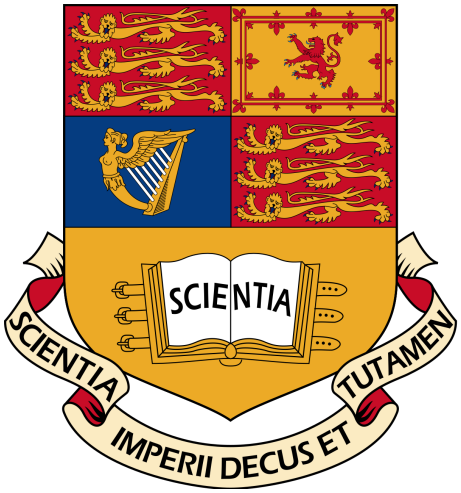
Imperial College London

Department of Electrical and Electronic Engineering



A Collective Understanding of Happiness

Author:

Nima Karsheneas

CID:

01500753

Lecturer:

Prof. Esther Rodriguez-Villegas

June 2023

Final Report Plagiarism Statement

I affirm that I have submitted, or will submit, an electronic copy of my final year project report to the provided EEE link.

I affirm that I have submitted, or will submit, an identical electronic copy of my final year project to the provided Blackboard module for Plagiarism checking.

I affirm that I have provided explicit references for all the material in my Final Report that is not authored by me, but is represented as my own work.

Abstract

Acknowledgements

Contents

"Brother Gallio, all want to be happy, but when it comes to see clearly what makes life happy they are shadowed by obscurity" - Seneca, *De Vita Beata* [104]

1 Introduction

The goal of this project is to employ engineering techniques to provide a technical framework for broadening our collective understanding of 'happiness'. As a more personal aim, I'd like to produce something tangible, a tool that harnesses this understanding that anyone can use for the development of their own sustained, and robust happiness. For this reason, the project will be focused on implementing something actionable, a guide for the user, but ultimately leaving the journey of finding a sustained happiness down to the individual. The directions that can be taken are vast, and the one that is chosen, will be crucial to the outcome. The following section will lay the philosophical and scientific groundwork for the happiness space, and by examining this space, the optimal path will be determined. The background for this project is split into 2 sections, the first is the background required for project specification. The second is the exploration of the engineering space to draw techniques that can best address the outlined problem.

– – – – – – – – – – – –

Approaching the concept of happiness can be quite intimidating for an engineer, its entire essence is a sea of red flags for a discipline that thrives and depends on clear definitions, metrics, and some degree of objectivity to build from. As a result, very little research exists that aims to employ engineering techniques to further develop our understanding of this all-important notion.

On the other hand, due to its crucial role in driving our motives and decisions, and its inherent subjectivity, happiness has been the subject of fierce and consistent philosophical discussion for millennia. From Aristotle's viewpoint that happiness is derived from virtuousness, i.e. the practice of what is 'good' (e.g. courage, justice, and generosity) [11] to the Buddhist philosophy that happiness is the embrace of all the joy and sorrow that one encounters [78], i.e. a state of mind that is independent of life events—a plethora of definitions and interpretations of what constitutes and increases happiness has materialised over the course of time, and all over the world.

This, fundamentally, is a psychology and philosophy problem being approached from an engineering perspective, thus, before moving forward and further defining the problem space, it is important to establish a direct line of communication to translate philosophical and psychological concepts into engineering ones. 'Happiness' (see definition of the notion in section 2.1.2) experienced at time t, $h(t)$, can be considered as the output of a process, generated by a set of inputs $\varphi_1(t), \varphi_2(t), ...\varphi_n(t)$ and passed through a non-linear time-variant system B, where the semantics of each of these components can change depending on one's

perspective on the problem.

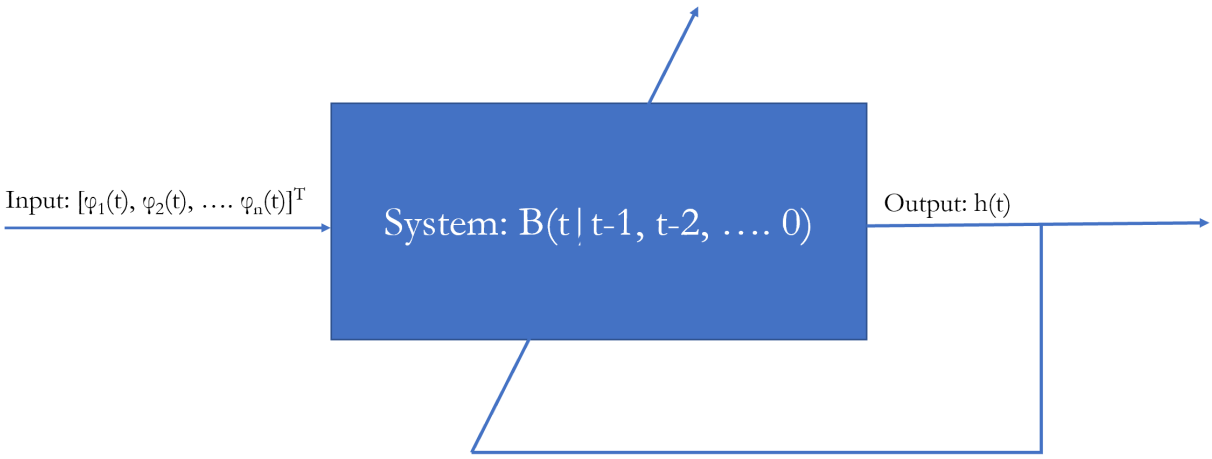


Figure 1: A diagram representing the happiness 'process' as a non-linear time-variant system.

The inputs can be considered as an individual's current position in the 'situation space', which is mapped by the range of possible values of variables that have relation to our happiness. These variables can take a variety of forms, it could be the weather that day, it could be the individual's current income, or it could be the current strength of our personal relations. Where we find ourselves in this space, is our current 'situation'.

The system B represents both the conscious and sub-conscious parts of our brains that maps our situation to our experienced level of happiness. It's a model that represents, for example, what kind of weather makes us happy, and how happy it makes us. This representation is not linear, given where we lie in the situation space, for the same input of weather, it may provide no contribution to our happiness or unhappiness, our mind may simply be elsewhere. Intuitively, its parameters are entirely dependent on the past and are time-varying; our brain is not resetting and regenerating itself at each new time instant, it is learning and fine-tuning its parameters constantly. In fact, the development of the brain, (the result of feedback mechanisms shown in Figure 1) is a complex amalgamation of genetic and experiential factors, and is enabled by Neural Plasticity [50]. The remainder of this project will investigate, design, and implement different ways of not only understanding more about the happiness system, but to provide users with inputs that will maximise their happiness output for this given system.

2 Background

2.1 Background for Project Specification

2.1.1 Blurred Lines

Often, the notion of happiness becomes convoluted with 'well-being'. Well-being has been defined as a combination of positive emotions such as contentment and happiness along with other measures, such as having agency over one's life, holding a sense of fulfilment, and experiencing strong and positive relationships [44]. It can thus be viewed as a study of *all* that leads to the 'betterment' of a person. It's important to note that this is not a measure of psychological state, but a weighted measure of psychological, social, and physical factors.

Just by examining this definition, the blurry lines make it easy to see where confusion arises. Well-being is defined as a combination of many dimensions, one being happiness (as described above), however, each of these dimensions themselves contributes to a sense of happiness (e.g. positive relationships are predictive of happiness [26]), suggesting that not only does happiness contribute to well-being, but well-being contributes to happiness; these are overlapping concepts that approach the human experience from different perspectives.

Happiness is undeniably tied to our own experience and internal biases, accordingly, we all have our own ideas of it, but yet it's a feeling we can all relate to [32], and for many, it's the most important measure of a good life. It provides an almost poetic scope on life, it's self-declared, and requires deep internal examination, it's rooted in our opinion of ourselves, and our opinion of what's around us; it's riddled with nuance. This subjectivity leaves a lot of room for uncertainty in an answer, can someone be lying to themselves about how happy they are to fit with an external agenda that has been imposed on them (peer pressure, emotional abuse, strict parents)? Could it be undesirable to be happy within a certain environment? What constitutes happiness for an individual?

On the other hand, well-being takes a more pragmatic approach, a top-down view of our human experience. It attempts to remove the aforementioned uncertainties, by applying context. How happy someone claims themselves to be is not the only indicator of how life is going, a look at objective metrics helps paint a clearer and more consistent picture. From happiness' perspective, well-being is the quality of soil from which happiness springs, whereas, to well-being, happiness is just one of it's many ingredients. In the context of scientific research, as a result of the ambiguity associated with 'happiness', the term 'subjective well being' is used instead, thus, when examining results from clinical study, conclusions regarding 'subjective well-being' will be taken as conclusions regarding happiness.

Nevertheless, our use of the term 'happiness' still remains abstract, informal, and somewhat ambiguous. In the following section, we examine the dominant schools of philosophic thought surrounding the subject, to develop a more robust and precise definition of the term.

2.1.2 The Philosophy angle

Philosophical literature regarding happiness can largely be divided into four schools of thought, *Hedonism*, *the life satisfaction theory*, *emotional state theory*, and *hybrid theories* [40].

The Hedonic viewpoint originates from the Greek philosopher Arstippus who claimed that the goal of life is to maximise one's pleasures and that happiness is the totality of such pleasurable experiences [81]. This can be interpreted as an aggregate of pleasant experiences over unpleasant experiences. It's up to an individual to determine the 'pleasantness' of an experience, but aside from that, it leaves little room for self-evaluation regarding their own happiness. Instead of an output $h(t)$ in the process outlined in Figure 1, it instead sees the process with an output $p(t)$, equal to the 'pleasantness' experienced at time t, with happiness now becoming $h(t) = \sum_{t=t_0}^{t} p(t)$, where a more pleasant experience results in a larger value of p. By and large, Hedonism concerns itself with the momentary feeling of happiness (pleasure is synonymous with momentary happiness, and overall happiness is the sum of that).

Life satisfaction claims that happiness is captured by how satisfied an individual is with their life as a whole. Tatarkiewicz compellingly argues that satisfaction with one's life as a whole is a satisfaction with all that has come, and all that is to follow [101]. Wayne Sumner also claims that happiness is intertwined with our expectations, our satisfaction with life is a measure of how well our life conditions match up with these expectations [97]. The holistic nature of this perspective bears some resemblance to 'well-being' (discussed in the previous section), however, the key difference is that this consideration is not done via an objective evaluation, but via self-reflection, it's still embodying subjectivity, feeling, and emotion.

Emotional state theorists identify happiness with one's emotional condition as a whole and can be interpreted as the opposite of depression or anxiety [40]. A distressed individual may lead a mostly pleasurable life, with reality matching up exactly with expectations, and satisfaction with life as a whole, but be subject to regular panic attacks and breakdowns. In this case, hedonists and life satisfaction theorists may claim the person to be happy, whereas emotional state theorists will claim the opposite. It is the psychological aspect of well-being and is a measure of happiness by looking at the 'health' of the parameters of the system B.

Hybrid theories claim that each of the described measures of happiness is not independent of each other, but actually coexist, and each contributes to the notion of happiness. It's a look at each component within the happiness process, depicted in Figure 1, sure, we are concerned with a high value of $h(t)$, but, only by examining the parameters of the system (emotional state theory, life satisfaction) can we conclude where we're heading (i.e. is the system stable?). There is some empirical research to support such an idea of coexisting theories, it has been shown that people that experience positive emotions over the course of a month (a hedonically 'happy' month), see an increase in their life satisfaction [23]. It seems very familiar (at least to me) that a month of pleasurable experience (e.g. coming back from

a holiday) would lead to a much more positive outlook on life, and as a re-enforcing feedback, this increased life satisfaction would enable the extraction of pleasure from day-to-day activities that wouldn't have been extracted beforehand. Happiness is a complex dance between each of the above theories, they're interdependent, a rise in one *likely* leads to the rise in the others, and vice-versa.

Although there are plausible arguments for each of these theories, and their likely co-existence, each is different in nature, thus the one on which focus is chosen to be placed will significantly alter the nature of the problem definition. The end goal is to produce a tool that increases an individual's sustained happiness. Hedonism, as previously explored, has a focus on momentary pleasures. One can't control the events of their life in the past, and so a pleasurable experience at time t is only ever a net positive; it is all that is ever really considered, and so when optimising, hedonism has little relevance to a sustained and robust happiness.

Emotional state theory is concerned with an evaluation of an individual's psychological state, this can be done via psychological assessments or tests. Psychological tests are the use of formal checklists and questionnaires, they are usually 'norm referenced', meaning they are standardised for consistency and repeatability, and have proven to be effective at measuring particular psychological traits and disorders [31]. Assessments are performed by psychologists, who compile and select various tests and data for a holistic evaluation [31], assessment is clearly out of the question in this project for practical reasons. Much like with well-being, happiness is determined by an entity independent of the individual, it introduces objectivity at the cost of personality.

Life satisfaction embraces personality, but perhaps relies too heavily on it. Although life satisfaction is an inherent capture of sustained happiness, its answer can be volatile due to its reliance on the individual, . Schwartz et al., for example, report that life satisfaction is considerably influenced by momentary contextual factors [86](the situation space composed of momentary variables are highly influential on the output of the system, i.e. the output $h(t)$ may not be indicative of a sustained happiness). This volatility could be viewed as stochastic disturbance, and to rectify this, consistent and frequent measurements smoothed with sliding Moving Average windows can be used, a common technique in stochastic time-series data analysis [6]. Another common argument against life satisfaction is that one can lead a very stressful and unpleasant life and still claim to be satisfied with their life [40], however, it could also equally be argued that this is where life satisfaction leaves room for the individual; they may not place any importance or value on pleasurable experience, but can still indeed be 'happy'.

The goal of the project is to build a collective understanding of sustained happiness along with a tool that individuals can use. By introducing objective measures of emotional state, we fix its definition to the assumptions of the metric itself, impeding on a collective idea of happiness from taking shape. Instead, by evaluating one's life satisfaction and embracing its

subjectivity, we give the individual the freedom to decide their own happiness based on their own biases and ideas. Through this, the underlying trends of cultures, minority groups, etc. can be better understood. This is not a disregard of all other theories of happiness aside from life satisfaction, they will be equally important in mapping a collective understanding of happiness. Life satisfaction will be used as the evaluative, and investigative measure, such that the resulting outcome of this research will tend towards optimising it.

2.1.3 Insights from scientific research

Much of current scientific research on happiness is directed towards building a clearer understanding of the inputs and outputs of the 'happiness process' outlined in Figure 1. Correlation research largely centres around identifying the most important variables that map out the situation space, but sometimes offers insights into the system B.

The *Hedonic Treadmill* is a popular and well-discussed theory of happiness within psychology, it was first introduced by Brickman et al., backed up by empirical evidence from lottery winners and accident victims, and claimed that people have temporary reactions to positive and negative events, and eventually return to a baseline level of happiness [17]. This theory claims that the system B dampens all inputs, leaving things rather bleak—it's a declaration that research to increase sustained happiness is obsolete. However, recent research offers hope, in a longitudinal study of 3,608 Germans, Diener et al. found that 24% of participants experienced a significant change in life satisfaction in the last 5 years compared to the first 5 years of the study [34]. In another study, Fredrickson et al. found that the practice of loving-kindness meditation (LKM), which focuses on loving and kindness towards oneself and others [71], enables an individual to reach higher levels of baseline positive emotions and life satisfaction [33]. This suggests that by shifting and then re-enforcing our perspectives regarding our interactions with the world, we can indeed break the hedonic treadmill to elevate our baseline happiness. Happiness is not entirely homeostatic, there is room for improvement, and we have personal agency for change.

The success of LKM falls well into the framework of Positive Psychology, more specifically the idea of 'authentic happiness' coined by one of the field's leaders, Martin Seligman. He outlines happiness as the resonance between three key areas in one's life experience, Engagement, Meaning and Positive Emotions [89] [88], which are summarised as following:

1. Engagement: Seligman emphasizes the concept of 'engagement' as being a key contributor to one's happiness. This refers to the state of being fully absorbed and deeply involved in activities that are intrinsically motivating. He alludes to the idea of "flow", which occurs when a person is highly engaged in a task, experiencing a sense of timelessness and immersion.

2. Meaning: A meaningful life, for Seligman, is being a part of something that is greater than yourself, which provides you with purpose and direction but also a sense of com-

munity. Engaging in altruistic behaviours and contributing to the well-being of others in a community is said to provide individuals with this sense of meaning.

3. Positive Emotion: Perhaps the byproduct, or from an alternative perspective, the backbone of the above two areas, positive emotions constitutes the cultivation of joy, gratitude, contentment, and love. Seligman provides a therapeutic framework for not only experiencing positive emotions in the present moment, but strategies to enhance and sustain these positive feelings [88].

To begin with, Seligman et al. provide clinical studies that highlight the positive effect on individual happiness as a result of a focus on positive psychology [52]. The idea of 'meaning' bears a strong resemblance to the idea of life satisfaction, it exists as the foundation for experience, it's a state of Being that serves as the viewpoint of experience. Positive emotion is synonymous to the emotional state theory, and so offers nothing really new to what has already been discussed. However, the final key area, engagement, offers a new perspective (LKM serving as empirical evidence for its positive effect), and this is essentially the application of philosophy or one's state of mind to activity; it's the manifestation of meaning into experience and the interplay of the two that is deemed to yield happiness.

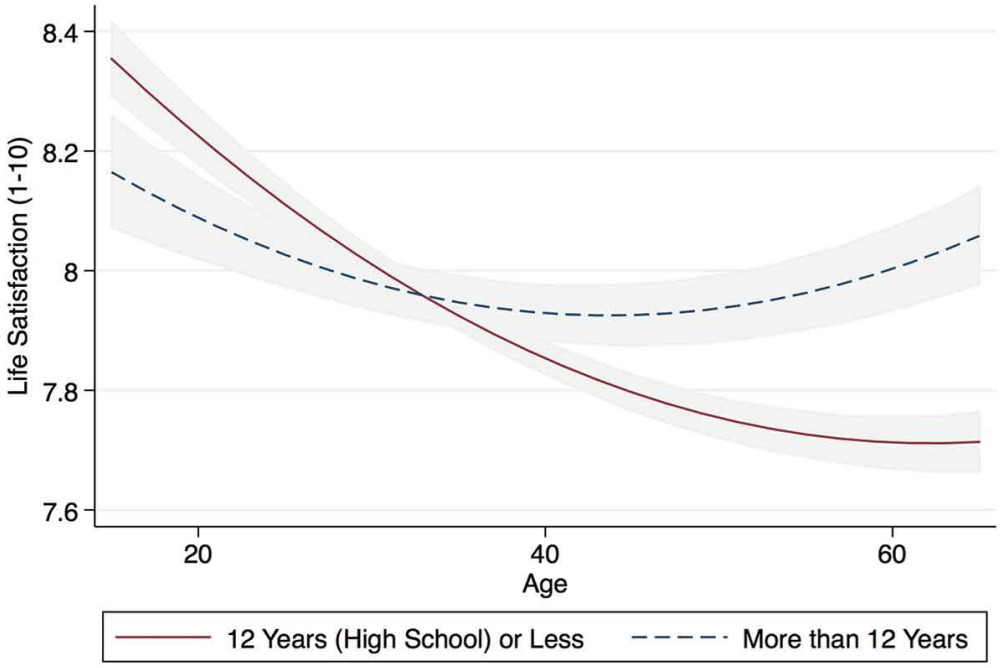


Figure 2: Figure showing the predicted life satisfaction (model based on data gathered on Australian population) at sample means for different ages, separated into two categories: people with more than 12 years schooling, and people with less than 12 years. [65]

As for correlation research, the relationship between income and life satisfaction, for example, has been shown to be relatively linear [48]. On the other hand, Figure 2 offers a deeper insight into the complexity of the happiness system. The difference in life satisfaction according to education level not only presents education as a dimension in the situation space,

but also offers insights into the time-variance of the system B. It can be seen that people with more than 12 years of education begin adulthood with lower life satisfaction, but over time, experience a higher level of life satisfaction compared to those with less than 12 years of education. Education no doubt plays an important role in income [65], which has been shown to increase life satisfaction [48], however, this by no means is the only contributing factor, if it were, then the life satisfaction of the more educated, below ~ 35 years old wouldn't be lower than those with lower-level of formal education. Education's fundamental value is that it *enables*. It gives us the tools to rationalise and understand the world around us, which ultimately, over time, leads to a greater exposure to ideas, and an agency over such ideas that could potentially be shaped to resonate with our own experience of the world.

There has also been wide research on the effect of personality on happiness. In particular, a study conducted by Anglim et al. which combines data from 465 datasets, including over 300,000 participants. This data is focused on the impact of personality traits, particularly extraversion and neuroticism, on individual subjective well-being over time [10]. The research suggests that personality traits have a sustained impact on happiness and well-being. However, the causal mechanisms and specific pathways through which personality influences well-being are complex and multifaceted and as a result, it's difficult to formalise the nature in which these personalities affect the manner in which happiness is obtained. Nevertheless, the study makes a strong case for the *existence* of a relationship between personality and happiness, and it is certainly a relation to consider and potentially investigate moving forward.

