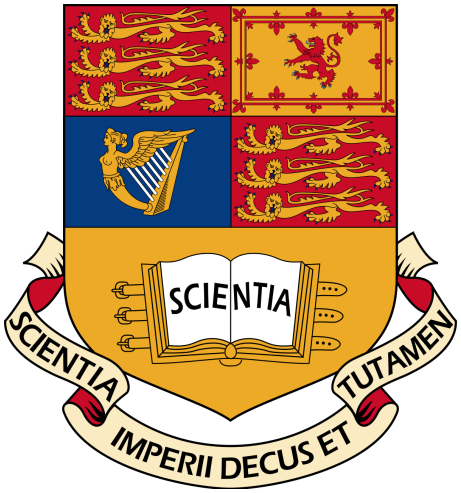
Imperial College London

Department of Electrical and Electronic Engineering



A Collective Understanding of Happiness

Author:

Nima Karsheneas

CID:

01500753

Supervisor:

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.

I have used ChatGPT, as an aid in the preparation of my report. It was sometimes used as a starting point for understanding complex topics, nonetheless all technical content, references, and code come from my own research and writing.

Abstract

Due to its crucial role in driving our motives and decisions, and its inherent subjectivity, happiness has been subject to fierce, and consistent philosophical debate for millennia. As a direct consequence of this subjectivity, the topic has seldom been approached from the lens of engineering. However, with the rise of Big Data and Natural Language Processing (NLP) [48], the inventory of tools available that allow for us to approach such a problem from an engineering standpoint is becoming ever-increasing. This project aims to implement a recommender system that will serve as the backbone for a user-interface that seeks to optimise individual experienced happiness, as well as providing a means for furthering our collective understanding of this all-important notion. A dual recommender system, based on a Deep Advantage Actor-Critic Reinforcement Learning framework, is designed, and implemented, which presents to a user an activity to complete, and an excerpt from a philosophical text to contemplate. According to feedback given by the user on the effect of these recommendations, the parameters of the model are updated in order to optimise the user's happiness. The designed model allows for salient qualitative inferences to be drawn by observing network parameters, as well as demonstrating promising results in a semi-simulated environment, outperforming other baseline models. This project serves as a proof-of-concept for the use of data, and Artificial Intelligence to aid our collective understanding of happiness.

Acknowledgements

I would like to dedicate this project to my parents, Amir and Mojgan, whose unwavering love, support, and selfless sacrifice I will be eternally thankful and indebted towards, the example you set is one I strive to follow each day.

I would also like to thank my flatmates, Geoff, Max, Baptiste, and Rohan, whose kindness and humour kept me smiling through what has been a challenging academic year.

Finally, I'd like to thank Professor Rodriguez Villegas for providing me with the wonderful opportunity to work on such an interesting project, and whose support and guidance has been valuable throughout this project.

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* [114]

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 this project 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) [10] to the Buddhist philosophy that happiness is the embrace of all the joy and sorrow that one encounters [84], 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.

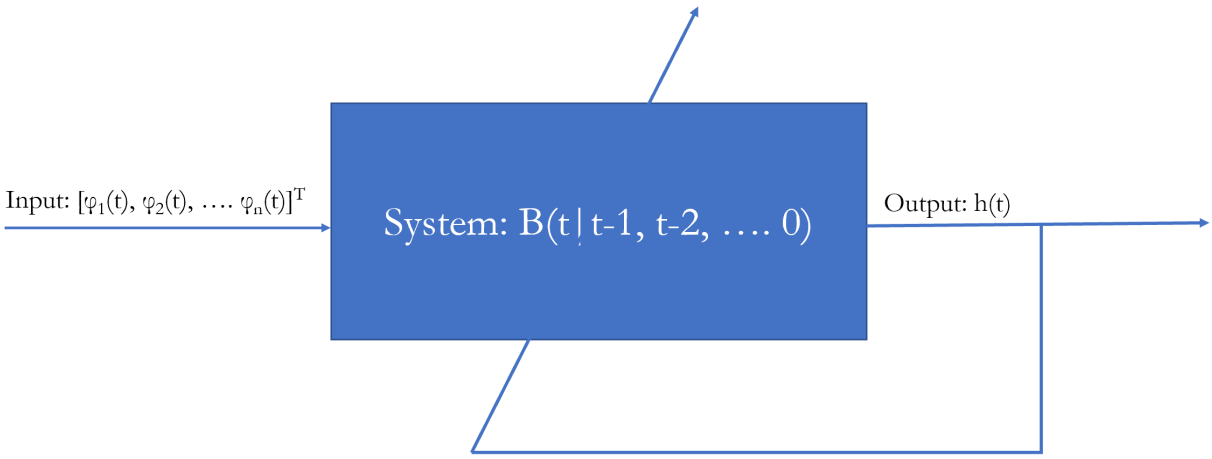


Figure 1: A diagram representing the happiness 'process' as a non-linear time-variant system.

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 [53].

The remainder of this project will investigate, design, and implement a technical framework to not only understanding more about the happiness system, but to provide users with suggested inputs that aim to maximise their individual happiness for their own 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 [45]. 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 its many ingredients.

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 in-depth understanding of what the term 'happiness' constitutes.

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* [41].

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 [88]. 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 [110]. 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 [105]. 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 [41]. 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. Furthermore, 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 [93](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 could be smoothed with sliding Moving Average windows, a common technique used in stochastic time-series analysis [5]. 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 [41], 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 an evaluative, and investigative measure, since from an analytical standpoint, it is the best representation of a sustained happiness.

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.

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 [77], 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 [96] [95], 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 community. 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 [95].

Seligman et al. provide clinical studies that highlight the positive effect on individual happiness as a result of a focus on positive psychology [55]. The idea of 'meaning' bears a strong resemblance to the idea of life satisfaction, 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 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.

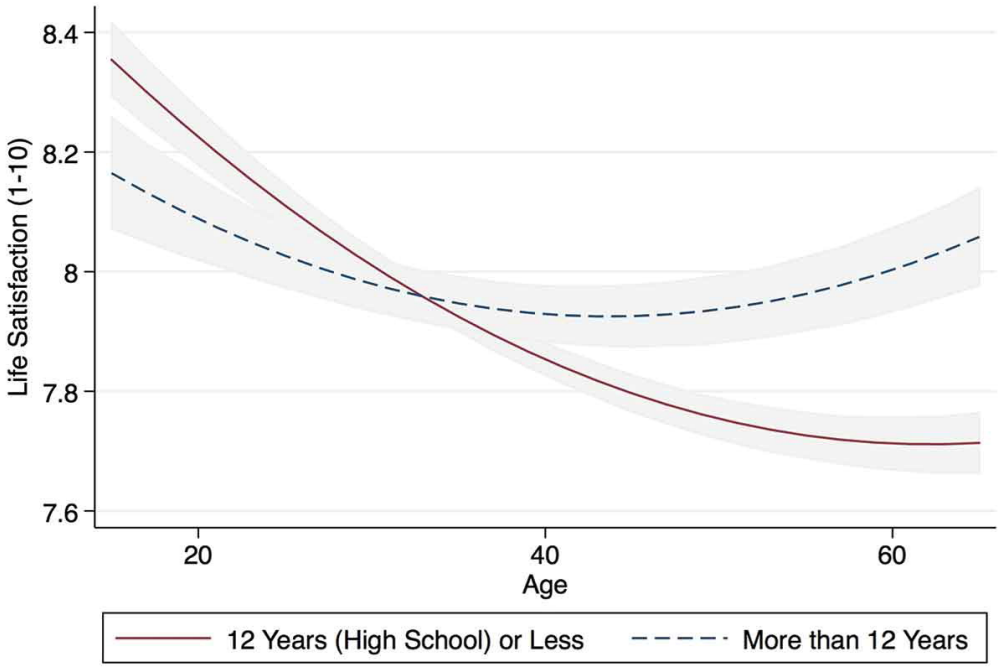


Figure 2: Figure showing the predicted life satisfaction (model based on data gathered on Australian population) at sample means for different ages, separated into two categories: people with more than 12 years schooling, and people with less than 12 years. [71]

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 relationship between income and life satisfaction, for example, has been shown to be relatively linear [51]. 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 [71], which has been shown to increase life satisfaction [51], however, this by no means is the only contributing factor, if it were, then the life satisfaction of the more educated, below $\sim$ 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 [9]. The research suggests that personality traits have a sustained impact on happiness and well-being. Despite this, the causal mechanisms and specific pathways through which personality influences both happiness and subjective 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 the subjectivity associated with 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, there is a strong indication of 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.

The final key conclusion was regarding 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, geographical, and physical 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 activities, 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 [56], i.e. the recommender system should be recommending relevant ideas and activities 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, is to provide an app for users, in which, alongside recommendations, will serve as a diary, where they can record the ideas and activities 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 offer some consideration to the integration of the following enhancements, 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 and activity spaces

To arrive at a three-dimensional map of philosophical texts, and likewise activities, like the one sketched in Figure 3, we will first require the positional encoding of both the activity and philosophy/idea spaces, necessitating the completion of the two following technical milestones:

- A sentiment-aware high-dimensional embedding of activities and 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. Similarly, the nature of an activity is far more dimensionally complex than a word, the meaning of 'football' on its own carries no information, but if we look at its use in the context of other words, a conceptual understanding of it can be built (from a mathematical perspective). Furthermore, with text, and activities, now being represented numerically, it will enable the use of them with mathematical models.

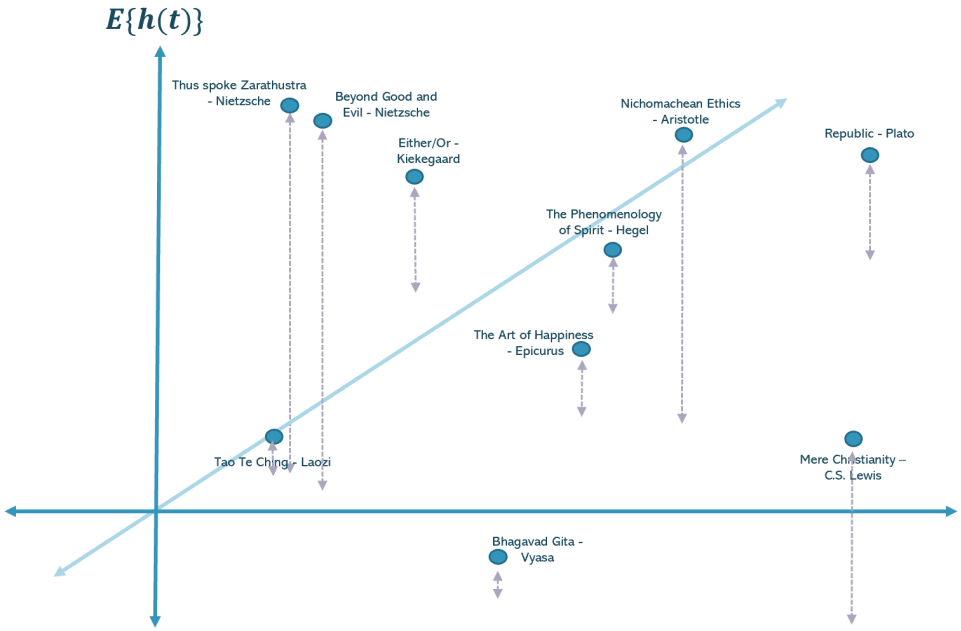


Figure 3: Figure showing a rough outline of what a map of philosophy texts may look like.

Although not necessary, the reduction of the high-dimensional vectors to the two-dimensional space allows 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 can 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 that undoubtedly exist 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 [87] [73], but also form the basis of more sophisticated techniques designed specifically for document, sentence, and paragraph-level corpora. The current most widely used, state-of-the-art technique for word-level encoding is the Bidirectional Encoder Representations from Transformers (BERT) [27], which borrows from the influential Transformer architecture [113], all of which will be discussed in further detail. Prior to BERT, Word2Vec [68] was one of the most widely adopted word embedding techniques.

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 [103]. 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 [89] 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 [68].

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

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 [68], 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.

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

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

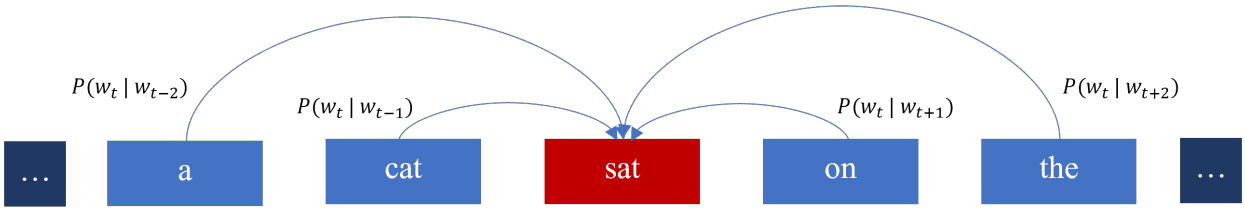


Figure 4: Figure demonstrating a CBOW example

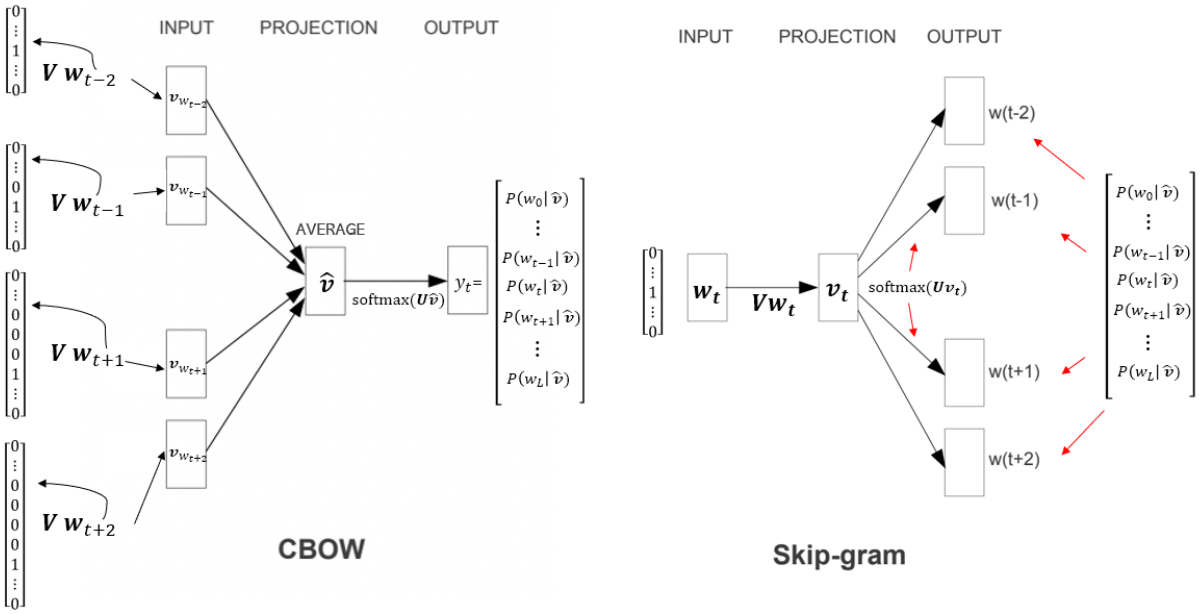


Figure 5: Diagram demonstrating the two Word2Vec architectures

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 [102].

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

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.

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 [67]. 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 [1]) 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 [98]. 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 [115] [78]. 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 [113]. 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 [121].

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

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.

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

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

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 [113].

