A} \cdot \mathbf{B}$$

where θ is the angle between vectors $\mathbf{A}$ and $\mathbf{B}$

$$\text{sim}(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \cos\theta$$

where $\mathbf{A} \cdot \mathbf{B}$ is the dot product of $\mathbf{A}$ and $\mathbf{B}$, and $\|\mathbf{A}\|$ and $\|\mathbf{B}\|$ are the corresponding magnitudes of the vectors $\mathbf{A}$ and $\mathbf{B}$. It follows that the cos similarity ranges between -1 and 1 or, in some cases, between 0 and 1, where 0 means the vectors are orthogonal (they share little information), 1 signifies that $\mathbf{A}$ and $\mathbf{B}$ are proportional, and -1 indicates that the two vectors are facing in opposite directions. Please note that any discrete array can always be thought of as a vector, and, subsequently, treated as such.

- **Mutual information:** Mutual information is a measure of the amount of information that one random variable contains about another. It quantifies the reduction in uncertainty about one variable given knowledge of the other. It is therefore also a metric that measures surprise. [7]

  The mutual information $I(X;Y)$ between two random variables X and Y is defined in terms of entropy as:

$$I(X;Y) = H(X) + H(Y) - H(X,Y),$$

  where:

  - $H(X)$ is the entropy of X, quantifying the uncertainty of X.
  - $H(Y)$ is the entropy of Y, quantifying the uncertainty of Y.
  - $H(X,Y)$ is the joint entropy of X and Y, representing the uncertainty of X and Y together.

An equivalent expression for mutual information is:

$$I(X;Y) = H(X) - H(X \mid Y),$$

where $H(X \mid Y)$ is the conditional entropy of X given Y. This form in particular highlights that mutual information measures the reduction in uncertainty/surprise about X after observing Y, encompassing its nature.

Furthermore, mutual information is always non-negative and symmetric:

$$I(X;Y) \geq 0 \quad \text{and} \quad I(X;Y) = I(Y;X).$$

It equals zero **if and only if** X and Y are **independent**, rendering it a similarity metric worth considering for most applied stochastic ventures.

- **Spearman's rank correlation coefficient:** Usually denoted as r_s, Spearman's rank correlation coefficient can be thought of as a variation of the Pearson r value, where the covariance and standard deviation of the variables **ranks** are computed rather than their values, the rank being their respective index in a **sorted array**.

$$r_s = \frac{\mathrm{Cov}(R(x), R(y))}{\sigma_{R(x)}\sigma_{R(y)}}$$

As opposed to the "standard" r value, the r_s value is not limited to linear correlations, and rather focuses on *monotonic* relationships. Needless to say, the r_s value ranges between -1 and 1.

2.1.3 Linear Algebra

As mentioned in the previous chapter, any discrete array may be thought of as an n-dimensional tensor of some sort, meaning one may encode information to said tensors and operate based on them. For this purpose, however, it is key to understand some **tensor operations** [8] [9] [10] :

- **Addition and Subtraction**: Element-wise operations where corresponding elements of tensors are added or subtracted.

$$\mathbf{a} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

$$\mathbf{a} + \mathbf{b} = \begin{bmatrix} 1+3 \\ 2+4 \end{bmatrix} = \begin{bmatrix} 4 \\ 6 \end{bmatrix}, \quad \mathbf{a} - \mathbf{b} = \begin{bmatrix} 1-3 \\ 2-4 \end{bmatrix} = \begin{bmatrix} -2 \\ -2 \end{bmatrix}$$

- **Scalar Multiplication**: Multiplying each element of a vector by a scalar value.

$$\mathbf{a} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad \text{Scalar} = 3$$

$$3 \cdot \mathbf{a} = \begin{bmatrix} 3 \cdot 1 \\ 3 \cdot 2 \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \end{bmatrix}$$

- **Dot Product**: Summing the products of corresponding elements of two vectors.

$$\mathbf{a} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

$$\mathbf{a} \cdot \mathbf{b} = 1 \cdot 3 + 2 \cdot 4 = 3 + 8 = 11$$

- **Element-wise Multiplication (Hadamard Product)**: Multiplying corresponding elements of two vectors.

$$\mathbf{a} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

$$\mathbf{a} \circ \mathbf{b} = \begin{bmatrix} 1 \cdot 3 \\ 2 \cdot 4 \end{bmatrix} = \begin{bmatrix} 3 \\ 8 \end{bmatrix}$$

- **Transpose**: Re-arranging the dimensions of a tensor, swapping axes. Please note that this is usually not needed for vectors, and mainly used for higher-dimension tensors.

$$\mathbf{a} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$\mathbf{a}^\top = \begin{bmatrix} 1 & 2 \end{bmatrix}$$

- **Reshaping**: Changing the shape or dimensions of a tensor while keeping the same total number of elements.

$$\mathbf{a} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

$$\text{Reshape to } 2 \times 2: \quad \mathbf{b} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

- **Cross Product**: An operation on two 3-dimensional vectors that results in a vector perpendicular to both.

$$\mathbf{a} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\mathbf{a} \times \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\text{General formula: } \mathbf{a} \times \mathbf{b} = \begin{bmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{bmatrix}$$

Later on, these techniques are applied without much mention of their use, as they are trivial.

2.1.4 Algorithms

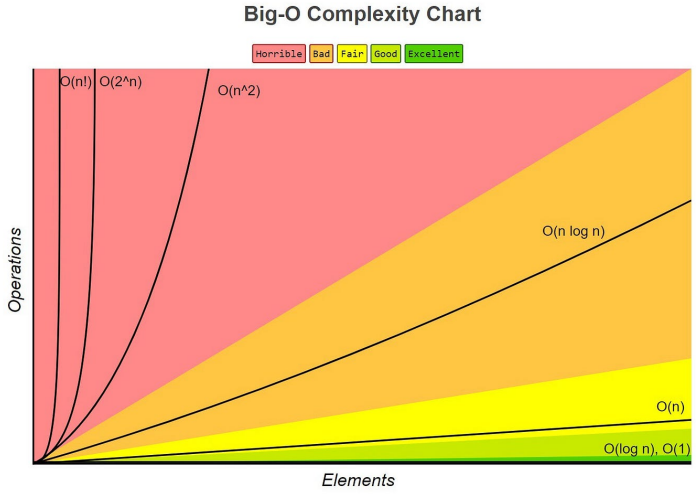
In mathematics as well as computer science, an algorithm is a finite set of strict instructions to be performed on an input to yield a specific output. Logically, there are an infinite number of possible algorithms to solve a particular task. Nevertheless, some algorithms are better than others. [11] [12]

How is that measured?

- **Time Complexity (Big O):** The time complexity of an algorithm, usually denoted $O(\cdot)$, describes how the runtime of an algorithm grows with respect to the size of its input. It is used to classify algorithms according to their performance or efficiency.
  As an example, consider an algorithm designed to sort a deck of n cards until it is in sorted descending order. If this algorithm simply shuffles the cards randomly, it would, on average, require $n!$ shuffles to sort the deck in the desired order. Therefore, its time complexity would be $O(n!)$. Note that the $O(\cdot)$ notation is only an approximate representation of an algorithm's actual runtime, as it is typically simplified to represent only the highest-order term. E. g.

$$O(n^2 + n) \to O(n^2)$$

  Note that some people might include more or less information in the $O(\cdot)$ they provide for a certain algorithm.



[13]

The image above may facilitate the general understanding of how generally "good" or "bad" algorithms may perform. It goes without saying that some algorithms cannot be further improved despite a conventionally "excellent" runtime.

- **Accuracy and Error Eate:** As is often the case in life, speed is not everything; quality matters, too. And, while one algorithm might outperform another in terms of time complexity, it might sacrifice precious accuracy in the process. The accuracy of an algorithm can be measured based on the result it returns as opposed to what the correct solution to the problem may be. In some cases, like engineering, an error of + or - 100% might be acceptable, whereas, in every other even remotely respectable science field, an error of just .1% might be detrimental. In this example, engineers, if they knew how, could opt to use a faster algorithm while sacrificing some accuracy, whereas all the others would need to wait a bit, as a more precise algorithm might take longer to finish. Subsequently, both parties could compute the respective error rate of their algorithm, and reconsider their life choices, or look down on engineers, correspondingly. Let it be noted that an algorithm superior to another in time complexity does not necessarily have to be less accurate.

2.2 Relevant Application Prerequisites & Specifications

For the purpose of constructing a synthetic influencer, a number of key decisions & assumptions have to be made beforehand, the essence of which may be split into the following categories:

- **Assumptions about TikTok's recommendation algorithm:** A general idea of how TikTok tends to recommend content to its users
- **The required software engineering methods:** A vague outline of how a synthetization could be made possible in a digital environment
- **Python repositories** The most viable toolkit for the operation

2.2.1 Presumptions regarding TikTok's recommendations

Devising an algorithm to fit an unknown case is, as logically inferable, difficult and might fall into the domain of impossibilities, or, if nothing else, it falls into the domain of gambling, which has little to do with science; therefore, it is of high import to obtain an overview of how TikTok's recommendations work in terms their correlation with the users' actions. For this purpose, the following reasonable assumptions have been made such to lay a foundation for all future work. The below assumptions are backed by TikTok's own disclosure [14] and personal experience of influencers [15] as well as logical deductions:

- **Positive correlation between likes & views:** The assumption is that a post that is receiving a high influx of interactions (*likes*) will be or is, at a given point, receiving a proportionally high influx of views and vice versa. These events should exhibit high correlation over time with a small Δt. Additionally, the assumption is that a post that is receiving a disproportionally high relative influx of interactions should not be further pushed, as the set of people who leave an impression belong to a subset people who view the post. This indicates the post might be ineffective at reaching a broader audience.

- **TikTok knows what you want:** The assumption is here that upon viewing posts from one category multiple times, say meme videos for example, TikTok automatically suggests posts of the same caliber, which then appear on the user's feed.

- **Some things just work:** The assumption is that there are certain categories of posts that simply score higher than others in a view-count ranking. Furthermore, it may further be assumed that those categories remain somewhat constant over a short period of time with but slight changes in their popularity distribution.

- **What works should stay, what doesn't should go:** Here, the assumption is that if certain categories of posts tend to receive more views as well as interactions relative to other categories, it stands to reasons to push that category to receive even more views, whereas categories that receive few views or impression should be reduced in future posts.

- **Positive feedback loop:** It may be reasonable to assume that the more views a post receives, the more views its successor will receive, which, in turn, means that, if done correctly, the view curve influences its derivative positively, pushing itself to a higher-order fit.

2.2.2 Necessary software engineering methods

Apart from mathematical theory, there is a considerably sizable set of software engineering tools required to build this project from ground up. The main categories include the following:

- **Web scraping:** Needed to get the necessary material for posts
- **LLM knowledge and API integration:** Needed to label the material
- **AI model creation, training, fine-tuning:** Needed for voice-overs
- **Data structures and indexing capabilities:** Needed for large data control
- **Event-based and time-based automation:** Needed for the program to run without human supervision
- **API handling:** Needed to work with TikTok's API
- **Data curration knowledge:** Needed for handling the data - methods include everything mentioned in the *Theory* section
- **At least minimal programming capabilities:** Needed to piece those parts together in one solid application
- **Web server maintenance knowledge:** Needed to have the program run at the desired time

2.2.3 Python repositories in question

To further actualize the prerequisites above, the following python packages will be used. Obviously, there is no one correct way to implement those ideas, and every other programming language & alternative packages would do just as well; however, the following ones are the ones that this project is based on, which, in case of a replication attempt, might facilitate the work required:

- **Web scraping:**
  - Selenium - A tool for automating web browsers
  - Requests - A simple HTTP library for making requests
- **LLM knowledge and API integration:**
  - OpenAI Python API - Python client library for OpenAI's API
- **AI model creation, training, fine-tuning:**
  - Coqui TTS - Text-to-Speech library for creating and training AI models

- **Data structures and indexing capabilities:**
  - Pickle - Python's built-in object serialization library
  - Pandas - Data structures and data analysis tools
- **Event-based and time-based automation:**
  - Flask - A micro web framework for Python
  - datetime - Provides convenient access to various tools regarding time
- **API handling:**
  - Requests - A simple HTTP library for making requests
- **Data curation knowledge:**
  - NumPy - Library for numerical computing in Python
  - SciPy - Library for scientific and technical computing
  - scikit-learn - Machine learning library for Python
- **Web server maintenance knowledge:**
  - Flask - A micro web framework for Python

3 Theoretical application

The theoretical application itself consists of 2 main parts, the first of which is essentially the idea and the corresponding pipeline, and the second of which is the concrete approach to each step in the pipeline. Consequently, the concrete approach can only be as good as making use of the general idea's maximum potential, wherefore meticulous work is to be deployed with respect to every seemingly insignificant feature in the final project's structure, as any logical fallacies may lead to failure.

3.1 Project structure

The overall idea of the project is to assemble TikTok videos out of either solid image memes or stories. Upon retrieval, the data is labeled with 3-5 categories out of a predetermined set, after which a distribution amongst the assigned categories is created. The labeling is conducted by an LLM model. The data is subsequently saved into a database and later assembled into a video of some optimal format, during the process of which the text contained in the memes is converted to clear text using a text recognition software and read aloud by a custom TTS model, or, in the case of stories, the text is rendered to voice using the same TTS model. subsequently, a final distribution of categories amongst all the memes (*or story*) included in the video is created and added to a second database, from where it could later be posted to TikTok. However, due to the time constraints put on this project, no actual posting can take place. Therefore, only the steps to immediately before posting are mentioned. Instead of real-world testing, experimentation is conducted in a synthetic environment to offer proof of concept. However, when applied, posts would be conducted at fixed intervals, and each post would be tracked for a duration of about 5 days, after which the weighted sum of interactions and views would be parsed alongside the video's distribution to the main algorithm. This algorithm is supposed to optimize the tag distribution vector to find the direction of the fastest ascent in terms of interactions and views. Additionally, it is supposed to ascertain the 3 categories with the highest $\frac{Interactions\&Views}{Appearance}$-ratio. In short, for every next generation, the category distribution vector of the video is attempting to match the algorithmically determined current ideal as closely as possible while retaining new information such as imperfections while matching the ideal, a *misc* category, or noise[16] added to the ideal vector to prevent overfocusing. Excessive testing is conducted to prove the effectiveness of the methods synthetically while operating on advocable assumptions about the nature of real-world circumstances to render the devised methods applicable.

To lay this out legibly, the project's structure may be broken down to the following 4 categories:

1. **Data retrieval & labeling:** The project consists of 2 pipelines, the data for which has to be collected from different sources as so:
   (a) **Memes:** The memes themselves are gathered from Reddit's most popular meme-focused communities, which in turn guarantees a certain degree of success, as the memes collected are only those that have been trending in a certain close time frame (*a week or a month at most*).
   (b) **Stories:** As for the stories, they are also gathered from Reddit. This includes mainly the *writing prompts* community, but there are others that can be accessed, too. Here, it is also possible to create stories using an LLM later on.
2. **Optimal video formatting:** For the experiment to achieve success, it is vital to create videos in a format enjoyable for the average recipient. Popular videos might share traits like:
   (a) 30sec to 1min duration
   (b) Animated background
   (c) Live subtitles (for stories)
   (d) Certain popular hashtags
   (e) Link to a popular song within TikTok (more on that later)
   (f) A small range of audible background songs (more on that later)
   (g) etc...

   Traits like the above ought to be incorporated meticulously into every Post produced to maximize the potential success.
3. **Feedback retrieval:** In an actual scenario, the feedback to a video would be measured by taking a weighted sum of the interactions and views from a post after a specified amount of time (*3-5 days*), which could be achieved through means of TikTok's API. The feedback would subsequently be saved in a history file to aid subsequent analysis and optimization.
   In this synthetic case, however, feedback is calculated and saved by a function designed to resemble that which could be expected from a platform such as TikTok.

4. **Algorithmic content optimization:** Roughly speaking, the central algorithm takes *nd*-vectors as input, where said vector includes the category distribution of a post amplified by the weighted sum of interactions & views. Additionally, dithering is administered with every addition to avoid overfocusing and aid dynamic adaptation to the current trend. For every post, the vectors are added towards the "ideal" vector currently presented by the algorithm in a manner of EMS (*i. e. exponential moving average*). Ideally, the vector should change direction when it is met with an input of a successful video to incorporate more parts of that and, consequently, assemble a video of the next generation, the composition of which is closer to the successful video than the generation before. For this, it is essential to understand that no video assembled perfectly resembles the "ideal" vector and thereby exhibits a certain degree of deviation, which is essential for the algorithm's development. To further support this mechanic, the vector is normalized each time it receives an input; plus, it receives random white noise as well.
   Apart from the features above, the algorithm is also keeping track of the vector's development with respect to the correlation of its gradient over time and the development of each component over time. This procedure returns the *3-5* most popular categories at any point in time with respect to the relative rise of popularity (*first derivative*). Categories yielded by this method are further pushed by amplifying their presence by a constant factor in the definitive "ideal" vector for the next generation.

3.1.1 Data retrieval & labeling

Retrieval In every data science project, data retrieval is one of the main issues. In this case however, however, it is exceedingly easy to obtain large amounts of data from the internet itself seeing as how popular the concepts of picture-memes and Reddit stories are. Very little research reveals that most memes and stories posted on TikTok do, indeed, originate from Reddit's popular communities:

- **memes:** r/memes, r/meme, r/meirl
- **stories:** r/writingprompts, r/aita

Having ascertained the sources of data, it is further advisory to apply certain rules to determine which part of the data is relevant. Seeing as the primary goal is to reach as many views/interactions as possible when reposting the retrieved content, it becomes apparent that memes/stories with already high impression rates on Reddit have high chances of securing similar popularity elsewhere. Therefore once per a certain time period (ideally once a month), the memes/stories with the highest impression score are stored to a local database in addition to a number of the most popular of all time. With every post stored, the first 10 comments are stored, too.

