A system to influence/manipulate the emotions of twitter users through auto-generated posts made by a machine-learning enhanced bot.

UFCFXK-30-3 Digital Systems Project

Callum Robert Jackson

16024212

Word count: 8602

Abstract

The goal of this project is to warn its user how automated machine learning based systems can manipulate people at scale on Twitter. My practical project will display just how effective these systems can be and offer insight into how, by changing different parameters, the system can yield certain reactions. Alongside this, my research will show that by extending some of the processes in which the system learns, these techniques can influence people's emotions so precisely as to sway things people usually believe they have autonomy over, such as their political voting patterns or the products they buy, and outline the part computational sentiment analysis has to play in this, followed by a breakdown of how this is implemented in my project. I will also cover the fundamentals of natural language processing and machine learning and how they are used to do sentiment analysis.

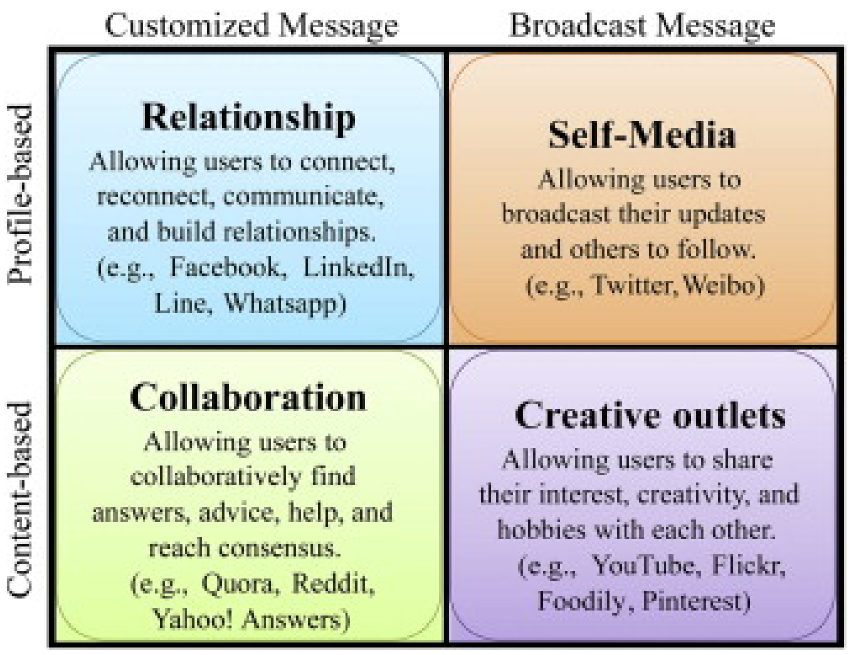
Social media and emotional manipulation

Audience labour theory

In 1867 Karl Marx laid out a structure of the relationship between labour, value and commodity. He pointed out the distinction of abstract and concrete labour and the importance of the abstract value labour which has become so prevalent in the service and entertainment economies today. Then in 1977 Smythe called communications the eponymous *Blindspot* in Marxist theory. He pointed out that, as he was concerned with TV and radio, there were two commodities being sold:

1. the goods of the companies that would be sold to customers through the running of adverts on the platform
2. the “audience and readership” as goods sold to the companies.

This process essentially makes the media company in the middle an attention broker. Under Marx’s rule, this means that any time spent watching or listening by the audience is time spent working. This idea is known as the audience labour theory; the idea that the audience, while they are consuming entertainment, their attention is a part of an economy as a commodity. Rather than getting paid for this work with money Smythe suggests that consumers are paid with entertainment. In fact now, with our use of social media it has gone far beyond entertainment, and well into the realm of satisfaction of deep human emotional needs.
Yu-Qian and Houn-Gee (2015) considered the human needs that satisfying events trigger as outlined by Sheldon et al (2001) and how these needs are met with social media.
They considered 4 different types of social media:

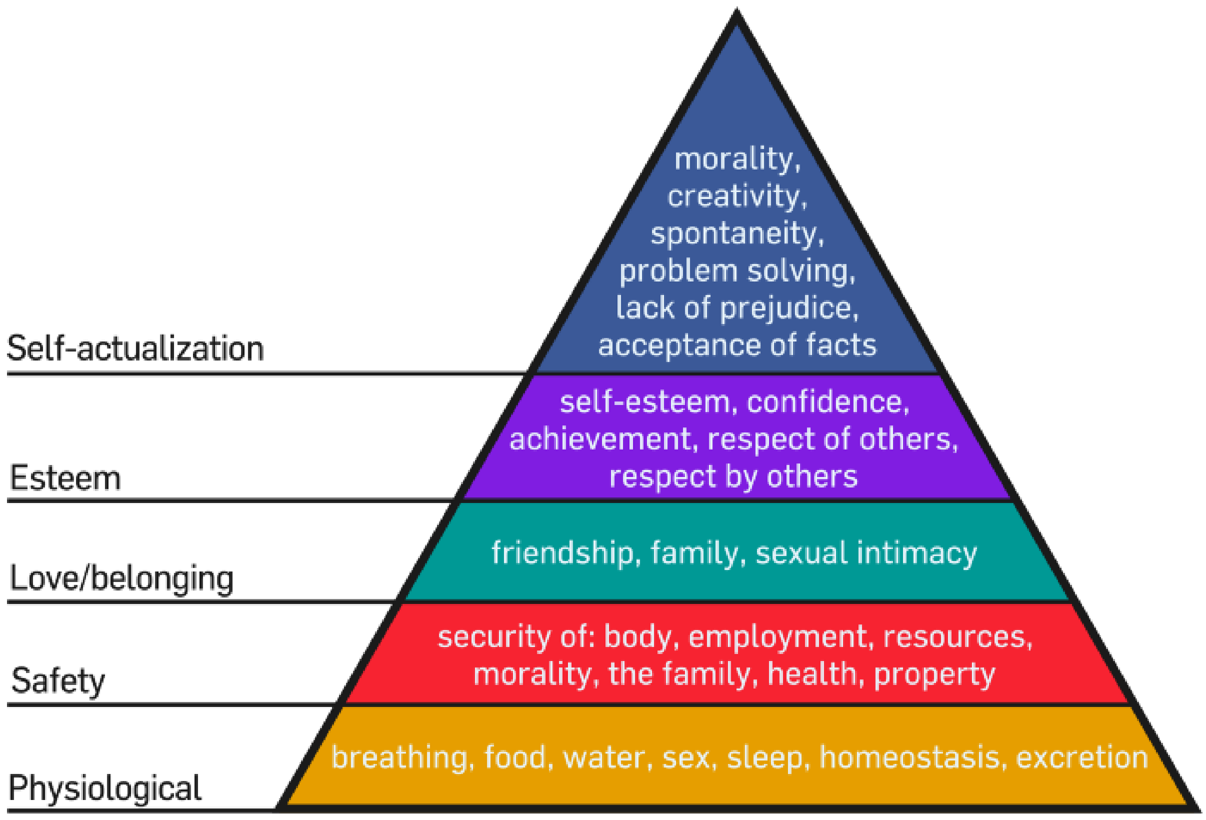

Customized Message
Broadcast Message
Profile-based
Content-based
Relationship
Allowing users to connect, reconnect, communicate, and build relationships. (e.g., Facebook, LinkedIn, Line, Whatsapp)
Self-Media
Allowing users to broadcast their updates and others to follow. (e.g., Twitter, Weibo)
Collaboration
Allowing users to collaboratively find answers, advice, help, and reach consensus. (e.g., Quora, Reddit, Yahoo! Answers)
Creative outlets
Allowing users to share their interest, creativity, and hobbies with each other. (e.g., YouTube, Flickr, Foodily, Pinterest)

My research will focus predominantly on relationships and self-media with some consideration of collaboration and creative outlets.