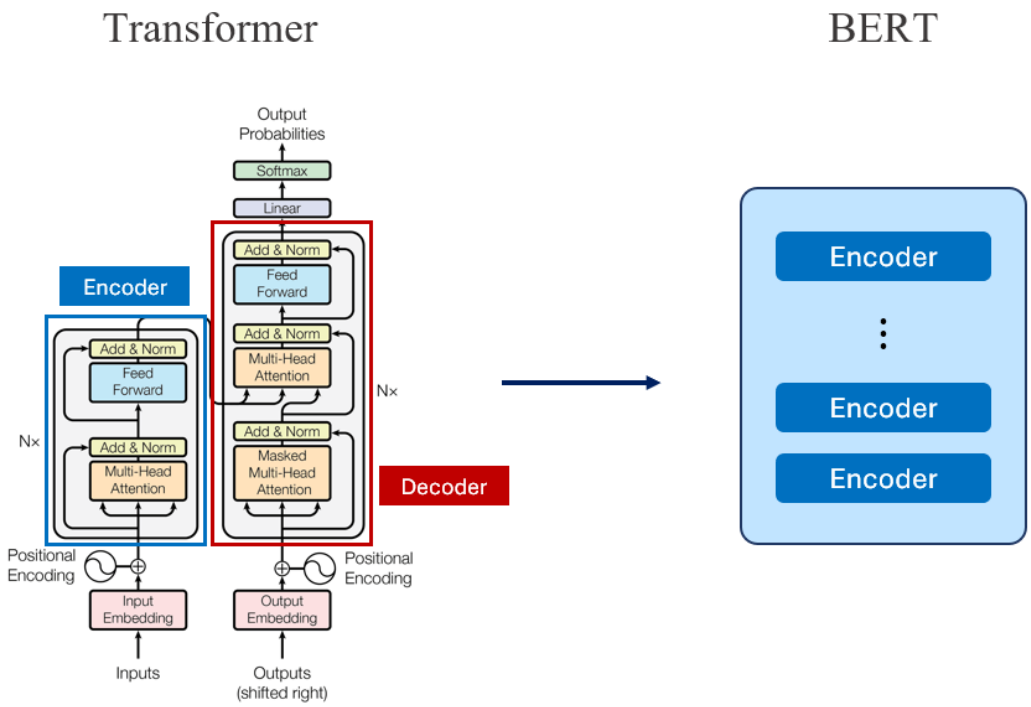


Figure 6: Diagram showing the BERT and Transformer architectures.

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

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 [113]. 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, Key, and 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 [6]. The $\mathbf{W}^Q$ and $\mathbf{W}^K$ matrices are learnt, and re-scale these similarity 'scores', learning, and informing the model what other words in the sequence to attend to for each specific word embedding. Softmax (equation 12) ensures the resulting weights add up to one,

preventing 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 [80]. Normalisation maps features onto a similar scale, it prevents exploding weights and thus helps stabilise the network [7]. 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 [80].

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

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

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 [119]), 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.

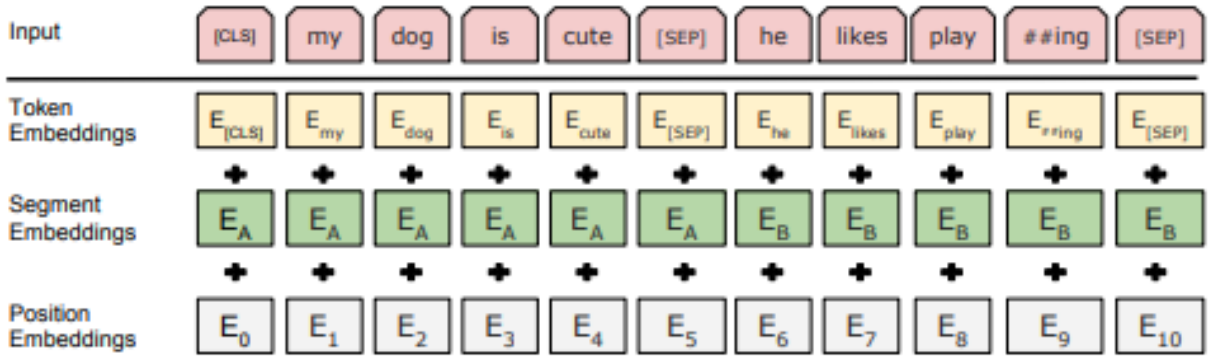


Figure 7: Diagram showing how input embeddings are constructed in the BERT architecture.

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 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 [60] [8] [82].

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 [76] on sentence-level tasks [82]. SBERT fine-tunes BERT pre-trained weights on a siamese-triplet network [91], 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 [82]), 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] [118]: *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]. It will be difficult to evaluate the performance of this method 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. Be that as it may, with the depth of training that has been done to obtain these embeddings, and its proven state-of-the-art performance in sentiment-based tasks, the SBERT embeddings will be used to obtain the high-dimensional embeddings of the activity and philosophy spaces that we are looking for.

2.2.4 Dimensionality Reduction

This section will define two methods that can later be used to reduce high-dimensional philosophical text embeddings 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 its n largest eigenvalues [49]. The Euclidean axes that define the space in which the data exists, 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 [49].

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

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 [120]. 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 [116], to tourism [37], movies and TV [13] [99] etc. Recommender systems can be broken down into two main approaches, content-based filtering, and collaborative-filtering methods, summarised in Figure 8.

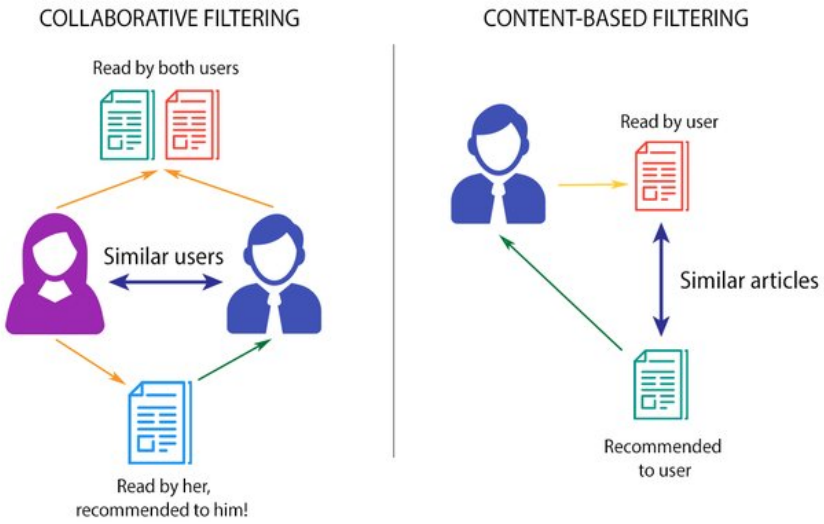


Figure 8: An overview of Content-Based Filtering and Collaborative Filtering recommender systems [112].

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 [86]. This can be extended to the context of our project, and, for example, philosophical texts exist on the 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, or cosine similarity 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 insights 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. Nevertheless, 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 [85]. 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 [90]. Memory-based approaches to CF, 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 outlining his solution to the Netflix prize competition [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 $\mathbf{U}$ and $\mathbf{V}$ 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 collective experience in order to understand more about happiness. A model based on CF is directly learning about the relationship between users and the content, whilst simultaneously improving 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 that have been discussed, should be adequately 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, 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.). Accordingly, information from both perspectives of the user-item paradigm in can be exploited in a 'hybrid' recommender system to provide better recommendations [70].

In the context of this project, supervised approaches will not be suitable due to the unavailability of pre-existing data. Taking this into account, semi-supervised approaches must be considered such that the model can adaptively learn 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 was 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 [83]. 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, in the past. 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:

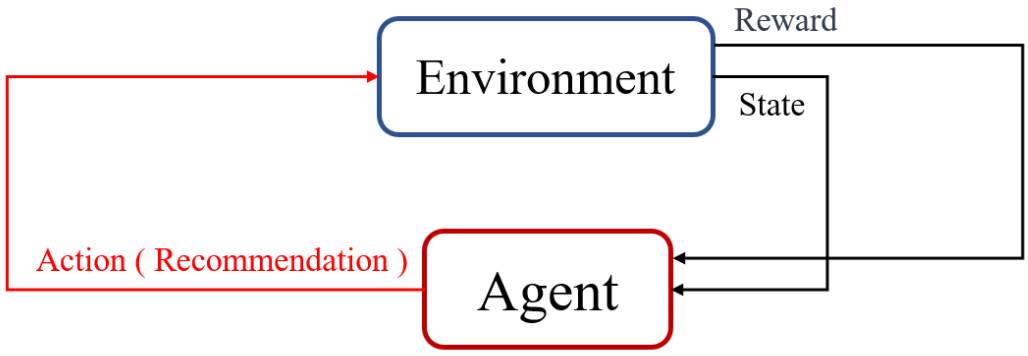


Figure 9: A diagram demonstrating a general RLRS.

More specifically, the main components of an RLRS are defined as follows [107]:

- **Policy:** Policy, denoted by π, 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 or 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. In general, the policy can be considered as 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 there is no existing model of the environment, it can be learnt by the agent through experience [107].

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. 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. This could, for example, be done by taking the action that yields the maximum Value [11]. 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. Finally, γ is the discount factor which 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 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 [117]:

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

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 [85], 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 [124], 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 [54].

Policy-based methods Policy-based methods, or often referred to as policy-gradient methods, such as Williams' REINFORCE, do not learn the value function and instead work with parametrised policies, and optimise the cost (either the discounted reward or the average reward) over the parameter space of the policy [54]. 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:

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

where r_t is 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).

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta J(\theta) = \theta_t + \alpha \cdot \hat{Q}(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t) \tag{22}$$

Where α is suitably chosen such that the update satisfies the descent direction condition. 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 [108]. 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 combine the strong points of both methods in a dual policy-value learning algorithm [54].

Reinforcement Learning: Actor-Critic Models Actor-Critic (AC) models combine Policy (actor) and Value (critic) based methods in a dual optimisation problem. The policy is parametrised as before, denoted by $\pi_{\theta_\pi}(a|s)$, 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.

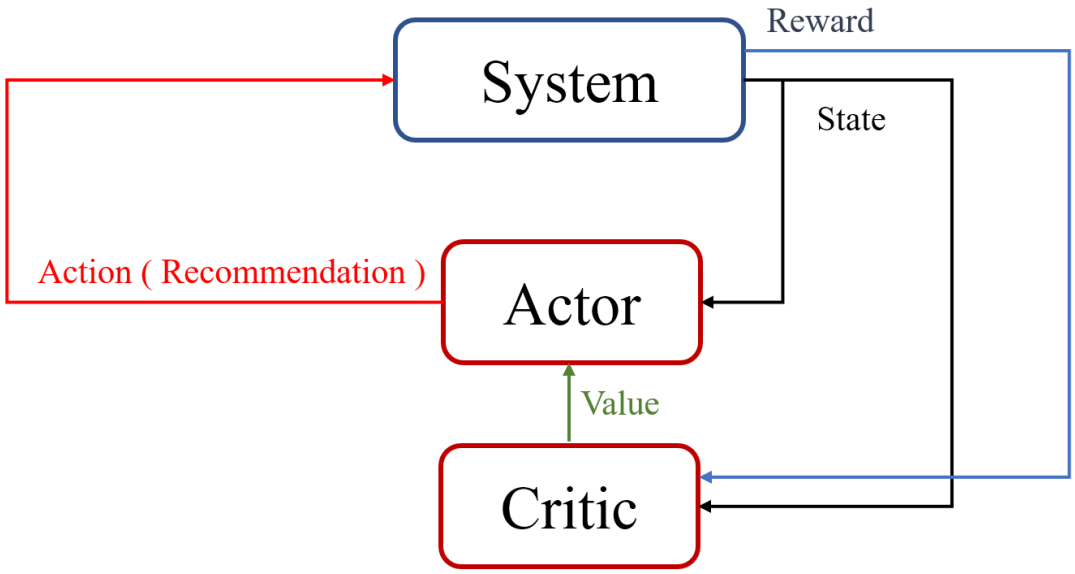


Figure 10: A diagram demonstrating the general pipeline of an AC learning algorithm.

The actor parameters θ_π are updated using the policy gradient, as in equation 22, except now the Q-function is learnt in a separate critic update algorithm:

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

$Q(s, a)$ 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 [106] 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 by following policy π_{θ_π} 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 possible, 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 how users and items are interacting.

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 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 cumulative future happiness, which can be likened to something closer to a focus on life-satisfaction.

Deep Actor-Critic Despite the improvements that it offers over Value and Policy based methods, the same problems associated with the sparsity of the user-item space still apply to the Actor-critic. A major milestone in RLRS has been the integration of deep learning methods, which have shown significantly improved performance when used in the huge state and action spaces present in some recommender systems [83]. 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 Actor-Critic method can be parametrised via Neural Networks to help produce better results. As aforementioned, 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 [44] [24], are investigated. 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 [79]. 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.

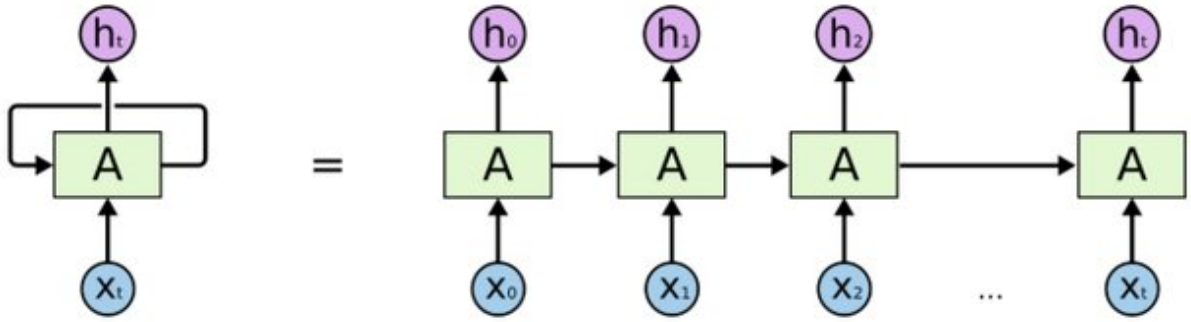


Figure 11: Diagram showing an unrolled RNN [79]

A forward pass through the RNN is characterised by the set of equations 26-29 [39], 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 [97]. 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 [39] [43]. 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 [39]. 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 [43]. 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 [44], the schematic of which is shown in Figure 12.

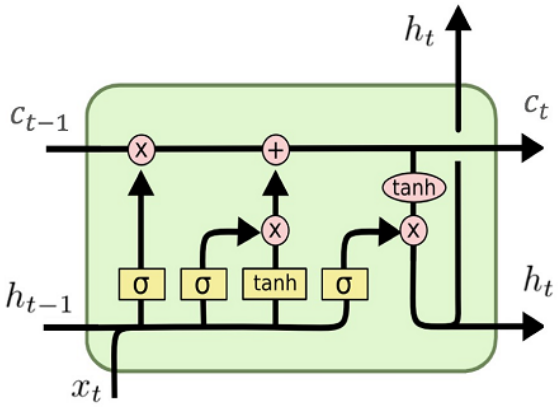


Figure 12: A visualisation of an LSTM model [72]

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.