2.1.4 Project Specification

So, what have we learned from the discussion thus far, and where does it leave us?

Perhaps the most important resolution procured was the importance of embracing subjectivity regarding happiness. If no philosopher can agree upon what happiness consists of, then focus on a certain viewpoint only reduces the dimensionality of our understanding, it imposes a bias that only the holder of truth (no-one) can impose. By valuing and embracing the opinion of the individual, we instead allow for the characteristics of the collective understanding of happiness to surface and take shape. From the somewhat loose thought experiment derived from the relationship between education, life satisfaction, and age, but more conclusively, the success of LKM, indicated the importance of shifting perspectives in order to reach higher levels of happiness. It also showed that this shift in perspective could originate outside an explicit focus on happiness.

The philosophy space has outlined, discussed, critiqued and concluded upon an ether of ideas regarding happiness, and all that may that contribute to it. From Plato to Camus, from the Buddha to Nietzsche, we can harness the depth and breadth of all this philosophical discussion, to see which ideas resonate with individuals.

Finally, the interplay between activity, and idea, implied by Seligman's framework for positive psychology, which highlighted the importance of bringing the conceptual into the physical realm of experience. Without the implementation of ideas, happiness cannot exist, and for sustained change, the departure from intellectual discussion is crucial. To adhere to each of the above conclusions, a dual recommendation system is proposed. A recommender system that periodically (daily, or weekly) recommends an idea ('food for thought') as well as an activity to carry out during said period. The following set of points will draw from all that has been discussed into a set of design requirements for the recommender system that has just been suggested:

- A suitable set of data, for the 'idea' space that draws from the vast array of philosophical discussion, should be acquired. The nature of this data should be discussed and decided upon (found in the design section) with an emphasis on covering a variety of ideas.
- Likewise, a suitable set of data for the activity space must be obtained with consideration of the personality and geographical diversity that will be required. Both sets of data should suffice for a proof-of-concept, and they by no means should serve as the final set of data, the focus of this research is providing a technical framework for understanding happiness.
- Both sets of data should be reduced to a common plane which will allow for a more robust understanding of their relationship, this objective will also be crucial when implementing mathematical models for recommendation.

- Following feedback from the user (based on a suitably designed question), the recommender system should update its parameters to improve its suggestions, i.e. suggestions should not be fixed, and the model should be adaptive.
- Model parameters should be 'transparent' such that salient conclusions can be drawn regarding the resonance people feel towards certain ideas and actions, this will ensure that a collective understanding of happiness can be obtained.
- The model should attempt to capture, or account for the time-varying (non Markov), non-linear description of the happiness system B.
- The model should have design considerations for the 'cold start' problem in recommender systems [53], i.e. the recommender system should be recommending relevant ideas and actions prior to receiving any feedback from users. The initial recommendation (given to a new user) should improve as the feedback is received from users, and should harness the (hopefully) growing understanding of happiness as a result of its use.

The above items provide an actionable set of objectives that will be considered throughout the design of the recommender system. They account for the majority of the conclusions that have been drawn throughout the prior discussion, as well as respecting the overarching goal of this project, which is to provide a tool that individuals can use, as well as building a collective understanding of happiness.

As a slight diversion of thought, but as context for these objectives in the wider vision of this research endeavour, the ultimate goal, beyond this specific project, it to provide an app for users, in which, alongside recommendations, will serve as a diary, where they can record the ideas and actions in which have brought them happiness or unhappiness, all of which will be used by the recommender system to provide better suggestions. Analytics regarding patterns and the nature in which happiness is brought to each individual will help promote not only a collective understanding of happiness from a research point of view, but can be fed back to the user to allow for them to strengthen their individual understanding of their own happiness. Undoubtedly, the recommender system that will be designed will serve as the backbone for this vision.

Now, the next set of objectives outline objectives of lower priority, these will help enhance the model that is being designed, but by no means are crucial to this project as a proof-of-concept. The model, at least, should be designed to provide the flexibility to integrate the following enhancements into the model as future work.

- Provided any legal and ethical issues are considered, the model, especially when dealing with the 'cold start' problem, should take into account additional parameters such as gender, nationality, education level, age, and as aforementioned (see Section 2.1.3) most

importantly, personality. These will not only help enhance the initial suggestion and model initialisation, but will help provide a deeper insight into the effect of each on happiness, provided the objective for model transparency is met.

- The model should be able to trivially expand its set of recommendable actions and ideas, such that users can help contribute to the dataset by providing new actions and ideas that they have resonated with. Not only will this ensure that a wider area is covered by the idea and activity space, but helps provide a sense of community and communication with the tool.

Perhaps in its most fundamental form, this project will be laying the groundwork for a community project; through the collective consumption and meditation on philosophical ideas, paired with the implementation of those ideas through experience, a collective understanding of happiness will take shape.

2.2 Technical background

2.2.1 Mapping the philosophy space

To arrive at a two-dimensional map of philosophical texts, like the one sketched in Figure 3, will require the completion of two technical milestones:

- A high-dimensional embedding of the philosophical texts.
- Dimensionality reduction of embeddings onto a two-dimensional space.

The philosophy texts themselves are simply a set of words, with the underlying style, topics, and ideas that determine the structure, order, and nature of these words. A high-dimensional (of order far less than the text length, but far greater than two) fixed-length vector embedding of these texts will serve as a dimensionality reduction process, where the text is condensed to a representation of these topics, styles, and ideas. Furthermore, with the text now being represented numerically, it will be far easier to work with mathematical models.

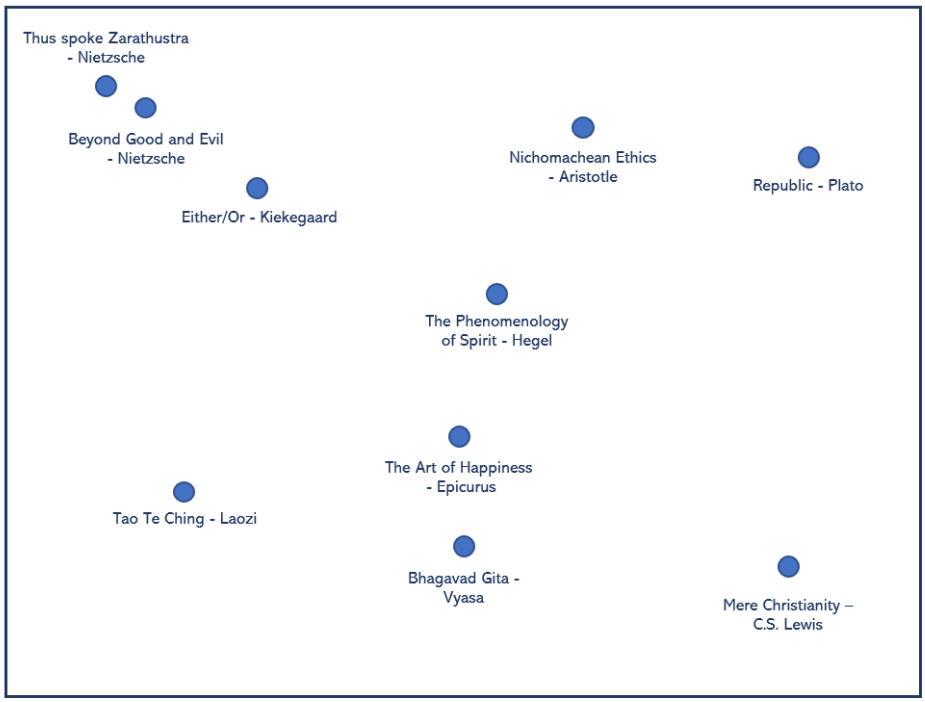


Figure 3: Figure showing a rough outline of what a map of philosophy texts may look like.

The reduction of the high-dimensional vectors on to the two-dimensional space allow for a more intuitive visual representation of the relationships between these texts, as seen in Figure 3. The dimensionality reduction of the embedding space will only serve for visual purposes, it will help guide the 'understanding' part of this project, however when making predictions, data will remain in the high-dimensional space to prevent loss of information, and allow the model to learn the complex patterns in the data.

2.2.2 Word-level embedding techniques

Averaged word-level embeddings can indeed perform well in sentiment analysis tasks on the sentence and document levels [80] [67], but also form the basis of more sophisticated techniques designed specifically for document-level corpora. The current most widely used, state-of-the-art technique for word-level encoding is Bidirectional Encoder Representations from Transformers (BERT) [27], which borrows from the influential Transformer architecture [103], all of which will be discussed in further detail.

The simplest of word embedding techniques is one-hot encoding, where a vector of the same length as the vocabulary of the corpus (number of unique words), has its values all equal to zero apart from the index corresponding to that particular word, where it's equal to one [95]. In this case, two semantically similar words such as 'sad' and 'upset' would be orthogonal to each other, and would entail the same cosine-similarity [82] as a semantically antonymous word such as 'happy'. Both BERT and Word2Vec, which will be discussed in this section, are effective in rectifying this. In general, semantically similar words are close in cosine-similarity, performing well in approximating a relationship such as the one outlined in equation 1 [62].

$$\text{'Paris'} - \text{'France'} + \text{'Italy'} = \text{'Rome'} \tag{1}$$

Word2Vec The Word2Vec model can be configured to one of two algorithms, both of which operate on the same principles but in an opposite way, the Continuous Skip-gram Model (CSG) and the Continuous Bag-of-Words Model (CBOW). The CBOW model aims to maximise the likelihood of the target word, given the context of the $n/2$ words before it and the $n/2$ words after it [62], demonstrated in Figure 4 and 5. Words in the vocabulary of the corpus are one hot encoded, and then matrix multiplied by $\mathbf{V}$ (to be learned), which is a matrix whose columns are formed by the vector embeddings of each word, $\mathbf{v}_{w_0}, \mathbf{v}_{w_1}...$, the result of the multiplication, because of one-hot encoding, is the learned embeddings of the corresponding context words. These are then averaged and then multiplied with another learned matrix, $\mathbf{U}$ whose rows are composed of the learned vectors $\mathbf{u_{w_0}}, \mathbf{u_{w_1}}...$, which produces a sort of 'score' matrix, with the higher score corresponding to the likeliest word, the softmax is then taken to assign probabilities to each word in the vocabulary.

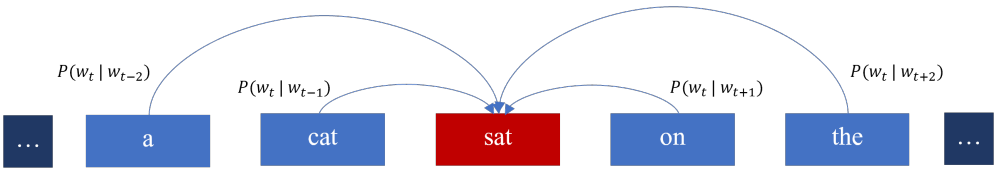


Figure 4: Figure demonstrating a CBOW example

The likelihood of the target word, given the context words around it, can be written as:

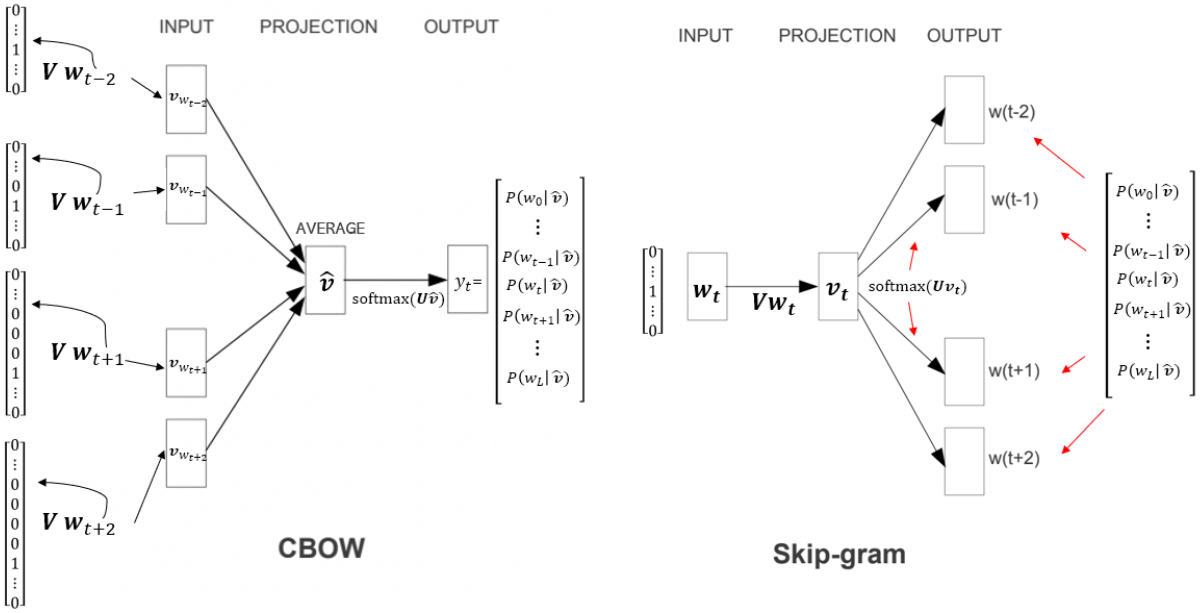


Figure 5: Diagram demonstrating the two Word2Vec architectures

$$L(\theta) = \prod_{t=1}^{T} P(w_t \mid w_{t-1}, w_{t+1}, .., w_{t+n}, w_{t-n} \; ; \; \theta) \tag{2}$$

Given the fact that the logarithm function is strictly monotonically increasing [12], such that $\arg\min_{\theta} \; L(\theta) = \arg\min_{\theta} \; \log(L(\theta))$, we can maximise the log-likelihood function and achieve the same value of θ. This transformation is performed to simplify calculations. The log-likelihood is then multiplied by -1 to turn it into a minimisation problem (this is the convention in optimisation). The cost function of CBOW can thus be written as:

$$J(\theta) = -\sum_{t=1}^{T} \log P(w_t \mid w_{t-1}, w_{t+1}, .., w_{t+n}, w_{t-n} \; ; \; \theta) \tag{3}$$

Where θ is a long vector containing the set of unknowns, the vectors that form $\mathbf{U}$ and $\mathbf{V}$, such that:

$$\theta = \begin{bmatrix} \mathbf{v_{w_1}} \\ \vdots \\ \mathbf{v_{w_L}} \\ \mathbf{u_{w_1}} \\ \vdots \\ \mathbf{u_{w_L}} \end{bmatrix} \quad \in \mathcal{R}^{2dL}$$

Now, with reference to Figure 5, noting that $\hat{\mathbf{v}}_t$ is the average of the vector representations of the context words at time t, it can be said that:

$$\begin{aligned} J(\theta) &= -\sum_{t=1}^{T} \log P(w_t \mid w_{t-1}, w_{t+1}, .., w_{t+n}, w_{t-n} \; ; \; \theta) \\ &= -\sum_{t=1}^{T} \log \frac{\exp(\mathbf{u}_{w_t}^T \hat{\mathbf{v}}_t)}{\sum_{j=1}^{L} \exp(\mathbf{u}_{w_j}^T \hat{\mathbf{v}}_t)} \\ &= -\sum_{t=1}^{T} \mathbf{u}_{w_t}^T \hat{\mathbf{v}}_t + \log \sum_{j=1}^{L} \exp(\mathbf{u}_{w_j}^T \hat{\mathbf{v}}_t) \end{aligned} \tag{4}$$

Stochastic Gradient Descent (SGD) is then used to update θ iteratively [19], according to equation 5. The remaining terms of the cost function are thus all the rows of the matrix $\mathbf{U}$ and the context vector representations that contribute to $\hat{\mathbf{v}}_t$ (see equation 4). This means that at a particular time instance t, each variable contained in θ is updated apart from those vector representations of the words outside the context window. The derivation of the analytic derivatives is long and tedious and thus is omitted from this report, but can be found in [94].

$$\theta_{\mathbf{t+1}} \leftarrow \theta_{\mathbf{t}} - \alpha \nabla_\theta J(\theta) \tag{5}$$

The CSG Model works with an almost identical but reversed workflow, it instead aims to maximise the likelihood of the context words according to a center word, as depicted in Figure. Either method can be used, or even the results of the two algorithms could be averaged. The differences in performance between these can be investigated upon implementation.

Some general remarks can be made regarding the properties of the Word2Vec algorithm from the above analysis. The first is that there is no consideration for order in the set of context words, meaning that some information can be lost because of this [61]. Another conclusion drawn is that this is an unsupervised algorithm, there is no need for labelled data, which is scarce for philosophical text data, and for this particular problem where there is no real ground truth. Larger corpora will lead to a better, more generalised representation of words (trained on more contexts), this will consequently increase the size of the vocabulary. The sizes of $\mathbf{U}$, and $\mathbf{V}$ are proportional to the size of the vocabulary L and the dimension of

the embedded vector d, for a vocabulary in the order of 10^4 (Oxford English dictionary has words in the order of 10^5 [2]) and an embedding vector in the 100s, this would result in 10^6 parameters that would are updated at each iteration. Another limitation is that there clearly isn't any way to handle out-of-vocabulary words. Given the simplicity of the architecture, this algorithm comes with some limitations, however, the task it is designed for (the semantic spatial vectorisation of text), matches well with the task at hand, and can serve as a good starting point.

Bidirectional Encoder Representations from Transformers (BERT) BERT is a far more sophisticated model compared to Word2Vec, it's a Bidirectional Transformer based architecture, which like Word2Vec considers bidirectional context words, but also takes their order into consideration. The network is traditionally pre-trained on two different tasks, Mask Language Modelling (MLM) and Next Sentence Prediction (NSP), in order to get a general understanding of language, it is then fine-tuned according to the specific problem [27]. MLM is a task that aims to predict randomly masked (hidden) words, NSP aims to distinguish the actual succeeding sentence from a randomly chosen sentence from the corpus [91]. Pre-training has proved effective in many Natural Language Processing(NLP) tasks, and just as importantly, it has been proven to speed-up learning significantly [105] [72]. Before understanding the architecture of BERT, and the MLM and NSP tasks, it is important to understand how the encoding layer of a Transformer works, since this module forms its foundation.

The transformer is depicted in Figure 6. During training, all n input words/tokens are passed simultaneously through an encoder, which is a stack of N encoder modules (the output of one feeds into the input of the next). Each of these modules has two submodules: a self-attention layer and a fully connected ReLU activation layer [103]. Both layers are followed by an "add and normalize" step—"add" refers to the residual connection of the input of each layer to its output, and "norm" refers to normalization, performed according to equation 6, where $\mathbf{v}$ is the encoded vector and γ and β are learned parameters [111].

$$\text{LayerNorm}(\mathbf{v}) = \gamma\frac{\mathbf{v}-\mu}{\sigma} + \beta \tag{6}$$

Input tokens are initially embedded using Byte Pair Encoding (BPE), a form of encoding that is based on what was originally a compression technique, where the most common pair of consecutive bytes of data is iteratively replaced with a byte that does not occur within that data [36]. They are then positionally encoded, this gives the encoder information on the ordering of the words. Positional encoding is implemented according to equations 7 and 8, where "pos" is the index of the embedded token in the sequence, "i" is the i^{th} dimension of the embedding space, and d_{model} is the dimension of the embedded token.