The needs Sheldon et al laid out are:

Autonomy	Feeling like you are the cause of your own actions rather than feeling that external forces or pressures are the cause of your actions.
Competence	Feeling that you are very capable and effective in your actions rather than feeling incompetent or ineffective.
Relatedness	Feeling that you have regular intimate contact with people who care about you rather than feeling lonely and uncared for.
Physical thriving	Feeling that your body is healthy and well taken care of rather than feeling out of shape or unhealthy.
Security	Feeling safe and in control of your life rather than feeling uncertain and threatened by your circumstances.
Self-esteem	Feeling that you are a worthy person who is as good as anyone else rather than feeling like a “loser.”
Self-actualization	Feeling that you are developing your best potential and making life meaningful rather than feeling you are stagnant and that life does not have much meaning.
Pleasure-stimulation	Feeling that you get plenty of enjoyment and pleasure rather than feeling bored and under-stimulated by life.
Money-luxury	Feeling that you have plenty of money to buy most of what you want rather than feeling like a poor person who has no nice possessions.
Popularity-influence	Feeling that you are liked, respected, and have influence over others rather than feeling like a person whose advice or options nobody is interested in.

They found that all 4 categories of social media cover a unique set of these needs but that all of them are covered by at least one and that, unsurprisingly due to the “social” aspect of it, a feeling of relatedness was the most common amongst them. They conclude by stating that “To be successful in social media campaigns, you must possess precise insight regarding your product, your campaign, the social media platform you are using, and the social media users you are targeting. Gaining precise insight is not easy, as there are many nuances involved”. Social media is a massive industry with almost 4 billion, or over half the world’s population, now using it regularly. This is nearly double what its user base was in 2015 according to Chaffey (2021).When considering Maslow’s hierarchy of needs - which categorises needs with basic survival at the bottom - as seen here:


Self-actualization
morality, creativity, spontaneity, problem solving, lack of prejudice, acceptance of facts
Esteem
self-esteem, confidence, achievement, respect of others, respect by others
Love/belonging
friendship, family, sexual intimacy
Safety
security of: body, employment, resources, morality, the family, health, property
Physiological
breathing, food, water, sex, sleep, homeostasis, excretion

We can see that all of the need satisfactions stated above are placed highly. And as more and more people are lifted out of absolute poverty as outlined by The World Bank (2018) and with that the number of people who can afford to use social media regularly - as Chaffey showed above - the market of emotional-attention-labour available on social media is booming and waiting to be exploited. Furthermore, Alsunni and Latif (2021) measured emotional investment in social media among university students against the Hospital Anxiety and Depression scale from World Neurosurgery 2015 (as opposed to previous research which has only considered frequency and duration of social media usage) and found significant increase in the likelihood of becoming anxious or depressed by factors of 1.76 and 1.48 respectively with per unit increase in emotional investment.

This has not gone unnoticed. The two biggest current exploits of this audience labour are marketing and politics, and following this I will give some examples of how different social media sites can be manipulated so that the designations assigned by Yu-Qian and Houn-Gee can be overridden thus increasing the audience labour extracted from the site's users.

Marketing

At the time Smythe in the 1970s was writing people would spend 9 minutes watching adverts for every hour of TV which has doubled to 18 minutes for contemporary TV, and people were expected to see an average of 500-1000 adverts a day. In 2021 it is estimated that we see between 6000 and 10000 adverts per day, almost ten times as much as what Smythe would have been considering according to Carr (2021). The key difference between marketing on social media in the 2010s versus on television in the 1970s is that it can be tailored to best suit the specific person viewing it. Where traditional media advertising would require analysis of viewing figures and polling to determine audience demographics, states Bozios (2001), social media has much simpler access to that kind of information. Some of this information is given consciously: age, gender, home address etc is all submitted when signing up to a new platform. However users also submit far more detailed personal information subconsciously to complex algorithms that analyse their usage statistics: what accounts they follow, how long they spend looking at certain pages etc. This is how social media has become the new hotbed for marketing, as Chandler (2019) shows. Having this level of insight into their users is how Facebook and YouTube are able to target their adverts so much more deeply than ever before. Historically an advertising company would have a limited knowledge of the average viewer of a certain TV show, so they would have to design their adverts to reach a wide audience so as not to exclude potential viewers. Social media has overcome this problem by only showing adverts to those it has deduced will be interested in it, so adverts can be designed for specific types of person while minimising the exclusion of potential customers. This reduction in exclusion is how these websites have become so efficient in delivering adverts. Of the 9-18 minutes per hour of adverts on TV there is very little that actually piques the interest of a viewer - if they are even watching at all as Heath (2007) reveals 20-40% of TV viewers leave the room during adverts. If the TV show has a diverse viewership then its likely they could be showing car adverts to people too young to drive or makeup to men. This is as opposed to online where Google can show you adverts related to products you have actually searched for or Facebook can show you adverts for something specific to your hometown. According to Fisher and Fuchs (2015) expenditure on advertising on social media has been growing by 17.6% annually as opposed to a decrease of 1.4% for newspapers. Not only is this potential disparity of product to viewer mitigated, but even the way in which a product is sold can be fine tuned: advertising companies will now make multiple differing adverts for the same product, but only show whichever advert is most appealing to a specific demographic.

Sentiment analysis plays an important role in this form of marketing. As I will show with my practical project sentiment analysis of written text is relatively straight forward. There are plenty of datasets online of text that have been analysed for sentiment, and training a model to understand this has been made easy by modern computational frameworks. The level of sentiment analysis that is done to understand a users engagement of an advert is more complex. Facebook will calculate how long their users take to scroll past a certain advert to determine how effective an advert is - GMI Blogger (2017). Similarly YouTube will measure whether a user immediately skips the advert after 5 seconds or whether they continue watching - YouTube support (2021). However both platforms are most concerned with maximising click-through ratios; the number of people that have seen the advert versus the number of people that actually click on the advert according to Dragilev (2019). A signal of success similar to click-through ratio is if a user later searches for a product after seeing it advertised. This is all about trying to hold the users attention for longer or maximising audience labour and selling this labour of attention for advertising revenue.

Appel et al (2020) discuss how marketing on social media has become less about specific technologies or platforms but about the people, what they do and how they act on these platforms, and most importantly figuring out how to monetize this audience and their activities, harking back to the ideas of audience labour theory. One of the key methods they discuss in this study is how people around the world generally use social media for three main purposes:

1. Digital communication with known others i.e. family and friends
2. Digital communication with unknown others who share common interests
3. Digitally accessing and contributing to content consisting of news, gossip and user generated product reviews

Almost all of these are, in one form or another, word of mouth based activities, this will become a key part of the political influence done on social media later on. This idea of word of mouth on social media is extensively backed up by Herhausen et al. (2019), Stephen and Lehmann (2016), Villarroel Ordenes et al. (2017). Appel et al go on to show how prevalent the influence of social media is on its users decision-making process from a marketing perspective due to its “omni-sociality”. This is primarily due to the activation of “need recognition”, the idea that there is a gap between someone's current circumstances and their desired circumstances says Watson (2016). This is where sentiment analysis can massively help with automating marketing; if you can trigger the right emotional response you can activate this need recognition. Dwivedi et al (2020) talk about one massive potential exploit for marketing teams: the implementation of virtual reality (VR) and augmented reality (AR) technologies. These technologies can enable customers to try before they buy, whether it be putting an AR piece of IKEA furniture in a room or trying L'Oreal's virtual makeover as shown by Animalz (2020). There has already been leaps and bounds made and the tech is still in its infancy. In the context of sentiment analysis and activating need recognition, tracking eye movement and heart rate could be used to assess the emotional impact of an advert, much like how analysis click-through rates and read times is done now. Hackl (2020) shows how, once AR and VR have become more prevalent, they will undoubtedly be incorporated into social media marketing.