Labeling As mentioned before, every meme/story needs to be tied to a corresponding vector (tag distribution). Labeling is done using an LLM (gpt-4), which is provided with a starting prompt as follows as well as the comments and the extracted text from the post [3] if possible. The title of the post is also provided to the LLM, after which the GPT API returns a vector with a distribution over the 5 most prominent tags in the input, where the rest is padded with zeros.

3.1.2 Optimal video formatting

While the optimal format settings for a shot-form post may be highly disputed, shared characteristics between successful video may be discerned. To be precise, those characteristics include:

1. **Duration:** Usually, as per TikTok's own report, videos ranging between 21 and 34 seconds score the highest engagement as well as views. However, to make actual money posting, it is vital to post videos of length 60 seconds or more, for which reason, 60 seconds is chosen as the desired video duration in this project.

2. **Background Music:** Each post on TikTok's platform may be linked to any song or sound within their library, which may result in a post's heightened engagement or view count simply based on the connection to a popular song and the subsequent push by TikTok's algorithm. Note that there may only be one linked song or sound per post. Note additionally that a referenced song is not necessarily one audible in the video, as one may simply reference a song while keeping their own audio within their

[3] Whole story or extracted text using tesseract within a meme

original post (This is the approach taken in this case). The song referenced in every post is *Love You So* by **The King Khan and BBQ Show**, as it has some of the highest engagement rates all while also being tied to the most popular videos every week, as of now, as well as generally occupying the usage charts on TikTok no lower than top 10. All things considered, *Love You So* seems to be the most common song tied to posts of type meme/Reddit story, which is why it is employment is beneficial to further boost TikTok's suggestion algorithm on each post tied to it.

3. **Information Transfer:** [17] Information may be transferred from a post to a viewer via a multitude of channels such as:

   - audio - *what they hear in the video*
   - visuals - *the imagery they see supporting what they hear, keeping the engaged*
   - text - *preferably a transcription of what is being said or key words*

   The combination of all the above categories is what keeps a viewer engaged and glued to a screen. Research suggests that providing a person with subtitles/captions in addition to something they hear may relax them, heighten their cognitive engagement, subconsciously suggest an option of more accessibility (as a video with subtitles may be stopped at any given point if needed to process the information received and/or for re-reading purposes), all while allowing the viewer to retain the conveyed information better and longer. This might not seem as important in this specific case; however, it is crucial to remember that videos that are more pleasant to watch tend to receive higher engagement. Therefore, live, small-chunk subtitles are employed for Reddit story readings.

4. **Background Imagery:** In addition to the core contents of each post, there needs to be, as mentioned above, some form of engagement ongoing in the background to keep the viewers brain and senses fully occupied at all times, subsequently leading to a higher watch time. At this point, there needs to be found an equilibrium between under-stimulation and over-stimulation:

   - **under-stimulating** videos might appear boring or too long to be watched, leaving the viewer dissatisfied and consequently, losing their attention/watch time

- **over-stimulating** videos with too many elements playing into different directions at once may strike a viewer as confusing or *too much*, rubbing them unpleasantly, losing their attentions/watch time as a direct consequence

Having understood those concepts, the conclusion may be reached of providing the user with slow, colorful, and mostly predictable visual stimulation in the background of the post. Note that the provided content needs to be pleasant on the eyes even if watched in a dark room. For this purpose, a highly successful background video in this branch of TikTok is used: *Slow Minecraft Parkour*. Incomprehensible as it might sound the first time considering this option, it needs to be noted that it is a very solid one, being a familiar concept to crushing majority of this project's target subset of the TikTok audience while checking all the above criteria. The success of this concept is also evident, appearing in a large percentage of the most popular meme/Reddit story posts.

5. **Hashtags:** Hashtags, even though they used to be a vital part of social media activity, have become almost obsolete in recent times. Often, videos without hashtags or a description *go viral* or receive comparable influx in attention to posts with hashtags and proper descriptions. They are not considered in this project, as their importance is fleeting

3.1.3 Feedback retrieval

In view of the fact that this is not done, as this project lacks the time to implement months worth of experimentation, no concretized layout of the concept is needed. This category is fully substituted in this project by an analogous synthetic counterpart that operates on the assumption that there exists a combination of tags that are liked more by a given audience, leading to a rise in positive feedback. Provided this assumption translates into the real world, then feedback is handled analogously by collecting *"likes"*[4].

[4]Or, an evaluation score in terms of synthetic testing

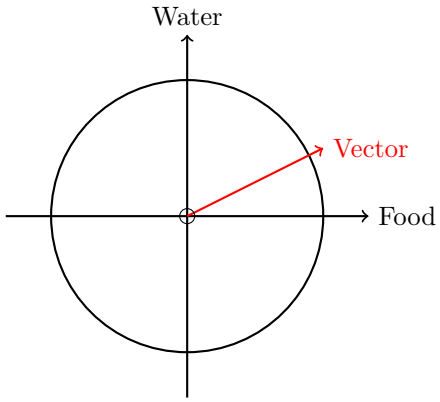
3.1.4 Algorithmic content optimization

As previously stated, the content optimization hinges on a vector-fitting scheme; however, within the general idea it is also foreseen for it supported by highest-similarity component combination $\frac{d_{(c_1+c_2+c_3)}}{dt}$ hyper-scaling with respect to the $\frac{d_{eval}}{dt}$ function to achieve a higher-order or, synonymously, a *"faster"* growth in evaluation. The above idea might sound complicated at first sight; however, it is not as suffocating in reality. In order to comprehend what the egregiously sounding idea described above encompasses, a broken down pipeline might prove convenient.

1. Conversion of tags to a vector
2. Finding best fit from database
3. Retrieving evaluation
4. Treating the vector
5. Calculating similarities to the $\frac{d_{eval}}{dt}$ function with respect to the individual vector component functions
6. Supporting and scaling the vector accordingly

Tag conversion As discussed above, the tag array is nothing else than a distribution over 25 categories adding up to 1. It may be interpreted as a vector and treated as such if and when necessary. This has certain useful qualities in this case, allowing for the application of vector distance metrics such as cosine similarity. To understand why it is possible to semantically interpret the distribution as a vector in this case, it is vital to consider what the exact goal is. Essentially, the vector is trying to find the direction of the highest reward, as it is looking for a sort of optimum, where the individual tags' weights measure how heavily the vector is incorporating their importance. A 2d equivalent could look as follows[5]:

[5]The y-axis represents the weight of water in the lunch distribution, while the x-axis represents the weight of food. Note that both of these values are positive, forcing the vector into the first quadrant. This holds true for any nd space



Optimal Lunch Composition

The concept when extended into 25 dimensions stays effectively the same. An interpretation as a vector is further convenient when regarding its changes over time. Seeing as it needs to be able to look for an optimum based on the input it receives, it needs to consequently move within said 25-dimensional space. A vector interpretation makes tracking of this movement very easy as the changes in each component *(In the example above: the percentage of water and food)* can be tracked individually, subsequently interpreted as a function of time or another vector. Therefore, the tag distribution is represented in a $1x25$ array. Two of these vectors are of relevance here:

1. **The main vector:** The main vector is the vector representing the instantaneous optimum, based on which a subsequent post is selected with respect to its own distribution to fit the current *"ideal vector"* the closest.

2. **The individual vector:** The individual vector is the vector assigned to each post within the database. It represents its corresponding tag distribution. It serves as a base for post selection.

Finding best fit from database Having interpreted tag distributions as vectors, finding the closest fit from the database poses little challenge, as almost any similarity metric may be applied reliably, the most interesting of which are:

- Cosine similarity (sim): Perfect for identifying closely related vectors

- Any linear correlation metric such as:
  1. Pearson correlation coefficient
  2. Spearman correlation coefficient
  3. The correl-factor
- Mutual information, as the problem is dealing with distributions

Note that there are other metrics such as *DTW (dynamic time warping)* or *Distance correlation*; however, there is no need to resort to these methods when dealing with a situation of no warping [6] as well as no τ in the system [7]. Experimentation is needed to discern the most reliable metric; however, it is highly improbable that any particular one of those mentioned above would fail to prevail in this task.
After determining the closest vector in the database, the corresponding post is selected for processing, as it follows the current ideal, pointing in the direction of highest expected positive feedback.

Retrieving evaluation In a real-world scenario, evaluation retrieval is about the simplest task out there. After a certain period since epoch, a program collects a combination of likes and comments as a scalar value. However, in synthetic testing, evaluation retrieval might pose a slight challenge since it has to be computed based on assumptions. For the moment, the assumption is that there exists a combination of categories/tags that are enjoyed by a given audience more than others, and that, as a direct consequence, more positive feedback (likes) is given to a post including these categories as a reaction. Whether they scale linearly to the proportional weight of the desired tags in a given distribution or quadratically or exponentially, and whether or not there is a maximum to be hit is, in the grand scheme irrelevant. The only assumption of importance is that which states that positive feedback scales in some proportion p reciprocally to the distance between desired and provided distributions.
After calculating the evaluation for each individual vector, it is, in turn, saved as a scalar value for subsequent use in vector treatment as well as in a separate storage file as a function of time/iteration to be used as means for later optimization, as it is that which is to be maximized.

[6] warping means stretching or compressing the function in time
[7] τ is the δt in a particular system. It is not present here as it would symbolize a shift in distribution, which is to be avoided.

Treating the vector The idea in treatment of the **main vector** is the following:

1. Let $\vec{v}_m$ be the current main vector
2. Let $\vec{v}_{sim}$ be the instantaneous most similar vector from database
3. Let e be the evaluation scalar of $\vec{v}_{sim}$ given by $e = f_{eval}(\vec{v}_{sim})$
4. Let $w[n]$ be the discrete white noise sequence with low amplitude, where n is a dimensionality component such that $n = dim(\vec{v})$
5. Let $\mathcal{L}(\vec{v})$ be a *low-pass* filter
6. Let $\vec{v}_{ema}$ equal an exponential moving average component computed by [8]

$$\vec{v}_{ema} = \frac{\mathcal{L}\left(w \cdot \vec{v}_m + (1-w) \cdot (e \cdot \vec{v}_{sim}) + w[\dim(\vec{v}_{sim})]\right)}{\begin{aligned} &sqrt((\mathcal{L}(w \cdot \vec{v}_m))^2 + (\mathcal{L}((1-w) \cdot e \cdot \vec{v}_{sim}))^2 + (\mathcal{L}(w[\dim(\vec{v}_{sim})]))^2 + \\ &2 \cdot \mathcal{L}(w \cdot \vec{v}_m) \cdot \mathcal{L}((1-w) \cdot e \cdot \vec{v}_{sim}) + 2 \cdot \mathcal{L}(w \cdot \vec{v}_m) \cdot \mathcal{L}(w[\dim(\vec{v}_{sim})]) + \\ &2 \cdot \mathcal{L}((1-w) \cdot e \cdot \vec{v}_{sim}) \cdot \mathcal{L}(w[\dim(\vec{v}_{sim})])) \end{aligned}}$$

The vector $\vec{v}_{ema}$ computed above is one that is suitable for further operations, and is equipped the following qualities:

- It is a composition of the old main vector and a recent vector from the database in an exponential manner, reacting to underlying trends
- Susceptibility to short-term spikes has been removed through filtering
- Overfocusing is being combated by adding white noise
- It is a unit vector

The above characteristics are vital for aligning the vector with the direction of highest reward as well as convenient for further processing. At this point the vector could almost be used as a reliable metric; however, it is still too susceptible to short term fluctuations. Therefore, one more step is needed in order for it to function as intended.

[8] This is a smoothed exponential average normalized s. t. $\|\vec{v}_{ema}\| = 1$

Calculating similarities to the $\frac{d_{eval}}{dt}$ function with respect to the individual vector component functions In order for this step to be understood, it is important to reiterate what the vector is essentially constructed to optimize. The fact is that rather than maximizing a function itself, it is often a much more practical idea to focus on its derivative. In other words, the vector is constructed to optimize the positive change in evaluation over time - $\frac{d_{eval}}{dt}$. Having the development in terms of t of the above vector as a prerequisite, a prior assumption needs to be considered as well: *There are certain categories that are more liked than others and hence receive more positive feedback.* This assumption prompts to decide on how many categories should be labeled as most relevant. An arbitrary yet most probably sufficient choice would be 3.
To ascertain which categories belong to that subset, the following steps are conducted in each iteration of the algorithm:

1. The development over iterations of the main vector is extracted as a function of time from epoch to the current point in time
2. The function is decomposed into its components, creating 25 individual functions of time of each component, respectively
3. The function of evaluation over iterations is extracted as a function of time
4. The component functions as well as evaluation functions are differentiated
5. Every possible linear combination of three vector component function is computed: which leaves $\binom{25}{3}$ or 2300 combinations [9]
6. The linear combinations are compared to the differentiated evaluation function iteratively to discern one with the highest similarity

Note that, again, these functions may be interpreted as vectors and compared using a vector similarity metric. Other metrics mentioned above work as well. The semantics of why this step would yield the correct functions may not be evident at first. However, consider that evaluation goes up when the relative weight of either of 3 given components goes up, whereas it stays low/unaffected (or noisy, respectively) when either of the remaining components' weights rise of fall, slowly eliminating uncertainty with each iteration, inevitably leaving

[9] Note that this is a large number in terms of computing anything; however, there is no need for speed, as even moderate computation time serves this purpose sufficiently

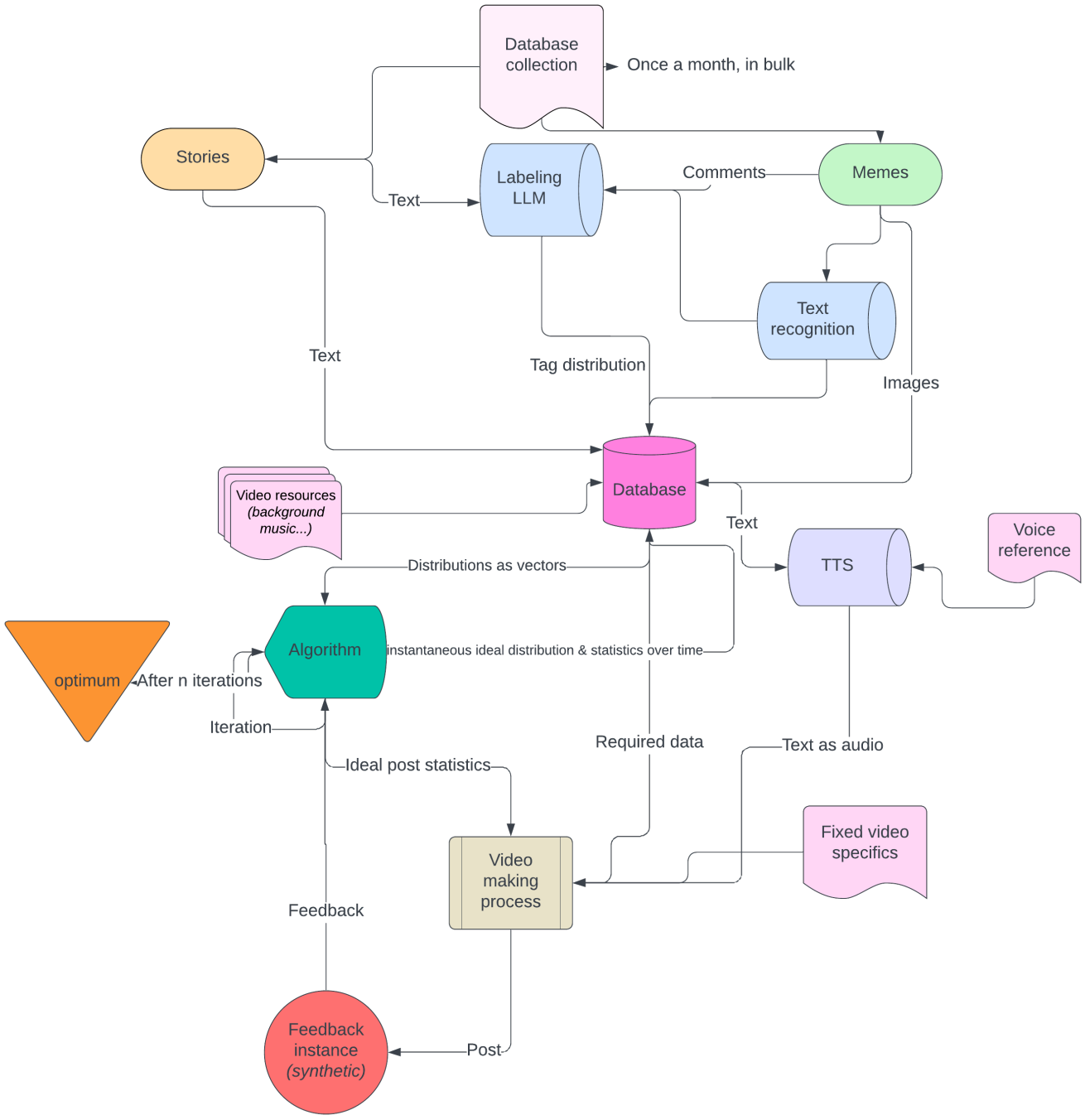
one linear combination of components superior to the remainder in similarity of behavior to the evaluation function. This works even if the 3 components are scaled unequally in the desired distribution.

Supporting and scaling the vector accordingly Having discerned the instantaneously most probably *3 favorite tags*, it is time to incorporate this insight into the previously prepared main vector. This is done by scaling their weights in the distribution by some factor q, the specific value of which has to be determined during experimentation. Afterwards, the vector is normalized again in order to remain a unit vector, and saved as a new element in its development over iterations.

This concludes theoretical rough outline of the main algorithm.

3.2 Notes regarding implementation

Taking a step back from rigorous theory is necessary when constructing a project above a provided complexity threshold in order not to lose track in the chaos of ideas, plans, and mutually interacting variables. As such, dedicating effort to a concrete implementation plan is advisory. The following is a concrete visualization of a concretized outline/structure of the project in a flowchart. It should shed light on the question how everything could work together.