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right), \qquad PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{7}$$

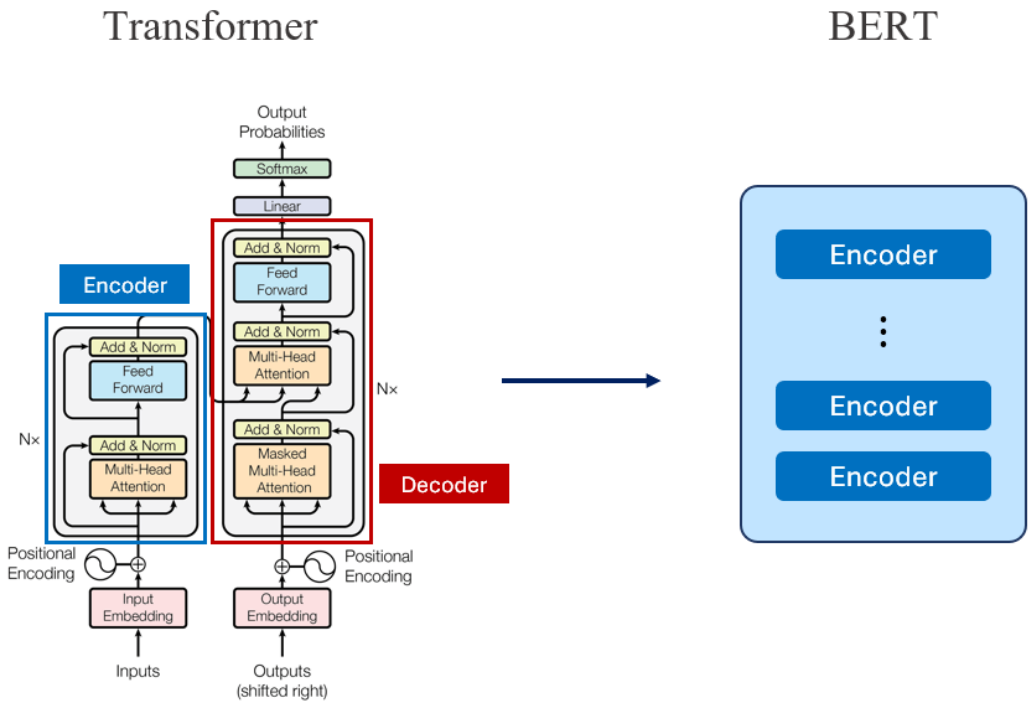


Figure 6: Diagram showing the BERT and Transformer architectures.

$$\mathbf{x}'_{pos} = \mathbf{x}_{pos} + \mathbf{PE}_{pos} \tag{8}$$

BP Encoded words are being offset by their corresponding **PE** vector. Each dimension of the embedding space is represented by a sinusoid, their wavelengths forming a geometric progression from 2π to $10000 \cdot 2\pi$. With reference to equation 9, PE values when offset by say, k positions, can be represented as a linear transformation of its original PE values, this allows the model to easily learn and attend to different positions [103].

$$\begin{bmatrix} \cos(\omega_i \cdot k) & \sin(\omega_i \cdot k) \\ -\sin(\omega_i \cdot k) & \cos(\omega_i \cdot k) \end{bmatrix} \cdot \begin{bmatrix} \sin(\omega_i \cdot pos) \\ \cos(\omega_i \cdot pos) \end{bmatrix} = \begin{bmatrix} \sin(\omega_i \cdot (pos + k)) \\ \cos(\omega_i \cdot (pos + k)) \end{bmatrix} \tag{9}$$

Perhaps the most defining and crucial feature of the transformer is the multi-head attention mechanism, which is a set of scaled-dot product attention layers in parallel, the outputs of which are then combined by concatenation and then multiplication with a learned 'blending' matrix [103]. The scaled-dot product attention is calculated according to equation 11.

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \tag{10}$$

$\mathbf{Q}, \mathbf{K}$ and $\mathbf{V}$ are referred to as the Query, Value, and Key Matrices, where $\mathbf{Q} = \mathbf{W}^Q\mathbf{x}$, $\mathbf{K} = \mathbf{W}^K\mathbf{x}$, $\mathbf{V} = \mathbf{W}^V\mathbf{x}$, and $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V$ are learned mappings and $\mathbf{x} \in \mathbb{R}^{d_{model} \times n}$ is formed by the embedded and positionally encoded input tokens.

$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{\mathbf{W}^Q\mathbf{x}\mathbf{x}^T(\mathbf{W}^K)^T}{\sqrt{d_k}}\right)\mathbf{W}^V\mathbf{x} \tag{11}$$

The term $\mathbf{x}\mathbf{x}^T$ is a matrix containing the dot products between the words, which, provided the Euclidean length of the vectors of each word are similar, is essentially measuring their similarity [7]. The $\mathbf{W}^Q$ and $\mathbf{W}^K$ matrices are learnt, and re-scale these similarity 'scores', learning and telling the model what other words in the sequence to attend to for each specific word embedding. Softmax ensures the weights of this heat map add up to one, to prevent exploding values. Finally, the resulting similarity score matrix is used to rescale the input embeddings via the learned matrix, $\mathbf{W}^V$.

$$\text{softmax}(x_i) = \frac{\exp(\mathbf{x}_i)}{\sum_{j=0}^{n}\exp(\mathbf{x}_j)} \tag{12}$$

Residual connections help propagate high-level information from previous sub-layers into the subsequent ones, but also allow gradients to propagate through the network upon backpropagation [74]. Normalisation maps features onto a similar scale, it prevents exploding weights and thus helps stabilise the network [8]. The matrix $\mathbf{x}$ is concatenated to the result of the multi-head attention submodule and passed through the feedforward submodule, which consists of 2 linear layers (equation 13), with ReLU activation (equation 14) in between [74].

$$F(\mathbf{x}) = \mathbf{W}^T\mathbf{x} + \mathbf{b} \tag{13}$$

$$\text{ReLU}(x) = \max(0, x) \tag{14}$$

The BERT architecture is simply a stack of N encoder modules, the base model, outlined in [27], for example, is a stack of 12 encoder modules. BERT makes some slight modifications to the transformer encoder architecture to handle sentence-level tasks, namely at the input representation stage. A special classification token ([CLS]) preludes input sequences (the 512 tokens that are passed in simultaneously), whose final output vector is a representation of the sequence as a whole, and is used for classification tasks [27]. Sentences are separated by a "[SEP]" token, which is also a part of the final output representation. Finally, the input embeddings that are fed into the first multi-head attention module, are composed of a summation of the initial token embeddings (using WordPiece subword segmentation [109]), learned segment/sentence embeddings, and the learned positional embeddings, as seen in figure [27]. BERT, in theory, is producing embeddings at 3 different levels, the token level, the segment (sentence) level, and the sequence (paragraph) levels.

Perhaps the most important novelty that BERT introduces is the power of pre-training the network on large, general datasets on the two tasks previously outlined, Mask Language Modelling (MLM) and Next Sentence Prediction (NSP). Since words have information regarding their position, and are being trained bi-directionally, if BERT were to be trained on a sliding window that masks every word, like Word2Vec, rather than learning the contextual

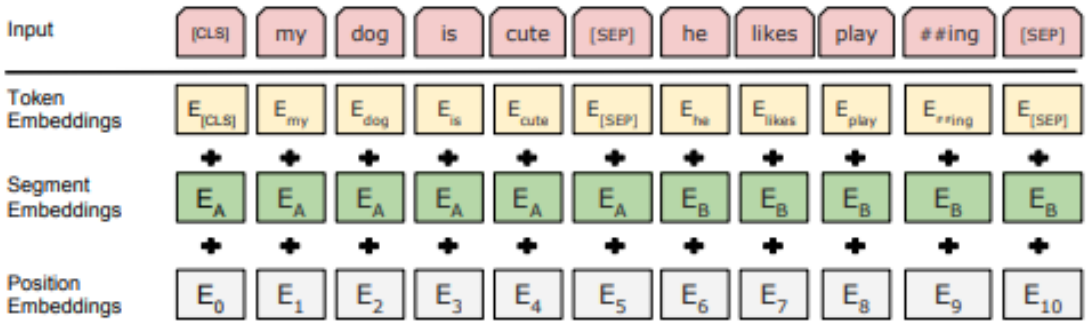


Figure 7: Diagram showing how input embeddings are constructed in the BERT architecture.

semantics of a word, BERT would learn how to extract the position of the word, i.e. it can predict the target word trivially [27]. To deal with this, MLM randomly select a token from the text and replace it with the "[MASK]" token 80% of the time, a random token 10% of the time, or leave the token unchanged 10% of the time. The "[MASK]" token does not always replace the randomly selected token to account for the fact that the "[MASK]" token does not appear during fine-tuning [27].

BERT derives the contextualised word embeddings based on the n tokens that are passed into it simultaneously, i.e. a particular word doesn't have a fixed embedding, this means that it is inherently dealing with polysemous words. Words are being trained simultaneously, and are able to understand the context in which they exist, meaning that word embeddings contain information regarding the set of tokens as a whole—this suits the problem of long philosophical text embeddings. Furthermore, BERT can deal with out-of-vocabulary words by representing them as subwords and characters [28]. Although the model is far deeper and more complex than Word2Vec, the number of parameters is independent of the vocabulary size and size of the dataset, therefore, it scales much better than Word2Vec to large datasets.

From a high-level perspective BERT serves as a trainable encoding block that can be pre-trained in an unsupervised manner, meaning that no labelled data is required. This quality, paired with its state-of-the-art performance in NLP tasks means that BERT is often used as a black-box module in a variety of tasks and architectures, and in the case of this project, paragraph and sentence-level embedding tasks [55] [9] [76].

2.2.3 Document/Paragraph level embeddings

Siamese BERT (SBERT) Despite the [CLS] token being likened to a paragraph-level embedding in the BERT architecture, it has no specific design considerations to implement this. As a result, it produces relatively poor sentence embeddings, often performing worse than averaged GloVe word embeddings [70] on sentence-level tasks [76]. SBERT fine-tunes BERT pre-trained weights on a siamese-triplet network [84], meaning that three sentences (two semantically similar sentences and one semantically opposite sentence) are passed into three

identical BERT networks with shared weights, which are then updated with a Triplet Objective Function(TOF) that aims to minimise the distance between similar sentences and maximise the distance between dissimilar sentences. Word embeddings are mean-pooled (proved to be more effective than using the [CLS] tokens or a max-over-time strategy [76]), the cosine-similarity between the output embeddings is computed, output to a 3-way softmax classifier that predicts between sentences-pair labels from the SNLI and MultiNLI datasets [16] [108]: *contradiction*,*entailment*, and *neutral*, model weights are then updated according to the TOF. SBERT is essentially updating BERT token embeddings such that when averaged pooled, semantically similar sentences are closer to each other in the embedded space.

SBERT embeddings are dependent on the pre-trained BERT parameters, meaning that embeddings will be relativised to the whole corpora it has been trained on, which is a huge dataset consisting of a plethora of text types (Wikipedia and BooksCorpus) [27]. This will likely lead to high cosine similarities between text embeddings, which can be rectified via normalisation techniques. SBERT will allow new texts of all types to be added to the dataset easily. It will be difficult to evaluate the performance of these two methods in the context of philosophical and activity representation, since there is no inherent ground truth and the space we are mapping to is somewhat abstract.

2.2.4 Dimensionality Reduction

This section will define a method that will enable the high-dimensional philosophical text embeddings to be reduced onto a two-dimensional plane, allowing for a more visually intuitive representation of the embedded vectors.

Principal Component Analysis (PCA) PCA is where data points are projected onto the n eigenvectors of the high-dimensional data's covariance matrix (15) that correspond to it's n largest eigenvalues [46]. Axes within the space spanned by the data are rotated to maximise the variance of the data, this set of rotated axes corresponds to the eigenvectors of the covariance matrix. A set of n of these axes are selected that maximise this variance, i.e. the axes corresponding to the largest eigenvalues. Data is projected onto the set of rotated axes, which preserves the maximum possible amount of information from the high-dimensional data [46].

$$\mathbf{R_{xx}} = \begin{bmatrix} r_{xx}[0] & r_{xx}[-1] & \dots & r_{xx}[-N] \\ r_{xx}[1] & r_{xx}[0] & \dots & r_{xx}[-N+1] \\ \vdots & \vdots & \ddots & \vdots \\ r_{xx}[N] & r_{xx}[N-1] & \dots & r_{xx}[0] \end{bmatrix}, \qquad r_{xx}(\tau) = \mathbb{E}\{x(i)x(i \pm \tau)\} \tag{15}$$

Eigenvectors are calculated by finding sets of λ and x that satisfy equation 16.

$$\mathbf{R_{xx}a} = \lambda \mathbf{a}, \qquad \mathbf{a} \in \mathbb{R}^d \tag{16}$$

PCA is a robust, well-defined method where the maximum amount of information from the original data is retained. New data-points can be added trivially by a simple projection onto the principal component axes.

Linear Discriminant Analysis (LDA) Linear Discriminant Analysis (LDA) can similarly be used for dimensionality reduction, its main caveat is that it's built on the assumption that all data is labelled into classes, i.e. in the case of philosophical texts this could be the school of thought, or the book at which the text is extracted from. Consider that n texts have already been embedded into d dimensional vectors, and each text is labelled with their school of thought, or author, giving K classes. LDA is designed to find a projection vector $\mathbf{w} \in \mathbb{R}^d$ that maximizes the between-class scatter while minimizing the within-class scatter [110]. The within-class scatter matrix $\mathbf{S}_w$ captures the sum of the within-class covariances for each data point, and for all classes, as shown in equation 17:

$$\mathbf{S}_w = \sum_{k=1}^{K} \sum_{i=1}^{n_k} (\mathbf{x}_i - \mathbf{m}_k)(\mathbf{x}_i - \mathbf{m}_k)^T \tag{17}$$

where n_k is the number of samples in class k, and $\mathbf{m}_k$ is the mean of class k:

$$\mathbf{m}_k = \frac{1}{n_k} \sum_{i=1}^{n_k} \mathbf{x}_i.$$

$\mathbf{S}_w$ captures the within-class scatter, but in order to maximise the between class scatter, the between-class scatter matrix $\mathbf{S}_b$ is now defined. It is computed as the sum of covariance matrices between class means:

$$\mathbf{S}_b = \sum_{k=1}^{K} (\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^T,$$

where $\mathbf{m}$ represents the overall mean vector:

$$\mathbf{m} = \frac{1}{n} \sum_{k=1}^{K} \sum_{i=1}^{n_k} \mathbf{x}_i.$$

The goal of LDA is to find the projection vector $\mathbf{w}$ that maximizes the cost function described by Fisher's discriminant criterion, given by:

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_b \mathbf{w}}{\mathbf{w}^T \mathbf{S}_w \mathbf{w}}.$$

$\mathbf{w}^T \mathbf{S}_b \mathbf{w}$ is a capture of the distances between class means, when projected onto $\mathbf{w}$, the larger its value the greater the between-class scatter. As for the term $\mathbf{w}^T \mathbf{S}_w \mathbf{w}$ this is a measure of the within-class scatter when projected onto $\mathbf{w}$, the smaller its value, the smaller the within-class scatter, or in Layman's terms, the data points in the same class are closer together, clearly maximising the cost function is optimising the problem definition.

The cost function is differentiated and set equal to 0, yielding the generalised eigenvalue problem, which when solved, gives the optimal $\mathbf{w}$:

$$\mathbf{S}_w^{-1} \mathbf{S}_b \mathbf{w} = \lambda \mathbf{w},$$

Where λ represents the eigenvalues and $\mathbf{w}$ is the corresponding eigenvector. The eigenvectors can then be ordered in descending order with respect to their corresponding eigenvalues. Projection onto the first n of these eigenvectors, reduces the data to n dimensions whilst minimising the within-class scatter and maximising the between-class scatter. This can prove to very powerful when attempting to assert the 'closeness' of philosophical texts in the 'idea' space, which correspond to the same author/school of thought, however, in the same vain there may be some unwanted bias towards same-class similarity, and may not be entirely reflective of the content of the text.

2.2.5 Recommender Systems: An overview

Now, to begin designing a recommender system that adheres to the outlined objectives, it is important to explore popular existing techniques to see where they fit within the framework of those objectives. Recommender systems have become undeniably intertwined with our every day lives, serving as the driver for the majority of our daily interactions on the internet, from e-learning [106], to tourism [37], movies and TV [13] [92] etc. Recommender systems can be broken down into two main approaches, content-based filtering, or collaborative-filtering methods, summarised in Figure 8.

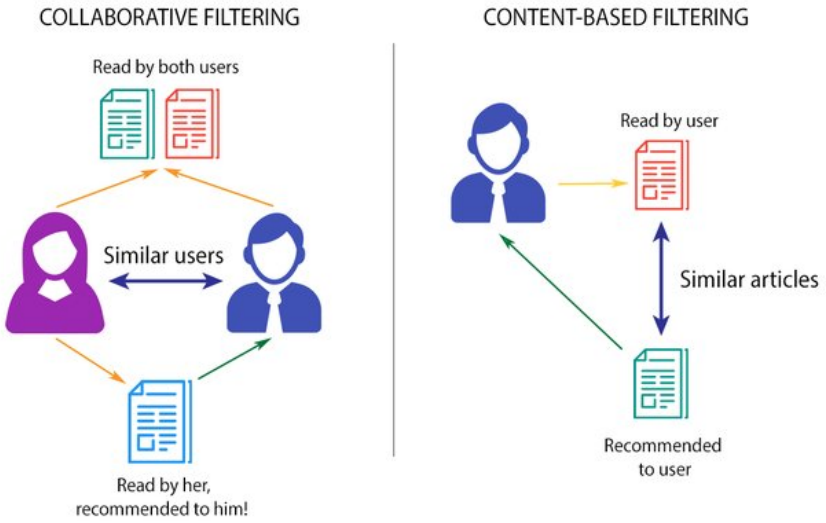


Figure 8: An overview of Content-Based Filtering and Collaborative Filtering recommender systems.

Content-based filtering In content-based recommender systems, items are grouped into profiles based on their features or descriptions. For example, books can be categorized by attributes like author and publisher, while movies can be categorized by director and actors. When a user rates an item positively, the system combines other items within that profile to construct a user profile, and recommendations are made based on content that match said profile [79]. This can be extended to the context of our project, and, for example, philosophical texts exist on the 2 Dimensional SBERT embedding space, with texts similar in semantics being closer together, when a user provides positive feedback for a particular text, texts closest in Euclidean distance could be recommended. This could equally be done with actions, if users react positively to a particular activity, then similar activities in its respective SBERT embedding space could be recommended.

The main limitation of this approach is its heavy dependence on detailed knowledge of item features, and the relevance of those features in providing good recommendations to the user. If applied to the context of our project, recommendations would be almost entirely dependent on the accuracy of SBERT embeddings, although auxiliary features such as author, release

year, or school of thought could be used to enhance recommendations. Now, there is room for adaptability of recommendations in such a system, for example, the positions of texts in the SBERT space can be adaptive, changing based on feedback from the user, however adaptability is not intrinsic to this approach. Since the model is basing recommendations solely on content, it offers no transparency to the dynamic between users and the idea and activity spaces. In this case, an understanding can be built using a separate technical pipeline, for example by logging interactions and performing data analytics to draw conclusions, however this is far from ideal with respect to the objectives of this project, where we would like the growing understanding, through increased interactions, to directly inform recommendations, and vice versa. The separation between the processes only allows room for the imposition of bias, which we want to steer clear of.

An additional limitation of the content-based approach is the lack of focus on 'exploration' of the idea space, recommendations, especially for an item space consisting of extremely large number of items. Recommendations will likely stay in the same region, preventing the exposure to 'new' ideas, especially in the context of the recommender system we are considering. To mitigate this, a stochastic decision can be made, with closer items being weighted higher in the probability distribution, or using something similar to ϵ-greedy policy in Q-Learning where a completely random recommendation is made with ϵ probability [1].Furthermore, unless a separate model is designed for an initial recommendation, content-based filtering has no design considerations for dealing with the 'cold start' problem, it is difficult to provide users with a good initial recommendation.

An advantage of this approach is that, provided there is a satisfactory framework for modelling content, which this method is dependent on, new items can be trivially appended to the item array, which satisfies one of the 'bonus' design requirements of the recommender system.

Collaborative Filtering Collaborative Filtering (CF) method is based on an almost opposite philosophy, where instead of modelling items, the interaction of users with items is modelled, and recommendations for new content is made according to the interactions of similar users [83]. Memory-based approaches to CF, to store user-item interactions in the user-item matrix, similar users are found by computing the similarity (e.g. cosine similarity) between the item interaction vectors of each user.One of the most widely used implementations of CF, which also provides an intuitive demonstration of its advantages and pitfalls, is the Matrix Factorisation technique, which was popularised after Simon Funk's blog post [35].

Matrix factorization decomposes the user-item rating matrix into two lower-rank matrices. Denoting the user-item rating matrix as, $\mathbf{R} \in \mathbb{R}^{m \times n}$ where m is the number of users and n is the number of items. Matrix factorization assumes the existence of latent factors or features underlying the user-item interactions, such that dimensionality reduction is possible whilst retaining sufficient information in order to make suitable recommendations [14].

Considering this, the user-item rating matrix $\mathbf{R}$ is decomposed into two lower-rank matrices of K latent factors, the user matrix $\mathbf{U}$ and the item matrix $\mathbf{V}$. The user matrix $\mathbf{U}$ has dimensions $m \times K$, and the item matrix $\mathbf{V}$ has dimensions $K \times n$. The user matrix $\mathbf{U}$ represents the user preferences for the latent factors, and the item matrix V represents the extent to which items possess the latent factors. Each entry in the user matrix, $\mathbf{U}_{u,k}$, represents the user u's preference for latent factor k. In the same manner, each entry in the item matrix $\mathbf{V}_{k,i}$, represents the extent to which item i possesses a latent factor k. In Funk's method, the matrices $\mathbf{U}$ and $\mathbf{V}$ are used to make predictions on ratings for the rating of user u and item i, according to equation 18:

$$r_{u,i} = \sum_{k=1}^{K} U_{u,k} V_{k,i} \tag{18}$$

$\mathbf{U}$ and $\mathbf{V}$ are then learnt such that the prediction error for the rating is minimised. This can be captured via a variety of loss functions, however Funk opted for a regularised absolute error loss between the matrix of predicted ratings $\hat{\mathbf{R}}$ and the actual ratings $\mathbf{R}$:

$$\min_{\mathbf{U},\mathbf{V}} \quad |\hat{\mathbf{R}} - \mathbf{R}| + \alpha||\mathbf{U}|| + \beta||\mathbf{V}|| \tag{19}$$

Where $||\cdot||$ represents the Frobenius norm [18]. The error can be minimised via a variety of iteration based optimisation techniques, e.g. Stochastic Gradient Descent. This objective function can be minimized using optimization algorithms, such as gradient descent, to update the entries in U and V iteratively. These optimised predicted ratings can then be used to recommend items to users based on those predicted preferences.