Facebook is king: with over 2.5 billion active users and a market cap nearing a trillion USD, it has become the world's largest attention broker. In its 2019 10-K filing (an annual report to the SEC of a company’s financial performance) Facebook reported an ARPU (average revenue per user) of $8.52 worldwide in Q4, states McFarlane (2020). This is made possible by their extensive data analysis capabilities of which, until 2018, was not fully realised by the public. The Cambridge Analytica scandal has revealed the true capability of a system in which people are willing or ignorant to their forgoing of their own privacy. Facebook had given access to 87 million users' personal data to data firm Cambridge Analytica, from Isaak and Hanna (2018). In 2013 researchers of psychometrics at Cambridge University had developed a method of data analysis that could tie a user's psychological profile to their Facebook activity, with regard to likes and shares. They had gained this psychological profile by posting a personality test in which volunteers would answer questions and get a result based on the big-5 personality trait model of psychology. Cambridge Analytica used this model along with a variety of other data sources to build profiles of over 5000 data points on 230 million US adults. They then combined this data with targeted messages in *Project Alamo* which was used to sway voters in the 2016 election toward the campaign on Donald Trump. This extended beyond Facebook through the selling of cookies to other websites, as well as blurring the lines between real news and misinformation. It also included similar levels of data (except crucially the results of the personality quiz) of all the users friends, says Venturini (2019)

By creating this level of integration and targeting, Cambridge Analytica have maximised the efficiency of the extraction of audience labour from Facebook's users. Using psychological profiling has allowed them to trigger the emotional needs outlined by Sheldon et al (2001) and surpass the designation of Facebook to the relationship category as assigned by Yu-Qian and Houn-Gee (2015).

Politics

While Facebook and YouTube's user bases and market caps are far beyond any of their competitors, other sites like Twitter and Reddit seem to be punching above their weight when it comes to political engagement and influence. In recent years Twitter has been at the heart of many social and political issues. Then president Donald Trump's usage of it made a hybrid media system, bringing Twitter to the forefront of news and politics, from Wells (2020). Movements characterised by their associated hashtag like Black Lives Matter have been fostered there, from Edrington and Lee (2018). In a similar vein to the aforementioned Project Alamo which was a focused political campaign with adverts targeted at specific people, further revelations regarding the interference by the Russian government in the 2016 US election poses a threat to the privacy of users of social media. Another scandal shrouded in dispute, this one with many pseudonyms, the Russian "socialbots", "Twitter trolls", "botnets" to name a few, had its reputability confirmed in 2019 when former Special Counsel Robert Mueller released his "Report On The Investigation Into Russian Interference In The 2016 Presidential Election" (2019). While Project Alamo was concerned with trying to persuade people to vote for Donald Trump, the method of this conspiracy was to try to widen an already vast political division by provoking discord and hatred, promoting both Trump and Bernie Sanders while sabotaging the campaign of Hillary Clinton according to Lee (2018). As well as this there were also reports of similar campaigning in Britain to encourage Brexit, however this remains unproven says Ellehuss and Ruy (2020), however if it were true then the idea behind the campaign would be the same.

Compared to Project Alamo this kind of campaign is more of a guerilla tactic. Similarly to how marketing firms will have a good understanding of their audience before they make an advert, the Russian engineers behind the bots did extensive market research before starting these troll farms. Similarly to Project Alamo, these bot farms were able to focus on far smaller demographics than typical marketing schemes. However this campaign had two major advantages over Project Alamo:

1. Normally a firm would analyse an adverts success and then make a new advert, this kind of campaign where bots are making multiple posts a day can review actual user response to each post in real time, and constantly improve their marketing.
2. People generally don't trust adverts. Yu-Qian and Houn-Gee (2015) state that as low as 15% of consumers trust social media marketing in the US, Traditional adverts had to work to try to make their audience listen and trust the company advertising, Project Alamo was trying to improve on this by using people's psychological profile and tailoring the adverts directly to them. This campaign tricked users into believing they were reading an actual user's post, and because people are more likely to believe and trust another person over an advert, they had essentially circumvented this problem.

This is where the idea of word of mouth being the most common usage on social media as per the research by Appel et al (2020) comes into play; people who use social media are looking to have interactions with other people, not adverts, which is what the Russian socialbots phenomenon has managed to tap into so effectively. Sentiment analysis in marketing would have to be done beforehand; then an advert can be changed or updated to include the more up-to-date research, with a system like this it can update and improve itself over time by reviewing and evaluating the reactions it garners.

That is not to say that it is all doom and gloom. The political manipulation and corruption on Twitter may be rising, and as stated by Putnam (2000) general engagement in politics (voting turnouts, joining of political parties and trade unions) is declining in developed countries. A study by Keating and Melis (2017) looked at 22 to 29 year olds in Britain who, according to Xenos et al. (2014), have relatively low turnouts for elections but high engagement with social media and showed that people, especially young people, are actually just expressing and engaging in politics in new and more creative ways, corroborated by Dalton (2008). A study conducted in by Ekstrom and Shehata (2016) looking at tweets made between 2012 and 2016 in Sweden shows that users engaged in political interactions on Twitter has doubled in some cases - now while this seems like it might just be attributed to an increase in general usage of social media as it has become more popular, engagement in other areas such as public production activities has actually stayed roughly the same. They also went on to show how - through the use of porous boundaries and low thresholds - regular social interactions on Twitter can very easily turn into political interactions, meaning that users likely are very conscious of politics at all times. While the actual percentage of adolescents engaging in political interactions is generally lower than that of adults, the rate at which their regular social interactions will turn into political ones has increased, as well as their overall average time spent having political interactions. More interestingly, the study also shows that the majority - especially in adults - of political engagement is characterised by consistent engagement over time, rather that "tune in, tune out" activity. Keating and Melis discovered that by dividing people into socio-economic classes those in the lowest class had the least political engagement, and that if social media were to be used as a tool to get more people engaged in politics this class would be the people you would target for having the most potential.

So perhaps this is the reason these websites are being targeted for political interference; political engagement has moved from traditional areas to a more open and potentially easier to interfere with platform with social media. Considering the research of Keating and Melis regarding the potential political malleability of people of low socio-economic status alongside the numerous studies that show that the average pro-Brexit voter was of such a status, namely Rodriguez-Pose (2018), Dorling and Tomlinson (2019) and Lichter and Ziliak (2017), it could very well be so that the Russians saw this same opportunity (although Abreu and Oner (2020) who discuss some potential methodological issues with conducting such studies). However as aforementioned there is still lack of concrete evidence to suggest the Russian bots really were intended to interfere with Brexit. Furthermore despite the general assumption that Trump drew in the working class white crowd in America, usually based on statistics showing that of all the white voters without a college degree 67% voted for Trump versus 28% for Clinton, the actual turnout of these people was less (aligning with the overall turnout) than in 2012 according to Hudak (2016).

Entertaining the idea that the goal of the Russian socialbots is to exploit this potentially political Twitter demographic, the idea is about maximising audience labour through emotional manipulation. So if to extract more labour from this audience requires simply manipulating this emotional investment, then we must consider how much emotional investment people have in politics, and how this investment can be increased. Gellwitzki and Houde (2021) looked at the 2015 imigration crisis in Germany and showed that a threat to people's national identity and European identity significantly increases their emotional investment in political events. They went on to show that the more the EU becomes politicized the more political actors become emotionally invested in it, and this emotional investment carries over to the populous. The parallels between this and Brexit and the potential for the Russian socialbots to capitalise on the high emotions of the British public is plain to see. Britain in the lead up to the 2016 referendum was a cornucopia of audience-labour ready for exploitation.

The consequences

To allow this kind of exploitation of our emotions and politics on social media by any actor, let alone malicious foregin actors is a relinquishing of our freedom. Sandoval (2015) says that the exploitation of labour must be counteracted by solidarity across occupational and national boundaries, which can be done using social media. Project Alamo and the Russian socialbots revelations are in direct opposition to this. There must be put in place some strong regulation that prevents nefarious manipulation through social media. However there has been surprisingly little uproar about what seems to be a direct attack on Western democracy. Are people really aware of the scale of it? Do people lack understanding of the consequences? Do people even care? To have this kind of regulation employed would first mean to have massive awareness raised as to the impact of these interferences. To raise this awareness I believe there must be educational tools in the media and beyond to show people how these systems work, what they can do and what the implications of them being so widespread are.