Database collection
Once a month, in bulk
Stories
Text
Labeling LLM
Comments
Memes
Text recognition
Text
Tag distribution
Images
Database
Video resources (background music...)
Text
Voice reference
TTS
Distributions as vectors
Algorithm
instantaneous ideal distribution & statistics over time
optimum
After n iterations
Iteration
Required data
Text as audio
Ideal post statistics
Fixed video specifics
Video making process
Feedback
Feedback instance (synthetic)
Post

3.2.1 Notes on intermodular interactions

From the graphic above, many connections and interactions can be discerned and it becomes apparent why so many prerequisites have to be met in order to be able to replicate this project. Consider that every interaction requires their own field of expertise, which extends beyond simply mathematics of the algorithm. Furthermore, a mistake in any instance of the pipeline could prove detrimental to the overall result. It is therefore necessary to define all assumptions used upfront, such as which sources to access for data collection, which video resources to utilize, which video characteristics to implement.

3.2.2 Notes on the main algorithm

Based on the general idea described in 3.1.4, numerous factors have to be decided on experimentally, as from a theoretical approach only so much can be decided. Namely, those factors include but are not limited to the following:

1. Weight distribution in exponential moving average
2. White noise amplitude and weight
3. Factor in supporting tags extracted by similarity to the evaluation function
4. Which metrics to save
5. Characteristics of the initial $\vec{v}_{m_0}$
6. Determining the amount of data required as well as computation time [10]

3.2.3 Possible problems and their respective solutions

Before starting experimentation and method testing, it is in every case recommendatory to regard possible points of failure and construct bases of solutions in order not to be held back if something goes wrong during testing, which is more often than not the case. In this scenario the following points could pose room for failure:

[10] As stated before, computation time of one iteration is not confined by seconds or minutes; however, it should not exceed 6h

1. Overfocusing of the vector due to too little variation in its movement, possibly leading to focusing on only one or two of the *sought after tags*
2. Failure to converge to a stable result due to too much variation in the vector's movement
3. Failure to reach a satisfactory result due to requiring too much data [11]
4. Failure to reach a satisfactory result due to too high time complexity

The respective solutions for each of the above hypotheticals have already been addressed to varying degrees and should not come as a surprise.

1. For the first two hypotheticals described, the solution is the employing of additional support through correlation enforcement to help convergence, and the implementation of a misc tag as well as white noise *(widely referred to as jitter)* to prevent overfocusing, which is additionally counteracted by the fact that the closest vectors from the database never match the *"ideal main vector"* perfectly. The key is moderation and balance of these aspects.
2. When dealing with optimization, it is never clear how much data is necessary for a method to work in advance, and the solutions vary from case to case. In this scenario, such a problem might be mended by adjusting the weight factors in the *exponential moving average* computation, or adjusting endorsement of the selected tags *(favorites)* in later stages of the algorithm.
3. Provided the computation turns out to take exponentially longer than initially expected, a multitude of approaches is available to employ:
   (a) Multiprocessing
   (b) Multithreading
   (c) Optimizing for complexity
   (d) Approximating
   (e) Switching to other languages away from python

[11] During synthetic testing, this poses no problem; however, a trait like this would render the method inapplicable in the real world. It is therefore to be avoided

4 Actual Application & Experimentation

4.1 Actual Application

This section discusses the concrete implementation of the aforementioned ideas in a concrete manner, showing and explaining the relevant code snippets as well as mathematical ideas (if present) corresponding to each step. For this purpose, it is structured as follows:

1. Data Retrieval & Text Extraction
2. Data Labeling
3. Text To Speech
4. Database Structuring
5. Assembling The Posts
6. Algorithmic Optimization

Note that real-world posting as well as feedback retrieval are not mentioned or discussed, seeing as no real-world testing has been conducted. Their implementation, however, is trivial and may be set up by any beginner programmer at their own discretion for any suitable practical use case scenario.

4.1.1 Data Retrieval & Text Extraction

The most difficult part in any data science venture is the initial and dynamic retrieval of data to operate on. Therefore, pickiness is discouraged, and the largest source of data is to be considered the best unless proven to be low quality. In this case, the most appropriate source of data would appear to be Reddit, having taken a prime spot for meme/story creation in the past decade, yielding innumerable numbers of original content each day for any individual to consume as well as any aspiring mathematically inspired content creator to copy. Note that it is advisable to not limit the retrieval to only one subreddit. [12] The retrieved data may be semantically divided into the following two categories:

[12] A subreddit is a widely used term for a community on Reddit comparable to a channel on YouTube

1. Title of a given post as well as the top 10 comments (provided 10 are present, otherwise all that are)
2. The post itself

The nature of the posts themselves is confined to:

1. memes from: r/memes, r/meme, r/meirl
2. stories from: r/writingprompts, r/aita

The retrieval works based on a Reddit API wrapper known as PRAW (Python Reddit API Wrapper). The retrieval would ideally happen in bulk (i.e. once a month from the top posts of the previous month, where the initial retrieval is manually run on the top of all time to stock up on material, as a considerable amount of data is needed to operate the algorithm).
The relevant code snippets for this step in the process are the following:
This function retrieves the URL of a Reddit post using *PRAW* [13]

```
def get_url(self):
    '''gets the url of the current post'''
    cred = Params.cred(self)
    idd = cred[0]
    secret = cred[1]
    uname = cred[2]
    password = cred[3]
    reddit = pw.Reddit(user_agent='agent',
                       client_id=idd,
                       client_secret=secret,
                       username=uname,
                       password=password)
    return(str(reddit.config.reddit_url) +
           str(reddit.submission(self).permalink))
```

This snippet retrieves the top posts and their comments from a specified subreddit using *PRAW*. It processes each post to collect its title, the top 10 comments, and skips any content marked as [gif]. [14]

[13] Note that the credentials have to be passed in individually
[14] This is done simply for the inconvenient nature of extracting text from gifs

```
post = reddit.subreddit(subreddit).top(time, limit=self)
for sub in post:
    body = ''''''
    body = body + 'title: ' + str(sub.title) + '\n' + '\n' +
    ↪  'comments: ' + '\n'
    url = Params.get_url(sub)
    for comment in reddit.submission(sub).comments[:10]:
        body = body + comment.body + '\n'
    if '[gif]' in body:
        body = body.replace('[gif]', 'skip')
    print(body)
```

This function downloads and saves an image from a Reddit post using *Urllib*. It retrieves the image from the post's preview and stores it in the specified location on the local filesystem using the function below.

```
def dump_img(self, loc: str):
    '''saves the image from a reddit link'''
    loc = os.path.join('../src', 'images', loc)
    url = self.preview['images'][0]['source']['url']
    urllib.request.urlretrieve(url, loc)
    return(loc)
    print('//image dumped')
```

This function extracts text from an image using the *pytesseract OCR* library and then uses GPT-3.5-turbo to fix any potential typos or spelling errors.

```
def clean(self, src):
    data = Image.open(src)
    user = pyt.image_to_string(data)
    answer = LLM.llm_call(self,
    ↪  user).removeprefix("ChatCompletionMessage(content='").removesuffix("',
    ↪  role='assistant', function_call=None, tool_calls=None)")
    return(answer.replace('\\n', ' '))
```

Note that in the code snippets above, it is shown how memes are treated. For stories, one would simply not save the image, but save the text contained within the post instead. Furthermore, a prompt is provided to the LLM to ensure the desired result. Here, the following prompt is used; however, note that any similar prompt would perform just as well.

Text Extraction Prompt Example

You are a **text extraction machine**. Your task is to extract the relevant information suited for reading from a given input. Below are examples and the rules you need to follow:

- **Input:** "F v @turtlekiosk self care is officially over we are doing drugs again 8:39 PM - 5/17/19 - Twitter for Android 35.8K Retweets 123K Likes"
- **Output:** "self care is officially over we are doing drugs again"
- **Input:** "9 Christina Warren & (@fim_git) — have discovered the nerdiest tool for taking screenshots of tweets in a seamless (and pretty!) way. You need Node installed but it is CLI and very good in almost all cases (a few edge case issues I've found but it is OSS so maybe I can fix) ukehorvatiscresnshot-twest Seraanshot a Tt e, Contouteto scrsenshot-twest development by creating an account on GitHub. o - `github.com` 414 pm - 6 Dec 2017 - Tweets ks"
- **Output:** "I have discovered the nerdiest tool for taking screenshots of tweets in a seamless (and pretty!) way. You need Node installed but it is CLI and very good in almost all cases (a few edge case issues I've found but it is OSS so maybe I can fix)"

Instructions:

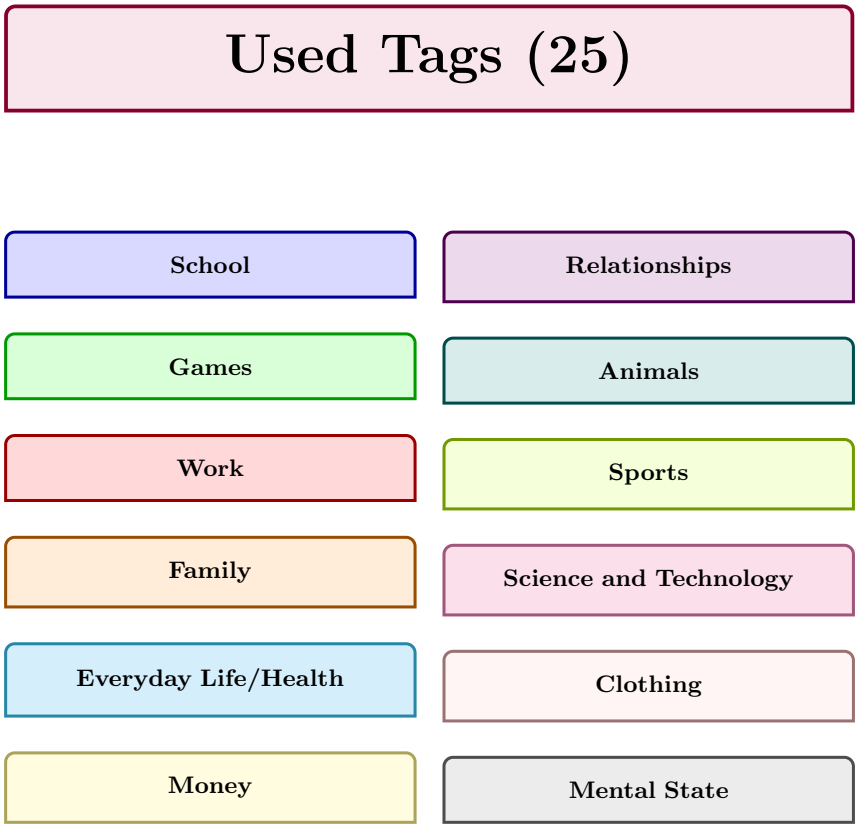
1. Keep all text that appears coherent, even within brackets.
2. Fix spelling mistakes and remove symbols from words if there is no reason for them to be there
3. Remove all brackets and nonsensical strings from the text.
4. Remove any @username at the start or end of the message unless they are relevant.
5. If the input doesn't fit the instructed format or is empty, return `False`.
6. Remove the author's name at the end of the text if present.

4.1.2 Data Labeling

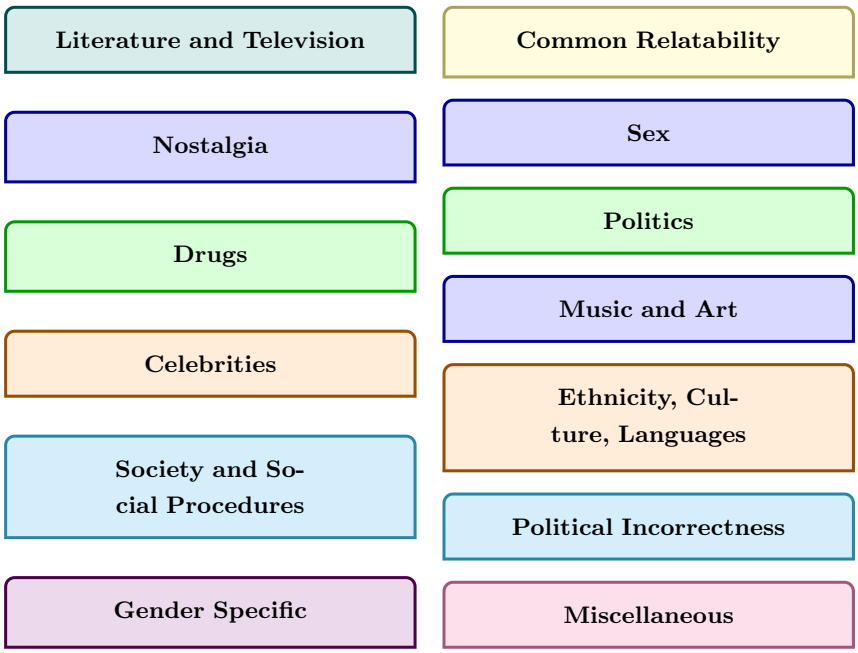
After the data is retrieved as discussed above, it needs to be properly labeled such to make further optimization work possible. Fortunately, due to the sudden rise of LLMs as well as their inferrable extensive linguistic capabilities, it is conveniently possible to do just that with but a few lines of code, the prerequisites for which as well as themselves are the following:

Prerequisites:

1. Have access to an LLM (local or remote) - here, OpenAI API is used [15]
2. Define solid classification tags - here, the following have been chosen :



[15] Note that minor costs may arise if a remote LLM is chosen (like here with the OpenAI API); however, a locally run LLM such as LLama may be used to circumvent this potential issue



16

3. A solid strategy on how to reliably label posts - here, the comments, title, and text potentially contained within the meme are used as a base for categorizing a post [17]

4. A solid starting prompt for the LLM [18] - here, the following prompt is attached to the LLM request:

[16] These tags have been determined by inspecting scarring numbers of short-form videos and have shown themselves to be the most prominent by my discretion

[17] Note that for stories, the entire story itself serves as sufficient input for the LLM

[18] Note that an assistant instead of a one time prompt works just as well, whereas fine-tuning the model doesn't seem to make a significant difference in this particular scenario

You are a meme-classifying machine. You are only to use the tags mentioned below.

You will receive content in the following format:

```
"title: text
comments: text
content: text"
```

Your task is to process the **comments** and output an array of **5 tags**, each with a percentage representing its prominence.

Example output:

```
[('tag_1', '31%'), ('tag_2', '14%'), ('tag_3', '30%'),
 ('tag_4', '21%'), ('tag_5', '4%')]
```

The percentages must always add up to **100%**, and no tag can appear more than once. Use only the tags listed below: {`tags`}

If fewer than 5 tags are applicable, assign `0%` to the remaining slots:

```
[('music_art', '50%'), ('nostalgia', '30%'),
('common_relatability', '20%'),
('society_and_social_procedures', '0%'),
('celebrities', '0%')]
```

If a tag is required but not provided, use `'misc'`. Strictly adhere to the format and use only the specified tags.

Having met all the above conditions, it is possible to successfully classify the previously scraped data using a few lines of code, the most essential of which are the following ones:

```
def llm_call(self, args: str):
    cred = Params.cred(self)
    oa = cred[4]
    client = OpenAI(api_key=oa)
    completion = client.chat.completions.create(
        model="gpt-3.5-turbo-0125",
        messages=[
            {"role": "system", "content": self},
            {"role": "user", "content": args}
        ]
    )
    output = str(completion.choices[0].message).removeprefix(
    'ChatCompletionMessage(content="').removesuffix('",
    ↪ role=\'assistant\', function_call=None,
    ↪ tool_calls=None)')
    return output
```

This function is designed to interact with OpenAI's GPT-3.5-turbo model. The function sends the aforementioned prompt along with the data in question. After receiving the model's response, it processes the output to clean up any extra formatting, ensuring the final result is further usable, as llm-responses tend to adhere to inconvenient formatting, which is not suitable for programming.

```
def pseudo_list(self):
    '''converts llm outputs to python arrays'''
    try:
        start = self.replace(' ', '').replace('),', ')
        ↪ ').removeprefix('[').removesuffix(']').split(' ')
        taglist = []
        for element in start:
            taglist.append(

            ↪ [element.split(',')[0].removeprefix("('").removesuffix("'"),
            round(float(

            ↪ element.split(',')[1].removeprefix("'").removesuffix("%')"))
            * 0.01, 2)])

```

```
13          return(taglist)
14      except:
15          return False
```

The `pseudo_list` function is used to further convert the outputs into a Python array format. It first cleans the raw output by removing unnecessary spaces and characters, then splits the string into individual elements. Each element is processed to extract the tag and its percentage, which is then converted into a decimal format and rounded to two decimal places. Technically, this can lead to the array's percentage not adding up to 1 exactly due to rounding; however, it is vital to understand that it is, factually, secondary whether or not the percentages add up to 1 since the array is later normalized regardless. Therefore, the prior is a non-issue. [19] The function returns a list of these pairs as tuples in form:

$$[(tag_0, percentage_0), (tag_1, percentage_1), ..., (tag_4, percentage_4)]$$

If an error occurs, it returns `False`, indicating that the conversion failed. [20] This function yields a usable format for subsequent treatment.

```
1  def checktags(self):
2      '''checks the output tags against predetermined array to rule
       ↪  out hallucinations'''
3      legal_tags: list[str] = ['school', 'games', 'work', 'family',
       ↪  'everyday_life/health', 'money', 'relationships',
       ↪  'animals', 'sports', 'science_technology', 'clothing',
       ↪  'mental_state', 'literature_television', 'nostalgia',
       ↪  'drugs', 'celebrities', 'society_and_social_procedures',
       ↪  'gender_specific', 'common_relatability', 'politics',
       ↪  'sex', 'music_art', 'ethnicity_culture_languages',
       ↪  'political_incorrectness', 'misc']
4      cleantag = []
5      try:
6          for tuple in self:
```

[19] Note that the fewer responses are discarded, the more economically efficient this process becomes

[20] Note that in this case, it would be good practice to raise an error and treat it accordingly in further steps, but seeing as there is only one exception possible in this code (which is format inadequacy), the differentiation is not needed

```
            cleantag.append(tuple[0])
        if set(cleantag).issubset(set(legal_tags)):
            return True
        else:
            diffs = []
            for tag in cleantag:
                if tag not in legal_tags:
                    diffs.append(tag)
            print(diffs)
            return False
    except:
        raise TagCheckError()
```

The above function checks whether the tags present in the LLM's response are contained within the set of *legal tags* mentioned in 2. It raises an error if they do not fulfill this condition. The labeled data is subsequently saved to the database, the structure of which is to be inspected below. This concludes labeling.