By examining the nature of the Matrix Factorisation algorithm, the pitfalls of such an approach become clear. The cold-start problem is especially significant for such an approach. Clearly, CF requires a significant amount of user-item interaction data to begin making accurate recommendations. Furthermore, when a new user or item joins the system and has limited or no interactions, there is no design consideration for making initial recommendations, a separate system would have to be used in order to make initial suggestions that are somewhat accurate. This is especially problematic in the context of our project, where there is no pre-existing user-interaction data.

As aforementioned, CF really suffers from problems associated with the sparsity of data. In the case of our model, which will likely have a large amount of users and items, and given that users typically rate or interact with only a small fraction of the available items, the sparsity of the user-item matrix will be almost guaranteed. An increasing sparsity of data makes it increasingly difficult to analytically extract patterns from the data, resulting in suboptimal recommendations, it also means that the computational complexity of operating such a model becomes huge. The system is also prone to shilling attacks, where malicious users or items can manipulate the recommendation process by providing fake ratings or artificially influencing the similarity measures. Such a model is also very limited in providing

suggestions for new or niche items because of the limited interactions.

There are, however, a number of positives associated with CF. In terms of the philosophy of the approach, it matches well with the goals of the project, which is to employ the collective wisdom in order to understand more about happiness. As a model based on collaborative learns, it is directly learning about the relationship between users and the content as it improves its suggestions. Since we know very little regarding the environment, CF is effected very little by this limitation, since embeddings are automatically learned. If such a model is to be used, the drawbacks of such a model that have been discussed must be addressed.

Deep-Learning, and Reinforcement Learning approaches Clearly, the two approaches that have been discussed are not mutually exclusive, modelling the item space can help inform and improve predictions that are rooted in CF, and vice versa. Deep Learning approaches to recommender systems have also become popular due to their high flexibility, ability to learn complex patters in the data, and automatic feature engineering. In the context of CF settings, deep neural networks are justified when there is a significant amount of complexity or numerous training instances. For example, using a multi-layer perceptron (MLP) to approximate the interaction function has shown performance gains over traditional methods like matrix factorization (MF) [21]. Aside from providing improvements to CF and Content-based methods independently, deep learning approaches, above all, provide the ability to arbitrarily combine relevant features, such as item-based features (SBERT embeddings for activity and ideas) and user-based features (user age, past interactions, gender etc.), in this way, information from both perspectives of the user-item paradigm in can be exploited in a 'hybrid' recommender system to provide better recommendations [64].

In the context of this project, supervised approaches will not be suitable due to the unavailability of pre-existing data. Taking this into, semi-supervised approaches must be considered, such that the model can be adaptively learnt as it interacts with the environment. Reinforcement Learning procedures offer a perfect solution to these requirements, offering promising results in recommendation tasks in dynamic environments which cannot be accurately can be modelled, as well as in cases where pre-existing labelled data is unavailable [21]. For example, in research done at Google, it is shown that RL can be used for more effective video recommendation on YouTube [20].

2.2.6 Reinforcement Learning for Recommender Systems (RLRS)

From the highest level of understanding, RL exists under the umbrella of semi-supervised Machine Learning techniques, where an agent (the recommender system in the case of our project) optimises its behaviour according to a closed-loop feedback signal, the reward function, which must be designed accordingly [77]. The RL problem is typically modelled as a Markov Decision Process (MDP), such that the state in which an agent finds itself in at time t is independent of the states it has found itself in the past, however, in the case of the happiness system, we have hypothesised that this is not the case, and there are ways to account for this, that will be discussed later. An RL algorithm, in the context of a recommender system (RLRS), can be best understood through the diagram below:

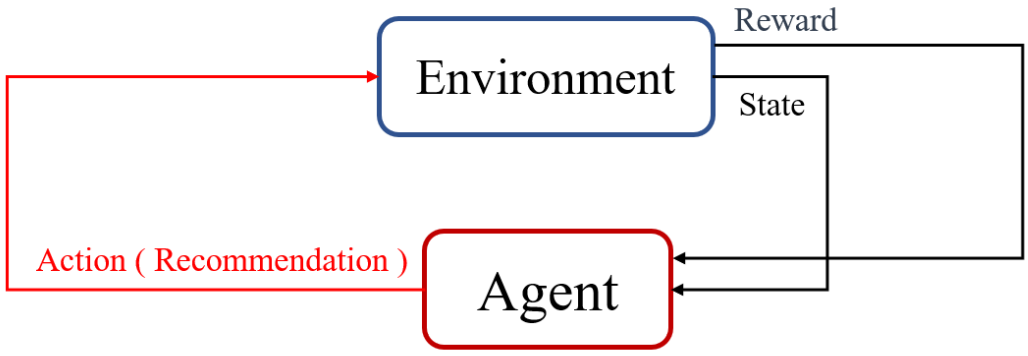


Figure 9: A diagram demonstrating a general RLRS.

More specifically, the main components of an RLRS are defined as follows [99]:

- **Policy:** Policy, denoted by π, which represents a probability distribution that defines the probability of taking action a out of the set of all probable actions $\mathcal{A}$ when in a state s. To contextualise this in our recommender system, the policy π defines the probability of recommending a particular activity and philosophical text, and the state is given by the current activity and text the user is putting into practice. RL algorithms are generally categorised into *on-policy* and *off-policy* methods. In *on-policy* methods, the RL algorithm attempts to improve the very policy that they are using to make decisions, which of course requires an initial model for the policy. In *off-policy* methods, the system aims to learn/evaluate a policy different from the one used to generate data. This is essentially the manner in which the agent picks a new action when in a given state.

- **Reward:** When an action is taken, i.e. a recommendation is made, the feedback (be that positive or negative) received from the user regarding the suggestion is captured via a suitably defined reward signal r, and is fed back to the model to improve the policy π.

- **Value Function:** The reward signal is an instantaneous feedback on the performance of the RLRS, the value function attempts to model the performance of the policy in the

long-run, this is particularly valuable to our project where we are focusing on providing a *sustained* happiness, the Value Function is a direct capture of that.

- **Model:** The model defines the environment in a way that predictions on the next state and reward can be made. If this available prior, it is learnt by the agent through experience [99].

Now that the general framework for a RLRS is defined, let's dive deeper into the different RL approaches to see which one best matches our project. The main approaches to RL problems are value-based iteration methods and policy based iteration methods.

Value-Based methods To begin with, Value-based methods are considered. Essentially, they are a class of algorithms used in reinforcement learning to estimate the optimal value function, and determine a policy that is derived from the optimal value function, for example by taking the action that yields the maximum Value (the maximum sum of future rewards). These methods aim to iteratively update the value function until convergence is achieved. The foundation of value-based iteration methods is the Bellman equation, which expresses the value of a state as the expected sum of discounted future rewards [29]. For a given policy π, the Bellman equation can be written as:

$$V^{\pi}(s) = \sum_{a} \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma V^{\pi}(s')\right]$$

where $V^{\pi}(s)$ represents the value of state s under policy π, $\pi(a|s)$ is the probability of taking action a in state s, and $p(s', r|s, a)$ is the transition probability of moving to state s' and receiving reward r after taking action a in state s, and γ is the discount factor which essentially serves as a weighting of current vs. future reward. Value iteration aims to find the optimal value function by iteratively improving an initial estimate. It starts with an arbitrary (randomly initialised) value function V_0 and updates it using the Bellman optimality equation:

$$V_{t+1}(s) = \max_{a} \sum_{s',r} p(s', r|s, a) \left[r + \gamma V_t(s')\right]$$

where $V_t(s)$ is the value function at iteration t. The algorithm continues iterating until the values converge, i.e., $\|V_{t+1} - V_t\| < \epsilon$, where ϵ is a small positive arbitrarily defined number. Clearly, a model is assumed here and the relevance of the Value function is dependent completely dependent on it, and is clearly unsuitable for our task. Instead, Q-Learning is a popular model-free value-based iteration method that directly estimates the optimal action-value function $Q^o(s, a)$, which represents the value of taking an action a in a state s, without requiring a model of the environment. Q-values are updated using the following rule, which is also derived from the Bellman Equation [107]:

$$Q_{t+1}(s, a) = Q_t(s, a) + \alpha \left[r + \gamma \max_{a'} Q_t(s', a') - Q_t(s, a)\right]$$

where $Q_t(s, a)$ is the Q-function at iteration t, α is the learning rate, r is the immediate reward received after taking action a in state s, s' is the resulting state, and γ is the discount factor.

Q-Learning iteratively updates the Q-values based on the observed rewards and transitions in the environment until convergence is reached. Now, exploration is not intrinsically part of the Q-Learning algorithm, actions are always chosen such that expected future reward is maximised, because of this, ϵ-greedy adjustment to the algorithm is employed, such that a random action is sampled with ϵ probability [1], this does help improve the coverage of the network, however the nature of the network does not allow for the implementation of more dynamic exploration strategies whilst maintaining stability. Q-Learning has proven to be sample inefficient, having slow convergence, and requiring numerous user-item interactions before reaching suitable results [114], this is a particular hindrance in the context of our problem where user-item are infrequent. Furthermore, since policy is entirely based on Value, this limits the ability of the model to learn the complexities of the policy, and therefore it lacks reliable guarantees of near-optimality for the derived policy from the value [51].

Policy Gradient (actor-only) methods Policy-Gradient methods such as Williams' REINFORCE essentially ignore the value function and work with parametrised policies, and optimise the cost (either the discounted reward or the average reward) over the parameter space of the policy [51]. As mentioned, first a parametrised policy, $\pi_\theta(a|s)$, is defined where θ is the set of parameters that define the policy. For the case of this example, the expected discounted reward is used as the cost function, and is defined as following:

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right], \tag{20}$$

r_t represents the reward received at time step t, and γ is the discount factor, as before. To update parameter values such that the expected reward is maximised, gradient ascent is performed. The gradient of the expected reward with respect to the parameters θ at a single time step is given by:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1} \hat{Q}(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t)\right], \tag{21}$$

Where $\hat{Q}(s, a)$ is an estimate of the Q-function when being in state s and taking action a. Gradient ascent is then applied to parameters (equation 22), such that convergence to local a local optima of the cost function is reached (provided some conditions of optimality are met).

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta J(\theta) \tag{22}$$

Where α is suitably chosen such that it satisfies the first order necessary conditions of optimality. Immediately, it can be seen that the manner in which policy-gradient methods learn is not completely independent of the value function, and convergence is still dependent on a suitable estimation of a value, e.g. $Q(s, a)$, however, the policy is directly being optimised to maximise the cost function, and so more is understood about the system that is operating within the environment. In the case of our problem, this would be getting an insight into the 'happiness system' B.

Where Q-Learning had a deterministic policy, taking the action with the highest Q-value, policy-gradient methods, which use a stochastic policy, inherently deal with the exploration-exploitation trade-off. Initially, the policy is random, this is where the model 'explores', however as the policy converges, the probabilities of optimal actions become closer-and-closer to 1, at which point the model begins 'exploiting'.

The main drawback of such an approach is the gradient must be estimated ($\nabla_\theta \log \pi_\theta(a|s)$ does not have an analytical solution), which often has a large variance [100]. The model also depends on a good Value estimation, which is certainly possible, but just how in value-based methods the policy was limited in its representation, so is the value of being in a certain state in policy-based methods. Actor-Critic methods aim to combing the strong points of both methods in a dual policy-value learning algorithm [51].

Reinforcement Learning: Actor-Critic Models Actor-Critic models combine Policy (actor) and Value (critic) based methods in a dual optimisation problem. The policy is parametrised as before, denoted by $\pi(a|s;\theta_\pi)$, and so is the Q-function (this could also be the value function if the problem is reformulated), denoted by $Q_{\theta_Q}(s,a)$, where θ_π and θ_Q are the set of parameters to be learnt. The job of the critic is to inform and supervise the actor, as demonstrated in figure 10.

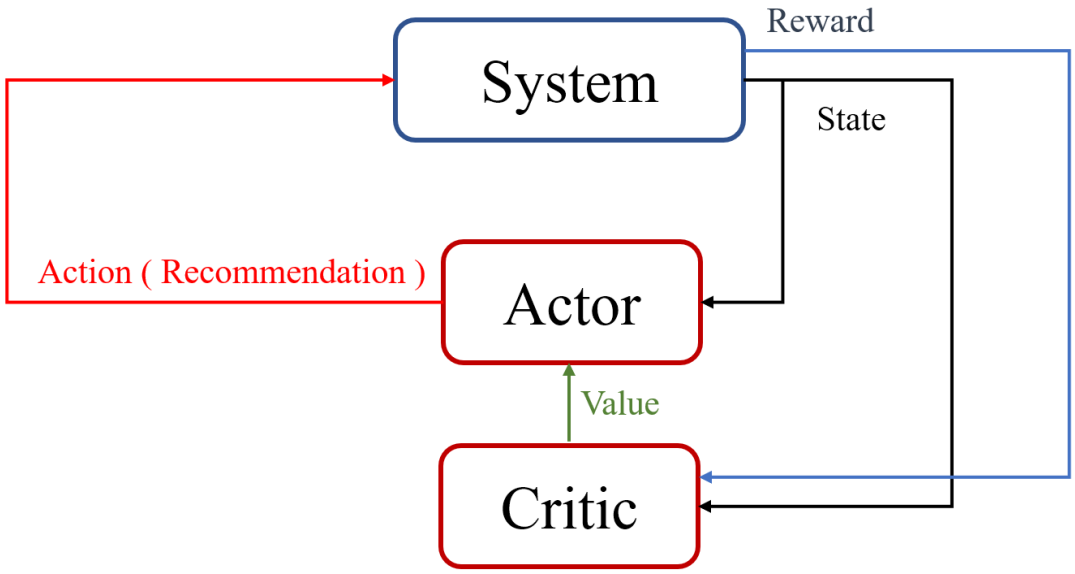


Figure 10: A diagram demonstrating the general pipeline of an AC learning algorithm.

The actor parameters θ^π are updated using the policy gradient, as in the previous section, except now the Q-function is learnt in a separate critic algorithm:

$$\nabla_{\theta_{\pi,t}} J(\theta_{\pi,t}) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1} Q_{\theta_Q}(s_t,a_t)\nabla_\theta \log \pi_\theta(a_t|s_t)\right], \tag{23}$$

$Q(s,a)$ can no longer be updated as before, since the policy is no longer taking the action that yields that maximum Q-value at the new state. Instead, the TD error [98] for the

action-value is computed:

$$\delta_t = r_t + \gamma Q_{\theta_Q}(s', a') - Q_{\theta_Q}(s, a) \tag{24}$$

Where a' and s' are found from following policy $\pi(a|s; \theta_\pi)$ in the new state s'. The TD-error is then used to update the critic parameters, such that:

$$\theta_{Q,t+1} = \theta_{Q,t} + \alpha_c \delta_t \nabla_{\theta_Q} Q_{\theta_Q}(s, a) \tag{25}$$

Of course, this method relies on the parameterisation of both the policy and the value/Q functions, however if this is available, the actor-critic framework helps combine the advantages of both policy and value-based approaches, i.e. increased stability, better sample efficiency, and a more natural exploration mechanism. To add to this, by directly parametrising and training both the policy and value functions, we maximise the transparency of the model, allowing us to get a clearer understanding of users and the items interact.

Discount Factor and Hedonism Since the Value (or Q function) function is a measure of future rewards, this means that by looking at equation 24 and especially at the term $\gamma Q_{\theta_Q}(s', a')$, it can be said that for a discount factor of close to 0, the agent becomes short-sighted, and thus places more importance on immediate reward, which is synonymous to a hedonic viewpoint of happiness. On the contrary, for a discount factor γ of close to 1, the agent becomes farsighted, and thus places a bigger importance on cumulitative future happiness, which can be likened to something closer to a focus on life-satisfaction.

Deep Actor-Critic Despite this, the same problems associated with the sparsity of the user-item space still apply to traditional RL methods, however, a major milestone in RLRS has been the implementation of deep learning methods, which have been able to perform well despite the huge state and action spaces present in recommender systems [77]. Deep Learning approaches allow for the flexibility of inputting a variety of features, ultimately allowing for better, more generalised predictions. Now, both the policy and value functions of the AC method can be parametrised via Neural Networks to help produce better results. As mentioned before, and as seen throughout the analysis of the RL methods, there is a dependency on the MDP assumption. In order to implement the time-dependency that has been hypothesised about the system B (and therefore of the recommender system), Recurrent Neural-Networks (RNNs), which have shown powerful results for time-series modelling and prediction [43] [24], are used. If deemed appropriate, RNNs can be used to model the time-dependency of user-preferences, and help boost model performance.

2.2.7 Recurrent Neural Networks (RNNs)

The main feature of RNNs that allows it to perform effectively in modelling non-markovian processes is the introduction of a circular feedback loop that allows for information from past-states to be propagated forward [73]. In this way, the feedback loop establishes a sense of memory in the system. Figure 11 provides a visualisation of this feedback loop, and how it influences future outputs.

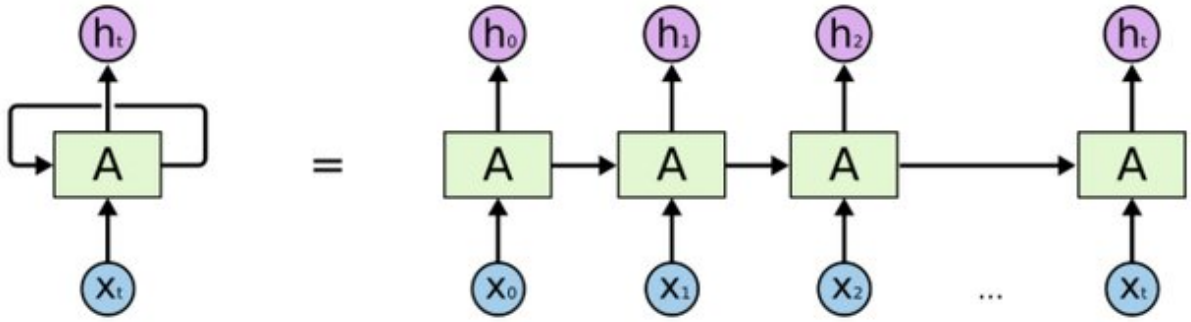


Figure 11: Diagram showing an unrolled RNN [73]

A forward pass through the RNN is characterised by the set of equations 26-29 [38], where $\mathbf{x}_t$ is the input vector at time t, $\mathbf{b}$ and $\mathbf{c}$ are bias vectors, $\mathbf{W}$, $\mathbf{U}$, and $\mathbf{V}$ are trainable weight matrices, and $\hat{\mathbf{y}}_t$ is the resulting 'activated' output of the network. An activation function applies a non-linear transformation to a vector, such that the non-linearities in the data can be modelled [90]. The activation functions σ_1 and σ_2 must be chosen accordingly according to the problem at hand, for example, in classification problems, softmax activation would be used for σ_2, such that a set of probabilities is outputted. As for σ_1, this activation defines the extent of which the hidden state of the current iteration is 'remembered', and is typically chosen to be the hyperbolic tangent.

$$\mathbf{a}_t = \mathbf{b} + \mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t, \tag{26}$$

$$\mathbf{h}_t = \sigma_1(\mathbf{a}_t), \tag{27}$$

$$\mathbf{o}_t = \mathbf{c} + \mathbf{V}\mathbf{h}_t, \tag{28}$$

$$\hat{\mathbf{y}}_t = \sigma_2(\mathbf{o}_t), \tag{29}$$

To put this into the perspective of our problem, in an actor-critic model, for the actor network $\hat{\mathbf{y}}_t$ will be a set of probabilities assigned to each possible action, where the critic network will output a single real number that represents the Value (or Q-value). The set of parameters, θ_π and θ_Q are defined by the $\mathbf{W}$, $\mathbf{U}$, and $\mathbf{V}$ matrices. The weights of each of these algorithms are learnt via the backpropogation algorithm, which uses the fact that the output of each node of the network is dependent on the previous layers, and this chain of dependency can be used to compute the gradients of the error with respect to the weights and biases via the chain rule. This is then used to perform gradient descent on each of the parameters, which is of course dependent on an appropriately defined cost function.