Sentiment analysis and natural language

Natural language processing (NLP) is essentially made of two components, each having had extensive research into how to do them:

1. Natural language generation (NLG)
2. Natural language understanding (NLU)

NLG

One common distinction between types of NLG is real versus template-based generation. Van Deemter et al (2005) actually state that this distinction is purely superficial; “real” NLG is actually just a complex template. One technique they have discussed here is like a template but more of a structural guide to NLG:

Content determination > Referring expressions > Aggregation > Lexicalisation > Linguistic realisation.

The final part, linguistic realisation, is where they state the main difference between real and template-based NLGs is: a template based system would have to be far too deep to actually generate sentences and account for all the nuances of language, where a real NLG would be able to complete that final step.

McDonald (2010) corroborates that because “computers are dumb” it makes more sense to do template based NLG. He also states that, as opposed to scientists studying language generation, computer systems and their authors, even with artificial intelligence systems, tend to miss the richness and fullness of language and any emotional or rhetorical attitudes, because computers have no basis for making the decisions that go into such natural utterances. Ratnaparkhi (2000) uses trainable surface NLG, a template-based system with *attribute-value* pairs so that his system can map from semantics to words. Most of these older studies are using techniques to try and minimise the amount of work their computer system does by manually analysing sentences and building a quasi-universal template. This quickly becomes overly complex and cumbersome, requiring massive amounts of human input to reach any semblance of universality. Gatt and Krahmer (2018) recognise this, stating that it is all too common for a system to need to interact in a novel way. They point to journalistic systems that automatically generate text from non-linguistic data: taking information and filling in the semantic gaps between keywords. They talk about how machine learning can be used alongside content selection with regard to Duboue and McKeown (2003) - this appears to be the most succinct and most straightforward to implement.

NLU

Neethu and Rajasree (2013) manually classified tweets from April and May 2013 as positive or negative (600 of each). This level of classification is probably quite accurate for the tweets they selected, however the dataset seems very small. However they have shown that this is clearly very effective through comparisons of a variety of tests. They use a feature vector of these 8 parts:

- Part of speech
- Special keyword
- Number of positive keywords
- Number of negative keywords
- Presence of negation
- Emoticon
- Number of positive hashtags
- Number of negative hashtags

An automation of this level of analysis could be implemented relatively easily. In 2012 Twitter held a much different position with relation to the political world, however from a technical standpoint sentiment analysis would be much the same. Martinez-Camara et al (2012) looked primarily at the use of emoticons, which doesn't initially seem very reliable, however is corroborated by Davidov et al (2010). Since those two papers have come out usage of emojis has drastically increased as well as the number and diversity of possible emojis, so perhaps that technique would be even more relevant now. Another interesting thing of note is that since the increase in character limit from 140 to 280, the average length of tweets has actually gone down slightly, only 1% of tweets hit the 280 character limit, and only 12% go over the original 140 characters from Perez et al (2018). This would be a very important metric to implement when trying to simulate a human tweeting to make the bot seem more human.

With specific regard to Twitter (and other social media sites) there are two key analytical features that could be used to make the sentiment analysis part of NLU much more accurate:

- Responses/likes/retweets ratios
- Reading other users' account histories

If properly implemented these could give a much deeper insight into a user base according to Zou et al (2017). One of the biggest draws to social media is how easy it is to express your liking or agreement of something; far more people express this through likes than responses, and retweets offer an even greater potential depth to the level of a users agreement. Furthermore, much like the Russian bots that were given initial knowledge of issues related to the American public but with even greater accuracy, a system could be designed to go into a specific user's past and figure out what kind of things they like or dislike.

Machine Learning

So clearly there are many different ways to do both NLG and NLU, another method that has become far more easily accessible in recent years is the use of machine learning. To build a system like the Russian interference in the 2016 election we would need to consider these two components separately. The Russian bots were doing natural language generation (NLG) with a supervised set of rules that Russian agents knew worked when it came to riling up Americans. They could then evaluate the effectiveness of the bots by doing sentiment analysis, which is primarily a challenge of natural language understanding (NLU). So what about if they didn't have that prior knowledge of how to rile up Americans? Would it be possible to make a system that could do this unsupervised? Would it also be possible to have it be more general-purpose, rather than the set goal of getting Donald Trump elected? The design and implementation of the following system will be done with the intention of teaching, in a cautionary manner, how machine learning systems can use computation sentiment analysis techniques to manipulate people's emotions on Twitter.

System requirements

Functional requirements

- The sentiment analysis knowledge base must learn how to analyse sentiment using the dataset of tweets
- The NLP will use the sentiment analysis knowledge base for two independent parts of the system
  - The NLU will use the SA knowledge base to read/understand tweets it has received
  - The NLG will use it to write/generate tweets
- The RLA will initially explore how to gain different responses based on altering its input
- The RLA will create tweets at regular intervals, with a default state of trying to receive neutral responses from the reader
- The RLA must take requests from its user and, using its previously explored knowledge, change how it acts to best complete the users request
- The user will interact with a GUI that allows them to control parameters using sliders to change what kind of response they want the RLA to aim for
- A user shall be able to view all the tweets being posted on the simulated Twitter forum
  - Tweets generated by the RLA
  - The "emotional response" generated by the simulated Twitter user who has "read" the tweets generated
- The RLA must get its reward from the simulated Twitter users evaluated response

Non-functional requirements

- The GUI should be responsive to user input
- The GUI must be clear and simple to use

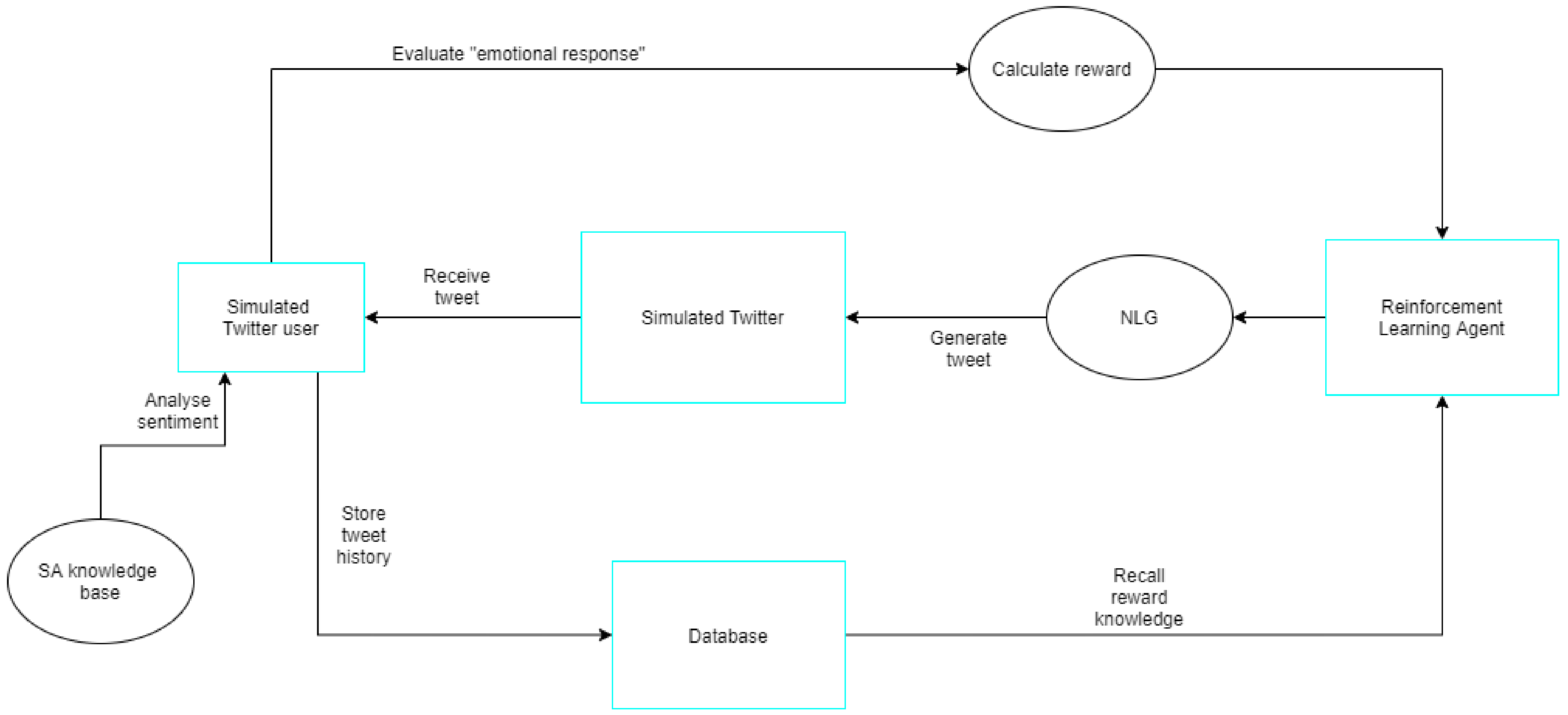
- The design of the system must follow proper development practices such that further functionality can be added following research into improvements
- The system must be designed to require minimal changes to deploy it into the real world; using input from actual Twitter rather than a simulation
- The code must run efficiently so that it will still function at scale
- The RLA must be able to receive further learning from other datasets for potential future developments