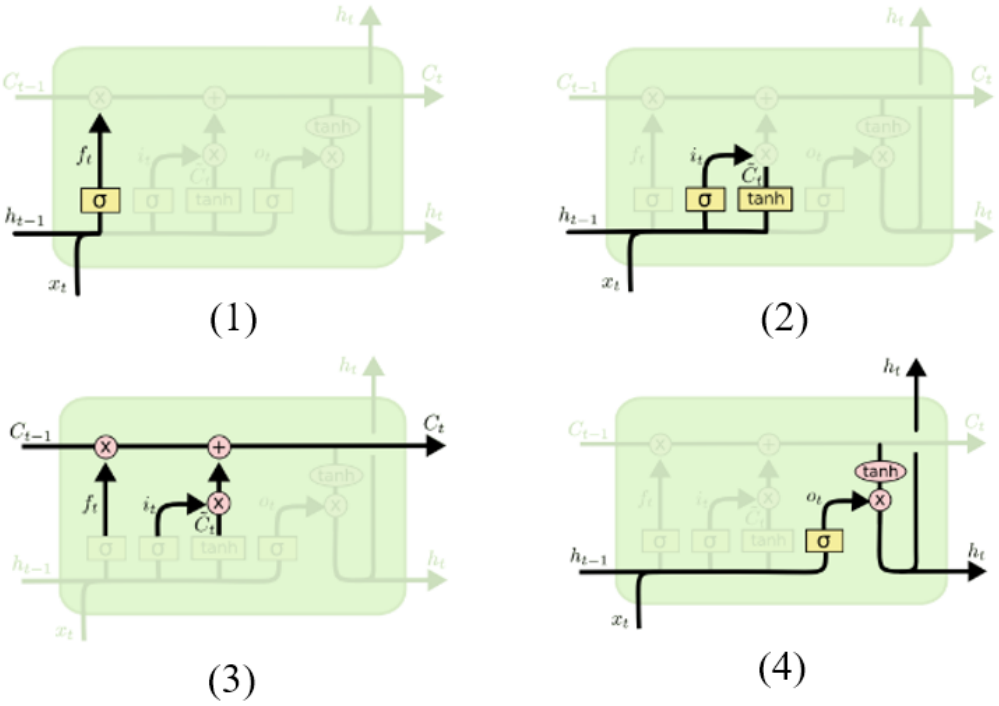


Figure 13: A decomposition of a forward pass in the LSTM model [72]

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

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

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:

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

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^l, 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 is independent of $\mathbf{c}_{t-1}$. Considering this, 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 [74]. 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 [39]. 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.

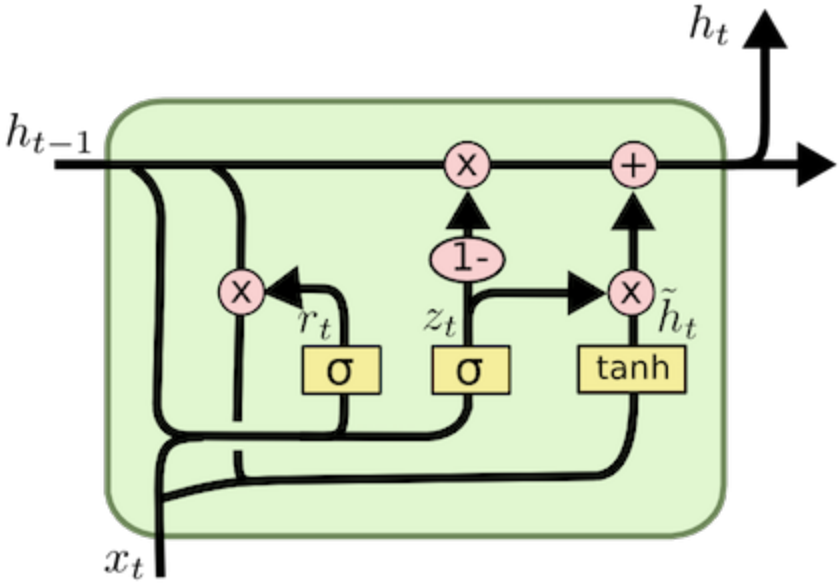


Figure 14: Schematic of a GRU [123].

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 capture this. One of the most widely adopted scales is Lyubomirsky et al.'s Subjective Happiness Scale (SHS) [63].

The SHS consists of four straightforward questions that capture different aspects of subjective happiness. Individuals rate each item on a seven-point Likert scale [50], 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 indicate 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 [65] [62] and showed high internal consistency as well as stability across all samples [62].

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 16. From a high-level perspective, this is a dual recommender system, an 'idea' and an 'activity' are being recommended to the user, owever, these are not independent recommendations, 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. Equally, if no resonant points exist in actuality, the model will also reflect this.

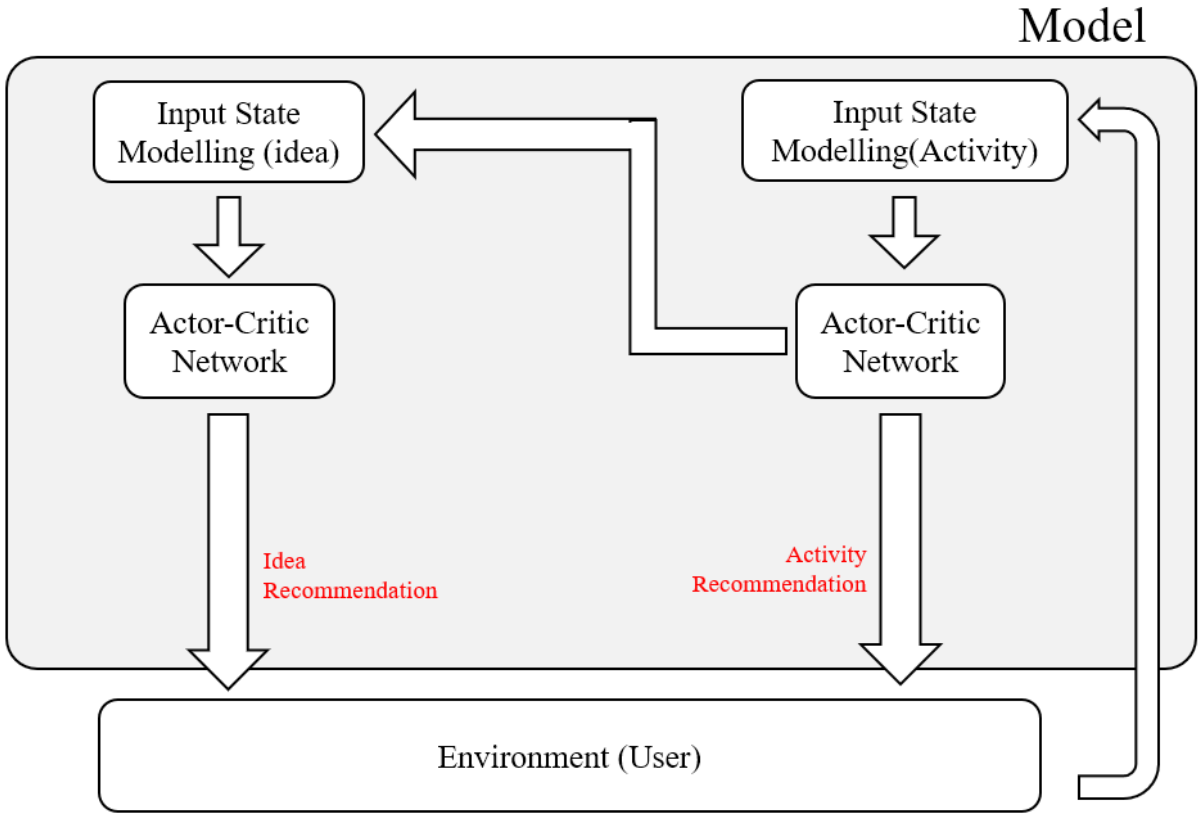


Figure 15: Diagram showing the design of the recommender system architecture.

Despite the inspiration from Seligman's theory of positive psychology, our exploration of this relationship is by no means rendering the theory as true, 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 this project

in particular, the pipeline in Figure 15 is implemented.

In Figure 15, there's an assumption that the idea and activity spaces have been defined, this is not the case, and 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).

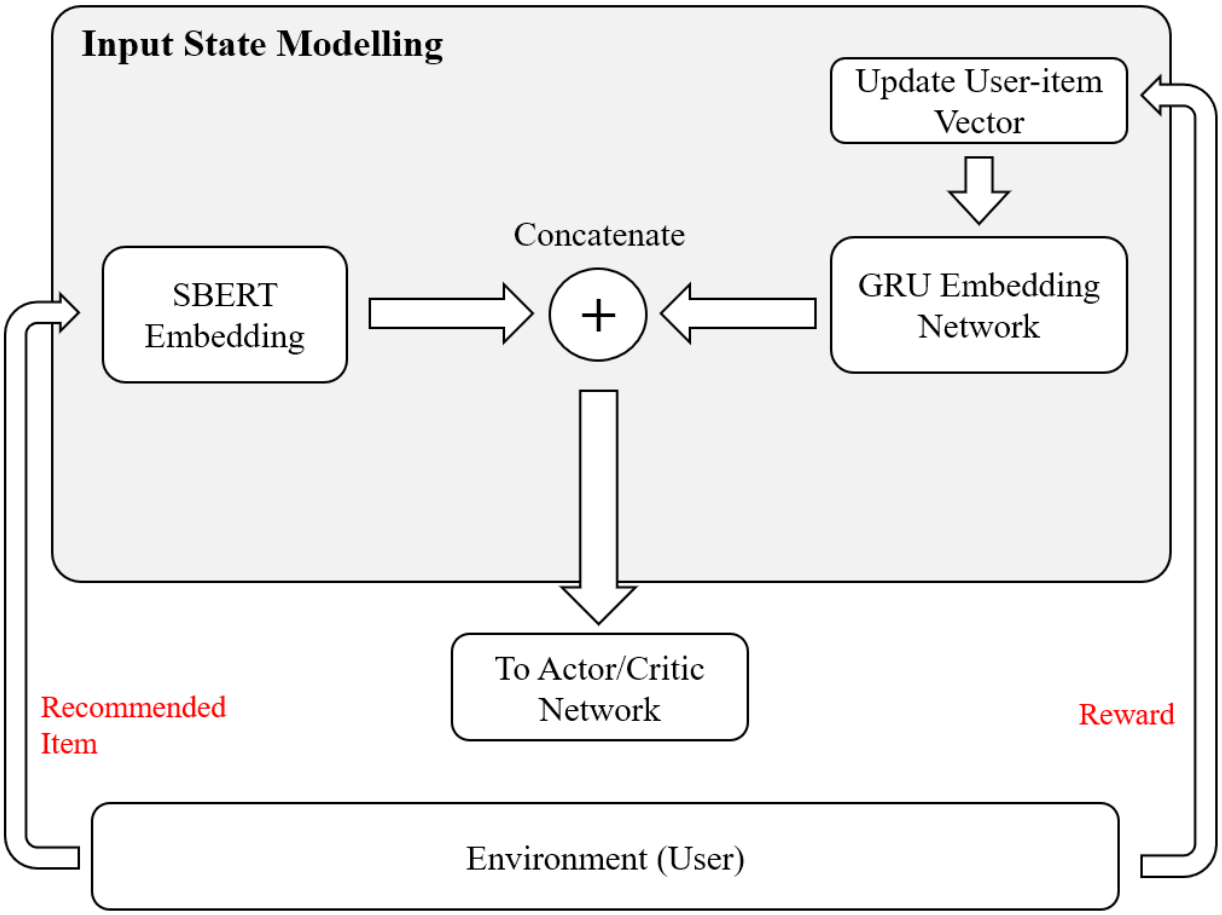


Figure 16: Diagram showing the design of the Input state modelling pipeline.

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. 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 a similar philosophy to MF is considered, 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) is 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 [126]. In this way, users with similar preferences and patterns will be mapped to similar output vectors.

The embedded user and item vectors for activities are obtained, 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 output of the actor network yields a probability 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 their 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 an appropriately defined loss function for the actor and critic networks. This loss is then backpropagated through the network 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 [3] [2] 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, [4].

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 [57] providing the best results from a qualitative standpoint. BART is a transformer-based sequence-to-sequence model, with a BERT-like encoder and GPT-like decoder, that has been fine-tuned on extractive summary tasks.

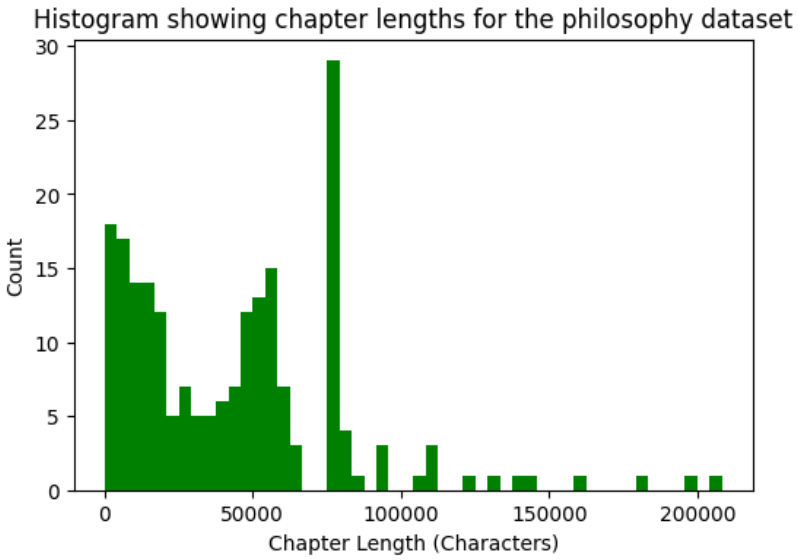


Figure 17: Histogram showing the lengths of chapters in the philosophy database.