Despite this, vanilla RNNs have one major flaw, and this is the vanishing or exploding gradient problem in long sequences [38] [42]. To understand how this problem arises, the calculation of the hidden layer is simplified to:

$$\mathbf{h}_t = \mathbf{W}\mathbf{h}_{t-1} \rightarrow \mathbf{h}_t = \mathbf{W}^t\mathbf{h}_0 \tag{30}$$

Now, assuming that $\mathbf{W}$ admits eigendecompostion it follows that:

$$\mathbf{W} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T \tag{31}$$

Since $\mathbf{Q}$ is orthogonal, the recurrence relation can be further simplified to:

$$\mathbf{h}_t = \mathbf{Q}\mathbf{\Lambda}^t\mathbf{Q}^T\mathbf{h}_0 \tag{32}$$

The eigenvalues that compose the diagonal matrix $\mathbf{\Lambda}$, are raised to the power of t, causing the eigenvalues with magnitude less than one to decay to zero and the eigenvalues with magnitude greater than one to explode to infinity [38]. Any component of $\mathbf{h}_0$ that does not correspond with the largest eigenvector will eventually be discarded. In fact, it was shown that in order for the RNN to store memories in a manner that is robust to small perturbations, the model inevitably must enter a region of the parameter space where gradients either vanish or explode [42]. The LSTM (Long Short Term Memory) was designed with a topology that directly addressed this vanishing gradient problem. This was done by introducing internal loops, that helped produce paths for gradients to propagate [43], the schematic of which is shown in Figure 12.

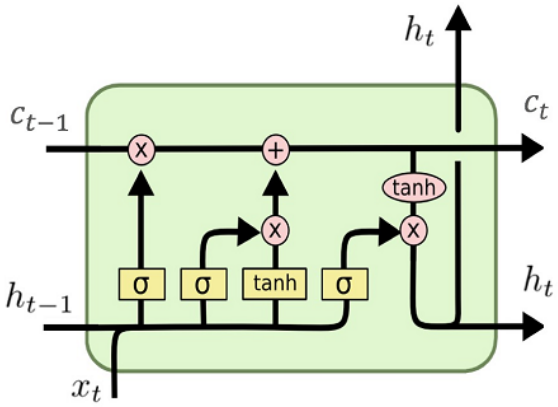


Figure 12: A visualisation of an LSTM model [66]

The key characteristic of the LSTM is the cell state (c_t) that propagates information from previous states with only minor linear interactions. The remainder of the LSTM is composed of three gates that control information flow, shown in Figure 13. Diagram (1) shows the 'forget' gate, here the model decides which features from the previous output to 'forget' and which to propagate forward. The input gate, signified by Diagram (2), decides which information is to be stored in the cell state such that it is propagated forward. Finally, the output gate (Diagram (4)) combines information from the cell-state $\mathbf{c}_t$, the output of the previous layer $\mathbf{h}_{t-1}$, and the input $\mathbf{x}_t$ to produce an output.

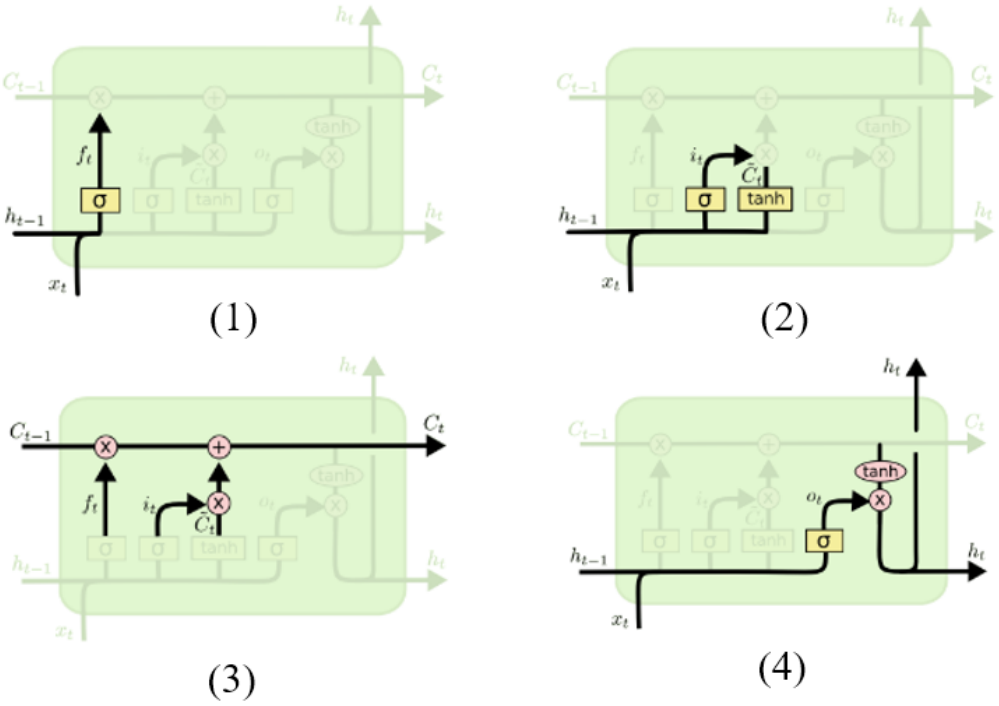


Figure 13: A decomposition of a forward pass in the LSTM model [66]

With reference to Figure 13, the output of the forget gate at time instant t is given by $\mathbf{f}_t$ where:

$$f_t^l = \sigma\left(b_f^l + \sum_j \mathbf{U}_f^{(l,j)} x_t^j + \sum_j (\mathbf{W}_f^T)^{(l,j)} h_{t-1}^j\right) \tag{33}$$

where $\mathbf{x}_t$ is the current input vector and $\mathbf{h}_t$ is the current, hidden layer vector (output of the previous layer), and $\mathbf{b}_f$, $\mathbf{U}_f$, $\mathbf{W}_f$ the biases, input weights and recurrent weights for the forget gate respectively. Note that superscripts l and j represent indexes of their respective vectors/matrices. Again referencing Figure 13, the LSTM cell internal state is updated as follows:

$$c_t^l = f_t^l \cdot c_{t-1}^l + i_t^l \cdot \sigma\left(b^i + \sum_j \mathbf{U}^{(l,j)} x_t^j + \sum_j \mathbf{W}^{(l,j)} h_{t-1}^j\right) \tag{34}$$

where b, $\mathbf{U}$, $\mathbf{W}$ denote the biases, input weights and recurrent weights of the input gate respectively, and i_t^l is the output of a similar computation to the forget gate (can be seen in Figure 13). Within the cell state c_t^i, described in equation 34, the gradient of the state with respect to the state from the previous layer $\frac{\partial \mathbf{c}_t}{\partial \mathbf{c}_{t-1}}$ is the only propagation of gradient through time, and by considering the change of their values with respect to each other, the problem of the vanishing gradient can be investigated, noting that the second term in equation 34 independent of $\mathbf{c}_{t-1}$ it thus follows that:

$$\frac{\partial c_t}{\partial c_{t-1}} = \frac{\partial f_t c_{t-1}}{\partial c_{t-1}} \tag{35}$$

$$= \frac{\partial \left[\sigma\left(b_f + \sum_j \mathbf{U}_f^j x_t^j + \sum_j \mathbf{W}_f^j h_{t-1}^j\right)\right] c_{t-1}}{\partial c_{t-1}} \tag{36}$$

$$= c_{t-1} \cdot \frac{\partial \left[\sigma\left(b_f + \sum_j \mathbf{U}_f^j x_t^j + \sum_j \mathbf{W}_f^j h_{t-1}^j\right)\right]}{\partial c_{t-1}} + \tag{37}$$

$$\frac{\partial c_{t-1}}{\partial c_{t-1}}\left[\sigma\left(b_f + \sum_j \mathbf{U}_f^j x_j^{(t)} + \sum_j \mathbf{W}_f^j h_{t-1}^j\right)\right] \tag{38}$$

$$= \sigma\left(b^f + \sum_j \mathbf{U}_f^j x_t^j + \sum_j \mathbf{W}_f^j h_{t-1}^j\right) \tag{39}$$

and thus finally, for $t' >> t$ it follows that:

$$\frac{\partial c_{t'}}{\partial c_t} = \prod_{k=1}^{t'-t} \sigma\left(b^f + \sum_j \mathbf{U}_f^j x_t^j + \sum_j \mathbf{W}_f^j h_{t-1}^j\right) \tag{40}$$

This shows that the gradient of the cell-state which is propagated through time is only dependent on the output of its forget gate, this gives the network far more control over the gradients in which it is propagating through time. It also means that preventing vanishing and exploding gradients is a part of what the LSTM unit is optimising, and for this reason, the model is far more robust to this problem [68]. This ultimately allows for the model to be more stable, and to converge faster. Now, relating the LSTM unit back to our problem, the input vector can be parametrised by the current recommendation, as well as additional features, the hidden state h_{t-1} corresponds to the previous action taken (i.e. the previous item recommended), and c_t is an abstract memory cell state.

The Gated Recurrent Unit (GRU) adopts a similar topology to the LSTM, but has been shown to have the same or better performance than the LSTM with fewer parameters [22], the schematic of which can be seen in Figure 14. With reference to the schematic, the hidden state of the GRU is given by:

$$h_t^i = z_{t-1}^i h_{t-1}^i + (1 - z_{t-1}^i)\sigma\left(b^i + \sum_j \mathbf{U}^{i,j} x_t^j + \sum_j \mathbf{W}^{i,j} r_{t-1}^j h_{t-1}^j\right) \tag{41}$$

Where,

$$z_t^i = \sigma\left(b^{i,u} + \sum_j \mathbf{U}_u^{i,j} x_t^{(j)} + \sum_j \mathbf{W}_u^{i,j} h_t^j\right) \tag{42}$$

$$r_t^i = \sigma\left(b^{i,r} + \sum_j \mathbf{U}_r^{i,j} x_t^j + \sum_j \mathbf{W}_r^{i,j} h_t^j\right) \tag{43}$$

The reset gates (r_t^i) are essentially controlling the flow of state information h_{t-1}, and therefore dictating its contribution to the next state h_t, introducing an additional, learnable, nonlinear transformation between past state and future states [38]. The main design disparity between the LSTM and GRU is that in a GRU, all temporal information flow is held in one vector, and thus only a single gating unit controls this temporal flow. For its comparable performance, and its superiority in terms of model complexity, the GRU will be preferred to the LSTM.

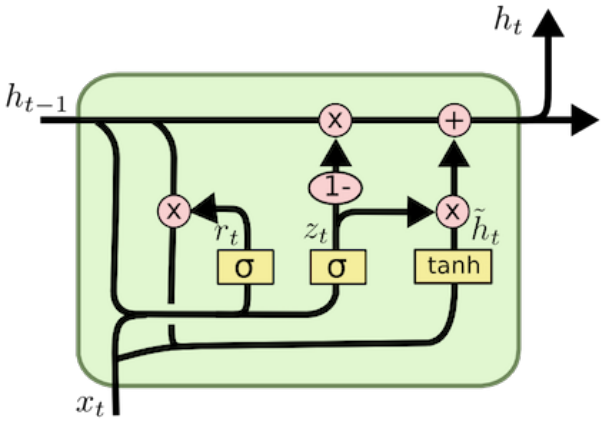


Figure 14: Schematic of a GRU [113].

2.2.8 Measure of Subjective Happiness

In the aims of the project, we have outlined and argued for a focus on a subjective measurement of happiness, therefore we will focus on investigating existing measures that focus on this. One of the most widely adopted scales is Lyubomirsky et al.'s Subjective Happiness Scale (SHS) [57].

The SHS consists of four straightforward questions that capture different aspects of subjective happiness. Individuals rate each item on a seven-point Likert scale [47], ranging from 1 (strongly disagree) to 7 (strongly agree). They are as follows:

1. In general, I consider myself a very happy person.
2. Compared to most of my peers, I consider myself less happy.
3. Some people are generally very happy; they enjoy life regardless of what is going on, getting the most out of everything. To what extent does this characterization describe you?
4. Some people are generally not very happy; although they are not depressed, they never seem as happy as they might be. To what extent does this characterization describe you?

To calculate an individual's subjective happiness score, the ratings for items 1 and 3 are summed, and the ratings for items 2 and 4 are subtracted. Unsurprisingly, higher scores indicating higher levels of subjective happiness. In fact, despite its brevity and its somewhat vague questions, the measure has shown strong positive correlation with more objective measures of happiness [59] [56] and showed high internal consistency as well as stability across all samples [56].

3 Recommender System Design

Now that the appropriate background for the project is set, and consolidating all that has been discussed thus far, the design for the architecture of our recommender system is now established, which can be visualised in Figure 15 and **??**. From a high-level perspective, this is a dual recommender system, an 'idea' and an 'activity' are being recommended to the user, however, these are not independent recommendations, and the 'activity' recommendation serves as the input state for the idea recommender system. This is such that a dependency between the two is asserted, and provided that the feedback that is asked of, and received from the user, is indicative of the relationship between the two, then the network will learn to find, and recommend the 'resonant' points between the two, inspired by the positive psychology of Seligman [89] [88].

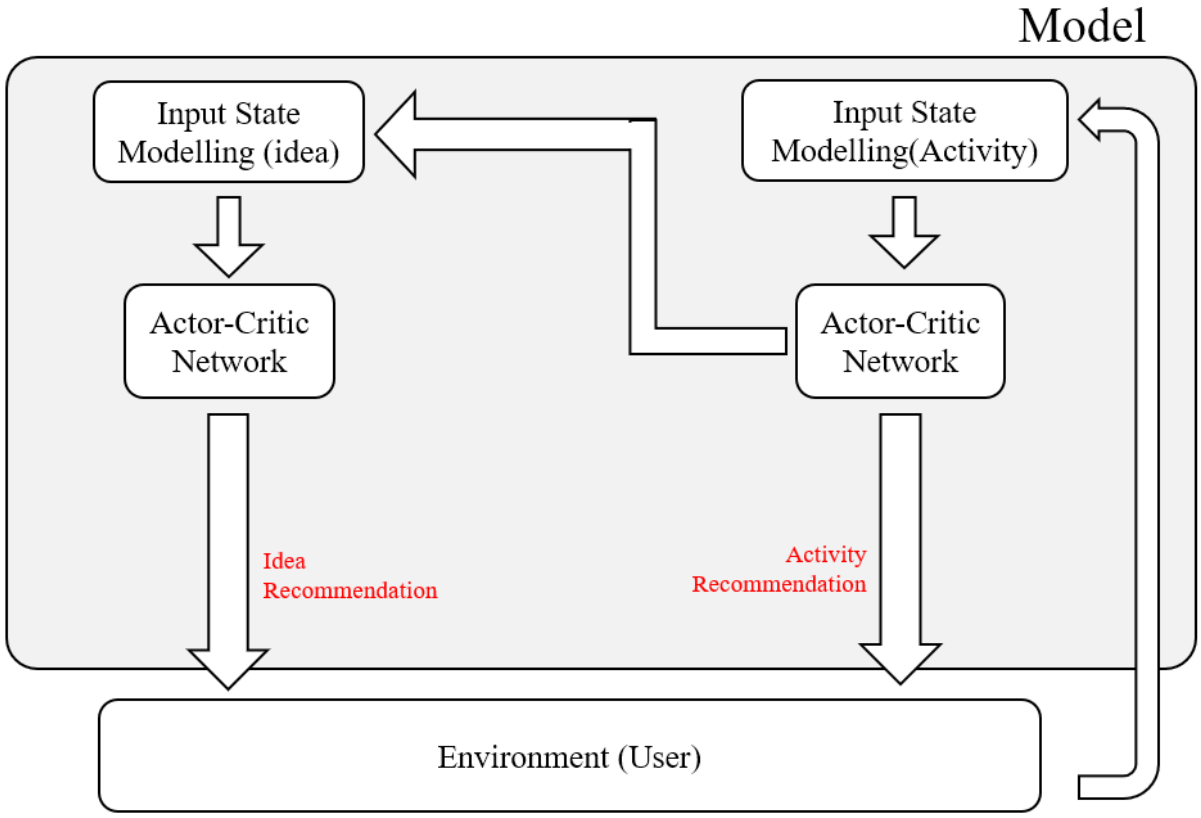


Figure 15: Diagram showing the design of the recommender system architecture.

This is not to affirm Seligman's theory as true, however, investigating both spaces will only provide better coverage in terms of our understanding, and there may indeed be something of value to be learnt about the interplay between thought and experience in terms of our happiness. It's worth noting that this could be reversed, an idea could be recommended, and then an activity, however, as a design decision, it was deemed more suitable to have ideas supporting actions, rather than actions supporting ideas. In fact, it could be that both pipelines are implemented to see which design architecture performs better. For the sake of this project, however, the pipeline in Figure 15 is implemented.

In Figure 15, there's an assumption that the 'idea' and 'activity' spaces have been defined, however, this will be the first task for implementation. It must first be decided the nature of the data that will be collected, i.e. the length of philosophy text, in what format the text will be in (raw philosophy text or summarised philosophy text). This will have to be a matter of discussion, and should take into account the logistics for the user, and the fact that the model is data-hungry (interactions need to be frequent).

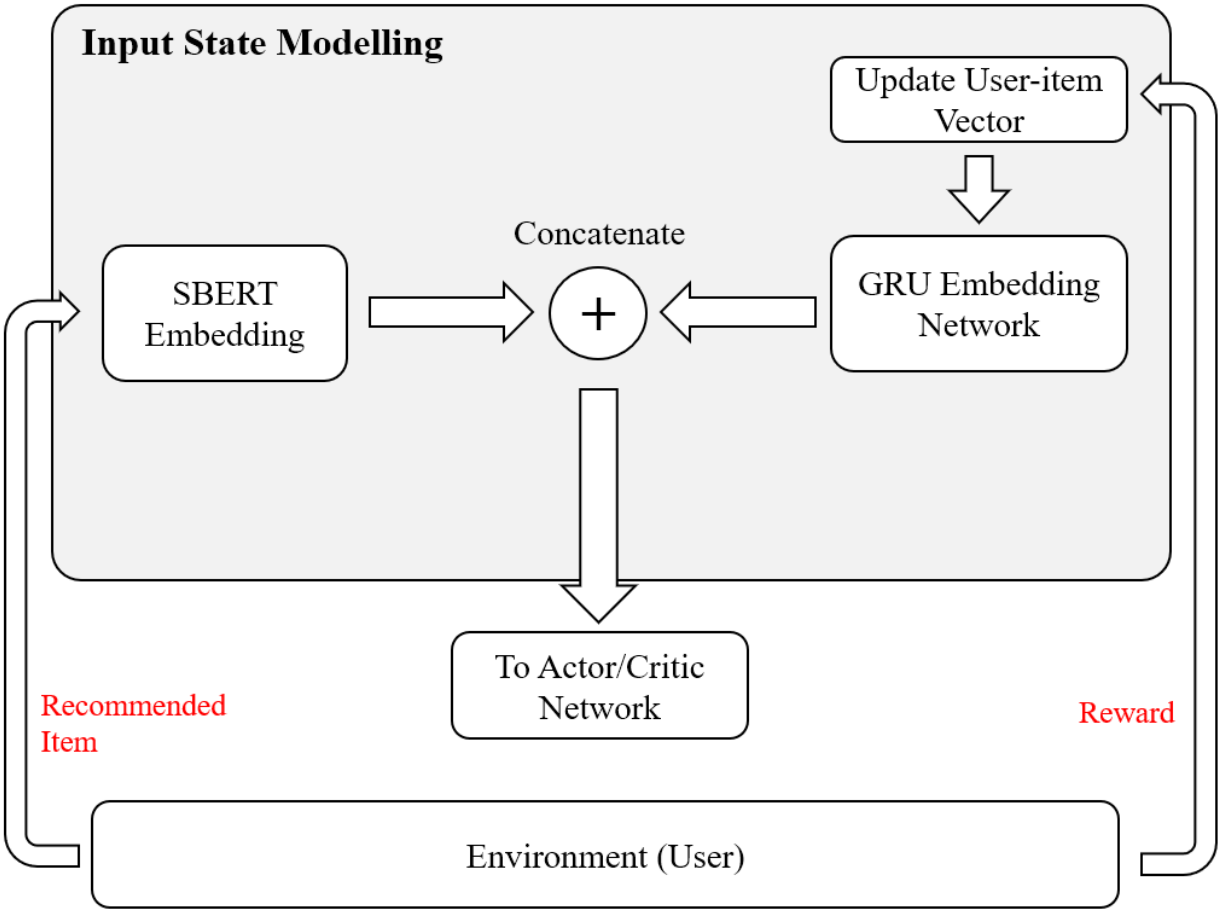


Figure 16: Diagram showing the design of the Input state modelling pipeline.