Basic structure

Rather than building NLP tools from scratch, I have set out to build a system that can learn how to generate natural language and post tweets that can garner a desired emotional response. As previously mentioned, the idea is to build a system that doesn't need to be told anything about its audience. My system consists of a reinforcement learning agent (RLA) that interacts in a conversational manner with its environment, which is essentially a simulated version of Twitter. Initially the RLA starts by mostly trying out tweeting in very different ways as it learns what emotional reactions these tweets get, then over time as it builds up its knowledge base it continues to tweet and take inputs from a user through a GUI so that the user can see how the system works. This system, at a basic level, shows that machine learning techniques can be used to analyse sentiment and learn how to manipulate people's emotions through social media. To follow from my research into labour theory and attachment to social media, the system must try to maximise a variety of emotional reactions; there should be sufficient agreeable content that makes its reader feel happy and comfortable, but also content that the reader can be angry at and hate.

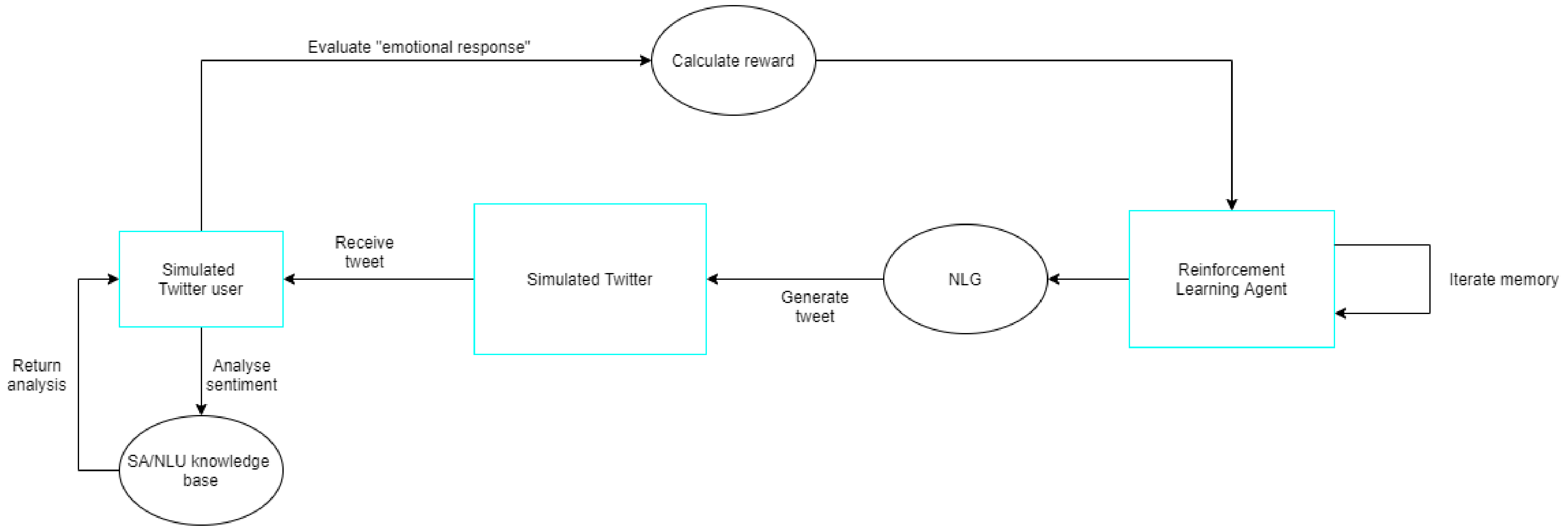
It would almost certainly be better to train a system like this in the real world; building a simulated version of Twitter will only detriment the performance and efficiency of the system. However it would be untenable from an ethical standpoint to potentially subject real people to emotional manipulation. As such I have opted for this approach whilst maintaining the prospect that the system would theoretically work similarly in the real world.

Here is my initial plan for the system during training phase:



Although the description of the system generally still fits, during the implementation of the system I have realised that the tweet history is not stored in a database then recalled by the RLA, instead the agent has a memory that is stored as one value which will be explained later. I also realised that the representation of how the simulated Twitter user uses the sentiment analysis knowledge base is not quite accurate so updated that as well.

This is my final diagram for the system during training phase:



It shows how the RLA sends a tweet to the simulated Twitter website and iterates its memory each time, then how the simulated Twitter user “reads” the tweet and uses the knowledge stored in its NLU model and returns an emotional response to the RLA which turns that value into a reward.

Reinforcement learning agent

Reinforcement learning agents are algorithms with a set of possible actions that they can take in a given environment. They are set up so they get a reward based on some outcome returned from the environment. They have a memory in which they store the actions they take along with the state of the environment when they took it and the reward that this state-action pair received. This process was first characterised by Bellman (1956) as part of his research into dynamic programming. It was first applied in control theory - theory concerned with the control of dynamic engineering systems - and later was applied in economics. This was his solution to the idea of optimality, which can be a variety of goals - minimizing travel time, minimizing cost, maximizing profits etc - in my case this goal is maximising emotional response .Rather than storing the entire memory of the system, which could very quickly become a massive list, the memory is instead represented as a function over time. The process in which a decision is made is known as a Markov decision process (MDP) - named for Andrey Markov - because Burke and Rosenblatt (1958) stated that the reward would only be derived from the current state-action pair. When considering continuous time-steps this idea is paired with the Hamilton-Jacobi equation - designed to identify conserved quantities i.e. make a continuous value discrete - to make a Hamilton-Jacobi-Bellman equation. Here we are concerned only with the Bellman equation as the time-steps are discrete.