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 [111], and filtered such that the remaining quotes were somewhat actionable on a personal level, leaving 184 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. 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
```

As for obtaining a list of activities, first, methods were investigated that could potentially extract direct 'advice' from philosophical texts, however, due to a lack of training data, there was no sophisticated model that was tuned for such a task. Instead, the internet was searched a definitive list that could be used, but there wasn't anything worth noting to be found, so instead, a brainstorming session was organised with peers from a variety of backgrounds. Although this is not ideal from a formality standpoint, it was deemed sufficient for proof-of-concept purposes. The result was a list of 84 preliminary activities, summarised in one, or a few words, e.g. 'Play Football', 'Do laundry'.

The full details on how to access this data, is included in the User Guide (9).

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 relationship of between each item is measured and stored. For embedding vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, the relationship is calculated via cosine similarity $c_{ij} = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{||\mathbf{x}_i|| ||\mathbf{x}_j||}$. Suppose there are L items in the activity set, then the resulting set of similarities for item i is stored in the vector $\mathbf{w}_i \in \mathbb{R}^L$ and the matrix whose columns (and rows) are formed by the similarity vectors $\mathbf{w}_i$, is denoted by $\mathbf{W}_1 \in \mathbb{R}^{L \times L}$, and is symmetric. For idea recommendations, the weight matrix is given by $\mathbf{W}_2 \in \mathbb{R}^{L \times M}$ where M is the number of ideas in the dataset.

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 most similar to i have a higher weighting, $w_{i,j}$, 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:

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

Contextualising this into our problem, activities define both the state and action spaces, meaning that the policy π_1 is parametrised by the cosine similarity 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 cosine similarities 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

Due to being a content-based approach, the described method doesn't suffer from the cold-start problem. Having said that, the model has no technical pipeline for making an accurate initial recommendation. 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 [75]. 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 [75]. Despite its popularity and widespread use, the reliability and validity of the indicator has been deemed as being limited by various studies [94] [81] [104].

The five-factor model of personality (FFM) is instead consulted, which has been widely accepted by the scientific community [40], and has shown to transcend language and cultural differences [15] [122].

The FFM is a set of five broad trait dimensions or domains, often referred to as the "Big Five": Extraversion, Agreeableness, Conscientiousness, Neuroticism, and Openness to Experience. Soto et al. describe each of the five factors as: "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 FFM was specifically designed to capture as much of the variability in individuals' personalities as possible, whilst minimising the number of trait dimensions. Many personality psychologists agree that its five domains represent the fundamental differences in personality traits, and that many models can be based off of, or at least related to, the FFM [101].

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 [64].

In a paper introducing the FFM and its applications, McCrae et al. provide lists of adjective descriptors for each of the five factors [66]. 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 cosine similarities between each activity and the factor descriptors will determine the probability of that activity being recommended to them, in the same manner as before. There is still a heavy reliance here on the accuracy of SBERT embeddings, but for a baseline model, it is deemed sufficient.

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 rewards at time t for the activity and idea recommendations, $r_{1,t}$, and $r_{2,t}$. where $\mathbb{E}[r_{1,t}] = \mathbb{E}[r_{2,t}] = 0$. For the activity recommendation, we assume the initial weight matrix $\mathbf{W}_1 \in \mathbb{R}^{L\times L}$, and for the idea recommendation, the weight matrix $\mathbf{W}_2 \in \mathbb{R}^{L\times M}$.

For a previously recommended activity i, and the newly recommended activity j, and newly recommended idea k, the probability weightings are updated as follows:

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

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

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.

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, details of where this can be found is discussed in the User Guide (Section 9). 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 GRU 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 Buffer**: The user-item buffer will be used to store the past 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 user-item buffer consists of a set of the past T user-item vectors, which hold the last given reward for each action. The user-item buffer is then passed through an embedding network, composed of GRU units such that a fixed, low-dimensional, temporally-aware embedding is obtained. The resulting embedding is then passed through the actor/critic networks.

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 [83]. This 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, since it has the ability to capture the inter-country and inter-culture influences on experienced happiness. Even if this additional feature does not help

enhance recommendations, it will at least provide an indication of the effect of culture/nationality on what ideas and activities bring happiness. It can be represented by a 2-Dimensional latitude and longitude number vector.

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	12 (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: Summary of input features into the densely connected Actor and Critic neural networks.

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 buffer $\mathbf{X}_t$ which stores the user-item vectors, $\mathbf{x}_t$ for each of the past T interactions, such that $\mathbf{X}_t = \{\mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \dots \mathbf{x}_{t-T}\}$. The user-item buffer is then passed through two stacked GRU layers (the second allowing for additional depth of temporal representation), each with T consecutive units (they are folded in Figure 19). The output of the input state embedding network produces a fixed 12-dimensional embedding, $\mathbf{h}_t^2$. The selection of the output dimensionality of 12 is justified in the Results section (5.2), for now it is assumed.

The previous recommendation, $\hat{y}_{t-1}$, 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. The SBERT network was made to be untrainable since it significantly increased the computational complexity of the model, and meant that training was far too slow. The SBERT embedding is then input into four consecutive densely connected neural network layers with ReLU activation (equation 14. It is then passed 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. Resultantly, the dimensionality of the final dense layer is equal to the dimensionality of the action space, (`action_dim`).

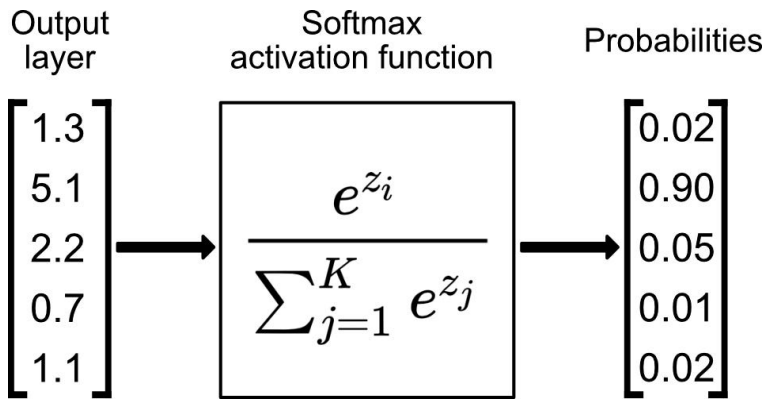


Figure 18: Diagram demonstrating the softmax activation function.

Prior to inputting data into the model, features are individually scaled with a MinMaxScaler, which scales an input vector according to equation 46:

$$\mathbf{x}' = \frac{\mathbf{x} - x_{min}}{x_{max} - x_{min}} \tag{46}$$

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, and allows for faster and more stable learning [42]. The architecture of the network is the same for both idea and activity recommender systems, the only difference being the output 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.

Batch Normalisation is performed between the second and third layer, as this has proven to help stabilise and accelerate learning [46].

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.

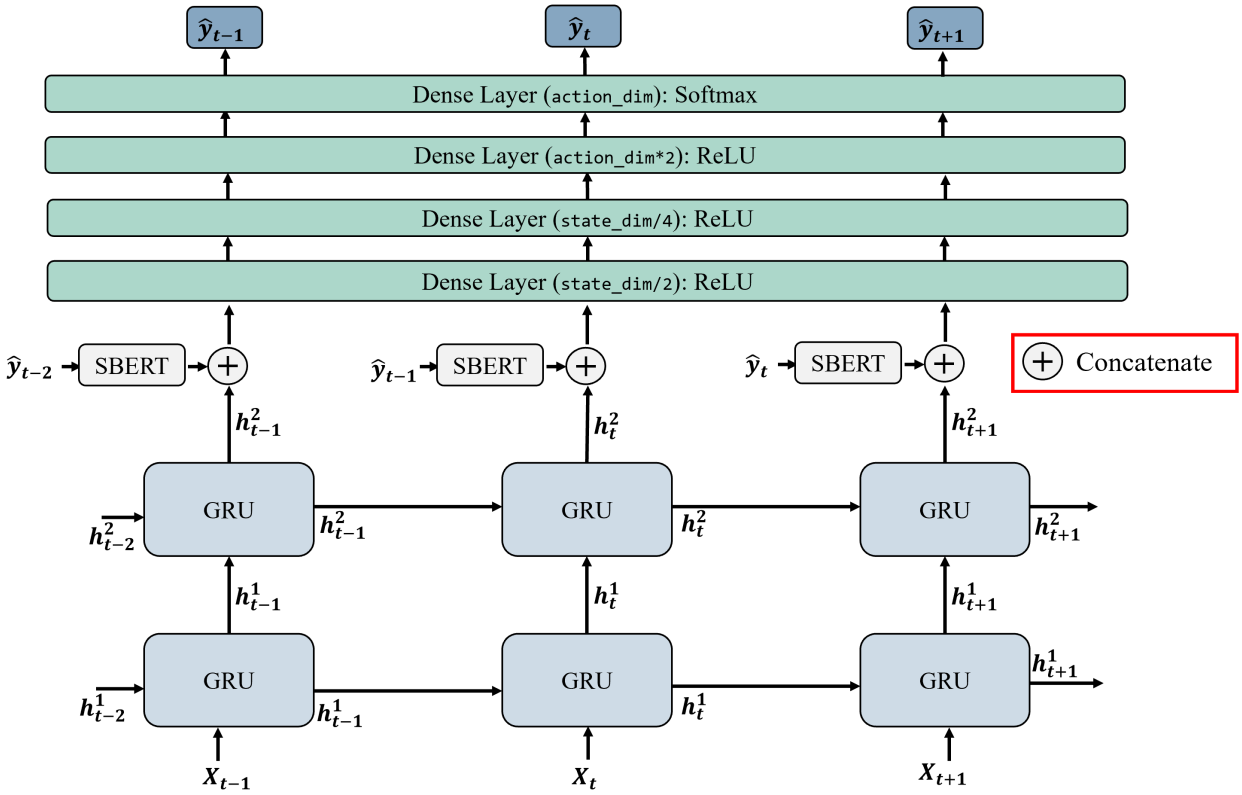


Figure 19: Schematic of the actor network.

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 [69], which includes the integration of the background in Sections 2.2.6 and 2.2.7, with a slight adjustment to the Q actor-critic methodology that was described. 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, $\hat{A}(s_t, a_t)$, which is the expected value of the Temporal Difference (TD) error (equation 24), or equally, the 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:

$$\hat{A}(s_t, a_t) = Q_{\theta_Q}(s_t, a_t) - V_{\theta_V}(s_t) \tag{47}$$

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

$$\hat{A}(s_t, a_t) = r_t + V_{\theta_V}(s_{t+1}) - V_{\theta_V}(s_t) \tag{48}$$

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

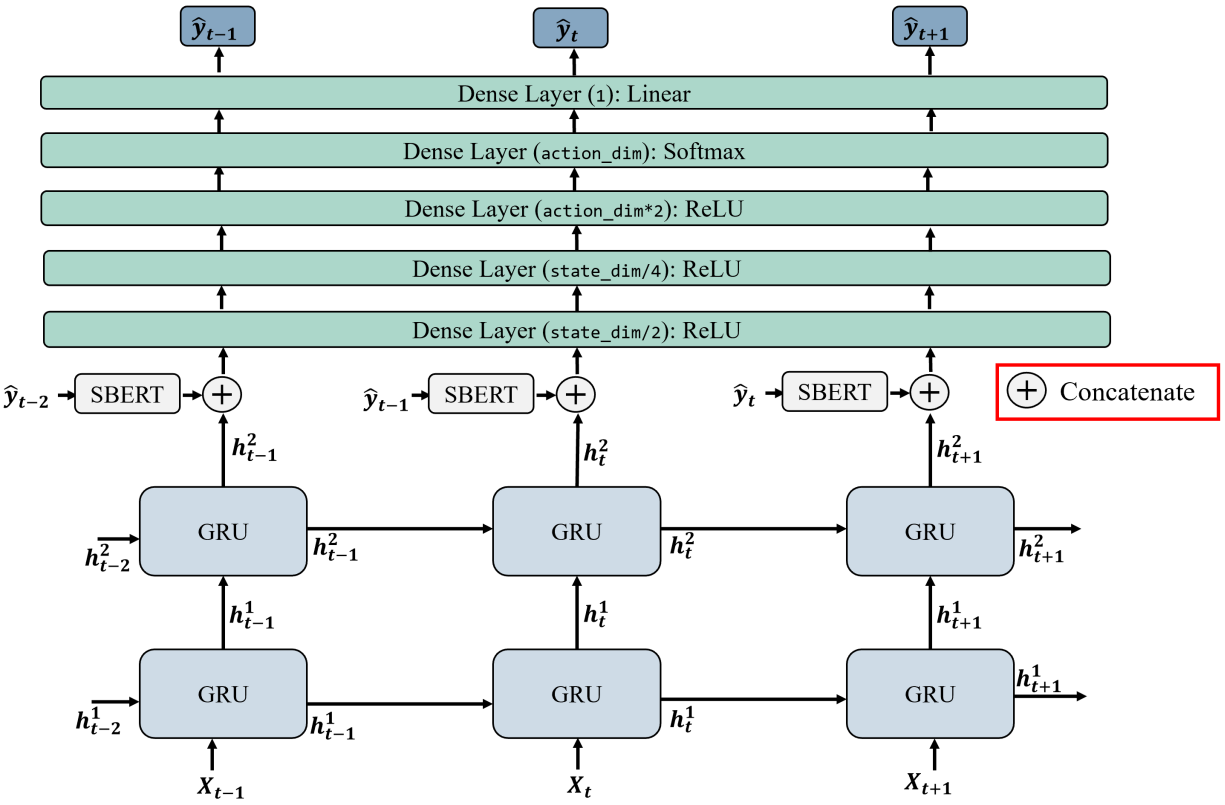


Figure 20: Schematic of the critic network.

variance, and is equivalent to the td-error. On the other hand, when the advantage is instead estimated using the sum of the discounted rewards (equation 49), it has a high variance and low bias [92].

$$\hat{A}(s_t, a_t) = R_t - V_{\theta_V}(s_t), \qquad R_t = \sum_{k=0}^{T-t} \gamma^k r_{t+k} \tag{49}$$

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 [92]. To begin with, consider k estimators of the advantage function with increasing order of contribution from discounted rewards, i.e.:

$$\hat{A}_t^{(1)} = \delta_t^V \qquad = -V(s_t) + r_t + \gamma V(s_{t+1}) \tag{50}$$

$$\hat{A}_t^{(2)} = \delta_t^V + \gamma\delta_{t+1}^V = -V(s_t) + r_{t+1} + \gamma r_t \gamma^2 V(s_{t+2}) \tag{51}$$

$$\vdots$$

$$\hat{A}_t^{(k)} = \sum_{t=0}^{k-1} \gamma^l \delta_t^V = -V(s_t) + r_{t+1} + \gamma r_t + \ldots + \gamma^{k-1} r_{t+k-1} + \gamma^k V(s_{t+k}) \tag{52}$$

Clearly,

$$\hat{A}_t^{(\infty)} = -V_{\theta_V}(s_t) + \sum_{k=0}^{T-t} \gamma^k r_{t+k} \tag{53}$$

As in equation 49, which is the difference between the empirical returns and the baseline value function [92] . The GAE(γ, λ) is defined as the exponentially weighted average of those k estimators, such that:

$$\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} = (1-\lambda)\left(\hat{A}_t^{(1)} + \lambda\hat{A}_t^{(2)} + \lambda^2\hat{A}_t^{(3)} + \dots\right) \tag{54}$$

$$= (1-)(\delta_t^V + \lambda(\delta_t^V + \gamma\delta_{t+1}^V) + \lambda^2(\delta_t^V + \gamma\delta_{t+1}^V + \gamma^2\delta_{t+2}^V) + \dots) \tag{55}$$

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

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

$$= \sum_{t=0}^{\infty}(\lambda\gamma)^l\delta_{t+l}^V \tag{58}$$

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 48 will be preferred initially at the cost of a high bias in the advantage estimation, since it allows for network parameters to be updated iteratively after a single observation of the environment. 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 (using policy gradient), but instead using the estimated advantage function as a guide, such that:

$$\nabla_\theta J(\theta) \approx \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot \hat{A}(s_t, a_t) \tag{59}$$

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 [52]. The value network is updated by minimizing the squared td-error [58]:

$$\mathcal{L}(\theta_V) = \sum_{t=1}^{T}(\delta_t^V)^2 = \sum_{t=1}^{T}(r_t + \gamma V_{\theta_V}(s_{t+1}) - V_{\theta_V}(s_t))^2 \tag{60}$$

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 designed, the full model was implemented in a class in python. Details of where to find the code digitally can be found in the User Guide (9).

4.4 Designing a reward function

Designing a suitable reward function that correctly captures the aims of our project 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) [63], 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. Accordingly, we 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 capture the quality of the recommendation, the relevance of the recommendation towards an individual's happiness, and their happiness in general, whilst allowing the room for reflection and subjectivity that we have outlined as being crucial.

4.5 Proposed solution to the cold-start problem

In this section, we leverage the a priori information in our arsenal to propose methods that aim to deal with the cold-start solution inherent to recommender systems, and thereby allow for more favourable performance in the early stages of model deployment where interactions with the environment have been few.