4.1.3 Text To Speech

As discussed, the posts need to contain an audible counterpart to the text contained within each post to stimulate longer attention as well as to conform to the standard form of such short-form posts. The Text To Speech (TTS) is done by feeding the cleaned text from a given meme to a pre-trained TTS model which is fine-tuned on voice samples from a selected source. Note here that it is important for the audible text to be conveyed in an attractive manner, meaning a suitable voice source has to be chosen and subsequently adapted to the model.

- **The TTS-model:** Due to its high quality and open source, the *Coqui* `XTTS-v2` model has been selected to act as a base
- **Voice Source:** Seeing as a large amount of clean studio-quality reference data is needed, the obvious pick would appear to be **video game characters' voice lines**. They meet all the above conditions while also being easy to retrieve and sample, granting a vast degree of modularity and accessibility.

Note that this step may take up some time.
The essential snippets of code for the above purpose are the following ones:

```python
def split_wav(input_file, output_dir, chunksize: int):
    # Create output directory if it doesn't exist
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    with wave.open(input_file, 'rb') as wav_file:
        frame_rate = wav_file.getframerate()
        frame_count = wav_file.getnframes()
        duration = frame_count / float(frame_rate)

        # Calculate number of 2-second chunks
        num_chunks = int(duration // chunksize)

        # Read and write 2-second chunks
        for i in range(num_chunks):
            output_file = os.path.join(output_dir, f"chunk_{i +
            ↪  1}.wav")
            start_frame = int(i * frame_rate * chunksize)
            end_frame = int((i + 1) * frame_rate * chunksize)
            wav_file.setpos(start_frame)
            frames = wav_file.readframes(end_frame - start_frame)

            with wave.open(output_file, 'wb') as out_wav_file:
                out_wav_file.setparams(wav_file.getparams())
                out_wav_file.writeframes(frames)
```

The above function is used to split a long WAV audio file into smaller chunks of a specified duration, which is vital for the TTS-model to function properly. This is done because the retrieved voice lines tend to be included in one coherent file. The function opens the input WAV file and calculates its frame rate, frame count, and total duration. Based on the specified chunk size (in seconds), it determines the number of chunks to generate. For each chunk, the function calculates the start and end frames, reads the frames for said segment, and writes them to a new WAV file in the specified output directory, creating an accessible mounting point for the subsequent TTS application. Note that the *2 second length* has been picked experimentally to work best in this case.

```
def convert(text: str, save_path: str, voice_model: str =
↪  "tts_models/multilingual/multi-dataset/xtts_v2",
            character: str = '', gpu_enable=False, exist: bool =
            ↪  True) -> None:
    if exist == True:
        print(voice_model, character, save_path)
        tts_engine = TTS(voice_model, gpu=gpu_enable)
        save_path1 = list(save_path.removesuffix('.wav'))[::-1]
        for index, element in enumerate(save_path1):
            if element != '/':
                save_path1[index] = ''
            else:
                break

        save_path1 = ''.join(save_path1[::-1])

        if os.path.exists(save_path1):
            try:
                tts_engine.tts_to_file(
                    text=text,
                    file_path=save_path,
                    speaker_wav=[os.path.join(os.getcwd(),
                    '../assets', 'models', character, entry)
                                 for entry in os.listdir(
                                 os.path.join(os.getcwd(),
                                 '../assets','models',
                                 ↪  character))],
                    language="en",
                    split_sentences=True
                )
            except ValueError or AttributeError as v:
                print(v, ' using predetermined model language as
                ↪  well as speaker instead')
                tts_engine.tts_to_file(
                    text=text,
                    file_path=save_path,
```

```
                    split_sentences=True
                )
        else:
            raise PathError(save_path1)

        print(f"Speech saved as {save_path}")
    else:
        with wave.open(save_path, "w") as wav_file:
            wav_file.setparams((1, 2, 44100, 0, 'NONE', 'not
            ↪ compressed'))
            for _ in range(9 * wav_file.getframerate()):
                wav_file.writeframes(struct.pack('<h', 0))
        print(f'empty speech saved as {save_path}')
```

The `convert` function facilitates the conversion of text into speech, leveraging a specified text-to-speech (TTS) model. It begins by checking whether the output file already exists and initializes the TTS engine. Depending on the presence of a predefined character (*one whose voice lines have been split into reasonable chunks aboe*) and model, it attempts to save the synthesized speech to the specified path. If the path does not exist, an error is raised. The function also includes an alternative mode to generate an empty audio file. This versatile implementation ensures that speech synthesis adheres to the given constraints, while also providing error handling for model or file-related issues. [21] This concludes TTS implementation.

4.1.4 Database Structuring

To make any program worthy of the name functional, a sensible database structure is needed. It goes without saying that different individuals tend to prefer different methods; however, for the sake of completeness, this section expands on how saving and accessing is processed in this particular example, serving as a facility for understanding the underlying mechanics if nothing else.

[21] Note that the voice resource selected is up to individual discretion. No suggestions are provided above or below due to legal issues.

```python
def fetch_name(path: str):
    '''Gets the current chronological name of an image.'''
    with open(path, 'rb') as f:
        number_array: list[int] = pkl.load(f)
    current_suffix: int = number_array[-1]
    number_array.append(current_suffix + 1)
    os.remove(path)
    with open(path, 'wb') as f:
        pkl.dump(number_array, f)
    name = f'image{current_suffix}.png'
    print('//name fetched')
    return name
```

The above function generates a new sequential name for an image using a pickled file. It reads the list of numbers, retrieves the latest suffix, increments it, updates the file, and returns the new image name. This ensures consistent chronological naming for images. Note that a list of all the integers is preserved instead of just saving an integer. This is done in case it is, in a later step, decided that an older file is not needed anymore, it can be deleted and there is evidence of its existence and potential references to other files can be traced back. Of course, just saving an integer or creating UUIDs would work just as well; however, this proved to be the most code efficient as well as suitable for human orientation.

```python
def dump_text(text: str, file_path: str):
    '''Saves the contained text from the respective image.'''
    with open(file_path, 'rb') as file:
        text_list = pkl.load(file)
    print('opened')
    text_list.append(text.replace('\n', ' '))
    os.remove(file_path)
    with open(file_path, 'wb') as file:
        pkl.dump(text_list, file)
        print('//text dumped')
```

The above function appends processed text (without line breaks) to a pickled list stored at a fixed file path. The function ensures that the updated list is re-saved, maintaining the state of stored text data across multiple calls.

```python
def dump_voice(text: str, file_path: str):
    '''Converts text to speech and saves it as an audio file.'''
    is_existing = text != 'False'
    model = 'tts_models/multilingual/multi-dataset/xtts_v2'
    convert(
        text=text,
        save_path=file_path,
        voice_model=model,
        character='dude1',
        exist=is_existing
    )
    print('//tts saved')
```

The `dump_voice` function converts a given text to speech *(using the TTS convert function)* and saves it as an audio file at the specified location, which is marked using the same integer as the corresponding image *(using the order integer suffix)* when the call is made. This provides a reference point. The corresponding image is saved in a similar way.

Indexing: The indexing of corresponding file batches is handled in the following way: Every batch is saved in the following format:

Index	0	1	2	3	4
Element	path_image	path_tts	tagarray	full_text	url

A separate pickled array with n of these sub-arrays is present. Once individual posts are combined into a video, a second indexing file is created with sub-arrays of the following form.

Index	0	1
Element	path/UUID.mp4	combined_tags

Where the tag combination is given by:

```python
def combine_tags(self: np.ndarray) -> np.ndarray: # tags are in format: ('tag', percentage as float)
    sum = np.sum([float(tag[1]) for tag in self])
    return(np.array([(tag[0], round(float(tag[1])/sum, 3)) for tag in self]))
```

The array consists of a normalized sum of all the individual posts' tags.
This indexing method allows for quick error catching as well as a solid structure of all the resources and posts present, facilitating access as well as modifications if needed as well as restoring potentially corrupt data. Note that this is an advantage if compared to say a single file `dict`-indexing.
This concludes the general process behind the database structuring.

4.1.5 Assembling The Posts

Having understood how the needed data is retrieved, treated, labeled, and subsequently saved, it is the time to inspect how the individual posts get turned into a watchable video for the users to behold. This step, too, requires, however, a few additional prerequisites. These being:

- A selection of popular background video resources to overlay the memes or stories on top of
- A selection of popular background songs to support attention and conform to common practices
- Finally, the stored pre-prepared data as well as the respective labeling

The `get_sorted_vid_and_audio` function processes a list of addresses to retrieve and organize video and audio file paths. It is designed for arrays in the format `[path_image, path_tts, [tagarray], full_text, url]` (which is the same format specified in 4.1.4).
It does this in the following way:

1. Load pre-indexed address data from the indexing file located at `'../src/indexing.pkl'`.
2. Filter the loaded data to include only the entry where the image paths match the provided `addresslist`.
3. Extract separate lists for image paths (`imgarr`) and audio paths (`audioarr`).
4. Return the organized lists as `[imgarr, audioarr]` for further processing.

```
1 def get_sorted_vid_and_audio(addresslist: list[str]) ->
  ↪ list[str]:
2     # Array in format: [path_image, path_tts, [tagarray],
      ↪ full_text, url]
3     with open('../src/indexing.pkl', 'rb') as f:
```

```
        addressars = pkl.load(f)
        addressars = [(address[0], address[1]) for address in
        ↪  addressars if address[0] in addresslist]

    # Extract separate arrays for images and audio
    imgarr = [address[0] for address in addressars]
    audioarr = [address[1] for address in addressars]

    # Return organized lists
    return [imgarr, audioarr]
```

A video may be assembled after this step using the following function.

- `make_video` generates a video by combining a series of images with background audio. It begins by loading the background video and extracting its size for proper image scaling, after which it retrieves a sorted list of image paths and corresponding audio clips using the function described above. A random background song is selected from the assets directory and is set to the same duration as the video (which makes sense, as one would want the visuals to match the length of what is audible.)

- The function processes each image and audio pair: for each pair, it creates an audio clip with a set start time, while each image is turned into a video clip with a matching duration. These clips are all added to a composite list. Finally, the audio tracks are combined, and the background video is set to play along with the audio, adding a small padding of 0.5 seconds between the vanishing of a meme from the screen and the appearance of the next one. [22]

- This function uses the `moviepy` library to handle video and audio manipulation, including resizing and positioning the images, and setting the audio duration. The `ix.Index.taglist` and `ix.FullIndex.combine_tags` methods are used to process the images before they are combined into the video. Note that `ix` comes from *import Index as ix*

[22] Note that for stories, this process becomes vastly easier, as there are no posts to resize or shuffle around.

```
# imglist in indexing form - universal path
def make_video(imglist:list[str], name:str,
↪ bkgrd=os.path.join('../assets', 'vids', 'bkgrd.mp4'),
↪ audio=os.path.join('../assets/music',
↪ random.choice(os.listdir(os.path.join('../assets',
↪ 'music'))))) -> None:
    background = bkgrd
    video = mpe.VideoFileClip(background)
    video_size = video.subclip(0, 1).size
    width = video_size[0]
    iml = imglist
    tlist = ix.Index.taglist(iml)
    tlist = ix.FullIndex.combine_tags(tlist)
    ix.FullIndex.index_video(name, tlist)
    lists = get_sorted_vid_and_audio(iml)
    iml = lists[0]
    audl = lists[1]

    random_audio = audio
    audio_background =
    ↪ mpe.AudioFileClip(random_audio).set_duration(video.duration)

    audcomplist = [audio_background]
    composite_list = [video]
    overallduration = 0
    for index, tts in enumerate(audl):
        audio = mpe.AudioFileClip(tts)
        duration = audio.duration
        audcomplist.append(audio.set_start(overallduration))
        image = iml[index] # only possible because the images are
        ↪ sorted the same way as the audio files
        vclip =
        ↪ mpe.ImageClip(image).set_start(overallduration).set_duration(
        duration).set_pos(("center",
        ↪ "center")).resize(width=width)
        composite_list.append(vclip)
```

```
29
30          overallduration += duration + 0.5
31      audio_background = mpe.CompositeAudioClip(audcomplist)
32      video.audio = audio_background
33      final = mpe.CompositeVideoClip(composite_list)
34      final.write_videofile(f"{name}", threads=32)
```

This concludes the assembling of an individual video.

4.1.6 Algorithmic Optimization

Having understood all the previous implementations, it is now the time to dive into the most important/interesting part of the project: the method itself - the main algorithm. This section includes parts of the testing code, too. This is to facilitate understanding as well as code legibility. Additionally, the following files are addressed often, as they play a key part in the database structure (this is their state at initialization):

- **memes.db**: Contains a serialized list of synthetic distributions, each created by the function `create_dist(6)`, repeated n times.
- **hist.db**: An empty serialized list, initialized for future use as a history storage.
- **eval_hist.db**: A serialized list initialized with a single value of 0 to set a starting point for evaluation at the origin.
- **main_vector.db**: Stores a normalized vector of random values (noise) based on the length of the `tags` list, serialized for later use in a main vector initialization. The noise is added to provide a point to start computation off from.

The function used for **memes.db** creation is the following. It simulates a combination of *it* individual memes in a post, combining their tags. Since during labeling, it has been defined that each meme is labeled with exactly a distribution of 5 tags, the same is done here:

```
1  def create_dist(it) -> np.ndarray:
2      arr = np.zeros(len(tags))
3      for _ in range(it):
4          randarr = np.random.randn(5)
```

```
5           arr[random.sample(range(0, len(arr)), 5)] += randarr
6       return normalize(np.abs(arr))
```

Furthermore, once a synthetic post is used, it is immediately deleted from the database to prevent the algorithm recycling the same distribution in a loop. This task is performed by the following function:

```
1   def pdb(ind) -> None:
2       with open(os.path.join('../', 'synth_test', 'memes.db'),
        ↪  'rb') as f:
3           db = pkl.load(f)
4       del db[ind]
5       with open(os.path.join('../', 'synth_test', 'memes.db'),
        ↪  'wb') as f:
6           pkl.dump(db, f))
```

The following is the function that matches the main vector as an argument to a distribution within the **meme.db** database using the highest cosine similarity between the main vector and a counterpart from the database over the entire db's contents.

```
1   def fit(arr) -> int:
2       with open(os.path.join('../', 'synth_test', 'memes.db'),
        ↪  'rb') as f:
3           db = pkl.load(f)
4       yarr = arr
5       simarr = [cos_sim(xarr, yarr) for xarr in db]
6       return db[np.argmax(simarr)], np.argmax(simarr)
```

The following function is the one that updates the main vector every iteration:

```
1   def update_vector(c, favorites):
2       path = os.path.join('../', 'synth_test')
3       with open(os.path.join(path,'main_vector.db'), 'rb') as f:
4           main_vector = pkl.load(f)
5       save_hist(main_vector)
6       dbarr, ind = fit(main_vector)
7       pdb(ind)
8
9       eval = get_likes(dbarr, favorites)
```

```
10      save_eval_hist(eval)
11      new_vector = normalize(smooth(WEIGHT*main_vector +
        ↪  (eval**2)*dbarr*(1-WEIGHT) +
        ↪  create_white_noise(len=(len(main_vector)), sum=AMP)))
12
13      if c > 5:
14          combs, combarr, evalhist = comb_eval()
15          hc = get_highest_cossim(evalhist, combarr)
16          for f in combs[hc]: new_vector[f] *= 5
17          new_vector = normalize(new_vector)
18
19      with open(os.path.join(path,'main_vector.db'), 'wb') as f:
20          pkl.dump(new_vector, f)
21          if c > 5 and set([tags[f] for f in combs[hc]]) ==
            ↪  set(np.array(favorites)): return 1
22          else: return 0
```

Broken down, it does the following:

- Loads the current main vector from `main_vector.db`
- Saves the current main vector to a history using `save_hist`
- Fits the main vector to the database (`fit`) and removes the best-matching element using `pdb`
- Evaluates the match using `get_likes` and saves the value to the evaluation history
- Computes a new vector by combining the main vector, database match multplied by the scalar value (eval) squared, and white noise in a manner of exponential moving average:
  - Weighted by a global parameter (`WEIGHT`).
  - Adjusted with `smooth` and normalized.

- If `c > 5`: [23]
  - Evaluates combinations (`comb_eval`) and identifies the highest similarity match.
  - Amplifies the weights of tags in the best combination by a factor of 5.
  - Normalizes the updated vector.
- Saves the updated vector back to `main_vector.db`.
- Returns `1` if the exact match of `favorites` is found after the given iterations (`c > 5`); otherwise, returns `0`. *This is done for testing purposes*

The smoothing function used in this case is the *gaussian 1d filter* with a low sigma, which generates a normal distribution as a system response for an *Dirac delta* parsed in. It is defined as follows [18]:

$$g_{\sigma,\mu}(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

In code, the implementation would look like so:

```
1 def smooth(arr, sigma=1):
2     return scipy.ndimage.gaussian_filter1d(arr, sigma=sigma)
```

Note that the choice of this specific low-pass filter is left to any individual's discretion and preference. Any respectable low-pass filter would perform just as well.