TD and Q-learning

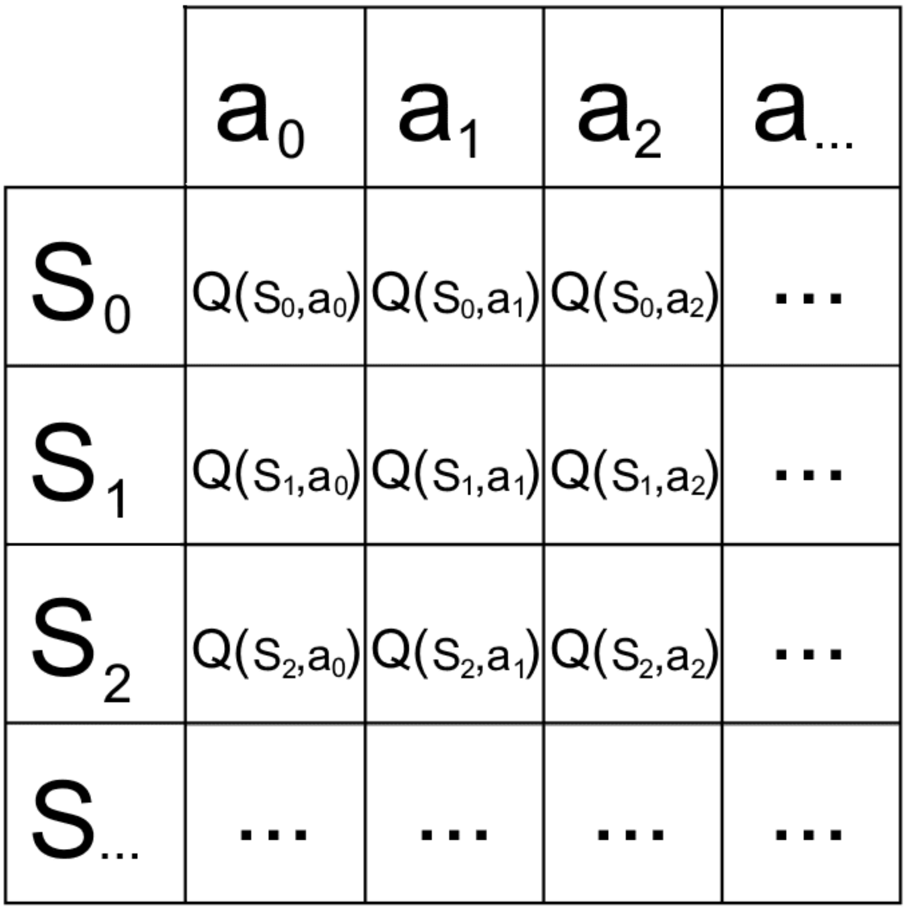
Temporal difference learning (TD) builds upon the Bellman equation and allows for the system to randomly sample from the possible actions it can take, so rather than always taking the best possible action, it can also be allowed to explore the environment it is in. Q-learning was devised by Watkins (1989) and is a form of TD. As the system is dynamic, it is given a learning rate; a number that deprecates the memory each time a state-action pair is decided, essentially slightly overwriting all the memory. This means that over time the older memories will have less and less impact on the current decision. It also incorporates an influence of the future by taking the maximum value action in the next state, which is also given a discount rate. The MDP is no-longer applicable because it now takes influence from the future and the past rather than solely from the present Gaon and Brafman (2019).

Q-learning is a model-free, bootstrapped, off-policy learning method.

- It is off-policy because it uses the value of the next state and its *greedy* action, as opposed to an on-policy learning method like SARSA which uses a pair of the next state and its *current* action assuming that the current policy will continue to be used as outlined by Sutton and Barto (2018)
- Brownlee (2018) explains that bootstrapping means that it takes random sampling from the potential actions it can take, and resamples its understanding based on its learning rate.
- Sutton and Barto (2018) go on to state that model-free means it starts by not having a model of its environment i.e. it doesn't know what it will encounter and has to learn

everything from scratch through what is essentially trial and error, making it very well suited to new environments.

The memory of a Q-learning agent can be represented as a Q-table, which stores all possible state-action pairs like this:



	a_0	a_1	a_2	$a_{...}$
S_0	$Q(s_0,a_0)$	$Q(s_0,a_1)$	$Q(s_0,a_2)$	...
S_1	$Q(s_1,a_0)$	$Q(s_1,a_1)$	$Q(s_1,a_2)$	...
S_2	$Q(s_2,a_0)$	$Q(s_2,a_1)$	$Q(s_2,a_2)$	...
$S_{...}$	...	...	...	...

As it takes each action, the position in the table relevant to that state-action pair is updated.

Deep Q-learning

Deep Q-learning is a method of improving the potential input size of an RLA, so rather than storing the state-action pair memories in a table it is mapped to the input layer of a neural network. This can be further extended to include a target network, where every N steps the weights from the working neural network are copied over to another network. This can increase the learning process's stability according to Wang (2020). This is the process I have used to develop an RLA in my project.

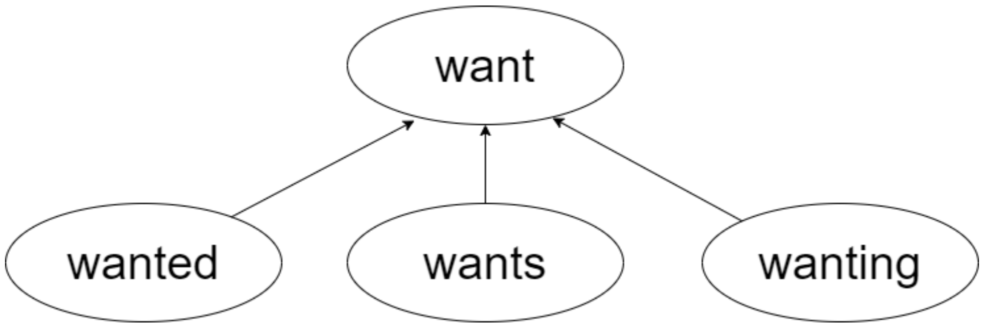
NLU as a learning environment

An RLA must have a learning environment to work in. To implement this I will build a neural network that can learn how to recognise sentiment from Twitter data and, upon reading a tweet generated by the RLA return a reward based on the sentiment it recognises - the higher the emotional impact of the tweet, the greater the reward.

Preprocessing

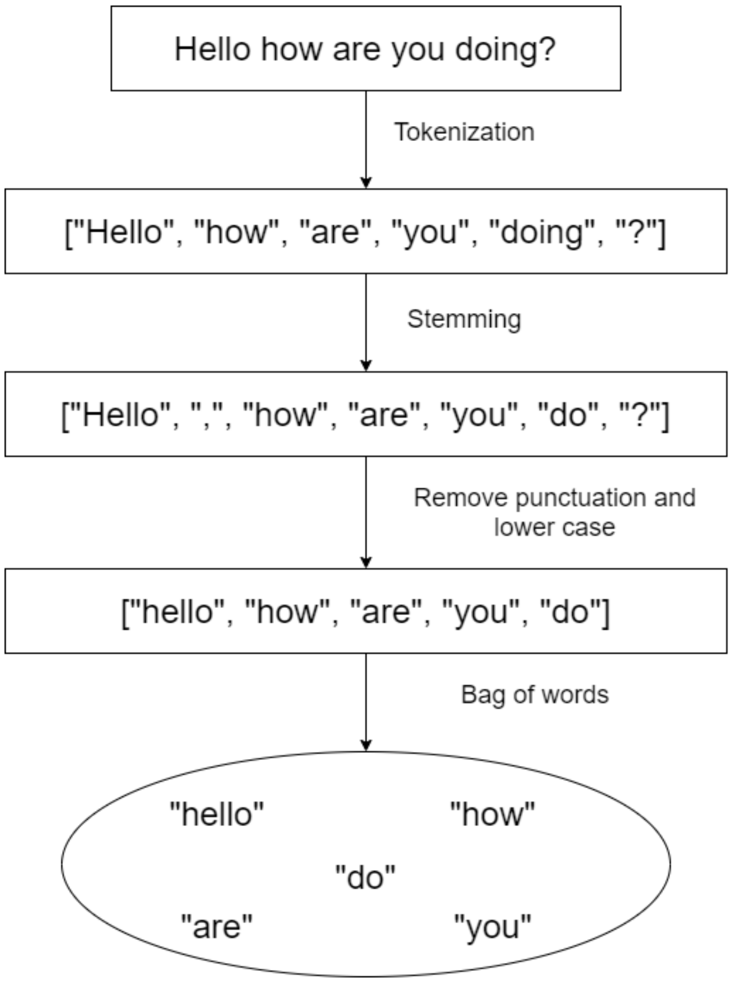
The first step is pre-processing, which is comprised of 3 sub-processes according to Nabi (2018):

1. Tokenization - Splitting a sentence/string into meaningful units, most commonly:
    - Words
    - Punctuation marks
    - Numbers
    - And if we were to implement some of the research of Martinez-Camara et al (2012) then this would also include emojis
2. Stemming - The process of finding the stem or root form of a word. Usually done with a crude heuristic for chopping off the ends of words.
3. Remove punctuation and set to lowercase.