One of the requirements of the user is to complete an FFM personality assessment prior to receiving recommendations—how can the information gathered through this process be leveraged into making better initial recommendations? A similar approach to the one explored in Section 4.2.2 is considered. Using the same list of adjective descriptors for each of the five factors given by McCrae et al. [66], the cosine similarity between those descriptors and each of the activities is measured, which essentially assigns a 'similarity score' between each personality factor and the activities. This process should, in theory, find which activities best match each personality factor. The top 5 activities for each of the personality factors is summarised in the table below:

Extraversion	**Agreeableness**	**Conscientiousness**	**Neuroticism**	**Openness**
Organizing	Volunteer	Organizing	Decluttering	Organizing
Volunteer	Organizing	Volunteer	Acting	Acting
Acting	Meet friends	Style clothes	Organizing	Style clothes
Style clothes	Cleaning	Cleaning	Vacuuming	Make music
Fundraising	Fundraising	Acting	Drawing	Draw

Table 2: Most similar activities to each of the five personality factors in the FFM.

Now, with the similarities between each activity and personality factor found, the user's percentile for each of the factors is used to weight the scores from each factor into a single score for the user, such that where an individual is more attributed to a factor, the weighting of receiving activities best suited to that factor, are higher. Similarities are then linearly scaled between -5, 5, obtaining a set of reward scores for each activity. The model can then be pre-trained on this set of rewards for the duration of time that it takes to complete the FFM questionnaire (5-10 minutes), such that the tool is ready to be used once the questionnaire has been completed.

Looking at Table 2, there seems to be a select few activities that appear in each of the five factors, and this highlights an inherent problem with the use of SBERT embeddings to compare items from separate contexts. Even though there may be an abstract connection between activities and personality descriptions, there is no guarantee that the model will capture this since it has been trained in a general sense, and not in the context in which we seek to use it. Nevertheless, since all information we have exists in the realm of language, and the inexistence of models that deal with such a problem, we are restricted to the use of state-of-art general purpose embedding models like SBERT embeddings.

Considering the above, a second solution to the cold-start problem is proposed. Prior to taking the personality tests, the user can be asked to list a few activities that bring them the most happiness for each of the four seasons, which do not necessarily have to exist within the activity space. The SBERT embeddings of the activities listed by the user are then used to find the cosine similarity between them and the activities available in the activity space, and these are used in the same way as before to construct a set of rewards. Since there are a few activities listed for each season, the maximum similarity between each activity (in the activity dataset) and the listed activities, given by the user, are used to determine their respective scores.

This set of rewards can then be used to pre-train the model over a year of simulated interactions, this process takes roughly 6 minutes on an NVIDEA GTX 1060 GPU, which is roughly the amount of time it takes to complete the FFM questionnaire, and thus is a suitable means of pre-training. Furthermore, these sets of rewards for each season, can be used as the input user-item vector at the beginning of each season to help improve performance even further. Since the items that are being compared via SBERT embeddings exist within the same context (activities), the performance of these comparisons is now far more satisfactory. For example, the closest activities within the dataset to 'Going to the park', are: 'Go to a festival', 'Go to the beach', 'Go on holiday', 'Gardening', all of which are similar, outdoor activities. Its least similar activities were given by: 'Vacuuming', 'Doing laundry', 'Write poetry', 'Paint nails' and 'Washing dishes', all of which are indoor activities that are qualitatively dissimilar to an activity such as going to the park.

For idea recommendations, a similar approach is proposed, the SBERT embeddings for the philosophy quotes are clustered into n clusters using the k-nearest neighbours algorithm [125] using the cosine similarity as the distance metric. The quotes closest to each of the cluster means are presented to the user, who are asked to feed back which of these they resonate best with, the selected quote will determine, as done with activity embeddings, the rewards assigned to each quote in the idea space, i.e. the closest quotes to the selected quote are assigned the highest score. These rewards are fed back to the model, independent of the activity input into the idea network, since there is little known of the relationship between the two that can be leveraged. The idea network, like with the activity network, is pre-trained in an environment based on these rewards.

5 Results

With the unavailability of real-world data for our given problem, and the time-limitations that meant that the model could not be employed under full testing conditions, in this section, the behaviour of the proposed model is tested in a semi-simulated scenario, the results of which are used to draw conclusions regarding the happiness of each user, and investigate the effect of personality embeddings. The agent (recommender system) exists in an almost infinitely complex environment, and so how is data to be simulated for a system whose behaviour we don't understand, and how can we draw conclusions from a ground truth that is founded upon a lack of understanding? The simplifications that will have to be made, mean that the complexities that we have attempted to capture via our AC model, essentially become obsolete. For this reason, an experiment is designed that takes responses from human subjects and places some additional assumptions in order to build the testing environment.

5.1 Experiment Design

The experiment that was carried out involved five human subjects, each presented with every activity in the activity space. They were asked to give a score for the following two items, which have been slightly adjusted from those discussed in the Section 4.4, to account for the fact that scores are being given in hindsight/foresight rather than in real-time:

1. On the whole, I believe this activity can contribute positively to my overall happiness.
2. This activity matches my interests well.

The miscellaneous item: "*On the whole, I am satisfied with my life, and I consider myself generally happy.*" is omitted since this question can only carry information temporally. Furthermore, we focus solely on the activity network. This came down to two main reasons, the first being that all the activities are familiar to the user, and have either been experienced by them in the past, or they have some conceptual understanding of what they entail, whereas many of the ideas are novel and unfamiliar to the user. The idea in itself is unattributed to happiness, instead, we aim to understand how these ideas manifest in experience, and thus it is difficult for the subject to give an evaluation of this when they have little a priori knowledge or experience with such ideas. The second reason is logistical, since the reward of ideas are dependent on the activity, this means that data must be collected over the joint activity-idea space, i.e. the user must evaluate the reward of every idea for every activity, in the case of our dataset, over 10,000 items. Nevertheless, aside from the spaces in which they operate, the models for both idea and activity are essentially identical, and so focusing on one is deemed sufficient in studying the performance of our proposed model.

To add some temporal depth to the experiment, without overwhelming subjects with items, they were asked to give separate answers for each season, leaving four data points per item. The response of each user for each season defines the reward that the agent will receive when

recommending that item. Now, to simulate some additional, and realistic, intra-seasonal temporal complexity, and to prevent the model from collapsing onto a single, repeated recommendation of the action with the highest reward, we propose the following property to the environment: if the same action is recommended n-times in a fortnight, then the reward, if positive, is multiplied by a factor of 0.5^n. If the reward is negative, then it is multiplied by a factor of 2^n, to penalise the model for repeated 'bad' recommendations. In technical terms, the past 14 actions are kept in a buffer, $b_t = \{a_{t-14}, a_{t-13}, ..., a_{t-1}\}$, and using Iverson bracket notation [47], it follows that the reward at time t is given by:

$$r_t = \begin{cases} r_{season,j,i} \cdot 0.5^{C(t)}, & \text{if } r_{season,j,i} > 0 \\ r_{season,j,i} \cdot 2^{C(t)}, & \text{if } r_{season,j,i} < 0 \end{cases} \tag{61}$$

Where $r_{season,j,i}$ is the score given to activity j, by user i for a particular season, *season*, and:

$$C(t) = \sum_{i=1}^{14} [a_{t-i} = a_t] \tag{62}$$

All subjects had the same age (23) and were from the same location (London). Their personalities were measured according to the International Personality Item Pool (IPIP) FFM questionnaire [38], and used as inputs into the model—an overview of the obtained data can be seen below:

User	**Mean reward per season**	**FFM Percentiles**
1	[0.46, 0.52, 0.84, 0.46]	[0.78, 0.48, 0.56, 0.62, 0.45]
2	[2.35, 1.98, 2.38, 2.2]	[0.68, 0.3, 0.58, 0.27, 0.41]
3	[-0.85, 0.28, 0.2, 0.12]	[0.46, 0.25, 0.01, 0.86, 0.06]
4	[-0.13, -0.53, -0.61, -0.12]	[0.65, 0.13, 0.89, 0.22, 0.76]
5	[0.94, 0.92, 1.11, 1.11]	[0.88, 0.69, 0.76, 0.09, 0.88]

Table 3: Table summarising the collected user data, with mean reward showing the average reward assigned to each activity for a particular season in the following order: *Winter, Spring, Summer, Autumn* (left to right). The FFM column shows the user's respective percentiles for each of the five factors, *Extroversion, Emotional Stability, Agreeableness, Conscientiousness, Intellect/Imagination* (left to right).

Using this semi-simulated environment, the behaviour of the model will be tested in a variety of experiments with a particular focus on its performance 'out-of-the gate', i.e. if it were to be employed, and used right now.

5.2 Hyperparameter Selection

We now seek to select the optimal hyperparameters for our model, according to the data that was collected, and the reward framework that was determined in the previous section. The hyperparameters that will be tuned are as follows:

- User-item embedding dimension: We seek to find the optimal user-item dimension, this will determine the amount of information that is carried forward into the respective actor and critic densely connected neural networks.
- Discount factor: Here, we seek to find the overall optimal (optimal over all users) weighting of future rewards in the td-error calculation.
- User-item buffer size/GRU Sequence length: This will determine the ideal amount of prior user-item interactions to consider, a buffer size of, T for example, will consider the last T user interactions when producing a user-item embedding that is fed to the actor or critic network. The buffer size corresponds to the resulting GRU sequence length when producing an embedding.
- Learning Rate: The learning rate determines the sensitivity of network parameters to their respective loss functions. A larger learning rate means that the network is more 'responsive', i.e. it will adapt to changing rewards quicker (for example at the turn of a season), but too large of a learning rate may cause divergence from optimal parameter values.

The procedure for empirically selecting the listed hyperparameters above, is as follows: when tuning a particular hyperparameter, all other hyperparameters are fixed, and the model is run consecutively over a period of one year (one recommendation per day) for each of the subjects, and the cumulative reward above the average reward for that particular season is used as the evaluative metric. The results from each of the experiments is summarised in Figure 21

The optimal embedding length/dimensions of the user-item vector was empirically found to be 12, with the performance dropping after a dimension of 16. The user-item embedding dimension must find a balance between providing sufficient depth of information without propagating redundant information to the actor and critic networks. For an embedding length of greater than 12, the embedding is increasing the complexity of the network whilst providing little extra, relevant information. Not only does this increase computational complexity, but the greater the number of parameters, the slower the network is to learn, hence the drop in performance. Interestingly, an embedding of length 2 offers better performance than an embedding of length 4 and 8, perhaps there is a 'sweet-spot' here in terms of the balance between simplicity and depth for this given environment.