Once the activity and idea space are mapped, the model can be implemented as designed in Figure 15. The model begins with a forward pass based on pre-training that should address the 'cold-start' problem, and will be discussed in detail in the following section. Prior to input into the Actor-critic models, the state space is modelled, and is done so according to the schematic seen in Figure 16. This stage of the model aims to combine both the Collaborative Filtering, and Content-based modelling of the user and item spaces, as discussed in Section 2.2.5. SBERT embeddings of items provide a dense, vectorised representation of items in the item space, and will allow for robustness to a growing item space. As for user-modelling, a model that adopts the same philosophy as the MF is adopted, where instead of the whole user-item matrix, the i^{th} column of the user-item matrix $\mathbf{R}$ (corresponding to the considered

user i) in input into a GRU network, that embeds the user-item vector into a fixed, low-dimensional vector that is able to capture the temporal-dependency of user-interactions with the item space [115]. In this way, users with similar preferences will be mapped to similar output vectors.

To begin with, the embedded user and item vectors for activities are obtained, and concatenated, and then passed through the Actor network, which consists of a simple densely connected neural network, where a forward pass corresponds to following the policy $\pi_1(a|s;\theta_{\pi_1})$ parametrised by the Activity Actor Network. The result of the actor network yields a probibility distribution for the action space, and an action is sampled from that probability distribution, which corresponds to a recommendation for an activity for the user to undertake. This result is embedded, and concatenated with the idea, user-item embedding, and serves as the input to the idea actor network. The resulting input data defines the state for the 'idea' RL algorithm. The input data is then passed through the idea actor network, which corresponds to following the policy $\pi_2(a|s';\theta_{\pi_2})$, parametrised by its corresponding network. An action is again sampled from the resulting probability distribution, which now corresponds to an 'idea' being recommended.

These recommendations are then presented to the user, who provides feedback regarding his reaction to each of these recommendations and the extent of their complementarity. This feedback is then interpreted and stored in two appropriate reward functions, one for the 'idea' AC network, and one for the 'activity' AC network. After n interactions (n forward passes) the stored state, action, and reward data is used to compute a loss function that is defined by the observed rewards, and the predicted value from the critic network. This loss is then backpropagated through the network and such that the parameters of each network are updated as to maximise the future expected reward, or in the case of our system, happiness.

4 Implementation

4.1 Data Collection and Preparation

As a part of discussion regarding the nature of recommendations, it was decided that recommended philosophical texts must be 'bitesize', i.e. with a length ranging from a sentence to a couple paragraphs. This is such that the interaction with the user can be daily. To recommend a whole chapter, or even a whole book, was decided to be too arduous of a task for a user, and would not only potentially deter future interactions, but mean that interactions would be too infrequent for the model to sufficiently learn.

With this in mind, some investigation was conducted into data collection. Since the data analysis of philosophy has not been widely researched, there exists very few pre-processed datasets for such a task. To make matters worse, most publicly available philosophy book databases such as Cambridge Core eBooks and Internet Archive [4] [3] contain books that are either not available for download or are in a scanned PDF format, where data can't be extracted. Project Gutenburg(PG) however has a free and somewhat extensive collection of philosophy books, and also provides the books in plain-text format, [5].

The goal of this project is certainly not to overwhelm the reader with overly-complicated and technical philosophical texts, the tool that is being built should be able to be used and enjoyed by the majority of people, not simply by experts in philosophy. To account for this, the list of the most popular philosophy books on the community book review and discussion website, *GoodReads*, was referenced, checked if available on PG, and downloaded if so. If a book is being widely read, it ensures at least some degree of universality regarding its contents. The most popular philosophy books are often based on western philosophy, thus the most popular books from other regions were searched for and included, to ensure a wide spread of ideas in the philosophy space that will be mapped later.

After some preliminary data was obtained, some cleaning of the data was required to get it in a suitable string format. Firstly, prefaces, introductions, and the user agreement licence is removed from the text file, this is text that is not part of the original book and may skew the resulting embedding vector away from a representation that reflects its contents. Once this is done, the data will, for example, look like the following snippet from Nietzsche's *Beyond Good and Evil*:

```
CHAPTER I. PREJUDICES OF PHILOSOPHERS\n\n\n1. The Will to Truth,
which is to tempt us to many a hazardous\nenterprise, the famous
Truthfulness of which all philosophers have\nhitherto spoken with
respect, what questions has this Will to Truth not\nlaid before
us! What strange, perplexing, questionable questions! It
is\nalready a long story;
```

All the text files from PG were in a similar format, titles, and paragraphs are separated by

two, or more, new-line symbols, and lines are separated by one new-line symbol. Data is split at any point where there are two new-line symbols, this produces a list containing separated titles and paragraphs. The beginning of each chapter, for each text, has its own specific signature, and so data was separated into chapters as to separate the 'ideas' in each text. The beginning of propositions in each chapter sometimes began with numbering (Arabic or Roman numerals), so this needed to be stripped from the data as well. The data cleaning process was implemented as a class in Python to help accelerate the process, and ensure new data could easily be added to the database.

At this point, the text was still far too long to meet our 'bitesize' requirement, to condense the ideas in the chapter, an extractive summary network was sought to be selected and used. A number of different networks were tried, with the Facebook AI's BART network [54] providing the best results from a qualitative standpoint. BART is a transformer-based sequence-to-sequence model, with a BERT-like encode and GPT-like decoder, that has been fine-tuned on extractive summary tasks.

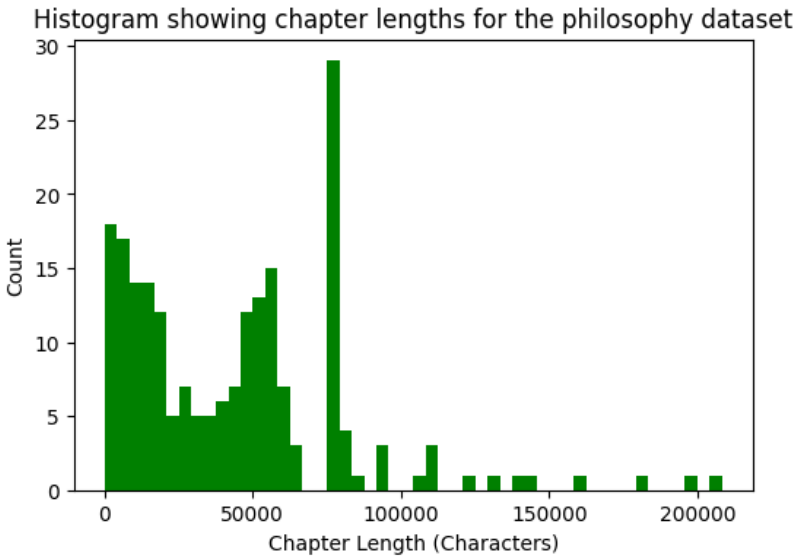


Figure 17: Histogram showing the lengths of chapters in the philosophy database.

This approach for acquiring the database had its limitations. As seen in Figure 17, the variance in chapter lengths was significant, leading to varying levels of summary qualities. Furthermore, the quality, and the coherence of the resulting summary was heavily dependent on the writer's style. For example, for Nietzsche's *Thus Spoke Zarathustra* the following is a summary of one of the chapters:

```
When Zarathustra had left the ugliest man, he was chilled and felt
lonesome. But behold, there were kine standing together on an
eminence, whose proximity and smell had warmed his heart. The kine,
however, seemed to listen eagerly to a speaker, and took no heed
of him who approached.
```

Nietzsche in this book presents his philosophy in novel style, to access the substance of the text requires one to have access to the context of the story, the meaning presents itself between the lines. This type of storytelling is not rare in philosophy, Philosopher's such as Voltaire, Camus, Hesse, etc. all adopt this style. For this reason, it was deemed unsuitable to obtain the data in such a way.

Instead, *GoodReads* was again consulted, where they have a page which contains all the philosophy quotes from texts that have been tagged by users, ordered by number of likes. The top 500 of these quotes were extracted via a python script that Web scraped [102], and filtered such that the remaining quotes were somewhat actionable on a personal level, leaving just over 200 quotes that will define the 'idea' network's action space. This number was deemed to be sufficient for proof-of-concept purposes, and will have to be increased significantly if this model is to be used commercially.

As for obtaining a list of activities, the internet was searched a definitive list that could be used, but there wasn't anything worth noting to be found, so instead, I organised a brainstorming session with friends from a variety of backgrounds. Although this is not ideal from a formality standpoint, it was deemed sufficient for proof-of-concept purposes. Below is an example of one of these quotes:

```
Holding on to anger is like grasping a hot coal with the intent of
throwing it at someone else; you are the one who gets burned.
- Buddha
```

4.2 Implementing a Baseline Model

The purpose of this baseline model is to implement a simple, adaptive, content based recommender system that will be used to compare the performance of the deep-actor critic network with. This model makes use of the SBERT embeddings of each item such that the position of each is relativised based on their semantics.

4.2.1 Model Design

The distance of each item from every other item is measured and stored. For embedding vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, the distance is calculated using the Euclidean norm $d_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||$. Suppose there are L items in the item set, then the resulting set of distances for item i is stored in the vector $\mathbf{d}_i \in \mathbb{R}^L$ and the matrix whose columns (and rows) are formed by the distance vectors $\mathbf{d}_i$, is denoted by $\mathbf{D} \in \mathbb{R}^{L\times L}$ and is symmetric. It's worth noting that if distances are stored as a 32-bit number, then for 1TB of storage, 1,000,000 items can be supported.

These distances define the probability weighting matrix $\mathbf{W} = \mathbf{J} \oslash \mathbf{D}$ where $\mathbf{J}$ is the matrix of ones and $\oslash$ is the Hadamard division operation. To put this into the context of the RL framework, given that the previous recommended item is item i, then we are considered to be in state s_i and the policy $\pi(a|s_i; \mathbf{W})$ is parametrised by the weight matrix $\mathbf{W}$, such that the items closest to i have a higher weigting, $w_{i,j}$ and correspond to a higher probability of being chosen. Following this, the probability of choosing item j when the current recommendation is item i, is thus given by:

$$\pi(a_j|s_i; \mathbf{W}) = p_{i,j} = \frac{w_{i,j}}{\sum_{k=0}^{L-1} w_{i,k}}$$

If we now contextualise this into our problem, activities define the state space and for activity recommendations, the activities define the action space also, meaning that the policy π_1 is composed of the distances between activities. As for the idea space, the state space is also given by the activity embeddings, however the action space is given by the idea embeddings, meaning that its policy π_2 is defined by the distances between ideas and activities. In this sense, just as with the designed AC network, first activities are recommended, and then ideas based on the recommended activities.

4.2.2 Initial Recommendation

This provides a reasonable solution to the cold start problem given an initial recommendation, however, how is this initial recommendation made for each person? Is there any information we can use to help get over that first hurdle? We recall the relationship between personality and happiness, this is an opportunity to both exercise and investigate this relationship.

The Myers-Briggs Type Indicator (MBTI) is a popular measure used to assess personality types based on the research of Carl Jung and his concept of "individual preference", which

suggests that seemingly random variations in human behaviour can be attributed to fundamental differences in how people think and feel [69]. Myers characterized these differences as distinct ways in which individuals prefer to utilize their cognitive processes. To measure these preferences, the MBTI presents a series of questions that gauge an individual's inclination towards one end of a spectrum in four different categories: energy, perceiving, judging, and orientation [69]. Despite its popularity and widespread use, the reliability and validity of the indicator has been deemed as being limited by various studies [87] [75] [96].

The five-factor model of personality (FFM) is instead consulted, which has been widely accepted by the scientific community [39], and has shown to transcend language and cultural differences [15] [112].

The FFM is a set of five broad trait dimensions or domains, often referred to as the "Big Five": Extraversion, Agreeableness, Conscientiousness, Neuroticism (sometimes named by its polar opposite, Emotional Stability), and Openness to Experience (sometimes named Intellect). Highly extraverted individuals are assertive and sociable, rather than quiet and reserved. Agreeable individuals are cooperative and polite, rather than antagonistic and rude. Conscientious individuals are task focused and orderly, rather than distractible and disorganized. Neurotic individuals are prone to experiencing negative emotions, such as anxiety, depression, and irritation, rather than being emotionally resilient. Finally, highly open individuals have a broad rather than narrow range of interests, are sensitive rather than indifferent to art and beauty, and prefer novelty to routine. The Big Five/FFM was developed to represent as much of the variability in individuals' personalities as possible, using only a small set of trait dimensions. Many personality psychologists agree that its five domains capture the most important, basic individual differences in personality traits and that many alternative trait models can be conceptualized in terms of the Big Five/FFM structure [93].

Early research into predictive aspects of personality traits reveals that extraverted individuals have a higher likelihood of experiencing positive life occurrences, while those scoring high in neuroticism have more negative experiences, suggesting a robust behavioural link between one's character and their encounters in life. Furthermore, the report discusses how an individual's personality shapes their subjective viewpoint of objective experience [58].

In a paper introducing the FFM and its applications, McCrae et al. provide lists of adjective descriptors for each of the five factors [60]. Prior to the first recommendation, it is suggested that users take the FFM questionnaire, and using the percentile of the user for each the five factors as weightings, the distances between each activity and the factor descriptors will determine the probability of that activity being recommended to them, in the same manner as before. Based on the feedback from the users, the weightings must be adaptively updated such that this recommendation is improved for the given personality. As a result, not only will the model become better at giving people the initial recommendation, but through looking at the convergence of these weightings, we can understand more about how different

personalities interact with activities in terms of their happiness.

4.2.3 Updating weights

The probability weightings will be updated according to a suitably defined reward function based on feedback from the user in response to a suitably designed question, which will be discussed later. For now, we assume a reward at time t for the initial recommendation $r_{1,t}$, the activity recommendation, $r_{2,t}$ and the idea recommendation $r_{3,t}$ where $\mathbb{E}[r_{1,t}] = \mathbb{E}[r_{2,t}] = \mathbb{E}[r_{3,t}] = 0$. For the initial recommendation, we assume the initial weight matrix $\mathbf{W}_1 \in \mathbb{R}^{L\times 16}$. For user u, with personality type l, and a recommendation taken such that activity i is recommended, it follows that:

$$w_{i,l,t} = \max\{\epsilon, \quad w_{i,l,t-1} + \gamma_1 r_{1,t}\} \tag{44}$$

Where γ is a hyperparameter and can be thought of as the learning-rate. The max operator is such that there are no negative weightings and thus negative probabilities, and ϵ is a very small number such that actions become very unlikely rather than impossible when people favour them badly. The MBTI weightings should be updated by all users, in a more CF approach to the initial suggestion. Similarly, for the activity and idea recommendations, respectively:

$$w_{i,j,t} = \max\{\epsilon, \quad w_{i,j,t-1} + \gamma_2 r_{2,t}\} \tag{45}$$

$$w_{i,k,t} = \max\{\epsilon, \quad w_{i,k,t-1} + \gamma_3 r_{3,t}\} \tag{46}$$

4.2.4 Model Properties

This model, despite its simplicity and heavy dependence on SBERT embeddings, has some favourable properties that are worth discussing. First of all, as the model keeps receiving positive rewards for a certain action for a given state, the policy will converge to a probability of 1 for that given action. The model parameters are indeed transparent, and the values of each of the probability weightings in their respective weighting matrices gives us an insight into the reaction people have to each item in terms of happiness. There is also consideration for the 'cold start' problem, which was another one of the outlined technical objectives. However, the algorithm is somewhat 'linear', aside from the complexity that SBERT embeddings provide, it does not have the complexity to model the non-linearity and time-dependency in the data, and so is insufficient for anything other than a baseline model for comparison.

4.2.5 Implementation

The implementation of the model was done in Python as a class, the code of which can be found in the Appendix (9.1). The Weight matrices can be loaded and saved to a file such that 'training' can be continuous and be held in multiple sessions.

4.3 Implementing the AC architecture

In this section, an initial set of input features for the actor-critic network are determined, as well as the specific architectures for the two recommender LSTM networks, as designed in Section 3.

4.3.1 Feature Engineering

For each of the actor and critic networks, the following set of features are proposed, and justified, they serve as an initial set of features that will help enhance predictions:

1. **User-Item Vector**: The user-item vector is used to store interactions between a particular user and the item space, it will be updated iteratively upon every interaction a user makes, such that recommendations are constantly adaptive to the user's changing 'taste'. The dimensionality of this vector is fixed to the size of the action/item space, but is passed through a GRU network to provide a fixed, low-dimensional, temporally-aware embedding, which is then passed through the main actor network.

2. **SBERT Embeddings**: Typically the state and action spaces are represented as one-hot encoded vectors, however, it is preferred that this model is robust to a growing action and state space, and one-hot encoded vectors not only carry no information regarding the state, but the size of the vector grows as the action space grows. For this reason, having a fixed, relatively low-dimensional embedding is preferred [77], not only allows for a more compact representation of the state space, but a relativised representation of the state space which holds useful information regarding items.

3. **FFM Percentiles**: As investigated earlier, the hypothesis that personality having an effect on happiness is utilised and investigated here. The FFM is used as the indicator for the users' personality, and is determined via a questionnaire prior to the first recommendation, the reasons for which were discussed in Section 4.2.2. The FFM consists of 5 categories, where users are given a score, and a population percentile for each, the percentile for each category is thus contained in a 5-dimensional vector, and is used as one of the network features.

4. **Age**: Although not previously discussed, but hinted at with the discussion of Figure 2, age is proposed as an important feature to include. It is particularly pertinent for activity recommendations where age plays as a clear physical barrier for some activities (Running, Climbing etc.). Furthermore, it only adds a single extra dimension to the input data.

5. **Geographical Location**: Geographical location is deemed to be an important feature to be included, this is its ability to somewhat capture the cultural influence on recommendations, as well as what activities may or may not be possible based on the location. Even if this additional feature does not help enhance recommendations, it will at least provide an indication of the effect of culture on what ideas and activities bring happiness. It can be represented by a 2-Dimnesional latitude and longitude number

These input features serve as a starting point for potential input features that define the state space, and again, itself is a research topic of its own, nevertheless, these features are deemed to be sufficient at providing a deep enough representation of the state space to produce effective recommendations, and are summarised in table below.

Input Feature	**Feature Dimensionality**	**Data Type**
User-Item	32 (After Embedding)	Real number
SBERT Embedding	768	Real number
FFM Percentile	5	Real Number $\in [0, 1]$
Age	1	Positive Integer
Location	2	Real Number

Table 1: Comparison of theoretical and empirical LMS misadjustments.

4.3.2 Actor-Critic Network Design

With the design schematic of the overall Actor-Critic architecture, as outlined in Figure 15, and the input features that were selected in the previous section in mind, the network that will parametrise the policy that maps states onto actions is now defined and implemented.

Deep Actor network The detailed schematic of the network is outlined in Figure 19. the input vector $\mathbf{x}_t$ corresponds to the user-item vector with dimensionality equal to the action/item space, which is passed through two GRU layers (the second allowing for additional depth of temporal representation), that produce a fixed 32-dimensional embedding, $\mathbf{h}_t^2$. The previous recommendation is passed through an SBERT encoding layer (non-trainable), and is concatenated with the embedded user item matrix, as well as the additional user-features. This is then input into three consecutive densely connected neural network layers with ReLU activation (equation 14) and then through a final dense layer with softmax activation [30] which converts the outputs of the final dense layer into a set of normalised probabilities (Figure 18) that corresponds to each possible action in the action space, therefore, the dimensionality of the final dense layer is equal to the dimensionality of the action space (`action_dim`).
Prior to inputting data into the model, features are individually scaled with a MinMaxScaler, which scales an input vector according to equation 47:

$$\mathbf{x}' = \frac{\mathbf{x} - \mathbf{x}_{min}}{\mathbf{x}_{max} - \mathbf{x}_{min}} \tag{47}$$

Where the subtract and divide are element-wise operations. MinMaxScaling allows for all features to occupy the same range, such that there is no numerical bias on a particular feature, which allows for faster and more stable learning [41]. The architecture of the network is the same for both idea and activity recommender systems, the only difference being the output

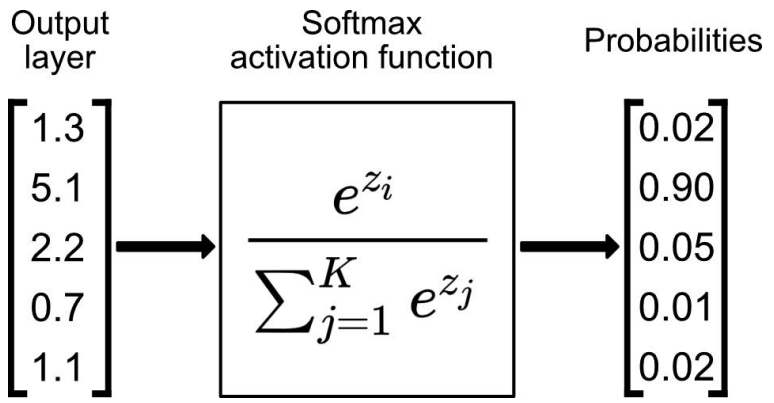


Figure 18: Diagram demonstrating the softmax activation function.

dimension of the network, where the two differ in the size of their action spaces. The model is implemented with the Python API, *TensorFlow*. Since the input of the current time instance is dependent on the output of the previous time instance (the previous recommendation defines what state we are in), we make use of the `stateful` flag in *TensorFlow*'s GRU module, and work in batch sizes of one, such that cell states, and hidden states are remembered at the end of each pass through of the network. This will slow down the training of the network, since working *TensorFlow* is optimised to work in batches rather than iterating through the data, however, due to this output-input dependency, there is no other choice.