To further understand the `update_vector` function, the used `comb_eval` function is to be regarded and understood. It is implemented in the following manner:

```
1 def comb_eval():
2     with open(os.path.join('../', 'synth_test', 'hist.db'), 'rb')
      ↪  as f:
3         hist = pkl.load(f)
4         vc = [[] for i in range(len(tags))]
```

[23] Here, c is a counter variable passed to the function per iteration. Only after 5 iterations completed, it starts to support with the closest fits from the database, as up to that point, testing revealed, trying to find a matching array is nonsensical, rendering it a waste of resources and time

```
        for v in hist:
            for i, val in enumerate(v):
                vc[i].append(val)
        vc = np.array(vc)

        combs = list(itertools.combinations(range(25),3))

    with open(os.path.join('../', 'synth_test', 'eval_hist.db'),
    ↪  'rb') as f:
        eval_hist = list(pkl.load(f))
        del eval_hist[0]

    combarr = [vc[a] + vc[b] + vc[c] for a, b, c in combs]
    combarr = [normalize(comb) for comb in combarr]
    combarr = [smooth(np.diff(c)) for c in combarr]
    eval_hist = smooth(np.diff(eval_hist))
    return combs, combarr, eval_hist
```

It performs the following tasks:

- Loads the historical data from `hist.db` and extracts individual components (tags) into an nd-array, `vc`, where each subarray corresponds to one component
- Generates all possible 3-element combinations of 25 elements using `itertools.combinations`
- Loads the evaluation history from `eval_hist.db` and removes the first entry, as, for this step, it is an obstacle, rendering the dimensions of the evaluation function and the vector component functions incompatible. It exists solely for semantic completeness when plotting
- Constructs combined vectors (`combarr`) by linearly combining the vector components in `vc` based on the priorly defined combination triples
- Normalizes each combined vector and applies a smoothing function to the differences of the combined vectors and the evaluation history
- Returns the combinations (`combs`), the processed combined vectors (`combarr`), and the smoothed evaluation history (`eval_hist`) without the 0th element

This concludes the implementation of the main algorithm.

4.2 Synthetic experimentation

Seeing as no real-world experimentation could be performed due to time being a highly constraining factor for this project, it is the more of the essence to perform a certain satisfactory amount of successful synthetic testing of the individual methods, namely and especially the main algorithm. This step is vital for setting expectations regarding potential future real-world testing as well as refining and tuning the tested methods. Additionally, it serves as a sort of *proof of concept*. What is to be noted is that, similarly to everything developed so far, the testing operates based on the initial assumptions regarding TikTok's suggestion algorithm in 2.2, which may therefore reinforce incorrect presumptions, provided that any were made; however, seeing as there are no far-fetched or under-researched aspects, it may be assumed that they are to a vast degree sensible, making them applicable in the real world as a direct consequence.

4.2.1 Simulation structure

In this simulation, TikTok is replicated based on all the assumptions made so far as well as the technical insights achieved thus far. At its core, the test:

- Takes tag distribution vectors as input
- Returns a number of positive interactions based on the relative presence of 3 endorsed tags in the vector - note that those tags, of course, are kept secret from the algorithm
- The algorithm attempts to redirect the vector in such a way as to conform to the simulation's requirements by taking "posts" (effectively, distributions) from a synthetic database of a reasonable size
- The development history of the vector is tracked and decomposed into the individual components, yielding their development as a function of time
- Using the aforementioned similarity metrics, the components are compared to the $\frac{likes}{t}$-function to determine the 3 best fits, endorsing them as a consequence.

Essentially, it is the above algorithm implemented in a synthetic environment. Testing is conducted in batches of different parameters in the algorithm to find the best fitting ones. The essential functions for this task are the following ones:

```
def get_likes(dist, favs):
    zd = zip(tags, dist)
    ev = 0
    for tag, num in zd:
        if tag in list(favs):
            ev += num
    return int(round(ev * LIKE_CAP, 0))
```

The above function, albeit short and simple in its nature, embodies immense importance, and one could go as far as to consider it the most important component of the entire testing stage. As such, it deserves thorough argumentation, as it the function which generates evaluation for every input vector. It operates solely based on the assumption that with *favored* tags present in a post evaluation goes up. The implications made as well as the assumptions taken through the above function are the following ones:

- There exists a positive relationship between relative *favored* tag weight and the evaluation score
- This relationship is for the sake of testing presumed to be linear; however, note that a higher power or exponential relationship would yield similar results on this scale
- Additionally, for the sake of reviewing stability, a global hard[24] `LIKE_CAP`. Note that this trait does not hold in a real-world application. It allows, however, for convenient reviewing in a synthetic environment in order to either heavily support or completely disprove a method.

Let it be noted that this mechanic, while not fully authentic, is reliable for testing purposes.
In its essence the function does the following:

- The function zips the `tags` with the values in `dist` (the input vector)
- Declares a variable `ev` at 0
- Iterates over the zipped pairs (`tag`, `num`) and adds `num` to `ev` if the `tag` is present in the `favs` list, creating a relative sum of all the *favored* tags present in the distribution

[24] a hard cap is a insurmountable maximum value, while a soft cap would considerably reduce acceleration after being reached

- Multiplies the final `ev` value by a constant `LIKE_CAP` and rounds the result
- Returns the rounded result to an integer for the sake of simplicity

The following function runs a simulation of *it* epochs of the synthetic equivalent of one account posting *it* posts.

```
def run(it, favs):
    Prep.prep(Prep)
    increment = []
    first_point = 0
    for i, c in tqdm(enumerate(range(it))):
        addition = update_vector(c, favs)
        if set(increment) == {0} and addition == 1:
            first_point = i
        increment.append(addition)
    save_inc(first_point, increment)
    with open(os.path.join('../', 'synth_test','eval_hist.db'),
    ↪  'rb') as f:
        eval_hist = pkl.load(f)
        eval_hist = smooth(eval_hist, sigma=5)
        return eval_hist
```

It performs the following tasks when executed:

- Initializes by calling `Prep.prep` to prepare the environment, cleaning up after its predecessor
- Creates an empty list `increment` and sets `first_point` to 0
- Iterates over a range defined by `it` and updates the vector at each iteration using `update_vector`
- Checks if `increment` only contains 0 and if `addition` equals 1. If true, stores the current iteration index in `first_point`
- Appends the result of `update_vector` to the `increment` list, where the result is an integer representation of the boolean function of whether the matching mechanic correctly identified all *favored* tags (mentioned in 4.1.6)

- Saves the `increment` list along with `first_point` to a file using `save_inc`, which is a metric for correct tag identification to inspect in experimentation evaluation
- Loads and smooths the evaluation history from `eval_hist.db` using a Gaussian filter (with σ=5, which serves general trend visualization)
- Returns the smoothed evaluation history

Consider this function the synthetic counterpart to a TikTok account posting *it* posts and optimizing them over the whole duration.
To conform to statistically acceptable practices, however, one account duration cannot be considered viable, wherefore a further order of magnitude exists, simulating the duration of n accounts:

```
def stat(n, runit):
    with open(os.path.join('../', 'synth_test', 'favs.db'), 'wb')
    ↪  as f:
        pkl.dump([], f)
    with open(os.path.join('../', 'synth_test', 'results.db'),
    ↪  'wb') as f:
        pkl.dump([], f)
    with open(os.path.join('../', 'synth_test', 'increment.db'),
    ↪  'wb') as f:
        pkl.dump([], f)
    for i in tqdm(range(n)):
        if os.path.exists(os.path.join('../',
        ↪  'synth_test','favs.db')):
            with open(os.path.join('../', 'synth_test',
            ↪  'favs.db'), 'rb') as f:
                favs = pkl.load(f)
        else:
            with open(os.path.join('../', 'synth_test',
            ↪  'favs.db'), 'wb') as f:
                pkl.dump([], f)
                favs = []
        favtags = random.sample(tags, 3)
        favs.append(favtags)
```

```
18          with open(os.path.join('../', 'synth_test', 'favs.db'),
            ↪  'wb') as f:
19              pkl.dump(favs, f)
20          ehist = run(runit, favtags)
21          if not os.path.exists(os.path.join('../', 'synth_test',
            ↪  'results.db')):
22              with open(os.path.join('../', 'synth_test',
                ↪  'results.db'), 'wb') as f:
23                  pkl.dump([ehist], f)
24          else:
25              with open(os.path.join('../', 'synth_test',
                ↪  'results.db'), 'rb') as f:
26                  results = pkl.load(f)
27                  results.append(ehist)
28              with open(os.path.join('../', 'synth_test',
                ↪  'results.db'), 'wb') as f:
29                  pkl.dump(results, f)
```

It performs the following tasks during execution:

- Initializes `favs.db`, `results.db`, and `increment.db` by dumping empty lists into them.
- For each iteration from 0 to `n-1`, it checks if `favs.db` exists [25]:
  - If it exists, loads the list of favorites
  - If it doesn't exist, creates an empty list and assigns it to `favs`
- Randomly selects 3 tags from `tags` and appends them to `favs`, assigning new *favored* tags to each *"account"*
- Saves the updated `favs` list back into `favs.db`
- Calls the `run` function over `runit` iterations and with the selected favorites as arguments
- Appends the evaluation history (`ehist`) to `results.db`. If `results.db` does not exist, it initializes the file with `ehist` as the first entry, leaving

[25] Note that this is done in case technically issues arise while the data crunching process is run unsupervised. This mechanic has the potential to save a sizable chunk of data in case of an unexpected fatal error

behind the development of evaluation over each account to be inspected during evaluation

This concludes the structure of the experimentation environment.

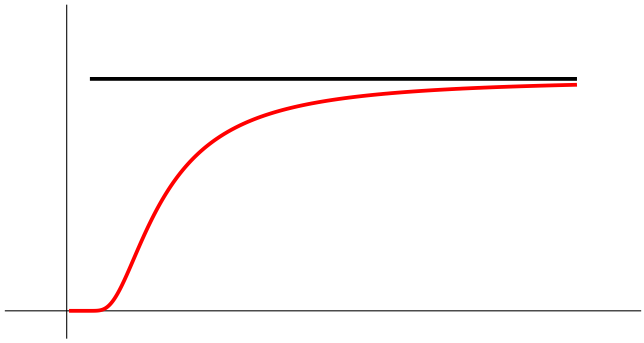
4.3 Expected results

Seeing as the testing stage is not only launched to prove the methods work but also to determine key factors of the process, the expectations weighing on its success are high, as it needs to provide satisfactory parameters for the algorithm's core functions and subsequently demonstrate its effectiveness in realistic circumstances. However, conjecture provides a rough idea of what the testing stage should yield given the extent of theory discussed thus far.

4.3.1 What is to be expected?

Through experience and theory, the following points are expected as results of synthetic experimentation:

1. The evaluation curve is expected to fit a slow rising exponential curve, while converging to a certain *"stability line"* [26] :



2. The weights of the exponential moving average are expected to move somewhere between 0.7 and 0.9 for the old vector and 0.3 to 0.1 for the new vector, respectively.

3. The required white noise during vector addition is expected to be low, as too much could negatively impact performance in finding stability

[26] The term "asymptote" would not be correct here as the resulting function of evaluation is by definition noisy and therefore, it surpasses the asymptote

4. The factor for supporting vector components with high similarity to the evaluation curve is expected to lie between 2 and 10

5. The "stability line" is expected to lie south of 75% of the global `LIKE_CAP`

4.3.2 Why?

In view of the fact that expectations are often tainted by severe guesstimation, it is the more crucial to reason rigorously and question initial conjectures, hunches, and expectations in order to not fall victim to ungrounded assumptions, the implementation of which could prove tedious and a setback. Therefore, the following is the reasoning behind the above expectations:

1. **Expected evaluation curve:** First, it is to be expected that the closer the vector is to incorporating the *favored* tags sufficiently, the faster it is going to be leaning into the trend of focusing on exactly those components due to a rising "reward". Combined with the additional support through the correlation mechanic, one would expect the function to rise very quickly once it catches onto the right track. Hence, the high-order rise. Furthermore, due to the late (after 5 epochs) implementation of the component support, which leaves the vector time to find a "right track", the function would be expected to start rising in the manner described above after those 5-8 epochs. Hence, the delay at the beginning. After the rise is completed, the function is expected to stabilize due to the even distribution of tags in the database as well as the implementation of a `LIKE_CAP`, as the vector still keeps other tags in its distribution while receiving a supply of them with each iteration. This would leave the ideal component weight to remain close to constant at this point (provided the database allows for it statistically [27]). Hence, the asymptote. Furthermore the curve is, of course, expected to be noisy due to statistical variance in the database as well as the jitter added to the vector in an attempt to prevent overfocusing. This trait is not depicted in the graph in 4.3.1, as the visualization serves the comprehension of a general trend.

2. **EMA weights:** The expectations regarding EMA weights are backed purely by experience. Generally, EMA is computed with weights of .9 of the old result and .1 of the new result; however, because this project is

[27] Statistically, if 90% of the database were to be used up, for example, a statistically relevant result could not be expected.

trying to achieve dynamic updating, 0.7 and 0.3 could fall as extrema into a range of what is sensible, as going beyond that would, in the majority of cases, introduce too much variation into the EMA, rendering stability impossible.

3. **Jitter:** Due to the potentially destructive properties of too much jitter, it would be an advocable idea to keep it low to moderate in order to preserve stability, while making use of its anti-overfocusing properties.

4. **Individual component support factor:** Considering that the reason this method is supposed to function effectively is the combination of a weighted EMA vector and the subsequent support by means of correlation of components, it is vital to not overshadow the influence of the weighted EMA vector by raising the influence of the supporting vector components too much. In doing so, the qualities of the EMA vector would be lost, leading to slower evaluation development. For this reason, a significant yet moderate factor is to be employed. The range from 3 to 7 is a solid choice in this scenario. However, for the sake of sufficiency, the range from 3 to 10 is considered.

5. **Location of the "stability line":** Due to the even distribution of tags in the database vectors as well as the ones present in the EMA vector as well as the moderate nature of specific component support, a supremum may be supposed that is by means of estimation somewhere below 75% of the like cap considering all the above influences.

4.3.3 Development Documentation

During development, trial and error is most often the only way to progress. This project is no exception to the rule. During developing and testing, numerous failures were encountered, the teachable properties of which are described in this section. This section is structured in the following way:

1. **Issues and insights regarding storing and indexing**

2. **Issues and insights regarding mathematical concepts**

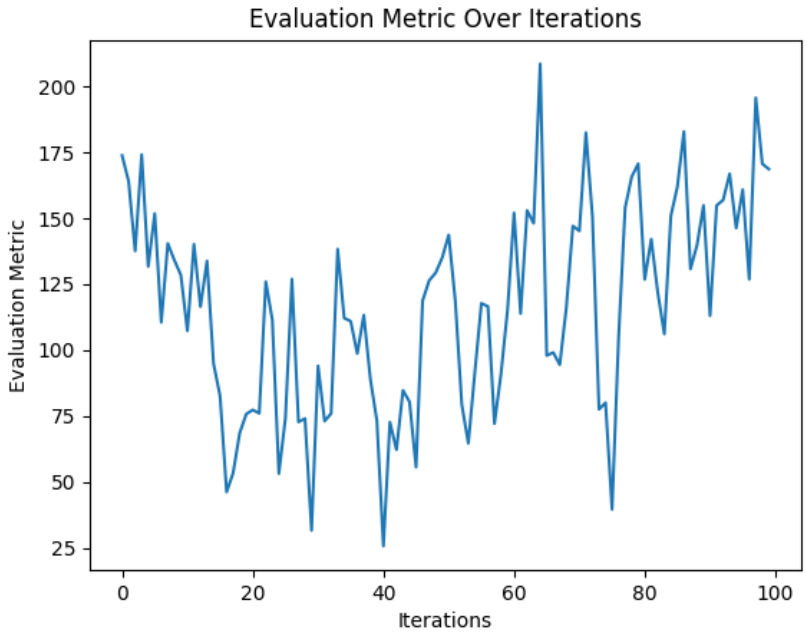
3. **Issues and insights regarding testing**

4. **Issues and insights regarding labeling**

Issues and insights regarding storing and indexing: During early stages of database construction, the idea was to store every aspect in one large object as opposed to several files; however, this proved to be immensely tedious to handle and implemented at least one extra step to every loading step while loading everything (even unneeded data) into ram, which, when dealing with larger amounts of data, is generally considered bad practice, as it can often lead to ram-overloads, crashes, and damaged data. The last point of these risks has proven to be especially detrimental in this case, as north of 1200 minutes of computation time have been irreparably compromised as a result of an unforeseen ram-overload. Seeing as everything was stored in one single object, nothing of value could be recovered.
The solution to this problem is the described database structure, which breaks the data up into several parts, of which only the ones required are loaded and subsequently unloaded when the need for them ceases. This method proved to work flawlessly for later data crunching. The implemented indexing method has been added as a safe-guard in case of partial data loss.

Issues and insights regarding mathematical concepts: During testing and developing of the mathematical methods, namely, the main algorithm, a multitude of setbacks has been encountered and surmounted, the most notable of which had been a an overfocusing issue of the vector, which could seemingly not be resolved by implementation of jitter (as is commonplace in development of AI with regard to the loss function [16]). Upon introducing jitter the development looked along the lines of the following example: Even though a slow upward trend would emerge in certain cases, it could not be considered sufficient, let alone reliable.
The solution to this setback has proved to be the introduction of supporting vector component weights through means of similarity with respect to the evaluation metric curve. Not only did it fix the issue, but as a combination with the exponential moving average vector, it outperformed the success of either one of the methods individually.
Furthermore, determining the most suitable distance function revealed that:

- *correl-factor* is suitable for the task at hand in over 90% of cases
- *cosine similarity* is suitable for the task at hand in almost 100% of cases
- *Pearson correlation* is suitable for the task at hand in over 90% of cases



- *Spearman's rank correlation* is suitable for the task at hand in over 90% of cases
- *combined similarity measure* is generally a nonsensical idea to pursue
- *distance correlation* fails in over 20% of cases, while being expensive to compute
- *dynamic time warping* fails in over 25% of cases, while being expensive to compute
- *mutual information* fails in over 15% of cases

Note that this could have been foreseen using theory; however, attempting it nonetheless offers teachable experience. Additionally, any of these metrics could be tweaked to serve the desired purpose reliable as a direct consequence of its supposed deterministic nature.