The optimal sequence length/buffer size was found to be 20 for this environment. A sequence length of 10 is insufficient in order to capture the time-dependent properties of rewards as a result of the 'fortnightly buffer' that is introduced into the reward formulation. A sequence

length of 30 allowed for the sufficient 'horizon' in order to model the fortnightly condition on rewards, however, the increased complexity it introduced, and the longer time it took for the buffer to fill up, meant that the overall performance dropped.

The results from changing the learning rates (of the Adam optimiser) can perhaps be best understood by inspecting Figure 22. Learning rates dictate the step size of the gradient descent algorithm. Too small of a learning rate means that parameters are stagnant, the policy isn't changing much, there is insufficient learning being done, and the results unsurprisingly provide very little improvement over the random policy, especially in the time-frame that we are considering. If updates to the model are very frequent, in the case where this tool becomes widely adopted, a lower learning rate will potentially allow for a sufficient speed of convergence, as well as for more stable learning. For too large of a learning rate, the AC algorithm performs close to no exploration and instead collapses onto a single prediction, the gradient vanishes, and no learning can be done (explained in more depth in the following section), and hence performs even worse than a random policy (note its cumulative reward was equal to -1123.12 but could not be appropriately demonstrated on the bar graph).

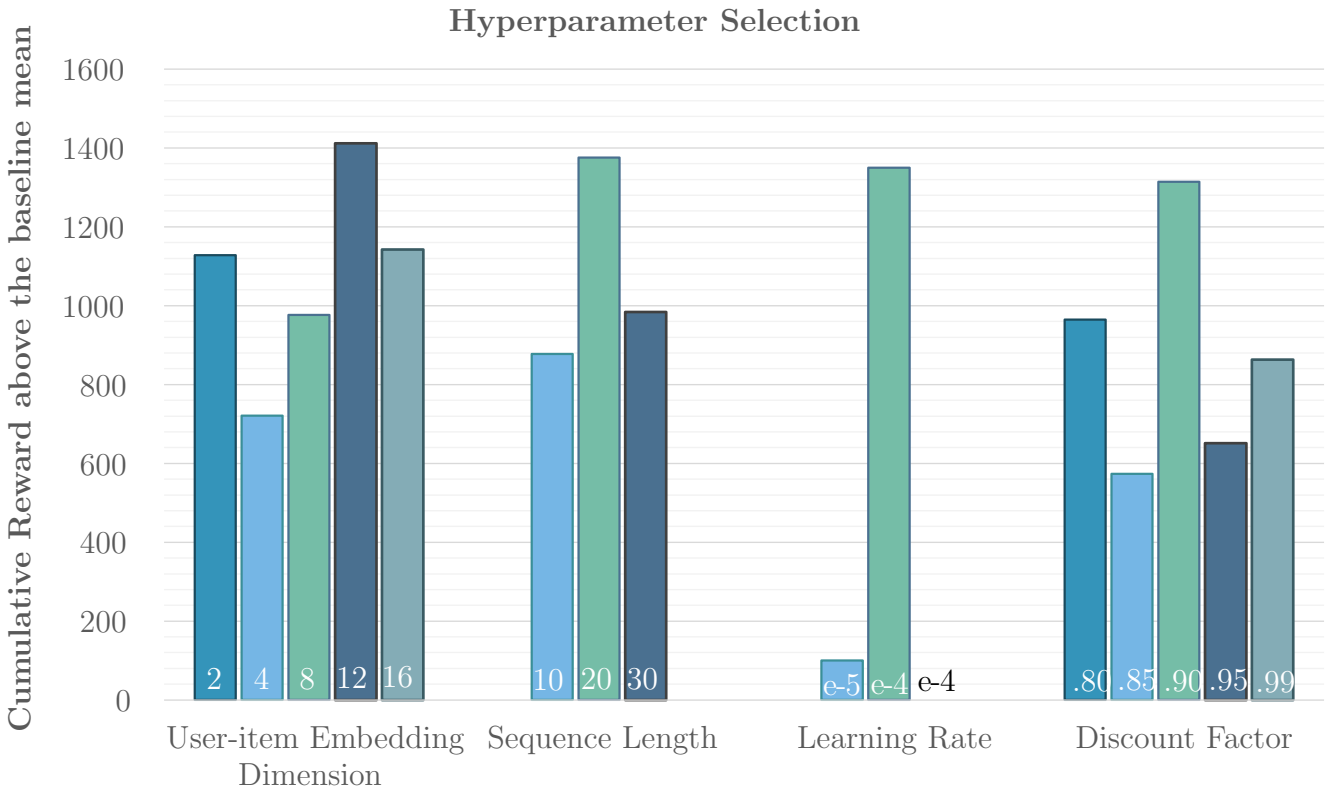


Figure 21: Bar graph showing the cumulative reward above the baseline value for different hyperparameters and hyperparameter values.

Finally, the discount factor determines the contribution of future rewards to the calculation of the value function. For our simulated environment, the future horizon that must be sufficiently captured by the value function in order to produce optimal conditions, is 14 days. Too large of a discount factor will not put enough weighting on those 14 days, i.e. the reward past those 14 days will carry too much of a weighting, whereas too small of a discount factor means that not enough of those 14 days is being captured by the value function. For example, the weighting of the 15th day on the value function for a discount factor of 0.99 is equal to $0.99^{15} = 0.86$, which is still substantial, whereas the weighting of the reward on the 15th day for a discount factor of 0.8 is equal to 0.035, which is close to no contribution. Clearly, from our results, a discount factor of 0.9 provides a good balance between the two.

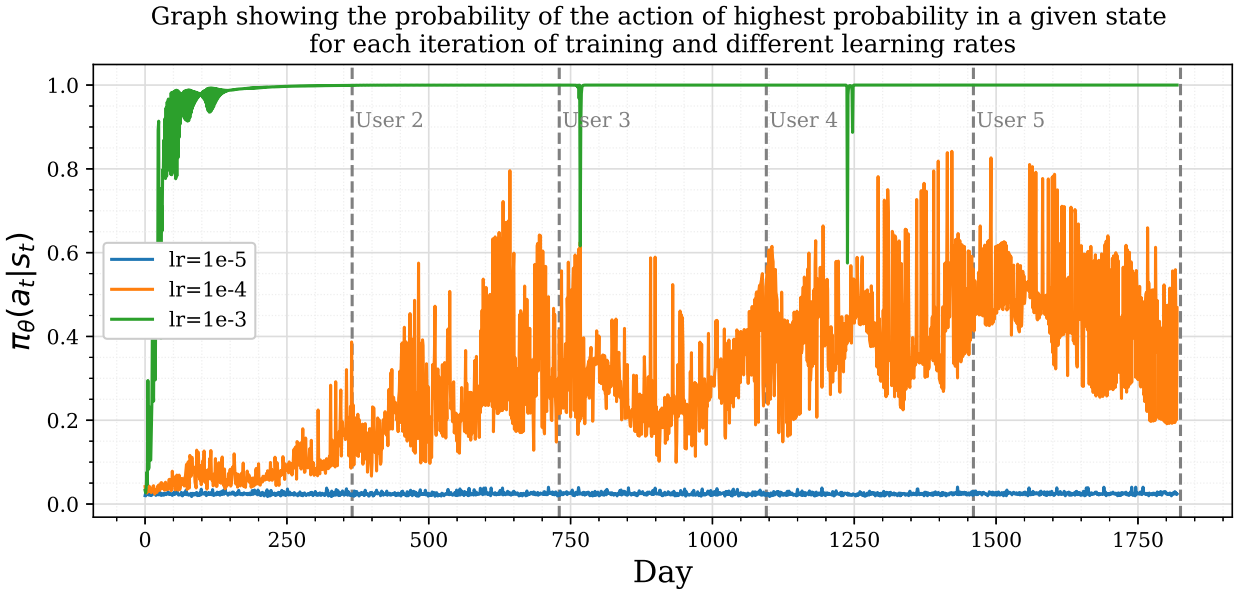


Figure 22: Plot showing the maximum probability for the policy $\pi_\theta(a_t|s_t)$ at each iteration of the conducted experiment for changing learning rates.

5.3 The role of reward in recommendations

When inspecting the recommendations given by the model, it was found that despite the penalty placed by our simulated environment on repeated activities, as training progressed, the model was increasingly taking the same action within the 14-day window, a common behaviour exhibited in Reinforcement Learning algorithms known as *policy collapse* [61]. This clearly shows that sufficient penalty is not being placed on the model for making repeated recommendations, and the Advantage Actor Critic model itself is not providing a sufficient deterrence against this behaviour.

The reason for this behaviour can be best understood by observing the loss function for the activity network (seen below). The first crucial observation is that the actor network is seeking a deterministic policy, since the loss is equal to zero when $\log \pi_\theta(a_t|s_t) = 0$ or, equivalently, when $\pi_\theta(a_t|s_t) = 1$. A consistently positive Advantage promotes the current policy, and as the parameters move in the direction of the current policy, and the model learns that taking a particular action often yields a positive advantage, then the probability of taking that increases, and the log probability begins approaching 0, causing the loss and therefore the gradient of the loss to vanish, thereby causing the policy to collapse.

$$\nabla_\theta J(\theta) \approx \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot \hat{A}(s_t, a_t) \rightarrow J(\theta) \approx \sum_{t=1}^{T} \log \pi_\theta(a_t|s_t) \cdot \hat{A}(s_t, a_t) \tag{63}$$

To account for this behaviour, the environment is adjusted such that the penalty for multiple recommendations of the same item within a fortnightly period is penalised further, specifically such that rewards are now negative for such a case. The following formulation of the reward is thus proposed:

$$r_t = \begin{cases} -5 + r_{season,j,i} \cdot 0.5^{C(t)}, & \text{if } r_{season,j,i} > 0, C(t) > 0 \\ r_{season,j}, & \text{if } r_{season,j} > 0, C(t) = 0 \\ r_{season,j,i} \cdot 2^{C(t)}, & \text{if } r_{season,j,i} < 0 \end{cases} \tag{64}$$

Where $C(t)$ is as before. The experiment is repeated for both environments for the same set of optimal hyperparameters as determined in the previous section. Since the distribution of rewards for the new proposed environment is skewed far more towards negative values, comparing the cumulative rewards above the mean for each user in each season, is no longer indicative of model performance. Instead, the difference between the cumulative reward when following the policy of the model, and the cumulative rewards when following a random policy (all actions have equal probability in every possible state) is used to compare the results, and is denoted by r' for simplicity. The two experiments are run, and the cumulative reward over using a random policy is calculated, and is plotted in Figure 23.

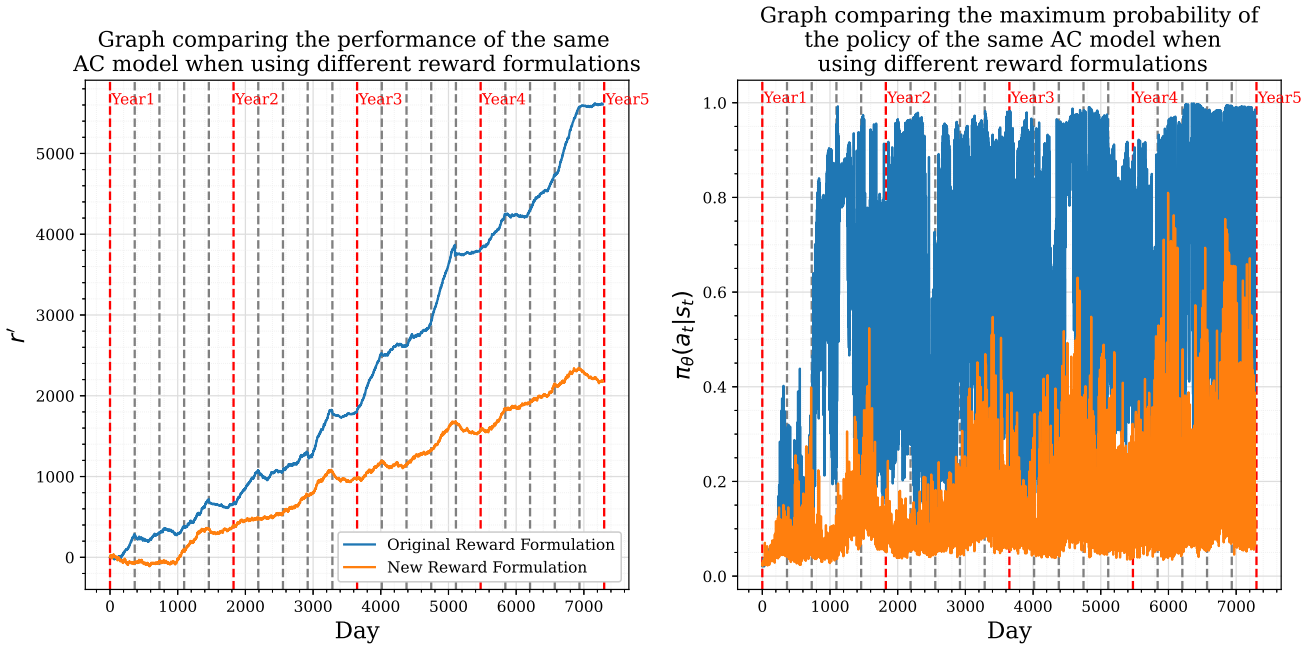


Figure 23: Plot showing the improvement in performance over a random for an increased penalty on repeated recommendations in the same fortnightly period (**left**), and the model's corresponding action of maximum probability at each time step (**right**).

By penalising the repeated recommendations, there is more pressure being put on the GRU to model the time-dependencies in the data, and the increase in time-complexity means that the model is slower to learn the optimal policy, hence the reduction in performance over the four years of simulation. This however gives a distorted view on the performance of the model under the new conditions. If for example, we look at table 4, we see that the new reward formulation unsurprisingly leads to far less repeated recommendations. Additionally, the new reward formulation leads to a far greater number of rewards equal to either 3, 4, or 5 (i.e. top recommendations without repetitions in the 14-day window) denoted as 'excellent' recommendations in the table below. This experiment demonstrated the importance of reward formulation on the nature of recommendations made by the model. The reward must be designed to target exactly what is being sought, and this will have to be done according to the environment. If this model is used in the intended environment, then the formulation of the reward must be selected carefully to achieve the desired performance.

Reward Formulation	**Number of repeated recommendations**	**No. of 'Excellent' recommendations**
Original	3217	790
New	1743	2292

Table 4: Table summarising the performance metrics for the two different reward formulations.

5.4 Investigating optimal discount factors for each user

In this experiment, the model is employed from scratch for each of the subjects, and each of their respective optimal discount factors are empirically found over the course of the year of simulation. This is an attempt to capture a hedonism metric for each of the subjects, however, it must be noted that since we have designed and imposed the time-dependency in the data, the optimal discount factors will not be reflective of the subject's actual level of hedonism, but will most likely be reflective of the fortnightly constraint placed on rewards. To confirm this, the experiment is repeated with the same parameters, but the fortnightly repetition penalty is reduced to a weekly one. Although no salient conclusions can be drawn from the subjects in this experiment, this is a demonstration of how this experiment can be repeated on real data to draw the conclusions that we seek with respect to hedonism. The experiment was carried out and the optimal discount factors for each of the users is summarised in the table below:

User	γ_{opt}: 14-day horizon	γ_{opt}: 7-day horizon
1	0.89	0.82
2	0.88	0.82
3	0.90	0.83
4	0.91	0.82
5	0.89	0.82

Table 5: Table showing the optimal discount factors for each user under two different reward formulations.

Now, the first obvious observation is the reduction in optimal discount factor across all users for a shorter penalty horizon. This is expected, the window of future rewards the model needs to consider is reduced, in fact $0.9^{14} = 0.206 \approx 0.82^{8} = 0.204$, meaning the model unsurprisingly performs best when it places the same importance on the relevant horizon. As for the inter-subject results, it seems that a smaller discount factor corresponds to a subject of higher mean reward, this is likely due to the fact that subjects with higher mean reward, have a 'deeper' bank of activities that make them happy, meaning they are less likely to be penalised by the constraint placed on repeated recommendations, and therefore can afford to pay less attention to it. This suggests that for the same temporal relationship with liked items, people with a wider range of activities that make them happy, are more inclined towards a hedonic sense of happiness.

5.5 Impact of personality embeddings on model performance

In this experiment, we investigate the prediction power of the FFM data by employing two versions of AC model, one where each user's FFM percentiles is input into the model, and one where FFM percentiles are omitted. The models are tested over 3 years of daily interactions with the environment defined in Section 5.1, the cumulative reward above the reward received by taking a random policy, r', is plotted for both models in Figure 24.

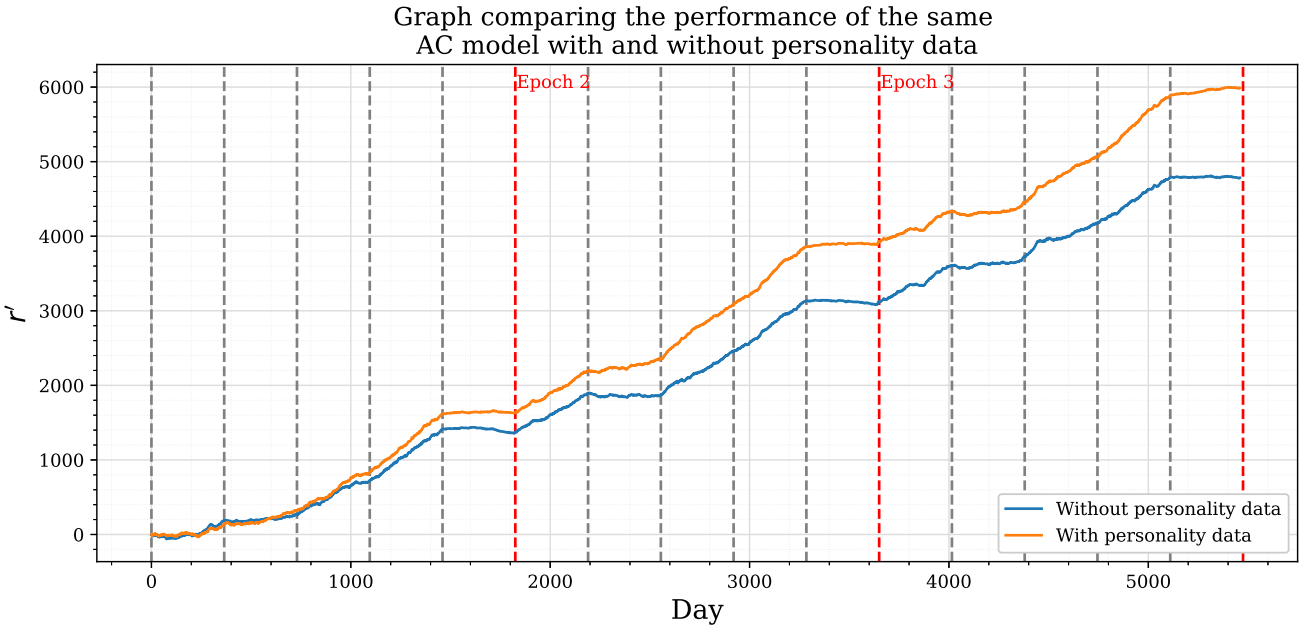


Figure 24: Plot comparing the performance of the model when including, and excluding personality features.

It can clearly be seen that the model that includes personality data is performing better than the model without, suggesting that personality is indeed predictive of what makes people happy. As the model interacts with the environment, the predictive power of personality is only increasing, and suggests that there is even more potential to draw from the relationship between our personalities and what makes us happy.

5.6 Inspecting the Value function

As afore described, inspecting the model parameters should give an insight into the relationship between the user, the recommendation, and happiness. We take user 5 for example, we input into the critic network (trained over 4 'years' of simulation): the empty user-item buffer, their personality data, age, location, and the SBERT embedding for each activity, resulting in a set of values associated with each activity that captures the expected discounted return (happiness) when making a particular recommendation. The result of this procedure (for user 5) can be visualised in Figure 25, where the x and z axes represent the projection of the activity SBERT embeddings onto its two most principle components, and the y axis shows the value for that activity, the two activities with the highest values are highlighted,

and labelled in red.

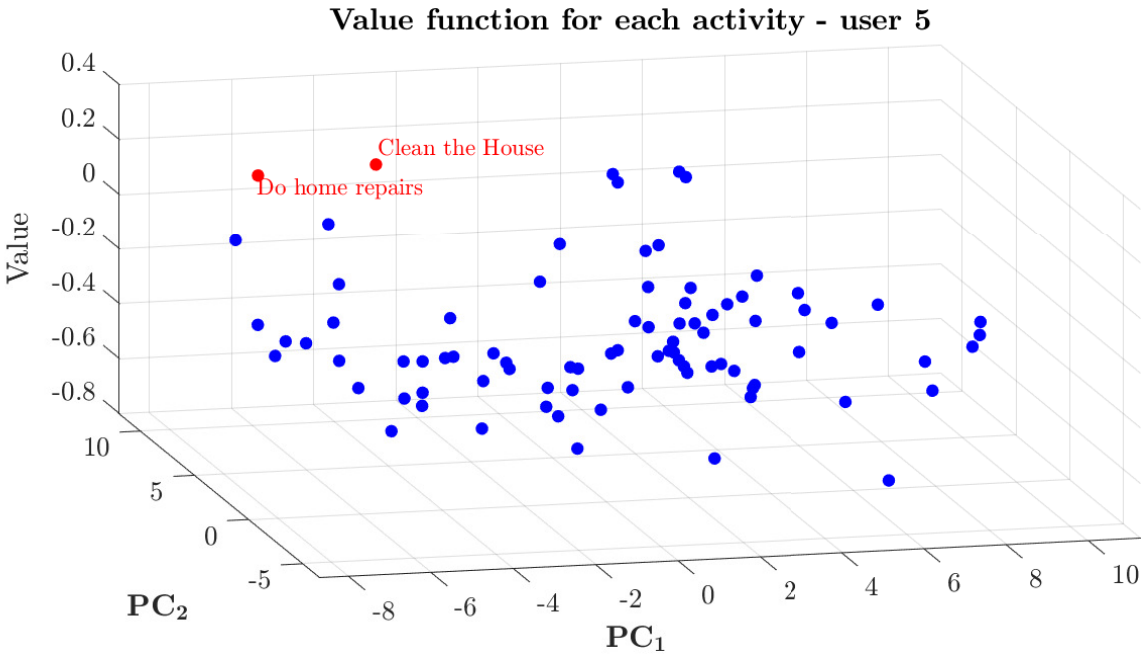


Figure 25: Plot of activity values when mapped onto the two priniciple components of the SBERT embedding space.

The average score given by user 5 over the four seasons for the ten activities with the highest value, in ascending order, left-to-right were: {0.5, 1, 3.25, 0.75, 2, 0, 2, 2.25, -2, -1}, essentially confirming that the optimal policy is yet to be learnt in the given environment. We expect the activities of highest value to be the ones that were assigned the highest score by the subject. When the model has not been trained sufficiently, and the optimal policy is yet to be learnt, the values will rather be an indication of how valuable an action was to the model in improving future recommendations. With more interactions (which were not possible, as a result of training time limitations), and when employed in a real environment, we expect that the value function will be more reflective of what we had initially anticipated it to be.

5.7 Cold-start problem Solution