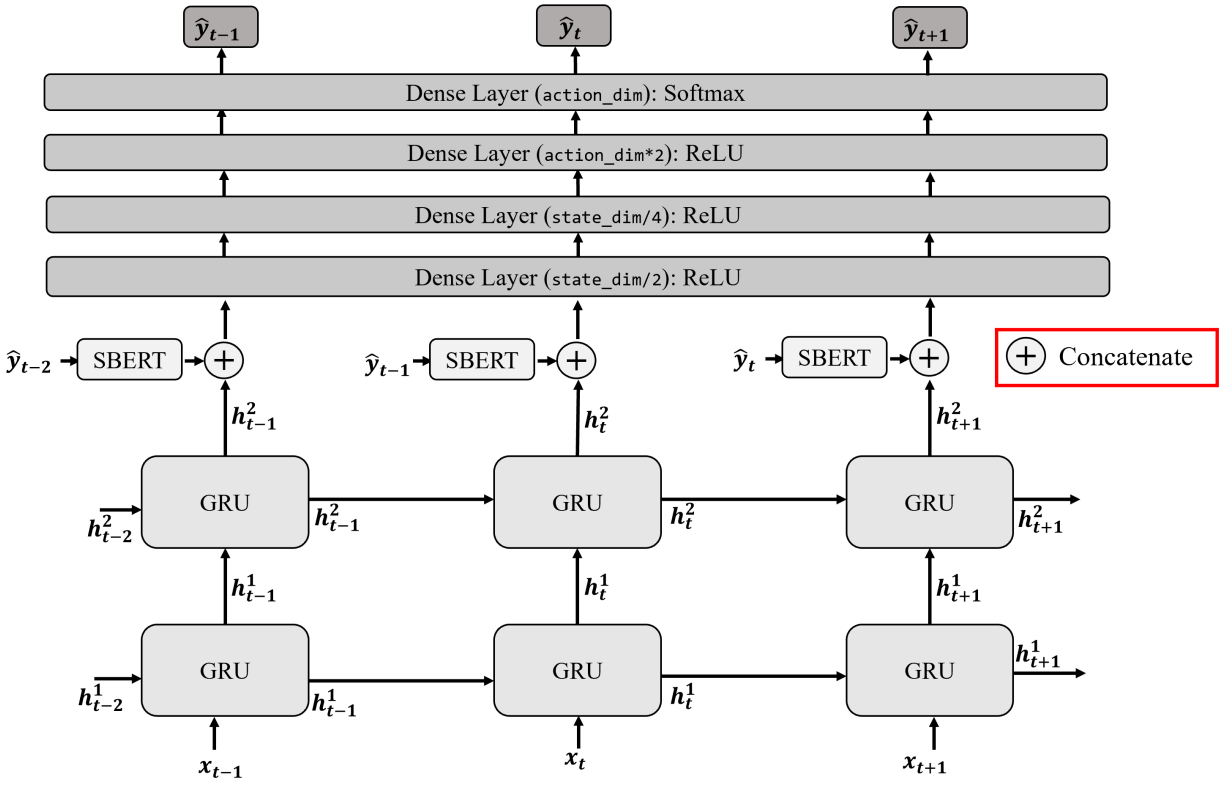


Figure 19: Schematic of the actor network.

Batch Normalisation is performed between the second and third layer, as this has proven to help stabilise and accelerate learning [45].

Critic LSTM network The critic network is used to predict the value for a given state, and has an almost identical architecture to that of its corresponding actor, however, since we are now predicting a single real number instead of a set of probabilities, the final dense layer consists of a single node with no/linear activation, as shown in Figure 20. As with the actor networks, the idea, and activity critic networks are identical.

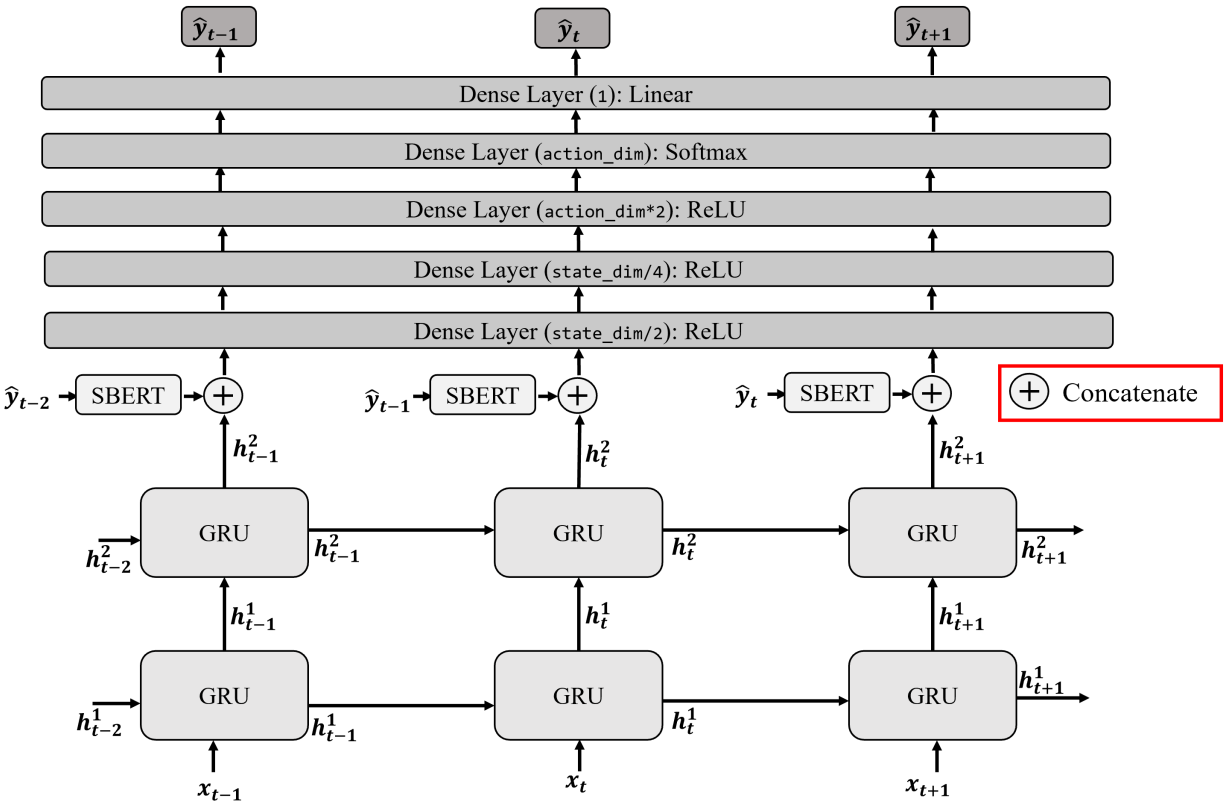


Figure 20: Schematic of the Critic LSTM network.

4.3.3 Update Procedure

With the networks for both the actor and the critic designed and implemented, they are now integrated within the Asynchronous Advantage Actor Critic (A3C) framework [63], which includes the integration of the background in Sections 2.2.6 and 2.2.7, with a slight adjustment to the vanilla actor-critic methodology. The actor and critic networks produce the policy $\pi_\theta(a|s)$ and the estimated value function $V_{\theta_V}(s)$, both of which will be assumed.

Instead of using the Q-function to update parameters, as done in equation 25, the advantage function, which is the expected value of the Temporal Difference (TD) error (equation 24), or equally, it can be seen as a measurement of how much better or worse the action a is compared to the average action value in state s, is used and is estimated by:

$$\hat{A}(s_t, a_t) = Q_{\theta_Q}(s_t, a_t) - V_{\theta_V}(s_t) \tag{48}$$

Instead of having two separate networks estimating the Advantage function, only the Value function can be used by replacing the estimate of the Q-function with its Value function equivalent, such that:

$$\hat{A}(s_t, a_t) = r_t + V_{\theta_V}(s_{t+1}) - V_{\theta_V}(s_t) \tag{49}$$

Clearly for the estimator to be unbiased, $V_{\theta_V} = V$, where V is the true value function, and as a result, this estimate of the Advantage function typically has a large bias and low variance. On the other hand, when the advantage is instead estimated using the sum of the discounted rewards (equation 50), it has a high variance and low bias [85].

$$\hat{A}(s_t, a_t) = R_t - V_{\theta_V}(s_t), \qquad R_t = \sum_{k=0}^{T-t} \gamma^k r_{t+k} \tag{50}$$

To find a trade-off between the bias and the variance, the Generalised Advantage Estimator (GAE) is introduced subject to a fine-tuning parameter λ that can be selected empirically. To begin with, consider k estimators of the advantage function with increasing order of contribution from discounted rewards, i.e.:

$$\hat{A}_t^{(1)} = \delta_t^V \qquad = -V(s_t) + r_t + \gamma V(s_{t+1}) \tag{51}$$

$$\hat{A}_t^{(2)} = \delta_t^V + \gamma\delta_{t+1}^V = -V(s_t) + r_{t+1} + \gamma r_t \gamma^2 V(s_{t+2}) \tag{52}$$

$$\vdots$$

$$\hat{A}_t^{(k)} = \sum_{t=0}^{k-1} \gamma^l \delta_t^V = -V(s_t) + r_{t+1} + \gamma r_t + \ldots + \gamma^{k-1} r_{t+k-1} + \gamma^k V(s_{t+k}) \tag{53}$$

Clearly,

$$\hat{A}_t^{(\infty)} = -V_{\theta_V}(s_t) + \sum_{k=0}^{T-t} \gamma^k r_{t+k} \tag{54}$$

As in equation 50, which is the difference between the empirical returns and the baseline value function [85]. The GAE(γ, λ) is defined as the exponentially weighted average of those k estimators, such that:

$$\hat{A}_t^{\text{GAE}(\gamma,\lambda)} = (1-\lambda)\left(\hat{A}_t^{(1)} + \lambda\hat{A}_t^{(2)} + \lambda^2\hat{A}_t^{(3)} + \dots\right) \tag{55}$$

$$= (1-)(\delta_t^V + \lambda(\delta_t^V + \gamma\delta_{t+1}^V) + \lambda^2(\delta_t^V + \gamma\delta_{t+1}^V + \gamma^2\delta_{t+2}^V) + \dots) \tag{56}$$

$$= (1-\lambda)(\delta_t^V(1+\lambda+\lambda^2+\dots) + \gamma\delta_{t+1}^V(\lambda+\lambda^2+\dots) + \gamma^2\delta_{t+2}^V(\lambda^2+\dots) + \dots) \tag{57}$$

$$= (1-\lambda)\left(\delta_t^V\left(\frac{1}{1-\lambda}\right) + \gamma\delta_{t+1}^V\left(\frac{\lambda}{1-\lambda}\right) + \gamma^2\delta_{t+2}^V\left(\frac{\lambda^2}{1-\lambda}\right) + \dots\right) \tag{58}$$

$$= \sum_{t=0}^{\infty}(\lambda\gamma)^l\delta_{t+l}^V \tag{59}$$

Now to find a balance between the bias and the variance, the fine-tuning of an extra parameter is required, this will be difficult provided the lack of training data, therefore, equation **??** will be preferred initially at the cost of a high bias in the advantage estimation, since it allows for network parameters to be updated with a single observation of the environment, allowing for iterative training on the network. The GAE can be later introduced if it is deemed that the results are unsatisfactory.

With an estimate for the advantage function in mind, the parameters of the network can now be updated. The actor network, which is learning the policy of the RLRS model, is updated as with the vanilla AC model, but instead using the estimated advantage function as a guide, such that:

$$\nabla_\theta J(\theta) \approx \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot \hat{A}(s_t, a_t) \tag{60}$$

Where $T = 1$ for iterative updates to the network. This gradient is backpropagated through the network via a popular Stochastic Gradient Descent (SGD) based optimiser, Adam [49]. The value network is updated by minimizing the mean squared error between the estimated state value $V_{\theta_V}(s)$ and the observed discounted returns R_t, such that:

$$\mathcal{L}(\theta_V) = \sum_{t=1}^{T}(R_t - V_{\theta_V}(s_t))^2 \tag{61}$$

Which is also backpropagated through the network via the Adam optimiser. With the update procedure for both actor and critic models set, and the networks that define the estimates for the policy and value function set, the full model was implemented in a class in python and can be found in the appendix.

4.4 Designing a reward function

Designing a suitable reward function that correctly captures the aims of our understanding is crucial to the manner in which this model will learn. Now, the design of such a question to correctly capture a person's happiness has been a subject of research for decades and is till ongoing [25]. As a result, the appropriate, systematic design of the items we propose to users requires its own research project, and is outside the scope of this project. Instead, we take inspiration from Lyubomirsky et al.'s Subjective Happiness Scale (SHS) [57], and look to implement something similar, but with a closer reference to the recommended items. We intend to capture the users' response to recommendations, as well as their overall happiness, we thus propose the following items:

For activity recommendations:

1. On the whole, this activity contributed positively to my overall happiness.
2. I found that this activity matched my interests well.

For idea recommendations:

1. On the whole, this idea contributed positively towards my overall happiness.
2. I found that this idea complimented my current outlook and thoughts.

Miscellaneous Feedback :

1. On the whole, I am satisfied with my life, and I consider myself generally happy.

Feedback is then given from a scale of -5 to 5 where -5 corresponds to a complete disagreement and 5 corresponds to complete agreement. Scores are then averaged for the questions that correspond to their respective AC network, yielding the final value for the reward function for that given state and action.

Although it is understood that these items lack the scientific foundation that is perhaps required, we believe they serve as a satisfactory starting point, and sufficiently captures the quality of the recommendation, the relevance of the recommendation towards their individual happiness, and their happiness in general, whilst allowing the room for reflection and subjectivity that we have outlined as being crucial.

5 Results

6 Evaluation

7 Future Work

8 Conclusion

9 Appendix

9.1 Baseline Model Code

```
1
2  import numpy as np
3  import pandas as pd
4
5
6  class BaselineModel:
7      def __init__(self, mbti_type, n_nearest_neighbours=0,
           update_weight=1,
8                   activities_data=("file", "activity_embeddings.
                       json"),
9                   quotes_data=("file", "quote_embeddings.json"),
10                  mbpti_data=("file", "personality_embeddings.json"
                       )):
11         """
12         activities ("file"/"df", "path/pd.DataFrame"): pass in a
               tuple specifying whether
13             the activity embeddings are a file path or a pandas
                   dataframe object.
14         quotes ("file"/"df", "filepath"/pd.DataFrame)     : pass in
               a tuple specifying whether
15             the quotes embeddings are a file path or a pandas
                   dataframe object.
16         """
17
18         self.mbpti_type = mbti_type
19         #------------------ LOAD ACTIVITIES ---------------------------
20         if activities_data[0] == "file":
21             self.df_activities = pd.read_json(activities_data[1])
22             self.activity_path = activities_data[1]
23         elif activities_data[0] == "df":
24             self.df_activities = activities_data[1]
25             self.activity_path = ""
26         else:
27             raise ValueError('activities must be a tuple: ("file
                   "/"df", "filepath"/pd.DataFrame)')
28         #------------------ LOAD QUOTES -------------------------------
29         if quotes_data[0] == "file":
30             self.df_quotes = pd.read_json(quotes_data[1])
```

```
31              self.quote_path = quotes_data[1]
32          elif quotes_data[0] == "df":
33              self.df_quotes = quotes_data[1]
34              self.quote_path = ""
35          else:
36              raise ValueError('quotes must be a tuple: ("file"/"df
                    ", "filepath"/pd.DataFrame)')
37          #------------------- LOAD MBTI ---------------------------
38          mbpti_order = ['ISTJ', 'ISFJ', 'INFJ', 'INTJ', 'ISTP', '
                ISFP', 'INFP', 'INTP',
39              'ESTP', 'ESFP', 'ENFP', 'ENTP', 'ESTJ', 'ESFJ', 'ENFJ'
                    , 'ENTJ']
40          if mbpti_data[0] == "file":
41              self.df_mbpti = pd.read_json(mbpti_data[1])
42              self.mbpti_index = mbpti_order.index(mbti_type)
43          elif mbpti_data[0] == "df":
44              self.df_mbpti = mbpti_data[1]
45              self.mbpti_index = mbpti_order.index(mbti_type)
46          else:
47              raise ValueError('mbpti_data must be a tuple: ("file
                    "/"df", "filepath"/pd.DataFrame)')
48          self.update_weight = update_weight
49          self.n_nearest_neighbours = n_nearest_neighbours
50
51          self.current_activity, self.activity_index = self.
                make_initial_mbpti_suggestion(self.df_mbpti,
52
53
54          self.prev_activity_index = 0
55          self.quote_index = 0
```

```
56          self.current_quote = ""
57
58      ## -------------
59      # CLASS METHODS
60      # ---------------------
61
62      # make suggestions
63      def make_initial_mbpti_suggestion(self, df, activities,
            n_nearest_neighbours=0):
64
65          if n_nearest_neighbours:
66              neighbour_indexes = np.asarray(sorted(range(len(df["
                    ProbabilityWeights"][self.mbpti_index])),
67                                             key=lambda x: df["
                                                    ProbabilityWeights"][self.
                                                    mbpti_index][x])[-
                                                    n_nearest_neighbours:])
68          else:
69              neighbour_indexes = np.arange(0, len(df["
                    ProbabilityWeights"][self.mbpti_index]))
70
71          probabilities = np.array([df["ProbabilityWeights"][self.
                mbpti_index][i] for i in neighbour_indexes.tolist()])/
                sum([df["ProbabilityWeights"][self.mbpti_index][i] for i
                 in neighbour_indexes.tolist()])
72          new_index = np.random.choice(neighbour_indexes, p=
                probabilities)
73          recommended_action = activities[new_index]
74          return recommended_action, new_index
75
76      def make_suggestion(self, n_nearest_neighbours=0):
77
78          if n_nearest_neighbours:
79              neighbour_indexes = np.asarray(sorted(range(len(self.
                    df_activities["ProbabilityWeights"][self.
                    activity_index])),
80                                             key=lambda x: self.
                                                    df_activities["
                                                    ProbabilityWeights"][self.
                                                    activity_index][x])[-
                                                    n_nearest_neighbours:])
81          else:
```

```
82              neighbour_indexes = np.arange(0, len(self.
                    df_activities["ProbabilityWeights"][self.
                    activity_index]))
83
84          # return list of probabilities for each of the n
                neighbours
85          probabilities = np.array([self.df_activities["
                ProbabilityWeights"][self.activity_index][i] for i in
                neighbour_indexes.tolist()])/sum([self.df_activities["
                ProbabilityWeights"][self.activity_index][i] for i in
                neighbour_indexes.tolist()])
86
87          # sample from that probability distribution
88          new_index = np.random.choice(neighbour_indexes, p=
                probabilities)
89
90          # get associated activity for the sampled index
91          recommended_activity = self.df_activities["Activity"][
                new_index]
92          self.prev_activity_index = self.activity_index
93          self.activity_index = new_index
94          self.current_activity = recommended_activity
95          return recommended_activity, self.make_quote_suggestion(
                self.df_quotes, new_index, n_nearest_neighbours=
                n_nearest_neighbours)
96
97      def make_quote_suggestion(self, df, index,
            n_nearest_neighbours=0):
98          if n_nearest_neighbours:
99              neighbour_indexes = np.asarray(sorted(range(len(df["
                    ProbabilityWeights"][index])),
100                                            key=lambda x: df["
                                                ProbabilityWeights"][index
                                                ][x])[-n_nearest_neighbours
                                                :])
101         else:
102             neighbour_indexes = np.arange(0, len(df["
                    ProbabilityWeights"][index]))
103
104         probabilities = np.array([df["ProbabilityWeights"][index][
                i] for i in neighbour_indexes.tolist()])/sum([df["
                ProbabilityWeights"][index][i] for i in
```

```
            neighbour_indexes.tolist()])
105         new_index = np.random.choice(neighbour_indexes, p=
                probabilities)
106         self.quote_index = new_index
107         recommended_quote = df["Quote"][new_index]
108         return recommended_quote
109
110     # Update Probability Weighhts
111     def update_prob_weights(self, activity_reward, quote_reward,
            export=True):
112         self.df_activities[self.prev_activity_index]["
                ProbabilityWeights"][self.activity_index] =  self.
                df_activities[self.prev_activity_index]["
                ProbabilityWeights"][self.activity_index] +
                activity_reward*self.update_weight
113         self.df_activities[self.prev_activity_index]["
                ProbabilityWeights"][self.quote_index] =  self.
                df_activities[self.prev_activity_index]["
                ProbabilityWeights"][self.quote_index] + quote_reward*
                self.update_weight
114         if export:
115             if self.activity_path:
116                 self.df_activities.to_json(self.activity_path)
117             else:
118                 self.df_activities.to_json("activity_embeddings.
                        json")
119             if self.quote_path:
120                 self.df_activities.to_json(self.quote_path)
121             else:
122                 self.df_activities.to_json("activity_embeddings.
                        json")
```

References

[1]

[2] About. URL: https://public.oed.com/about/.