Issues and insights regarding testing: Setbacks during testing were closely related to the storage management issues. After an ameliorated database structure had been implemented, they did not occur again. However, a minor setback in later stages of testing arose due to a logical fallacy in the code, rendering a statistical metric unusable. Fortunately, the damage turned out to be moderate, and the metric could be omitted during evaluation, having caused no serious detriment to the overall conclusion.

Issues and insights regarding labeling: Initially, labeling was supposed to be conducted with semantic image recognition with LLMs; however, just 7 months ago, semantic image recognition in widely accessible/affordable LLMs was either entirely unavailable or mostly ineffective. This led to a redefining of the labeling process to use text available in the form of comments, titles, and the contents of a given image in question. During early stages of this concept, the idea was to fine-tune a model available under the OpenAI API; however, this proved time-ineffective as well as needless when compared to using an assistant or a regular prompt, of which using a regular prompt proved to be less tedious to implement and handle than an assistant while moving around the same price point. Combined with defining satisfactory prompts, this maneuver turned out to be unexpectedly time-consuming. The time and effort invested were, nevertheless, not for naught, as the consequently established labeling method resulted in a largely reliable (as well as in a cost-effective) one.

4.3.4 Testing Synopsis & Conclusion

Overall, the testing revealed the desired parameters in a satisfactory manner, while keeping computation time moderate. The methods tested turned out to be effective, proving the concept proposed and affirming the provided thesis that optimizing positive feedback in the case of short-form content based social media, **is, in fact, possible** (provided that the few realistic assumptions made hold true in the real world as well). The results of experimentation follow in the next section.

5 Results & their respective evaluation

In order to support the claims and conclusions made above, it is a necessity to provide statistically sufficient proof. This section includes the results of over 60 simulated accounts over the time span of 100 epochs, respectively. This would translate to approximately 1500 days, 36000 hours, or just north of 4 years in a real-world testing scenario if conducted sequentially [28].

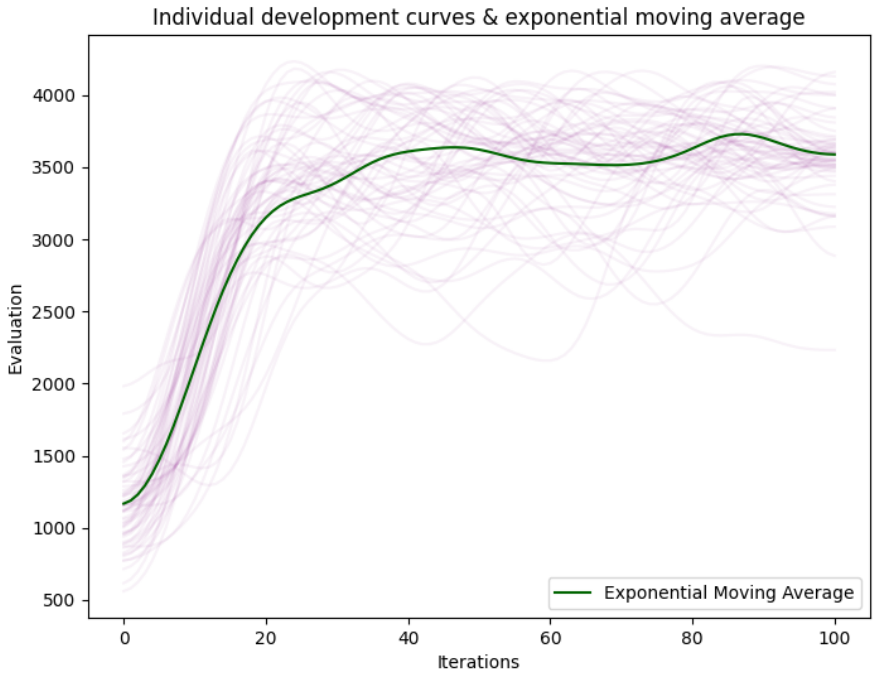
5.1 Overall results

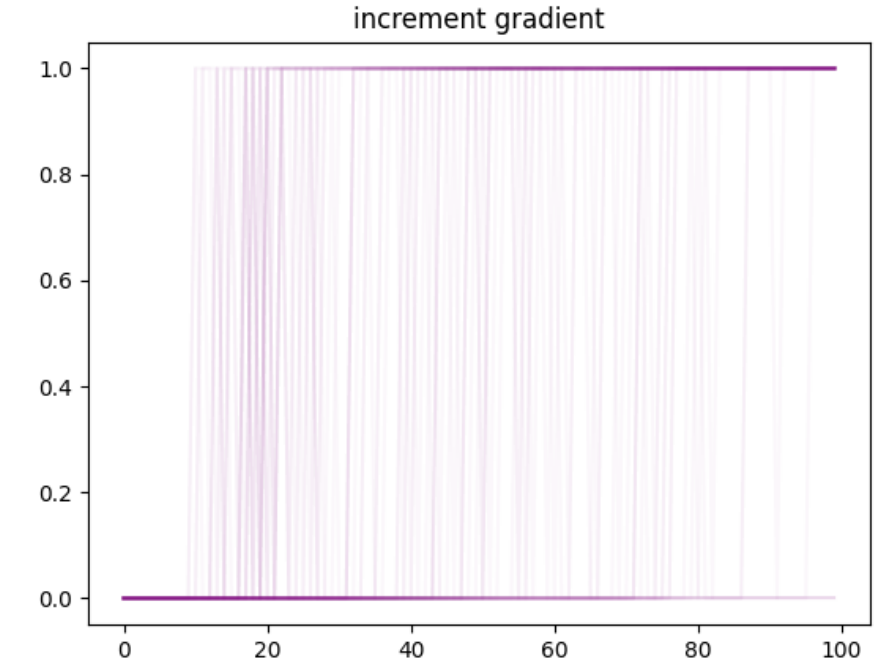
During testing, the following parameters and attributes were identified as the best (or first to perform sufficiently):

- The most reliable correlation metric in the questions of determining the closest distribution from the database with respect to the *main vector* is **cosine similarity** closely followed by the **correl-factor** and *Spearman's correlation coefficient*
- the weights in the question of exponential moving average were determined to be .75 for the current vector and .25 for the vector to be added
- The effect of white noise on the vector proved to be fleeting; however, the ideal value proved to be $\frac{1}{20}$ of the `LIKE_CAP`, which could be translated in a practical scenario to the same factor of a weighted, averaged sum over a recent evaluation interval. In this case, its existence is purely synthetically justified
- A factor of 5 when supporting the relative weight of highest similarity components with respect to evaluation showed itself to work adequately, and therefore, remained at that value

Furthermore, the methods proved to be successful in 100% of cases. For this, see the following plots:

[28] Assuming an account posts 4 times a day


Individual development curves & exponential moving average
Evaluation
Iterations
Exponential Moving Average
4000
3500
3000
2500
2000
1500
1000
500
0
20
40
60
80
100


increment gradient
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

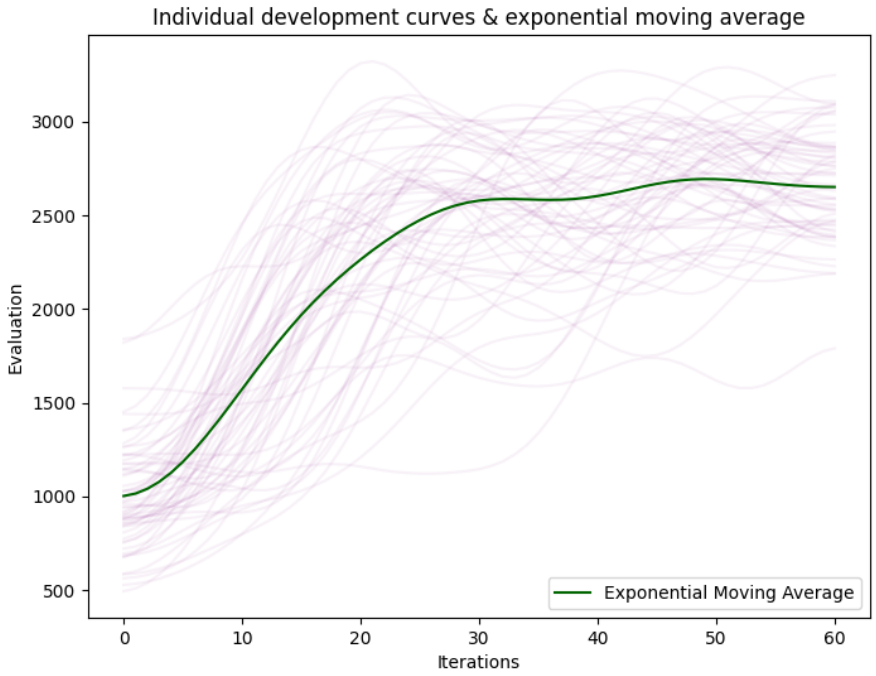
Depicted above are two representations of the results of the experimentation conducted [29]:

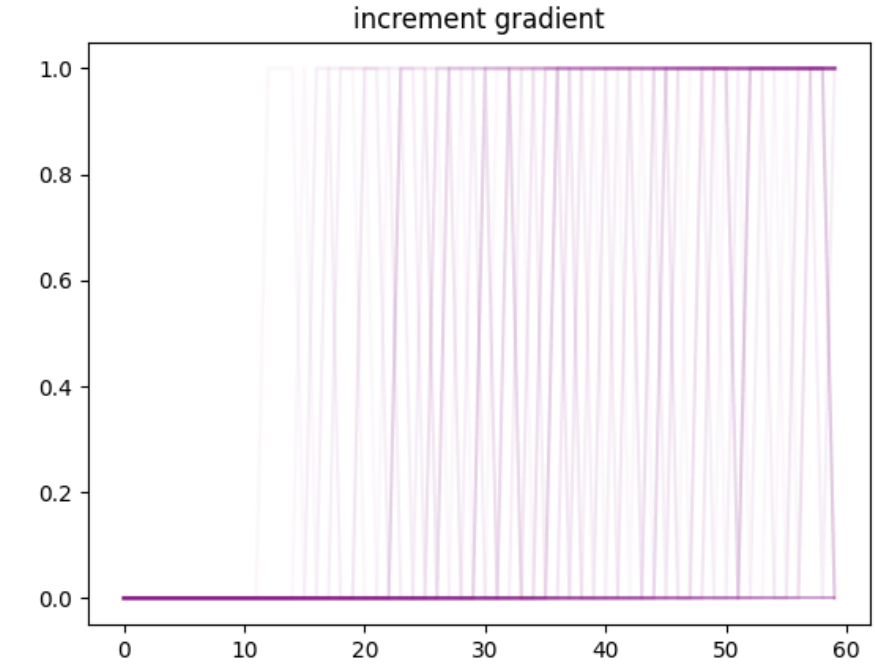
1. The first graph depicts smoothed evaluation curves in a low alpha (overlaid) as well as the corresponding exponential moving average (green), indicating an initial high order (potentially, exponential) ascend that begins to happen within the first 5 epochs leading to a stabilizing point at around epoch 20, after which the curve stays stable around a line to which it converges. This line is located at around 35% of the `LIKE_CAP` after smoothing and at around 40% before smoothing. Note that an ascend is visible in 100% of the cases. This does not mean that this method will always work, but much rather, it means that it is statistically immensely reliable to perform.

2. The second graph represents the weighted and subsequently summed-up occurrences of all the 3 correct *favored* tags having been identified by the algorithm, where 0 represents no hit and 1 represents a hit. Observe how towards later stages, a majority of the curves start converging to 1, whereas at the very beginning all of them start at 0. This additional metric goes to show that the process is statistically relevant as well as that the additional support through selected tags by means of correlation to the evaluation curve is highly effective and relevant.

Furthermore, computation for 60 accounts over 100 epochs combined with the large size of the database took just north of 10 hours, rounding to approximately $\frac{1epoch}{6s}$, which is more than manageable in the real world, considering posts have initially been set to 4 per day. Additionally, in a more realistic application the computation time rounds to approximately $\frac{1epoch}{1s}$, as the database is considerably smaller. The graphs for this scenario would look as follows:

[29] For aesthetic reasons these graphs have been generated with a large database: 25000 samples

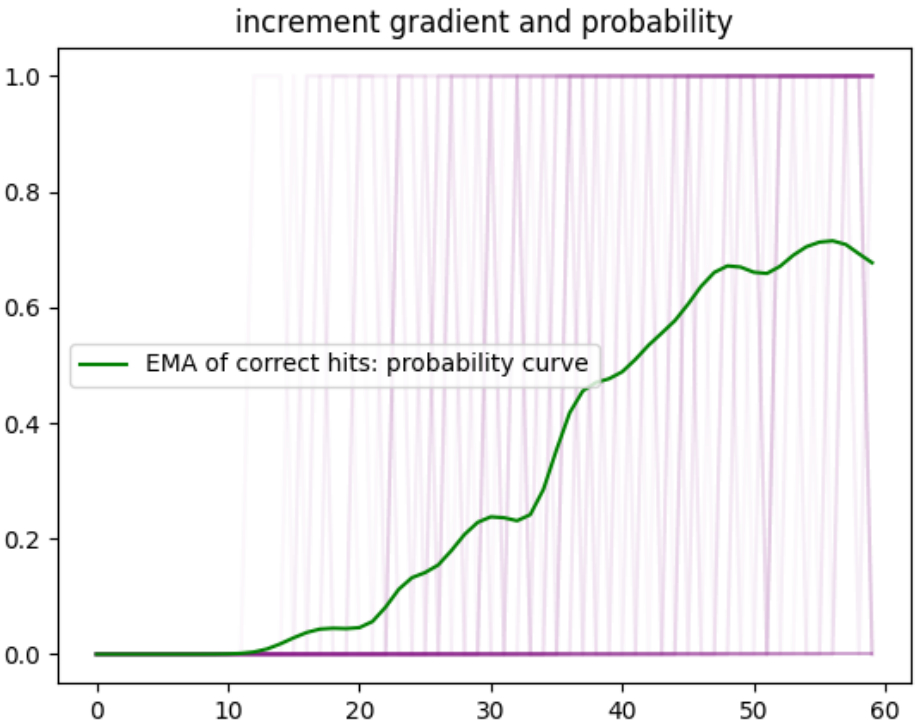

Individual development curves & exponential moving average
Evaluation
Iterations
Exponential Moving Average
3000
2500
2000
1500
1000
500
0
10
20
30
40
50
60


increment gradient
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50
60

The above are graphs representing the same scenario as the previous graphs, except that per account only 60 epochs were calculated and the database only included 1000 entries per account iteration. Note how an upward trend emerges in every case, too. The effects are, however, less steep than the ones achieved using a larger database, which makes sense. Notable is that the curve stabilizes around the same point in time (about iteration 20) as well as the characteristics of the ascend being similar, albeit of lesser order. Here, the curve stabilizes around 25% `LIKE_CAP` after smoothing (or 30% before smoothing). Additionally, note that the weighted increment graph (labeled increment gradient) behaves almost identically to its 25000-sample counterpart, looking like a zoomed-in picture of the previous graph cropped from 0 to 60, signaling a further statistical indication of the methods' effectiveness. Further statistics yielded by this testing stage yielded:

- The average first fully accurate pinpointing of all 3 *favored* tags happened after **27** (rounded) epochs
- On average, per *"account"*, the 3 *favored* tags were correctly identified **26.7%** of the time
- Computation time for this test was only approximately 45 minutes
- Furthermore, the accurate tag delineation function (essentially probability) looks as follows [30]

[30] it peaks at around y = 0.71 after smoothing or 0.74 before smoothing



As such, clear results are present in a more realistic scenario, too. As a matter of fact, their behavior is highly similar to that of its large-database counterpart, displaying high-order (possibly) exponential ascend within the same intervals as well as subsequent stability. Note that this is due to the nature of tag distribution within the database. In an actual scenario, there is no cap, and the favored and provided tags have an initial connection, as demand stimulates supply, rendering this theoretical success possibly tiny when compared to its real-world analogy. All facts and results considered, the tested methods have proven effective under synthetic circumstances. When faced with ideal (large) data supply, the methods displayed impressive adaptation potential, and when faced with a more realistic database scenario, the methods showed similar behavior to that displayed in the large-data testing, albeit less steep, **proving the concept synthetically.**

5.2 Evaluation in terms of expectations

Overall, the testing turned out as expected with some minor specifications that have not been accurately predicted:

1. The *"stability"* turned out to exist; however, it appeared considerably lower than 75% of `LIKE_CAP`
2. The nature of the functions ascend as well as the characteristics of it matched the prediction, where stability has generally been reached after 20 epochs, which was not thematized in the predictions
3. The influence of white noise was direly overestimated, as it proved to work interchangeably in higher and lower weights within addition
4. The weights within EMA were determined to lie within the specified range; namely, aat .75 and .25, which is surprisingly high (but not unexpected)
5. The functional range of the component support factor was slightly overestimated, as 5 proved to be the upper bound of what is functional, while everything above 5 started overshadowing the effect of the EMA vector, sacrificing efficiency

All in all, the results conform to the expectations set in a satisfactory manner. While slightly inaccurate assumptions were made during prediction, none of them were completely inapplicable. Therefore, this testing affirms the assumptions drawn during expectation and prediction setting. It is to be deemed satisfactory.