The performance of the proposed solutions to the cold-start problem as described in section 4.5 are now investigated. The first proposed method is trained on the for 365 iterations. Both methods are trained on for 365 iterations on their respective derived set of rewards when placed under the environment introduced in Section 5.3, and then deployed under the original environment defined in Section 5.1. This is done in order to, or at least to some extent, simulate the fact that pre-training is done in a different environment to the one in which it will be used. Again, due to the inability to appropriately simulate data for the idea network, only the proposed methods for the activity network are investigated.

The two proposed solutions to the cold-start problem are carried out for user 3, the results being shown in Figures 26 and 27, where r' is the same as before, the cumulative reward above using a random policy. As hypothesised in Section 4.5, the second method completely outperforms the first proposed method, which actually hinders the performance of the model, and is unable to accurately capture the preferences of the user. The second method successfully yields a substantial improvement in model performance over the course of the year, and thus is considered successful, and is advised to be used in potential future uses of this model. It is worth noting that the nature of the second solution is well suited to the environment, and thus its performance may be skewed.

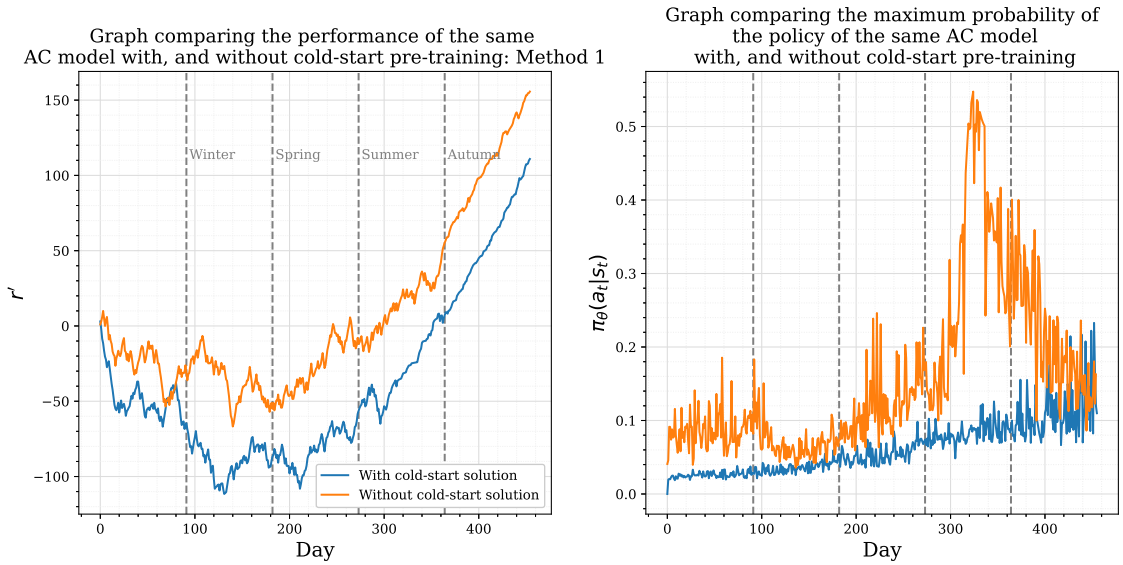


Figure 26: Plot showing the effect on model performance when implementing the first proposed solution to the cold start problem for User 3.

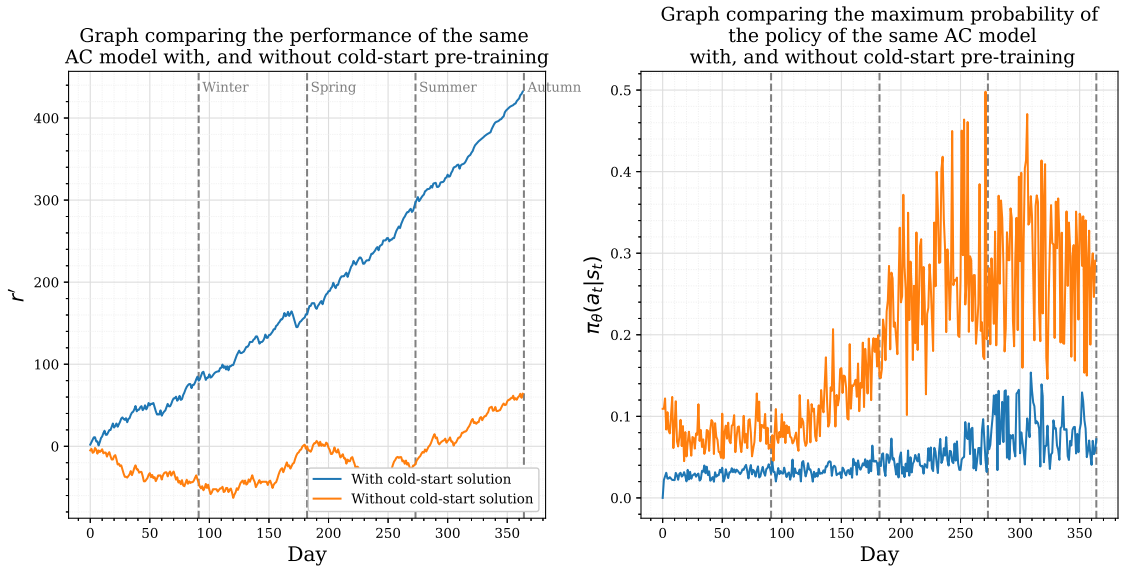


Figure 27: Plot showing the effect on model performance when implementing the second proposed solution to the cold start problem for User 3.

6 Evaluation

6.1 Performance comparison

With the performance of the proposed model investigated in a variety of scenarios, we now aim to evaluate our design choices by comparing our final model with different versions of the same model, and the baseline model that was proposed in Section 4.2. The model was also compared with an implementation of a Deep Q-learning network [109], but because of its poor sample inefficiency, it performed very poorly on this task, and is omitted from comparisons, and instead, details of its performance can be found in the appendix.

Three different versions of the AC network are implemented for comparison, the first is the proposed model with all the design features that have already been discussed. The second is the same AC network but with dense neural layers instead of GRUs for the user-item vector embedding. The final is the AC network with one-hot encoding of the previous recommended item instead of the SBERT embedding that is proposed in our model. The models are employed in the environment introduced in Section 4.4, to put an emphasis on their ability to capture the time-dependency present in reward signals. The models are employed on every user (consecutively), over a period of three years in the given environment. The results from this process are obtained, and plotted in Figure 29, where r' is as before.

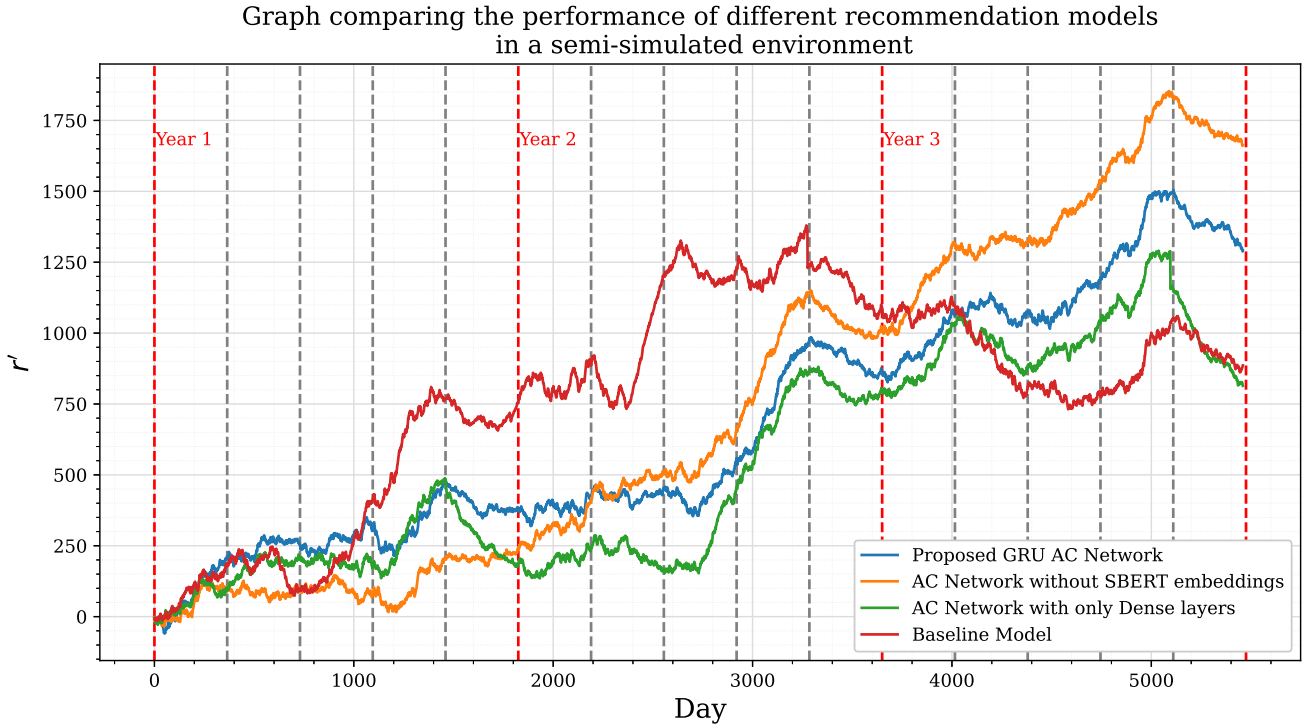


Figure 28: A graph comparing the performance of our proposed model with modifications of the same model, as well as a baseline model.

Interestingly, and in many ways unsurprisingly, both the baseline and the AC model with only dense layers seem to drop in performance after the third year, this is because they continue to update their parameters based on the assumption that the environment possesses

the Markov property, and thus are guaranteed to converge towards a suboptimal policy. The baseline model outperforms all models over the first two years, likely due to the fact that it has far fewer parameters than the other models, and thus can adapt to observations faster.

The best performing model out of all the models that were tested, is the AC network that uses one-hot encoding to embed the previous recommended item, instead of SBERT embeddings. In many ways this is also expected, since the activity space that we consider has 84 dimensions, and thus the one-hot encoded vector is 84-dimensional, whereas the SBERT embedding is 768-dimensional, meaning that the relativised embeddings that the SBERT introduce, do not yet outweigh the additional dimensionality that it introduces with it. Even though in this particular scenario the SBERT embedding hinders performance, it is far more robust to a growing activity space since its dimensionality remains constant no matter the size of the activity space, whereas the one-hot encoding will grow linearly in dimensionality, therefore, if the model is to operate within a more complex, and sparse activity space, then using SBERT embeddings will be preferred.

Despite Figure 29 affirming most of our design choices, there are still some observations worth noting in the grand scheme of the problem. The first is that despite its theoretical ability to model the non-linear time-dependencies within the environment, the model is still far from converging to the optimal policy for the given environment after 5,460 interactions. There are sudden changes (at the boundary of seasons) to the environment, that make it difficult for any model to capture, especially since the model has no objective sense of 'time', i.e. the day of the year, and thus can't store information regarding this change of environment within its parameters. As a result of this lack of convergence to the optimal policy, as explored in Section 5.6, it means that the Value function is not representative of the expected discounted future 'happiness', but rather the value of being in a state in regard to learning. Furthermore, after these 5,460 interactions, the best performing model only provides an improvement in reward of around 0.3 per recommendation (on average) over using a randomly initialised policy. It is likely that the model will require many more interactions with the environment before arriving at a point where salient conclusions can be drawn by inspecting the actor and critic models. As seen with the implementation of our solution to the cold-start problem, there are potential ways of addressing this slow convergence, but more work is likely needed to arrive at satisfactory performance in the early stages of model deployment.

6.2 Evaluating Model Properties

Aside from the performance of the model, we now evaluate the properties of the model in accordance with the initial outlined objectives. Firstly, as a result of the properties of the GRU, the model can be considered as non-linear time-variant [59]. Perhaps the most advantageous property of the model is the transparency of its parameters with respect to the qualitative conclusions that can be drawn by inspecting them.

The critic network of the activity model gives a direct evaluation of the expected future utility (happiness) when interacting with a given item (activity recommendation) for a given age, location, personality, and interests (user-item vector). With sufficient training, evaluation of the contribution of these variables towards the output prediction can be measured via models such as the DeepLIFT algorithm [100], that determine the importance of each variable towards the output, i.e. the maximisation of an individual's current, and future happiness.

The resulting probability distribution from the actor network can also be inspected to see which activity is best to follow the previous activity such that the individual's happiness is maximised (the activity corresponding to the highest probability). Furthermore, despite the discount reward being a hyper-parameter, i.e. it can't be learnt via SGD, data for a given user can be trained on with different discount factors and the one that maximises the reward over some given validation data, can be used to select an empirically optimal discount factor for that given user, which also provides an insight into the extent in which they experience Hedonic happiness over long-term happiness. The same conclusions can be made regarding the idea actor and critic models.

The model is also not restricted to only receiving feedback from recommended items, users themselves can input activities that they have done, along with the happiness they experienced from it, in order to train the recommender system. If the activity, or idea does not exist within the dataset, then the closest item, in terms of SBERT cosine similarity, can be input into the model instead.

The most limiting, and perhaps unavoidable property of the actor-critic model, is that new activities and ideas cannot be trivially appended to their respective item spaces. To make recommendations on new items, the model architecture must be suitably adjusted to account for the growth in the state and action spaces, namely the user-item buffer, and the dimensionality of the actor network output layer.

On the whole, our model design manages to integrate the vast majority of the initial objectives that we laid out for this project, all whilst achieving some promising results on the semi-simulated environment. As a result of the unavailability of data, we are restricted in the conclusions that we can make about actual, experienced happiness. Nonetheless, what we have done, is successfully provide proof of concept, as well as an end-to-end technical framework, which is ready to begin interacting with, and drawing conclusions from the intended environment.

7 Future Work

Given the fact that an AI and data-driven approach to happiness research is so novel, with there being close to no prior research, there is undoubtedly a need for further research into this topic. With the time spent on this project, a few key areas have presented themselves as lacking the sufficient depth in order to make the definitive, and salient conclusions that we seek.

In the context of this project, the first and most obvious area of work, is the need to collect data from the intended environment, under full test conditions. In order to make salient conclusions regarding the happiness of people, we require actual interaction between the model, and users, and not the semi-simulated conditions that we have investigated in the previous sections. In order to make this process seemless, it's recommended that a web user-interface should be designed and connected to a Python backend that makes function calls to the class containing the model that has been designed.

The second area that requires research, is the definition/design of the respective item spaces of the recommender systems. Although our design is based on some prior research, and was done somewhat systematically (according to some prior objectives), the selection and curation of these items was done manually, and undoubtedly contains unwanted bias. This was deemed sufficient for proof-of-concept purposes, however, if research on this topic is moved to a more formal phase, then more formal research needs to obtain a more definitive, and balanced set of items to present to the user. The appropriate selection of these items requires further data analysis, collaboration with philosophy and psychology experts, and a survey that considers the preferences of the public. Investigation should ideally answer the following questions: *Do people prefer receiving direct advice, or do they prefer texts that invoke reflection, such as the philosophical texts that we have used? What length of text do they prefer? How often do users prefer interactions to be? Are activity recommendations useful?*

From our investigations into the effects of the reward function, it was shown that the nature of recommendations is sensitive to the nature of the reward function, and what is deemed as a 'good' or 'bad' recommendation. Firstly, research is required, most likely via a psychological framework, into an appropriate and optimal set of questions to provide the user for feedback. The second stage of research should determine what is seen as a bad recommendation (negative reward). If, for example, there's an emphasis on giving recommendations that yield only high positive scores, then the reward function should be designed such that only those recommendations yield a positive reward.

There is also room for improvement with regard to the initialisation of the model for new users. Although our solution to the cold-start problem showed promising results, there is still potential for further improvement. The effectiveness of this solution will be crucial to the experience users will have in the early stages of deployment, and can be the defining factor

in user retention.

Further variations to the model could also be explored, the loss function of the Actor network could, for example, be modified to include entropy regularization in order to promote exploration and defend the model against policy collapse, as described by Li et al. [58]. The effect of implementing a separate GRU-based embedding network for SBERT embeddings could also be investigated.