[3] Internet archive: Digital library of free amp; borrowable books, movies, music amp; wayback machine. URL: https://archive.org/.

[4] Open access at cambridge university press. URL: https://www.cambridge.org/core/open-research/open-access.

[5] Project gutenberg. URL: https://www.gutenberg.org/.

[6] Data smoothing, Dec 2022. URL: https://corporatefinanceinstitute.com/resources/business-intelligence/data-smoothing/.

[7] Measuring similarity from embeddings nbsp;—nbsp; machine learning nbsp;—nbsp; google developers, Jul 2022. URL: https://developers.google.com/machine-learning/clustering/similarity/measuring-similarity.

[8] Normalization nbsp;—nbsp; machine learning nbsp;—nbsp; google developers, Jul 2022. URL: https://developers.google.com/machine-learning/data-prep/transform/normalization.

[9] Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. Docbert: BERT for document classification. *CoRR*, abs/1904.08398, 2019. URL: http://arxiv.org/abs/1904.08398, arXiv:1904.08398.

[10] Jeromy Anglim, Sharon Horwood, Luke D. Smillie, Rosario J. Marrero, and Joshua K. Wood. Predicting psychological and subjective well-being from personality: A meta-analysis. *Psychological Bulletin*, 146(4):279–323, 2020. doi:10.1037/bul0000226.

[11] Aristotle. Metaphysics. *Clarendon Aristotle Series: Metaphysics: Book* , 2006. doi:10.1093/oseo/instance.00258589.

[12] Ken G. Binmore. *Mathematical analysis: A straightforward approach*. Cambridge university press, 1993.

[13] J. Bobadilla, F. Serradilla, and J. Bernal. A new collaborative filtering metric that improves the behavior of recommender systems. *Knowledge-Based Systems*, 23(6):520–528, 2010. doi:10.1016/j.knosys.2010.03.009.

[14] Dheeraj Bokde, Sheetal Girase, and Debajyoti Mukhopadhyay. Matrix factorization model in collaborative filtering algorithms: A survey. *Procedia Computer Science*, 49:136–146, 2015. Proceedings of 4th International Conference on Advances in Computing, Communication and Control (ICAC3'15). URL: https://www.sciencedirect.com/science/article/pii/S1877050915007462, doi:https://doi.org/10.1016/j.procs.2015.04.237.

[15] Thomas J Bouchard and John C Loehlin. Genes, evolution, and personality. *Behavior genetics*, 31:243–273, 2001.

[16] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. *CoRR*, abs/1508.05326, 2015. URL: http://arxiv.org/abs/1508.05326, arXiv:1508.05326.

[17] Philip Brickman, Dan Coates, and Ronnie Janoff-Bulman. Lottery winners and accident victims: Is happiness relative? *Journal of Personality and Social Psychology*, 36(8):917–927, 1978. doi:10.1037/0022-3514.36.8.917.

[18] Albrecht Böttcher and David Wenzel. The frobenius norm and the commutator. *Linear Algebra and its Applications*, 429(8):1864–1885, 2008. URL: https://www.sciencedirect.com/science/article/pii/S0024379508002772, doi:https://doi.org/10.1016/j.laa.2008.05.020.

[19] Francois Chaubard. Cs224n: Natural language processing with deep learning, 2019. URL: https://web.stanford.edu/class/cs224n/.

[20] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H. Chi. Top-k off-policy correction for a reinforce recommender system. *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 2019. doi:10.1145/3289600.3290999.

[21] Xiaocong Chen, Lina Yao, Julian McAuley, Guanglin Zhou, and Xianzhi Wang. Deep reinforcement learning in recommender systems: A survey and new perspectives. *Knowledge-Based Systems*, 264:110335, 2023. doi:10.1016/j.knosys.2023.110335.

[22] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014. URL: http://arxiv.org/abs/1412.3555, arXiv:1412.3555.

[23] Michael A. Cohn, Barbara L. Fredrickson, Stephanie L. Brown, Joseph A. Mikels, and Anne M. Conway. Happiness unpacked: Positive emotions increase life satisfaction by building resilience. *Emotion*, 9(3):361–368, 2009. doi:10.1037/a0015952.

[24] J.T. Connor, R.D. Martin, and L.E. Atlas. Recurrent neural networks and robust time series prediction. *IEEE Transactions on Neural Networks*, 5(2):240–254, 1994. doi:10.1109/72.279188.

[25] Gabriela Delsignore, Alejandra Aguilar-Latorre, Pablo Garcia-Ruiz, and Bárbara Oliván-Blázquez. Measuring happiness for social policy evaluation: a multidimensional index of happiness. *Sociological Spectrum*, pages 1–15, 2023.

[26] Melikşah Demir. Close relationships and happiness among emerging adults. *Journal of Happiness Studies*, 11(3):293–313, 2009. doi:10.1007/s10902-009-9141-x.

[27] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018. URL: https://arxiv.org/abs/1810.04805, doi:10.48550/ARXIV.1810.04805.

[28] Dharti Dhami. Understanding bert-word embeddings, Jul 2020. URL: https://medium.com/@dhartidhami/understanding-bert-word-embeddings-7dc4d2ea54ca.

[29] Avinash K. Dixit. *Optimization in economic theory*. Oxford University Press, 1990.

[30] Rob A Dunne and Norm A Campbell. On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function. In *Proc. 8th Aust. Conf. on the Neural Networks, Melbourne*, volume 181, page 185. Citeseer, 1997.

[31] Michelle F. Eabon and Dan Abrahamson. Understanding psychological testing and assessment, Aug 2022. URL: https://www.apa.org/topics/testing-assessment-measurement/understanding.

[32] Ada Ferrer-i Carbonell and Xavier Ramos. Inequality and happiness. *Journal of Economic Surveys*, 28(5):1016–1027, 2013. doi:10.1111/joes.12049.

[33] Barbara L. Fredrickson, Michael A. Cohn, Kimberly A. Coffey, Jolynn Pek, and Sandra M. Finkel. Open hearts build lives: Positive emotions, induced through loving-kindness meditation, build consequential personal resources. *Journal of Personality and Social Psychology*, 95(5):1045–1062, 2008. doi:10.1037/a0013262.

[34] Frank Fujita and Ed Diener. Life satisfaction set point: Stability and change. *Journal of Personality and Social Psychology*, 88(1):158–164, 2005. doi:10.1037/0022-3514.88.1.158.

[35] Simon Funk. URL: https://sifter.org/~simon/journal/20061211.html.

[36] Philip Gage. A new algorithm for data compression. *C Users J.*, 12(2):23–38, feb 1994.

[37] Ángel García-Crespo, José Luis López-Cuadrado, Ricardo Colomo-Palacios, Israel González-Carrasco, and Belén Ruiz-Mezcua. Sem-fit: A semantic based expert system to provide recommendations in the tourism domain. *Expert Systems with Applications*, 38(10):13310–13319, 2011. doi:10.1016/j.eswa.2011.04.152.

[38] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[39] Michael Gurven, Christopher von Rueden, Maxim Massenkoff, Hillard Kaplan, and Marino Lero Vie. How universal is the big five? testing the five-factor model of personality variation among forager–farmers in the bolivian amazon. *Journal of Personality and Social Psychology*, 104(2):354–370, 2013. doi:10.1037/a0030841.

[40] Dan Haybron. Happiness, May 2020. URL: https://plato.stanford.edu/entries/happiness/.

[41] Simon Haykin. *Neural networks and learning machines, 3/E*. Pearson Education India, 2009.

[42] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural netsnbsp; and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 06(02):107–116, 1998. doi:10.1142/s0218488598000094.

[43] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi:10.1162/neco.1997.9.8.1735.

[44] Felicia A Huppert. Psychological well-being: Evidence regarding its causes and consequences. *Applied Psychology: Health and Well-Being*, 1(2):137–164, 2009. doi:10.1111/j.1758-0854.2009.01008.x.

[45] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. pages 448–456, 2015. URL: http://jmlr.org/proceedings/papers/v37/ioffe15.pdf.

[46] Ian T. Jolliffe and Jorge Cadima. Principal component analysis: A review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016. doi:10.1098/rsta.2015.0202.

[47] Ankur Joshi, Saket Kale, Satish Chandel, and D. Pal. Likert scale: Explored and explained. *British Journal of Applied Science amp; Technology*, 7(4):396–403, 2015. doi:10.9734/bjast/2015/14975.

[48] Matthew A. Killingsworth. Experienced well-being rises with income, even above $75,000 per year. *Proceedings of the National Academy of Sciences*, 118(4), 2021. doi:10.1073/pnas.2016976118.

[49] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[50] Bryan Kolb, Allonna Harker, and Robbin Gibb. Principles of plasticity in the developing brain. *Developmental Medicine amp; Child Neurology*, 59(12):1218–1223, 2017. doi:10.1111/dmcn.13546.

[51] Vijay R. Konda and John N. Tsitsiklis. Onactor-critic algorithms. *SIAM Journal on Control and Optimization*, 42(4):1143–1166, 2003. doi:10.1137/s0363012901385691.

[52] Angela Lee Duckworth, Tracy A. Steen, and Martin E.P. Seligman. Positive psychology in clinical practice. *Annual Review of Clinical Psychology*, 1(1):629–651, 2005. doi:10.1146/annurev.clinpsy.1.102803.144154.

[53] Angela Lee Duckworth, Tracy A. Steen, and Martin E.P. Seligman. Positive psychology in clinical practice. *Annual Review of Clinical Psychology*, 1(1):629–651, 2005. doi:10.1146/annurev.clinpsy.1.102803.144154.

[54] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461, 2019. URL: http://arxiv.org/abs/1910.13461, arXiv:1910.13461.

[55] Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. *CoRR*, abs/1908.08345, 2019. URL: http://arxiv.org/abs/1908.08345, arXiv:1908.08345.

[56] Sonja Lyubomirsky and Heidi Lepper. A measure of subjective happiness: Preliminary reliability and construct validation. *Social Indicators Research*, 46(2):137–155, Feb 1999. doi:10.1023/a:1006824100041.

[57] Sonja Lyubomirsky and Heidi S. Lepper. Subjective happiness scale. *PsycTESTS Dataset*, 1997. doi:10.1037/t01588-000.

[58] Keith Magnus, Ed Diener, Frank Fujita, and William Pavot. Extraversion and neuroticism as predictors of objective life events: A longitudinal analysis. *Journal of Personality and Social Psychology*, 65(5):1046–1053, Nov 1993. doi:10.1037/0022-3514.65.5.1046.

[59] Deborah Mattei and Charles E. Schaefer. An investigation of validity of the subjective happiness scale. *Psychological Reports*, 94(1):288–290, 2004. doi:10.2466/pr0.94.1.288-290.

[60] Robert R. McCrae and Oliver P. John. An introduction to the five-factor model and its applications. *Journal of Personality*, 60(2):175–215, 1992. doi:10.1111/j.1467-6494.1992.tb00970.x.

[61] Remya R.K. Menon, R Akhil dev, and Sreehari G Bhattathiri. An insight into the relevance of word ordering for text data analysis. In *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*, pages 207–213, 2020. doi:10.1109/ICCMC48092.2020.ICCMC-00040.

[62] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013. URL: https://arxiv.org/abs/1301.3781, doi:10.48550/ARXIV.1301.3781.

[63] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783, 2016. URL: http://arxiv.org/abs/1602.01783, arXiv:1602.01783.

[64] Ruihui Mu. A survey of recommender systems based on deep learning. *IEEE Access*, 6:69009–69022, 2018. doi:10.1109/access.2018.2880197.

[65] Boris Nikolaev and Pavel Rusakov. Education and happiness: An alternative hypothesis. *Applied Economics Letters*, 23(12):827–830, 2015. doi:10.1080/13504851.2015.1111982.

[66] Christopher Olah. Understanding lstm networks, Aug 2015. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[67] Shay Palachy. Document embedding techniques, Jun 2022. URL: https://towardsdatascience.com/document-embedding-techniques-fed3e7a6a25d.

[68] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1310–1318, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL: https://proceedings.mlr.press/v28/pascanu13.html.

[69] Woods RA;Hill PB;. Myers brigg, Sep 2022. URL: https://pubmed.ncbi.nlm.nih.gov/32119483/.

[70] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. URL: https://aclanthology.org/D14-1162, doi:10.3115/v1/D14-1162.

[71] Seph Fontane Pennock. The hedonic treadmill - are we forever chasing rainbows?, Aug 2022. URL: https://positivepsychology.com/hedonic-treadmill/.

[72] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018. doi:10.18653/v1/n18-1202.

[73] Gábor Petneházi. Recurrent neural networks for time series forecasting. *CoRR*, abs/1901.00069, 2019. URL: http://arxiv.org/abs/1901.00069, arXiv:1901.00069.

[74] Michael Phi. Illustrated guide to transformers- step by step explanation, Jun 2020. URL: https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0.

[75] Ken Randall, Mary Isaacson, and Carrie Ciro. Validity and reliability of the myers-briggs personality type indicator: A systematic review and meta-analysis. *Journal*

of Best Practices in Health Professions Diversity, 10(1):1–27, 2017. URL: https://www.jstor.org/stable/26554264.

[76] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084, 2019. URL: http://arxiv.org/abs/1908.10084, arXiv:1908.10084.

[77] Mehrdad Rezaei and Nasseh Tabrizi. Recommender system using reinforcement learning: A survey. *Proceedings of the 3rd International Conference on Deep Learning Theory and Applications*, 2022. doi:10.5220/0011300300003277.

[78] Matthieu Ricard. A buddhist view of happiness. *Journal of Law and Religion*, 29(1):14–29, 2014. doi:10.1017/jlr.2013.9.

[79] Deepjyoti Roy and Mala Dutta. A systematic review and research perspective on recommender systems. *Journal of Big Data*, 9(1), 2022. doi:10.1186/s40537-022-00592-5.

[80] Elena Rudkowsky, Martin Haselmayer, Matthias Wastian, Marcelo Jenny, Štefan Emrich, and Michael Sedlmair. More than bags of words: Sentiment analysis with word embeddings. *Communication Methods and Measures*, 12(2-3):140–157, 2018. doi:10.1080/19312458.2018.1455817.

[81] Richard M. Ryan and Edward L. Deci. On happiness and human potentials: A review of research on hedonic and eudaimonic well-being. *Annual Review of Psychology*, 52(1):141–166, 2001. doi:10.1146/annurev.psych.52.1.141.

[82] David Sarokin and Jay Schulkin. Information: A brief modern history. *Missed Information*, 2016. doi:10.7551/mitpress/9780262034920.003.0002.

[83] J. Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. *Collaborative Filtering Recommender Systems*, pages 291–324. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. doi:10.1007/978-3-540-72079-9_9.

[84] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. *CoRR*, abs/1503.03832, 2015. URL: http://arxiv.org/abs/1503.03832, arXiv:1503.03832.

[85] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. 06 2015.

[86] Norbert Schwarz and Fritz Strack. *Reports of subjective well-being: Judgmental processes and their methodological implications.*, pages 61–84. Well-being: The foundations of hedonic psychology. Russell Sage Foundation, New York, NY, US, 1999.

[87] David M. Schweiger. Measuring managerial cognitive styles: On the logical validity of the myers-briggs type indicator. *Journal of Business Research*, 13(4):315–328, 1985.

URL: https://www.sciencedirect.com/science/article/pii/0148296385900049, doi:https://doi.org/10.1016/0148-2963(85)90004-9.

[88] Martin EP Seligman. *Positive psychology in practice*. John Wiley & Sons, 2012.

[89] Martin EP Seligman et al. Positive psychology, positive prevention, and positive therapy. *Handbook of positive psychology*, 2(2002):3–12, 2002.

[90] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *Towards Data Sci*, 6(12):310–316, 2017.

[91] Wei Shi and Vera Demberg. Next sentence prediction helps implicit discourse relation classification within and across domains. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5790–5796, Hong Kong, China, November 2019. Association for Computational Linguistics. URL: https://aclanthology.org/D19-1586, doi:10.18653/v1/D19-1586.

[92] Choonsung Shin and Woontack Woo. Socially aware tv program recommender for multiple viewers. *IEEE Transactions on Consumer Electronics*, 55(2):927–932, 2009. doi:10.1109/tce.2009.5174476.

[93] Christopher J. Soto and Joshua J. Jackson. Five-factor model of personality. *Psychology*, 2013. doi:10.1093/obo/9780199828340-0120.

[94] Siniša Stanivuk. Understanding word2vec: Code and math, Aug 2020. URL: https://stopwolf.github.io/blog/nlp/paper/2020/08/17/word2vec.html.

[95] S. S. Stevens. On the theory of scales of measurement. *Science*, 103(2684):677–680, 1946. doi:10.1126/science.103.2684.677.

[96] Lawrence J. Stricker and John Ross. An assessment of some structural properties of the jungian personality typology. *The Journal of Abnormal and Social Psychology*, 68(1):62–71, 1964. doi:10.1037/h0043580.

[97] L. W. Sumner. *Welfare, Happiness, and Ethics*. Oxford University Press, 1996.

[98] Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1988. doi:10.1007/bf00115009.

[99] Richard S. Sutton, Francis Bach, and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT Press Ltd, 2018.

[100] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, page 1057–1063, Cambridge, MA, USA, 1999. MIT Press.

[101] WŁadysŁaw Tatarkiewicz. Analysis of happiness. page 140, 1976. doi:10.1007/978-94-010-1380-2.

[102] Tsaone Swaabow Thapelo, Molaletsa Namoshe, Oduetse Matsebe, Tshiamo Motshegwa, and Mary-Jane Morongwa Bopape. Sasscal websapi: A web scraping application programming interface to support access to sasscal's weather data. *Data Science Journal*, 20, 2021. doi:10.5334/dsj-2021-024.

[103] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL: http://arxiv.org/abs/1706.03762, arXiv:1706.03762.

[104] Paolo Verme. Happiness, freedom and control. *Journal of Economic Behavior amp; Organization*, 71(2):146–161, 2009. doi:10.1016/j.jebo.2009.04.008.

[105] Laura von Rueden, Sebastian Houben, Kostadin Cvejoski, Christian Bauckhage, and Nico Piatkowski. Informed pre-training on prior knowledge, 2022. URL: https://arxiv.org/abs/2205.11433, doi:10.48550/ARXIV.2205.11433.

[106] Shu-Lin Wang and Chun-Yi Wu. Application of context-aware and personalized recommendation to implement an adaptive ubiquitous learning system. *Expert Systems with Applications*, 38(9):10831–10838, 2011. doi:10.1016/j.eswa.2011.02.083.

[107] Christopher Watkins. Learning from delayed rewards. 01 1989.

[108] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. URL: https://aclanthology.org/N18-1101, doi:10.18653/v1/N18-1101.

[109] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL: http://arxiv.org/abs/1609.08144, arXiv:1609.08144.

[110] Petros Xanthopoulos, Panos M. Pardalos, and Theodore B. Trafalis. *Linear Discriminant Analysis*, pages 27–33. Springer New York, New York, NY, 2013. doi:10.1007/978-1-4419-9878-1_4.

[111] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. *CoRR*, abs/2002.04745, 2020. URL: https://arxiv.org/abs/2002.04745, arXiv:2002.04745.

[112] Shinji Yamagata, Atsunobu Suzuki, Juko Ando, Yutaka Ono, Nobuhiko Kijima, Kimio Yoshimura, Fritz Ostendorf, Alois Angleitner, Rainer Riemann, Frank M Spinath, et al. Is the genetic structure of human personality universal? a cross-cultural twin study from north america, europe, and asia. *Journal of personality and social psychology*, 90(6):987, 2006.

[113] Herong Yang, 2023. URL: https://www.herongyang.com/Neural-Network/RNN-What-Is-GRU.html.

[114] Yang Yu. Towards sample efficient reinforcement learning. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, 2018. doi:10.24963/ijcai.2018/820.

[115] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. Recommendations with negative feedback via pairwise deep reinforcement learning. *CoRR*, abs/1802.06501, 2018. URL: http://arxiv.org/abs/1802.06501, arXiv:1802.06501.