4. Bag-of-words - Representing a sentence as a bag of the words in the sentence rather than as a structured sentence, however keeping multiplicity.

This process is essentially trying to abstract the meaning away from the actual sentence and make it much easier to process. Here is an example of the pre-processing pipeline being done on a sentence:


Hello how are you doing?
Tokenization
["Hello", "how", "are", "you", "doing", "?"]
Stemming
["Hello", ",", "how", "are", "you", "do", "?"]
Remove punctuation and lower case
["hello", "how", "are", "you", "do"]
Bag of words
"hello"
"how"
"do"
"are"
"you"

Dataset

For the NLU to learn an adequate dataset must first be selected. The selection of this data determines what sentiment the NLU will recognise thus deciding what emotional reaction the RLA will learn how to trigger. As this project is purely conceptual I have chosen a relatively simple but large dataset of 1.6 million tweets in which each tweet has been assigned a sentiment value of positive or negative by KazAnova (2017). As such this will be the limit of the emotion the RLA will learn how to generate. For the purpose of my system I removed any of the unnecessary metadata of each tweet which consisted of an assigned ID, the date of the tweet, the username of the person who tweeted and a “NO_QUERY” value. I also removed any words containing @ at the start of them signifying that the tweet was a reply as the aforementioned functionality of considering who is being tweeted at is beyond the scope of this project. The remaining data was the text of the tweet and its assigned sentiment value where 0 = negative, 4 = positive.

Some analysis of a sample of tweets will reveal how their sentiment has been determined (I have numbered them as such for referencing purposes)

Negative		Positive	
1	too worried and tired to post tonight	4	cant wait for prom and after prom party on friday
2	oh noes Poor you! What a a shame	5	hey happy birthday!
3	it was fun times and much needed relaxation. Went by too quickly though	6	has passed 1st year

When using the feature vector outlined by Neethu and Rajasree (2013), which used these 8 features to identify positivity vs negativity:

- Part of speech
- Special keyword
- Number of positive keywords
- Number of negative keywords
- Presence of negation
- Emoticon
- Number of positive hashtags
- Number of negative hashtags

It is very easy to see the comparison where the number of positive vs negative keywords is very easily identifiable: tweets 1 and 2 having the words “worried”, “tired”, “poor” and “shame”, and tweet 5 having “happy”. Tweet 3 includes “fun” and “relaxation” and no explicitly negative words, but it is the presence of negation at the end “though” that would make it overall a negative tweet. Similarly with tweet 4 “can’t” and “wait” are both somewhat negative words independently, but together the use of “can’t” to negate “wait” makes it overall positive. Tweet 6 is a bit harder to pin any of the feature vector to, the word “passed” on its own could very easily be positive or negative without the human understanding of the context that it is good to pass 1st year.

Having the system set up like this means that to get the RLA to understand other contexts we would simply need to swap out the dataset that the NLU trains on, then allow the RLA to work conversationally with it. If I were to experiment with such an implementation, keeping in mind the political aspects of my research, there are two datasets that I would use to do so. One dataset of tweets contains analysis of people’s sentiment towards candidates for the next Prime Minister of India in the run up to their general election, created by Kumar (2020). Each tweet has been assigned a neutral (0), positive (1) or negative (-1) sentiment value.

1	the three codes modi cracked give india huge foreign policy jumpstart via
-1	will not wear heart while vote this timechances heart failure
0	nirav modis bail plea hearing begins

As the direction of the sentiment is more nuanced than simply positive or negative some of the analysis is harder. With these three examples however it is fairly easy to see how they have done the analysis. In the first tweet, the word “huge” in the context is certainly positive, especially when paired with “jumpstart”. In the second the negation and the word “failure” at those specific points in the tweet are quite obviously negative. The final tweet, as a neutral example, accurately shows very little alignment with any of the 8 features proposed by Neethu and Rajasree.

The other dataset of tweets has been analysed for sentiment to extract hate speech, specifically racism and sexism by Toosi (2019). Here the assignment of 1 denotes it is hate speech and 0 is not.

Hate speech		Not hate speech	
1	if you hold open a door for a woman because she's a woman and not because it's a nice thing to do, that's . don't even try to deny it	4	i want to teach you love like you've never felt it before. #quote #quotes #love #pakistan #allin216 #girls #boys #fashion #feelings
2	@user "the dying of the light" village green/townÂ² #antisemitism #hocoschools #columbiamd #hocomd	5	we lost another member of our family yesterday:-(#gutted #rosie
3	how many#pols passed by how many times and said nothing? #bluelivesmatter #draintheswamp #ferguson	6	sequoia is about the #weekend!

Again here there is a lot more context required to make judgements. With tweets 2 and 3 it is easy to assign the negative hashtag feature to “#antisemitism” and “#bluelivesmatter”. When comparing tweet 1 and 4 we can see where the analysis works especially well; although 4 uses hashtags about gender and nationality, it is not hate speech, whereas 1 is while only talking about gender.

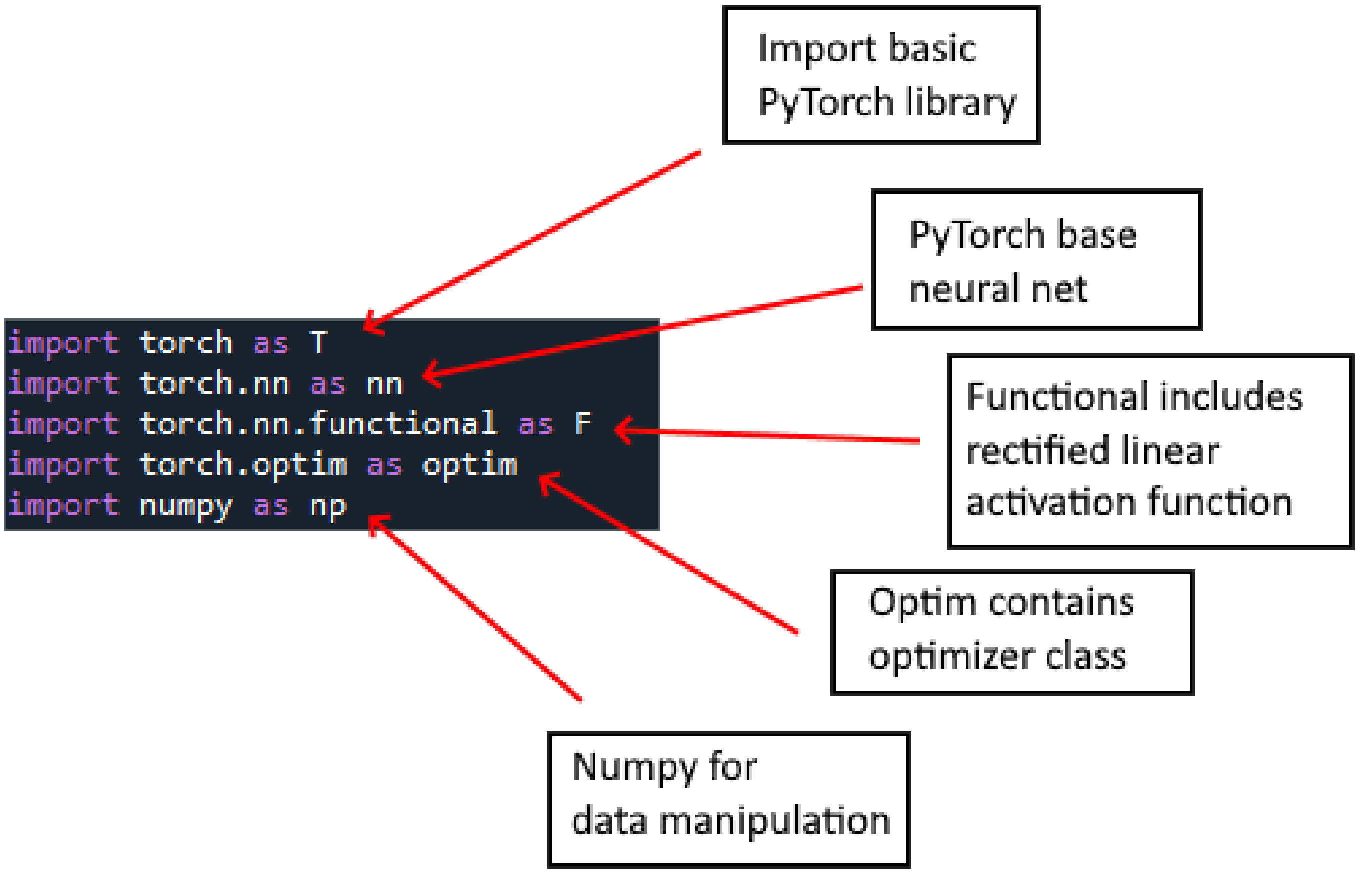
With these two alternative datasets it is apparent that it would be possible to extend my system to be able to generate politically oriented tweets, and rather than just generating tweets with negative sentiment it could trigger much stronger emotions by generating hate speech. Having the knowledge of both datasets combined would surely increase its capabilities much closer to that of the Russian socialbots.