The final recommended area of future work, is research into possible additional input features, this will require a more in-depth survey of correlation research into happiness. The current set of input features were sufficient for proof-of-concept purposes, however, the more relevant features that are used, the richer the model parameters will be in their representation of the system B, and the more we can learn about what, and how, different features contribute towards individual happiness.

8 Conclusion

By consulting existing psychology and philosophy research into happiness, and observing ways in which it can be brought into the engineering framework, we have successfully conceptualised, and implemented a completely novel approach to the problem, one that harnesses the prediction power of deep learning models, and garners our collective experiences of happiness to train a model that inherently learns from these experiences in order to maximise the individual's long term experienced happiness. The deep actor-critic network is specifically designed to operate within the semi-supervised, model-free environment that is defined by the problem, and has the ability to capture the non-linearities, and time-variance inherent to our experience of happiness.

In addition, the model parameters and outputs, such as the value, the policy, and the discount factor can be inspected to make direct inferences about people's experienced happiness according to different input features, such as their age, personality type or location. Deep learning provides the flexibility that allows for additional numerical, or categorical input features to be added to the set of input features, whether that be religion, the weather, or salary. The contributions of these input features to the output recommendation can then be measured by the DeepLIFT algorithm, a direct indicator of their importance towards our experienced happiness. In other words, the model we have designed, provides the means for a collective understanding of happiness to take shape.

Although we were unable to effectively simulate an environment where the idea and activity networks would be evaluated in tandem, we focused solely on the activity network, since both are identical in architecture. The model showed encouraging behaviour when tested in our semi-simulated environment, and although these experiments provided some insights into how it may perform in the intended environment, these results can provide nothing more than a proof-of-concept.

There is undoubtedly work still needed to be done on this topic in order to arrive at a place where reliable conclusions can be made regarding our relationship with experience, ideas, and happiness. This is a field of research within engineering in its infancy, and this project has laid the groundwork, and pathed a direction, for future research to build upon.

9 User Guide

All the relevant code, and data used in this project, is included in the following GitHub repository: https://github.com/nimakarshenas/happiness_project. All the models that are used in this project are contained in classes, with comments that explain input arguments and class methods. To minimise the inference, and training times, the models require a Pandas dataframe as input, which holds the precomputed SBERT embeddings for the activity and idea items. A script has been written to automatically obtain the required dataframe, and can also be found in the repository, denoted as `get_df.py`.

10 Appendix

10.1 Deep Q-Learning Network (DQN)

We implemented a Deep Q-Learning Network to compare our Actor Critic network with. Deep Q-Learning is perhaps the most widely used Deep Reinforcement Learning framework. Q-learning was discussed in Section 2.2.6. Instead of using a table to derive Q values, the same network as used with the Actor network (with linear activation instead of softmax activation in the final layer) can be used to predict Q values when in a given state. The network's loss is calculated by first obtaining the *target*:

$$target = r_t + \gamma \max_a \hat{Q}\left(s_{t+1}, a\right) \tag{65}$$

and then calculating the loss:

$$\mathcal{L}(\theta) = \left(target - \hat{Q}\left(s_t, a_t\right)\right)^2 \tag{66}$$

Where $\hat{Q}\left(s_t, a_t\right)$ is the output of the network for taking action a_t when in state s_t. To test the model, and see if it learns, it was employed in a simple environment, the same introduced in section 5.1, without the temporal penalty on multiple of the same recommendations. Due to the constantly changing environment (at the boundary of seasons), the model required a high degree of exploration for satisfactory results. Suitable parameter values for the epsion greedy algorithm were empirically found to be $\epsilon_0 = 1$, $\epsilon_{min} = 0.2, \gamma_\epsilon = 0.999$, where ϵ_0 is the initial value of ϵ, $\epsilon_{t+1} = \max\{\epsilon_{min}, \ \ \gamma_\epsilon \cdot \epsilon_t\}$. The largest learning rate for the Adam optimiser that allowed for stable learning was empirically found to be equal to 1e-4. The result from that experiment is shown below, where r' is as it was throughout the results section.

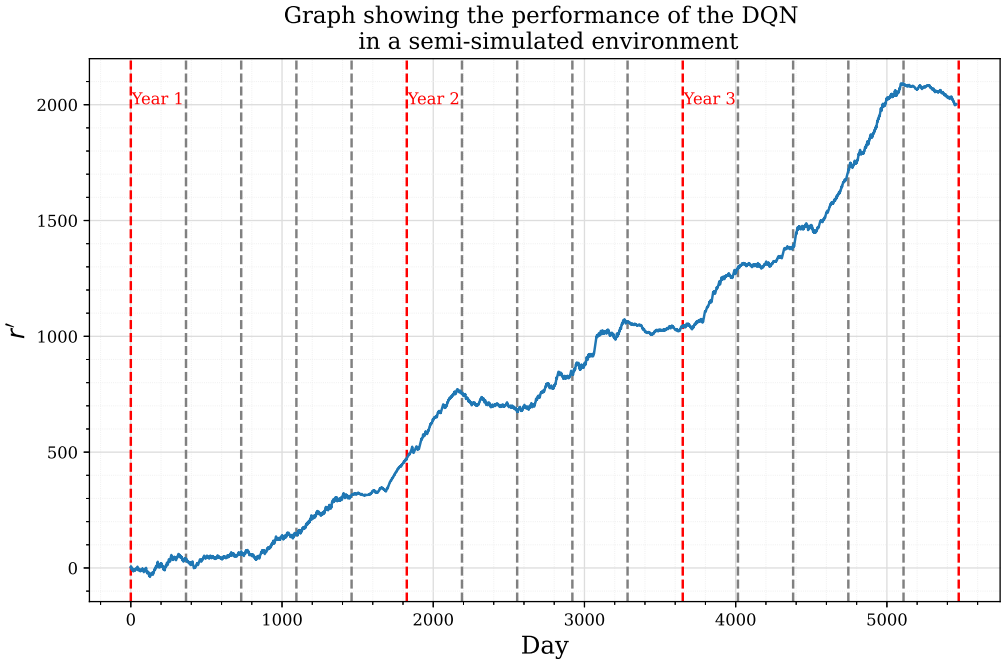


Figure 29: Plot showing the performance of the DQN in a simple environment.

Now, the model is employed in the same environment as the one used in the Evaluation section, with the results being shown below: The model is clearly struggling to adapt to the

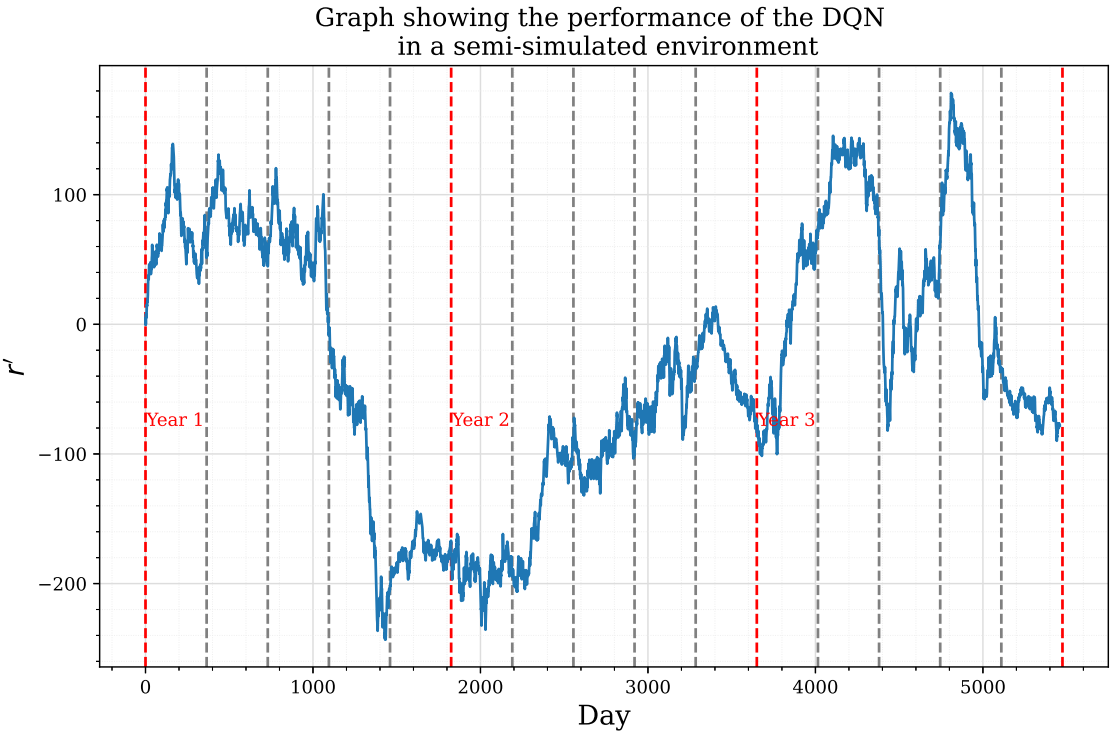


Figure 30: Plot showing the performance of the DQN in the evaluation environment.

harsher environment. The model shows some signs of sufficient learning in the second and third years of simulation, however it achieves very poor performance, and is far slower to learn the environment, despite having the same network architecture as the actor network.

References

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

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

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

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

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

[6] 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.

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

[8] 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.

[9] 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.

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

[11] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[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] Lewis R Goldberg, John A Johnson, Herbert W Eber, Robert Hogan, Michael C Ashton, C Robert Cloninger, and Harrison G Gough. The international personality item pool and the future of public-domain personality measures. *Journal of Research in personality*, 40(1):84–96, 2006.

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

[40] 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.

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

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

[43] 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.

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

[45] 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.

[46] 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.

[47] Kenneth E Iverson. *A Programming Language*. John Wiley Sons, Inc.605 Third Ave. New York, NYUnited States, Jan 1962.

[48] Zhijing Jin, Geeticka Chauhan, Brian Tse, Mrinmaya Sachan, and Rada Mihalcea. How good is nlp? A sober look at NLP tasks through the lens of social impact. *CoRR*, abs/2106.02359, 2021. URL: https://arxiv.org/abs/2106.02359, arXiv:2106.02359.

[49] 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.

[50] 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.

[51] 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.

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

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

[58] Rongpeng Li, Chujie Wang, Zhifeng Zhao, Rongbin Guo, and Honggang Zhang. The lstm-based advantage actor-critic learning for resource management in network slicing with user mobility. *IEEE Communications Letters*, 24(9):2005–2009, 2020. doi:10.1109/LCOMM.2020.3001227.

[59] Benjamin Lindemann, Timo Müller, Hannes Vietz, Nasser Jazdi, and Michael Weyrich. A survey on long short-term memory networks for time series prediction. *Procedia CIRP*, 99:650–655, 2021.

[60] 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.

[61] Yicheng Luo, Jackie Kay, Edward Grefenstette, and Marc Deisenroth. Finetuning from offline reinforcement learning: Challenges, trade-offs and practical solutions. 03 2023.

[62] 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.

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

[64] 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.

[65] 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.

[66] 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.

[67] 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.

[68] 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.

[69] 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.

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

[71] 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.

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

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

[74] 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.

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

[76] 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.

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

[78] 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.

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

[80] 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.

[81] 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.

[82] 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.

[83] 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.

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

[85] Eduardo Rodrigues Gomes and Ryszard Kowalczyk. Dynamic analysis of multiagentnbsp; nbsp; nbsp; nbsp; nbsp; nbsp; qnbsp; nbsp; nbsp; nbsp; nbsp; nbsp; -learning with -greedy exploration. *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009. doi:10.1145/1553374.1553422.

[86] 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.

[87] 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.

[88] 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.

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

[90] 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.

[91] 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.

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

[93] 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.

[94] 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.

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

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

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

[98] 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.

[99] 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.

[100] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. *CoRR*, abs/1704.02685, 2017. URL: http://arxiv.org/abs/1704.02685, arXiv:1704.02685.

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

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

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

[104] 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.

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

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

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

[108] 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.

[109] Chunxi Tan, Ruijian Han, Rougang Ye, and Kani Chen. Adaptive learning recommendation strategy based on deep q-learning. *Applied psychological measurement*, 44(4):251–266, 2020.

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

[111] 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.

[112] Lionel Tondji. *Web Recommender System for Job Seeking and Recruiting*. PhD thesis, 02 2018. doi:10.13140/RG.2.2.26177.61286.

[113] 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.

[114] 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.

[115] 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.

[116] 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.

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

[118] 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.

[119] 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.

[120] 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.

[121] 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.

[122] 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.

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

[124] 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.

[125] Zhongheng Zhang. Introduction to machine learning: k-nearest neighbors. *Annals of Translational Medicine*, 4(11):218–218, Jun 2016. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4916348/, doi:https://doi.org/10.21037/atm.2016.03.37.

[126] 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.