6 Conclusion & ethical statements

6.1 What is to be inferred?

Having attained success in the testing stage, proving that the methods do, indeed, work, the question of corresponding conclusions is begged. The core questions regarding what follows from the research are the following ones broken down:

1. *What has this research achieved?*
2. *How could it be used or implemented?*
3. *Why is it relevant?*

6.1.1 What has this research achieved?

Using a process involving a vast number of mathematical methods, this research managed to yield a method to reliably and relatively quickly achieve a theoretical attention/feedback optimization; albeit in theory, the assumptions upon which theoretical concept and methods rest are not so far-fetched as to render the described methods inapplicable, qualifying the yielded methods to be explored further and to be applied. This shows that there exists a simple manner, using which almost everyone can attain profit and/or publicity at this point in time without having the need to work for it. This fact alone makes the results of the conducted research attractive to anyone seeking quick attention in a digital landscape. This includes but is not limited to:

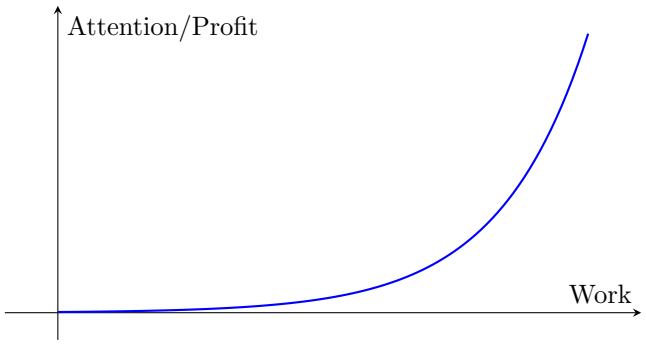
- Advertisers of any kind
- Politically engaged individuals
- Aspiring influencers lacking therefore required predispositions
- and more...

Apart from its obvious contents, this paper presents interesting implications worth exploring further, the most essential of which are addressed below in 7. To briefly foreshadow:

- The relative simplicity of this project indicates a high probability of such methods being widely used to keep people addicted to media for advertising/profit purposes

- Its quick adaptability to human interaction to maximize attention is a known concept and goes to show how much control and freedom large-data corporations hold over masses nowadays employing strategies similar in nature, begging many questions of morality and ethics
- The algorithms unequivocally imminent success in achieving what it has been designed to do, in combination with its simplicity, would corroborate a certain *dead internet theory* [19] or at least indicate a considerable probability of some of its core aspects being true (be it now or in the near future)

Therefore, the project has successfully confirmed the initial thesis that it is, in fact, not particularly difficult to find a large digital audience as an *"influencer"* in today's age of data science. By employing basic mathematical ideas from stochastics, statistics, information theory, and other fundamental mathematical concepts, it is possible to gain a sizable audience as well as subsequent profit without much work at all in the grand scheme of things:



While it may require a lot more work than simply creating original content at the beginning, the work required for maintaining and growing success approaches 0 towards later stages, as the application runs itself if implemented well, requiring little to no external maintenance. [31]

6.1.2 How could it be used or implemented?

By this point, it should be more than evident what the targeted use case for this research is/has been from the start; however, seeing as semantically broadening horizons should be a defining characteristic of any research project of this kind, it is possible to extend the discussed concept to various other applications.

[31] That is unless unexpected technical issues arise

Without needing to push the boundaries of commons sense too far, one could even go as far as to claim that any situation that requires optimization of feedback / attention influx could, in theory, make use of this method without much alteration.
Consider, for reference, the following scenario:

Supply-Demand-Optimization: Let *Lise-Lotte's Munchies (further denoted LLM)* be the name of a popular local ice cream shop. They sell all sorts of flavors: Chocolate, Vanilla, Strawberry, etc... The fact to be noted is that they sell a high number of flavors. For the sake of this argument, let the number of flavors be *25*. LLM, as many stores, have a limited amount of supply for each given period of time. They track their customers' favorites and make sure to have an adequate supply at hand to conform to the higher demand in a select few specific flavors that they observe. They also take into consideration long-term trends (such as seasonal demand). What they fail to consider, however, is that they may be losing out on customers, as they start adapting to new demand only so quickly as they start losing money. But fair enough, very few humans can pick up on changes that are not made apparent to them (Such as a collective shift to start preferring pistachio flavored ice cream, which, in this case, does not belong to their overall or seasonal favorites, wherefore they keep little supplied). This above failure is subsequently reflected in the number of recurring customers (seeing as the number of customers is a function of adequate supply). The fault in this case lies completely with LLM's managers, as they failed to adapt to a new business opportunity, and they even started losing profit because of it.
How could LLM's managers have foreseen this change in trend in time for them to adapt to it?
The answer should come as no surprise: *They could have used the optimization algorithm described above.* This could be implemented by tracking individual flavor sales at the checkout and computing the distance to the turnover function per day to the *n-th* combination of individual ice cream sales per day, which would slowly adapt with the arising trend of pistachio demand, potentially bringing in more (recurring) customers as well as not losing money due to supply shortage, keeping an upper hand amongst their competitors.

Entertainment-Preference-Optimization: In a very similar way, a club owner could track the visitor's over played song-genres to optimize the current

shift in preference, adapting to it, and potentially even making enough extra profit to hire a solid bartender.

Psychiatric-Treatment-Optimization: A psychiatric hospital could, for example, track patients with coinciding diagnoses of violent outbursts by algorithmically matching the methods of therapy administered/occurrence of common events per week to the number of violent outbursts, discerning the most suitable treatment methods as well as external mundane factors to mitigate said outbursts in each individual at, again, almost no extra work. This would pose further chances for data science in psychology, the interest for which is ever rising, according to [20], while keeping the required data relatively low as opposed to Stanton's criticism.

The above are just a few examples on how versatile methods to the likes of the one described in this paper are. Note that in each of these examples the following conditions are met:

1. A low to moderate effort way of consistent labeling exists
2. There exists a clear, unambiguous function of what is to be optimized
3. A sufficient amount of data can be procured in a given time frame
4. There exists an irrefutable causality between labeling categories and the target function [32]
5. There is room for optimization
6. The labeling categories are either complete [33], or they cover the area of causality to a vast degree [34].
   Note that if someone optimizes their yard incl. all the corresponding factors to support apple growth on their tree, they will still have no apples if there is a prominent detrimental untracked factor present, such as the neighbor stealing the apples in question.

[32] The one to be optimized
[33] Meaning they either describe the optimization category fully, such as in the ice cream example where every flavor is tracked
[34] Such as in the example of psychiatric hospitals

The above are the main conditions to follow in any particular application in theory. It needs to be noted that the above examples of application are pure conjecture that would fit a theoretical scheme. There is no saying whether or how an implementation of this sort would turn out, but it serves as just another conceptual visualization if nothing else.

6.1.3 Why is it relevant?

In today's age, where computational prowess is so high that the bounds of standard mechanics have to be disregarded to push to a new level with quantum computation, where every other vending machine is powered by essentially *magic* that nobody really understands that is AI, where the dominant manner of communication is an invisible net connecting paper-thin devices that next to everyone owns at least one of, where many lead a double life split between the physical and digital world, there is evident relevance to be discovered in researching optimization using exactly these techniques to ameliorate that which can be gained from all the above elements. This claim holds true, especially in the case of social media, which are a prime subject of economic interest for any corporation worth the title. Pursuing higher efficiency in marketing oneself digitally, be it in a ruthless and potentially unethical manner, highlights the particular opportunities offered to society by means of digital facilities while simultaneously accentuating the risks and ethical dilemmas that come with it, educating the public on what lies behind their daily life, bringing the thought of being manipulated by instances with all sorts of large data access of any digitally active individual to the forefront of thoughts. The relevance of this research lies not mainly with the devised methods themselves, but rather with the implications that it makes regarding the omnipresent digital surroundings as well as the frightening extent of the control that applied mathematics to data science can exert at any time for any purpose (be it self-serving, such as this one, or charitable in its nature). The relevance of researching the power that comes with understanding the science behind what most perceive as *magic* of ones and zeros is unequivocally one of the, if not the, most relevant subjects to thematize and support, especially as the world rapidly shifts into the quaternary sector, bearing "massive implications for globalized economic disruption" [21] if left as unattended as it has so far.

6.2 Legal and ethical implications

As is often the case in research, with each question answered, a new one arises. Especially in research regarding human behavior as well as how to exploit and utilize it, as is the case here, the question of morality, ethics, and legality should always be addressed. The following is a summary of what the discussed research provides / what it has been designed for:

> *The mathematical optimization of user interactions with videos is an economically driven concept designed to capitalize on users by keeping them engaged on a platform to generate income (more often than not through advertisements). The core foundation of such an optimization is a fatal human flaw, which manifests itself in a phenomenon widely known as doom-scrolling, fueled by an intrinsic human strive for dopamine production and cozy information consumption. Furthermore, in the aforementioned case, no proprietary original content is provided to the viewers, much rather the work for anyone who decides to employ such a method is minimized or, possibly, zero. This is due to a sort of digital arbitrage in which the "influencer" collects free content provided online by other people, and subsequently capitalizes on it by labeling it as their own.*

Ethically, this might sound questionable. Moral problems might arise for some from various aspects of the above concept, the most essential of which are the following:

1. *Is it ethical to capitalize on human flaws? (Especially so seeing as the effects prove harmful to the recipients)*
2. *Is it ethical to capitalize on the work of others?*
3. *Is it ethical to make a profit without having worked?*
4. *Is it ethical to employ such an unsustainable model of business?*

In an attempt to shed light on the above questions, it is vital to break them down adequately and regard any corresponding analogies. For this, each of the above questions needs to be dissected individually.

6.2.1 Is it ethical to capitalize on human flaws?

Phrased this way, many will instinctively oppose it. After all, that just sounds pure evil; however, it is vital to define what capitalizing on human flaws actually means. *What is a human flaw?* A human flaw is a defining characteristic of a person allowing for some sort of manipulation for external profit. This, however, does not mean in any way that the person who is being manipulated does not receive anything in return. On the contrary, one may think of the above situation as a sort of forced deal—but a deal nonetheless. One may see this exact strategy employed heavily in advertisement. According to [22], marketing is, at its core, exactly that: a deal forced through manipulation, which can be made apparent by inspecting a few of the following exhibits.

- **McDonald's:** McDonald's advertisement is often accompanied by all sorts of manipulation such as family togetherness [23], capitalizing on the social needs of humans, suggesting delicacy of food as well as hunger, capitalizing on humans' desire to eat delicious food. It is, however, vital to mention that the customers receive something in return, albeit disproportionally menial.

- **Coca-Cola:** One of the prime examples of exploitative behavior in marketing is certainly Coca-Cola. Their employing of manipulative advertisement has been a success story throughout the past decades, employing emotional suggestion on many levels [24]. Be it successfully connecting joyous childish aspects such as *Santa Clause* to their product, capitalizing on human melancholy, or portraying the nigh omnipresence of their product, leveraging the essential human need to belong [25] while feeding into the addictive properties of caffeine and sugar. Again, a sort of forced deal is provided to the audience, where, as a result, the consumer procures a product with which they may associate numerous externally implied sentimental qualities, which are not necessarily included in a caffeinated drink. The deal is, therefore, highly imbalanced.

- **Nike:** Nike is most known for their *Just Do It* campaign, which is also their slogan. In their advertisements, Nike often thematize people's intrinsic aspirations as well as their failure to reach it in suggesting that all that is needed is motivation, which, in turn, their product will provide to the people. To any reader, this statement, when written out this way, should seem exceedingly manipulative; and yet, it works. It does so by exploiting

> a fundamental human laziness - that being the desire to reach difficult goals such as weight loss in an easy and quick manner. Of course, no difficult goal may be cheated without proper corresponding work, which makes the above suggestion possibly even harmful to the audience, supporting a surreal illusion. Note that in this instance, too, the customer receives a product through a manipulation-enforced deal: Comfortable shoes, a good-looking shirt or an attractive headband, yet none of these products live up to the suggestions provided.

To answer the question of ethicality in the case of capitalizing on human flaws in this case, it needs to be stated that the user, albeit manipulated, receives exactly what they are expecting on a given platform, and, should there be no effective strategy employed, they would simply focus their attention on another *influencer*. Therefore:
Is it ethical? One could argue it is, as the deal, albeit forced and imbalanced, is apparent to any user: you get the kind of posts you like, and I get the money you generate while watching ads. An altruist, of course, would argue that the relationship between influencer and user is inherently imbalanced and therefore unethical. The decision hereof is left to every individual, provided the above facts.
But is it a known concept? Yes, absolutely, widely employed even. Manipulation to maximize attention is nothing if not synonymous to the core definition of solid marketing.

6.2.2 Is it ethical to capitalize on the work of others?

Again, put bluntly like in the phrasing above, many would condemn it. However, this question only comes down to which socioeconomic system you might prefer. A capitalist would argue that capitalizing on the work of others is as natural as breathing, whereas a socialist or communist would oppose that with all their might. For that reason, one could, in the shoes of a ruthless capitalist, argue here that if one is able to take another's work and market it efficiently enough, or improve on it slightly, passing it off as their own, then that is their right, as in doing so they are creating financial gain. That is also exactly the case in the procedure described above. Where the memes and foundations of the posts are taken directly out of the repertoires of others, one should not forget their composition in the definitive posts as well as labeling, reading aloud, and optimizing within the posts themselves. To answer then whether it is ethical

to capitalize this way on the work of others, one has to take into consideration the two main socioeconomic perspectives and decide which one suits them best. Note that to argue for ethicality in this case is synonymous with claiming that people shouldn't be rewarded for the work they do.
For visualization purposes: By pursuing the logic displayed above, the conclusion is reached that artists are not deserving of royalties if their music is shamelessly reused by somebody else.

Note that this question together with the answer above are closely tied to questions of legality, which are answered below; however, for the sake of reaching a satisfactory conclusion, foreshadowing cannot be fully circumvented: While the content used is not specifically protected by laws of intellectual property or copyright, it is to be noted that using the work of others without providing proper credit for the original creator is nevertheless, in most cases, regarded as highly unethical, which might be tied to the fact that neither western nor any other worldly society operates in capitalism of such ruthlessness as depicted above, yielding a verdict of predominant unethicality in this question.

6.2.3 Is it ethical to make a profit without having worked?

More often than not, undeserved profit is frowned upon, as, for many, a slight meritocratic tendency of regarding the work-reward system presents itself in their character. Every sensible person want their work to be rewarded accordingly: the more demanding the provided job, the higher of a reward is to be expected. In its essence, this argument is closely tied to utilitarianism, which incorporates a similarly structured ideal: the more useful somebody's work, the higher the reward should be. However, these ideals reach its limits rapidly when considering intermediation - a so called *middleman*. A intermediary (*or middleman*) is an individual or company who procures goods from a source for the purpose of reselling them later, marking up the price. [35]
Using both of the ideologies above, the role of a middleman appears needless or, even, more of an inconvenience, going against the world views mentioned above, rendering it unethical. Consider however the following:

1. *An intermediary does create value:* In spreading and promoting wares to audiences who would not have been able to procure them otherwise (or simply would not have). Any salesman is a sort of middleman.

[35] *Note that this project does exactly that - a digital arbitrage / intermediation*

2. *Society is built on middlemen:* According to [26], intermediation "played an important role in a historical perspective", as trade hinged and still does on distribution and marketing. A capitalist society requires this kind of distribution to advance (keeping competition between corporations active).

3. *Middlemen work, just differently* As an intermediary the work is based solely on wit and creativity to find what to procure and where to turn a profit, matching demand, creating value

Note that upon initial inspection, the role of an intermediary seems needless and expendable. To utilitarianism as well as meritocratic philosophy, a middleman would appear unethical at first. However, upon closer examination, the middleman proves to be useful and, in fact, even working, or rather, his wit is working for him. The above facts suggest a stronger component of ethicality in this case.

6.2.4 Is it ethical to employ such an unsustainable model of business?

To understand the dilemma in this case, it is first necessary to establish why this model of business is unsustainable. *What makes a business model unsustainable?* Non-sustainability shall be defined by satisfying one or more of the following criteria.

- **Long-term stability:** If a business model operates on a short-term trend, which is highly susceptible to slight variations in social demand, potentially vanishing or dramatically decreasing on a whim, it is to be considered instable and as such, unsustainable.
  As an example hereof, consider the sudden rise and dramatic fall of fidget-spinners in 2017. Anyone who produced large numbers or, worse, procured larger numbers to resell later towards the "unpredictable" end of their trend fell victim high losses.

- **Parasitic stability:** If a business model can only reach success by prying on another's, a host's, work it is per se unsustainable, seeing as if a larger number of businesses or individuals were to employ a similar or identical strategy on the same host, the source of income would become over-used, lose attractiveness, and "die".

Consider, for this argument's sake, the dead internet theory [19]: While questionable and classified as a conspiracy theory, it suggests that a crushing majority of internet traffic nowadays is carried out by bots and automated programs (recently, LLMs), suggesting that the host (the internet) is slowly "dying" due to an exceedingly high influx of computer-generated content to generate money parasitically, which is, in turn, pushed by a different kind of non-human interaction, which exists to raise human attention. This ends in a vicious cycle of random computer generation as well as computer-perpetrated positive feedback, slowly drifting away from what human users want to consume, making the internet less attractive, "killing" it. While the aforementioned theories and studies may seem dubious and exaggerated, they unequivocally bear logical implications and truths and shows that, provided everyone exploits the internet with non-original content, it will incontrovertibly cease to be attractive to humans, and, as such, it will crumble as a source of income.

- **Finite-resource-dependent stability:** The definition, in this case, is, essentially, self-explanatory: If a business is dependent on any sort of limited resource, it is bound to crash at a certain point, as one of the two following business-detrimental events may occur:

  1. The required resource runs out, rendering the business unprofitable.
  2. The required resource becomes so expensive as to make the business economically impracticable. [36]

  For this, consider the industrial use of natural resources such as oil and their subsequent effect on the atmosphere.

Of course, there are more aspects to what may render a provided business model unsustainable; however, the above are, for this project, the most relevant ones, of which it satisfies the criteria of **Parasitic Stability** 6.2.4, as it is neither heavily dependent on a limited resource nor is it build on the base of a short-term social trend [37]; however, as per theory mentioned, it depends on on a host of attractiveness (the internet) and its relative rarity to generate attention and income, all while simultaneously "killing" said host. Having established that the model of business is, without question, economically questionable, the question arises as to how that correlates to its ethicality.

[36] The latter of which is more common.
[37] Much rather, this is what it can best adapt to.

The ethical issue arises mainly due to intentional exhausting of the host, which is, in this case, any short-form content platform, but it may be applied to the entirety of social internet. Additionally, researching a method to raise the efficiency of the described parasitic behavior as well as subsequently publishing it for any person to see, understand, and, as a consequence, potentially implement is exceedingly difficult to justify properly, indicating a dominant aspect of unethicality in this question.

6.3 Legal Questions

Legality is an important question in dealing with AI and deterministic algorithms on social media in modern times. A recently big question is whether image generating AI models may be trained on publicly available images of unknowing artist's. This project, too, it raises question of legality. Not that this section mentions general legal concepts to defend its legality without focusing on a single countries law. The following question at hand is to be answered: *Is it legal to make money off other people's memes/stories:*[38] Generally, phrased like so: No.

1. However, it can be argued for implied license agreed to by the original poster by submitting the meme or story to a public forum without any marks of copyright. This would render commercial use legal, according to [27] (at least in US law)

2. Furthermore, it may be argued that integrating the story/meme in a video, combined with TTS and the context of other posts, could be considered derivative work, rendering commercial use legal [28] [29]

3. Additionally, the possibility exists to argue for intent to contribute to the public domain: By taking a meme template from the public domain, editing it with a caption, and subsequently publishing it on a public online forum without any markings of intent to preserve it as intellectual property, the intention to contribute to the public domain could be implied, as precedent suggests that memes or stories posted on Reddit under these circumstances are used commercially by others. This could be interpreted as an implied voluntary waiving of rights [30] [39]

[38] Note that there are no legal issues if no financial gain is procured
[39] Note that this is not legal advice and would need much more rigor to defend or prove

Considering the defense above as well as the impressive amount of precedent, it would likely not be deemed illegal in a court of law; however, this project does walk a line in a legal gray area, and as a consequence, needs a solid defense if implemented. Consider as a final point that *innocent until proven guilty* is a strong defense, as, in a digital world, it would prove very difficult to prove being an original creator of anything.

7 What follows in terms of future research?

Every time a question is answered, at least two more are raised, making research nothing less than a glorified hydra. Although this study provides theoretical solutions to the problems of whether and how it is possible to quantitatively maximize online attention and positive feedback, it should also raise many new ones. The following are the more evident of the aforementioned questions:

1. *How would these theoretical methods perform when actually applied to a real-world situation?*
2. *What is the exact psychology behind people's reactions to this kind of posts?*
3. *Can this idea be further refined generate more attention/profit?*
4. *Are short-form content platforms the only places where this idea would appear to be useful?*
5. *Are there perhaps even places where methods similar to these are in use to keep a larger number of people hooked on something?*

Or, even less obvious questions such as:

1. *Do the methods only work because of the initial assumptions of a pre-existing suggestion algorithm and if true, how exactly does TikTok's suggestion algorithm work? Is it any similar in structure to this?*
2. *If, in time, more non-human-powered content creation machines/programs emerge, what would happen? To what extent does the dead internet theory [19] hold true? Where does it fail?*
3. *Are social media doomed in just another tragedy of the commons?*

4. *How can larger AI models and deterministic models (similar in concept to the one described in this paper) acting on the internet be prevented from overfocusing due to bot interactions?*
   Paraphrased: *How could human content be preserved if it came so far as to the computer-generated side of the internet overpowering the one perpetuated by humans?*

Some elaboration on a certain subset of the above queries is needed for the sake of completeness, although some are self-explanatory. The part to be accentuated here is mainly the one encompassing **the tragedy of the commons** and the **dead internet theory**, as they pose a prominent obstacle in questions of ethicality concerning this research as pointed out in 6.2.4. It cannot be repudiated that, if launched en masse, programs like the one described in this study would yield quick profit for anyone daring enough to implement it; however, an exhaustion of the common good that is the attractiveness of the internet would become imminent. For this purpose, a further study on the limits and extent of these methods is to be conducted as well as their ultimate long-term effects. Furthermore, an additional study would be advisable, which concerns itself with development of neural network implementations or deterministic programs regarding this exact situation[40] is to be conducted while focusing on *false feedback*[41] and the subsequent divergence of the initial program from it is eventual goal and premeditated path. After inspecting the results of said study, a further step in the direction of proposing new guidelines for non-human interaction on social media is to be devised. In combination with all of the above, it wouldn't be a waste of time to conduct further studies on the relative non-human internet events as well as their overall impact, either corroborating, disproving, or stating its extent more precisely and as such, more informatively.

Arguing from a different angle, a study on the large-scale implementation of such methods could be conducted to shed light on today's marketing, the way social media operate in collaboration with their sponsors/advertisers, and the way people react to such subliminal *forcing of content* in the short-term as well as long-term span. This could yield the following insights:

[40] This being maximizing positive human feedback / human attention on social media
[41] One perpetuated by non-human instances

1. Further apprehension with respect to human attention as well as furthering the recently popular studies of cerebric reward system(s) including the role of dopamine and serotonin in stimulation and addiction [31]

2. Further understanding of economic opportunities regarding how to *force* products, keep a broad audience interested in oneself or one's own economic endeavor as well as how to utilize mathematics and data science to flexibly adapt to changes in trends and demand, "as the use of recent Data Science technologies for improving forecasting and nowcasting for several types of economic and financial applications has high potential. ", according to [32]

3. Further knowledge in terms of extent of the effect that large data corporations have on any digitally active individual / active masses, providing further implications regarding the scary degree of freedom that is left at any large corporation's discretion in today's digital age, seeing as "through their ownership of large digital platforms, Big Tech have amassed tremendous resources enabling them to redefine communication, commerce, and even culture", according to [33]

With the above suggestions and implications, it becomes evident that in spite of this paper's length, or perhaps, because of its length, the is much to be studied and uncovered in subsequent research. The topics at hand range from psychology over business and economics back to mathematics and data science yet again to complete the circle. There is always much more to be desired, and as such, exponentially more to be done.

8 Sources

References

[1] D. Herrera, "Discrete calculus and finite sums." https://princeton.learningu.org/download/bc7e982a-7974-459c-8286-c201d2236718/M352_Discrete%20Calculus%20and%20Finite%20Sums.pdf, April 2016. Accessed: 2024-09-06.

[2] M. S. University, "Directional derivatives and gradient vectors." https://users.math.msu.edu/users/gnagy/teaching/10-fall/mth234/w7-234-h.pdf, Fall 2010. Accessed: 2024-09-06.

[3] S. H. To, "Correlation coefficient formula." https://www.statisticshowto.com/probability-and-statistics/correlation-coefficient-formula/, n.d. Accessed: 2024-09-06.

[4] G. Margalit and J. Kable, "Cross-correlations and their applications to brain patterns and sound waves." https://courses.physics.illinois.edu/phys406/sp2017/Student_Projects/Spring16/Josh_Kable_Gilad_Margalit_Physics_406_Final_Report_Sp16.pdf, Spring 2016. Accessed: 2024-09-06.

[5] IEEE Computer Society, "Bulletin of the technical committee on data engineering." http://sites.computer.org/debull/A01dec/A01DEC-CD.pdf, December 2001. Accessed: 2024-09-06.

[6] C. Spearman, "The proof and measurement of association between two things," *The American Journal of Psychology*, vol. 15, pp. 72–101, January 1904. Accessed: 2024-09-06.

[7] M. Vu, "Lecture 1: Entropy and mutual information." EE194 – Network Information Theory, Tufts University, Electrical and Computer Engineering. Lecture 1: Entropy and Mutual Information, Tufts University, EE194, Network Information Theory.

[8] J. Chen, "Vector operations." https://people.math.harvard.edu/~jjchen/math21a/handouts/vector-ops.html, n.d. Accessed: 2024-09-05.

[9] S. University, "Matrix operations." https://ee263.stanford.edu/notes/matrix-primer-lect2.pdf, n.d. Accessed: 2024-09-06.

[10] E. Million, "The hadamard product." http://buzzard.ups.edu/courses/2007spring/projects/million-paper.pdf, April 2007. Accessed: 2024-09-06.

[11] K. C. of Computer Sciences, "Time complexity." https://www.khoury.northeastern.edu/home/rjw/csu390-f06/LectureMaterials/Time-Complexity.pdf, Fall 2006. Accessed: 2024-09-06.

[12] N. J. Higham, "Accuracy and stability of numerical algorithms." http://ftp.demec.ufpr.br/CFD/bibliografia/Higham_2002_Accuracy%20and%20Stability%20of%20Numerical%20Algorithms.pdf, 2002. Accessed: 2024-09-06.

[13] FreeCodeCamp, "Time complexity chart." https://paper-attachments.dropbox.com/s_2D428973624E7FC84C7D69D11421DE762BEA6B6F3361231FCDCAE0425D14526F_1664885448372_Untitled.drawio+17.png. Accessed: 8 December 2024, time complexity chart.

[14] TikTok, "How tiktok recommends content." https://support.tiktok.com/en/using-tiktok/exploring-videos/how-tiktok-recommends-content, n.d. Accessed: 2024-09-06.

[15] PeaceDuke_official, "Tiktok algorithm theory from the creator with 40k." https://www.reddit.com/r/Tiktokhelp/comments/153zqyi/tiktok_algorithm_theory_from_the_creator_with_40k/, 2024. Accessed: 2024-09-06.

[16] Z. Cai, C. Peng, and S. Du, "Jitter: Random jittering loss function," *arXiv:2106.13749 [cs.LG]*, June 2021. Last revised on 7 July 2021.

[17] T. Alabsi, "Effects of adding subtitles to video via apps on developing efl students' listening comprehension," 2024.

[18] A. I. Zverev, *Handbook of Filters*. New York: Wiley, 1967. Bibliography: p. 569-571. Library of Congress Control Number: 67017352.

[19] A. Ahmed, R. Qamar, R. Asif, and M. J. Imran, "Dead internet theory," *Pakistan Journal of Engineering Technology & Science*, vol. 12, pp. 37–48, June 2024.

[20] J. Stanton, "Data science methods for psychology," Feb. 2020. Last reviewed and modified on 26 February 2020.

[21] P. Cooke, J. H. Yun, and Zhao, "The digital, quaternary, or 4.0 web economy: Some implications." Based on Report to DGIST government project, Republic of South Korea. Jin Hyo Yun is the corresponding co-author.

[22] V. Danciu, "Manipulative marketing: Persuasion and manipulation of the consumer through advertising," *Theoretical and Applied Economics*, vol. XXI, no. 2(591), pp. 19–34, 2014.

[23] X. Yuxin and W. Weichao, "A case study on mcdonald's advertisements from the perspective of intercultural communication," *English Literature and Language Review*, vol. 8, no. 2, pp. 22–31, 2022.

[24] M. Pham, "How coca-cola maintained 'human-centric' insight throughout the pandemic," *Marketing Week*, October 22 2021. Available at `https://www.marketingweek.com/coca-cola-human-insight-pandemic/`.

[25] G. Winch, "The importance of belonging to a tribe," February 2020. The curative powers of group identity. From *The Squeaky Wheel*.

[26] L.-E. Gadde and I. Snehota, "Rethinking the role of middlemen," in *Proceedings of the IMP 2001 Conference*, (Oslo, Norway), Industrial Marketing, 2001. Presented at the IMP 2001 Conference, 9-11 September, Oslo, Norway.

[27] S. Vondran, "The implied license defense to copyright infringement." `https://www.vondranlegal.com/the-implied-license-defense-to-copyright-infringement`, Nov. 2024. AZ Bar Lic. #025911, CA Bar Lic. #232337.

[28] Federal Assembly of the Swiss Confederation, "Federal act on copyright and related rights (copyright act, copa)," Oct. 1992. Derivative works are protected as works in their own right. The protection of the works used in the derivative work remains reserved. Status as of 1 July 2023. English translation provided for information purposes only and has no legal force.

[29] Federal Assembly of the Swiss Confederation, "Federal act on copyright and related rights (copyright act, copa)," Oct. 1992. Where the author has transferred the rights to a copy of a work or has consented to such a transfer, these rights may subsequently be further transferred or the copy otherwise distributed. Status as of 1 July 2023. English translation provided for information purposes only and has no legal force.

[30] Swiss Federal Institute of Intellectual Property, "Using a work: Public domain." `https://www.ige.ch/en/protecting-your-ip/copyright/using-a-work/public-domain`. Accessed: 8 December 2024.

[31] O. Guy-Evans, "Brain reward system." Simply Psychology, Sept. 2023. Reviewed by Saul McLeod, PhD. Accessed: 2024-12-05.

[32] S. Consoli, D. R. Recupero, and M. Saisana, eds., *Data Science for Economics and Finance: Methodologies and Applications*. Ispra (VA), Italy; Cagliari, Italy: Springer, 2021.

[33] S. Khanal, H. Zhang, and A. Taeihagh, "Why and how is the power of big tech increasing in the policy process? the case of generative ai," *Policy and Society*, Mar. 2024. Corresponding author: Araz Taeihagh, Lee Kuan Yew School of Public Policy, National University of Singapore, Singapore.

List of Figures

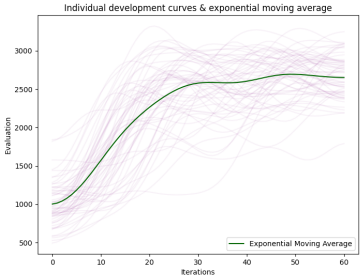


Figure 5: Self-created plot

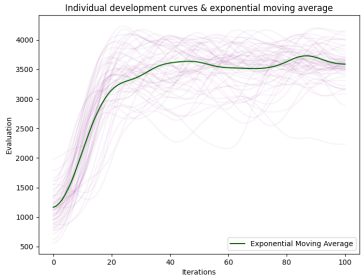


Figure 6: Self-created plot

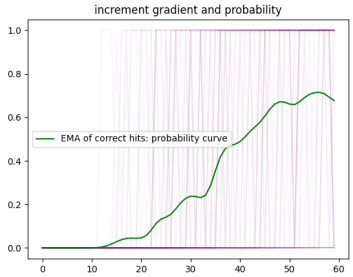


Figure 7: Self-created plot

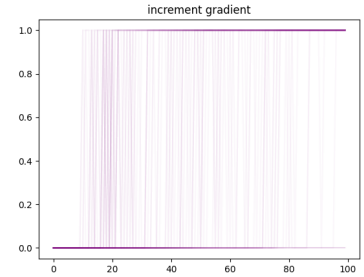


Figure 8: Self-created plot

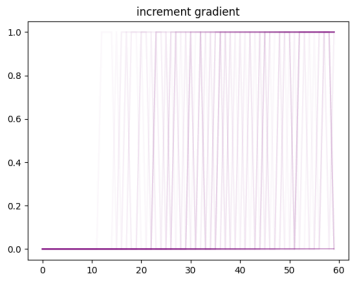


Figure 9: Self-created plot

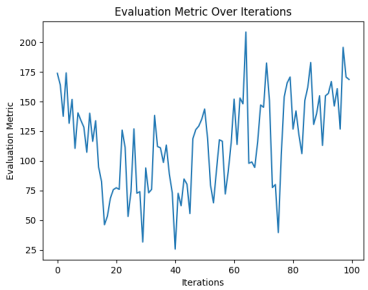


Figure 10: Self-created plot

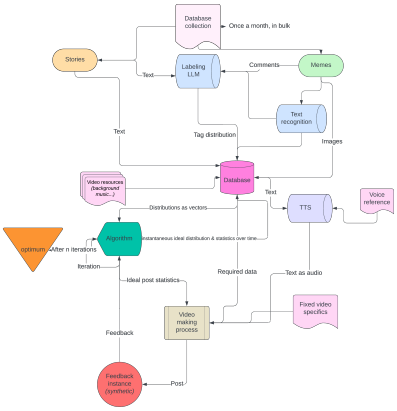


Figure 11: Self-created flowchart

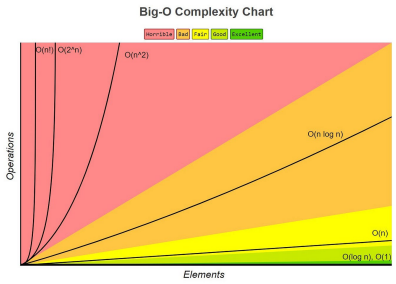


Figure 12: Time Complexity Chart, procured from FreeCodeCamp, accessed 8 December 2024[13]

Every other graph used in this paper has been created originally for it.

9 Acknowledgements

I would like to extend my gratitude to those who have supported me throughout the development of this work.

First and foremost, I wish to thank my supervising teacher, Mr. Martin Pfister, for his continuous support and insightful feedback during the course of this paper. His expertise and feedback have proven indispensable throughout.

I would also like to extend my sincere appreciation to Mr. Gian Peter Ochsner, for his constructive and detailed guidance regarding language and formality.

Additionally, I would like to thank my family for lending an ear whenever I needed to discuss a given aspect of the project with someone.

Not to forget, special thanks are extended to Christian Frick, without whom I would never have had the option the sufficient expertise to tackle this project, let alone complete.

Lastly, I am grateful to all those who have contributed, directly or indirectly, to the successful completion of this paper/project.

Authenticity Declaration
Maturitätsarbeit

Name: ..... First Name: .....

Class: ..... Supervising Teacher: .....

I hereby declare that I have completed the present work with the title

.....

independently, with the exception of proofreading by 2 individuals (Mr. M. Pfister & Mr. G. P. Ochsner) as well as my family's assistance in binding the paper, without unauthorized external assistance and in my own words. I declare that I have not used any sources or tools other than those explicitly stated (including AI tools) and that I have cited the authorship of any direct quotes (citations) or paraphrased statements and ideas from other authors or AI tools. I have also acknowledged all individuals who have made a significant contribution to this project/paper. Furthermore, I vouch for the scientific integrity of the research conducted. This includes:

- The methods created
- The results listed
- The theory provided

Appendix

GitHub Project

You can find the code used in this project (with possibly some minor alterations) deposited in the following github project:

https://github.com/AlbertDerWeise/MaturArbeit

QR Code

Below is the QR code for quick access:

Figure 13: QR Code for the project