Start of Q-learning stage 1

The deep Q learning agent has two main parts. First of all the agent has to be able to decide what actions it is going to take based on the current state of the environment. Secondly the agent has a deep neural network in which it stores its memory.

<u>Starting with the neural network</u>

The imports I have used for this project are:



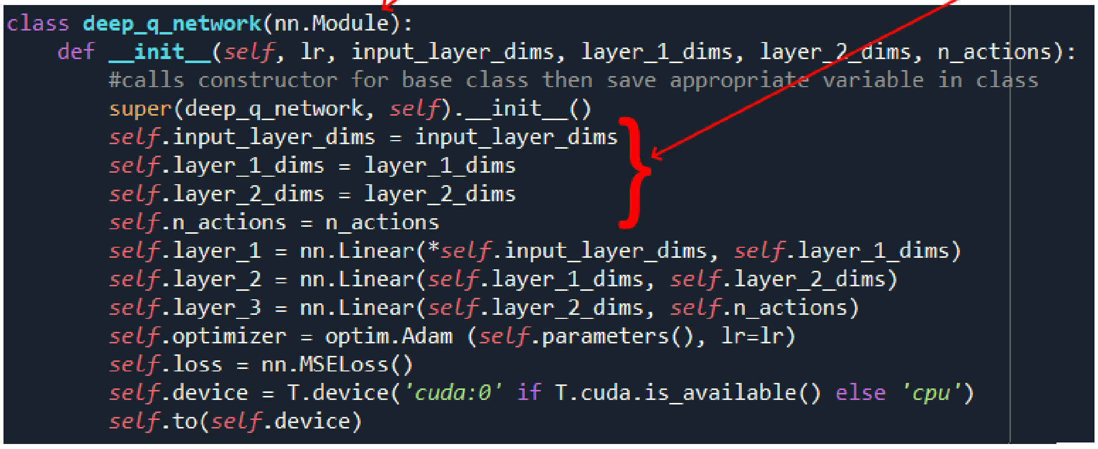
```
import torch as T
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import numpy as np
```

Import basic PyTorch library
PyTorch base neural net
Functional includes rectified linear activation function
Optim contains optimizer class
Numpy for data manipulation

For now I am just implementing a replay network but as aforementioned it would increase stability to use a target network as well - I have decided to go without for ease of implementation.

Build deep_q_network class by extending the built in PyTorch nn.Module object

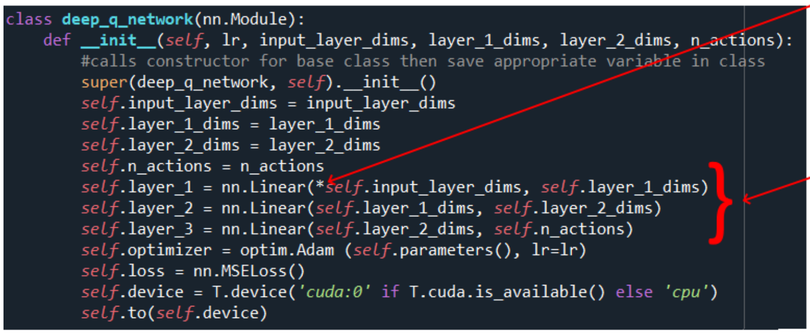
Initialise the dimensions of each layer of NN



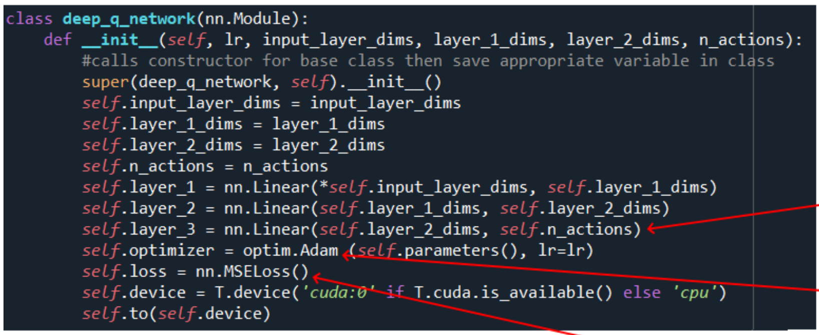
```python
class deep_q_network(nn.Module):
    def __init__(self, lr, input_layer_dims, layer_1_dims, layer_2_dims, n_actions):
        #calls constructor for base class then save appropriate variable in class
        super(deep_q_network, self).__init__()
        self.input_layer_dims = input_layer_dims
        self.layer_1_dims = layer_1_dims
        self.layer_2_dims = layer_2_dims
        self.n_actions = n_actions
        self.layer_1 = nn.Linear(*self.input_layer_dims, self.layer_1_dims)
        self.layer_2 = nn.Linear(self.layer_1_dims, self.layer_2_dims)
        self.layer_3 = nn.Linear(self.layer_2_dims, self.n_actions)
        self.optimizer = optim.Adam (self.parameters(), lr=lr)
        self.loss = nn.MSELoss()
        self.device = T.device('cuda:0' if T.cuda.is_available() else 'cpu')
        self.to(self.device)
```

* unpacks a list on inputs, this will allow for extension into a convultional NN in future

Each layer takes the layer before it's size as its own input size



```python
class deep_q_network(nn.Module):
    def __init__(self, lr, input_layer_dims, layer_1_dims, layer_2_dims, n_actions):
        #calls constructor for base class then save appropriate variable in class
        super(deep_q_network, self).__init__()
        self.input_layer_dims = input_layer_dims
        self.layer_1_dims = layer_1_dims
        self.layer_2_dims = layer_2_dims
        self.n_actions = n_actions
        self.layer_1 = nn.Linear(*self.input_layer_dims, self.layer_1_dims)
        self.layer_2 = nn.Linear(self.layer_1_dims, self.layer_2_dims)
        self.layer_3 = nn.Linear(self.layer_2_dims, self.n_actions)
        self.optimizer = optim.Adam (self.parameters(), lr=lr)
        self.loss = nn.MSELoss()
        self.device = T.device('cuda:0' if T.cuda.is_available() else 'cpu')
        self.to(self.device)
```

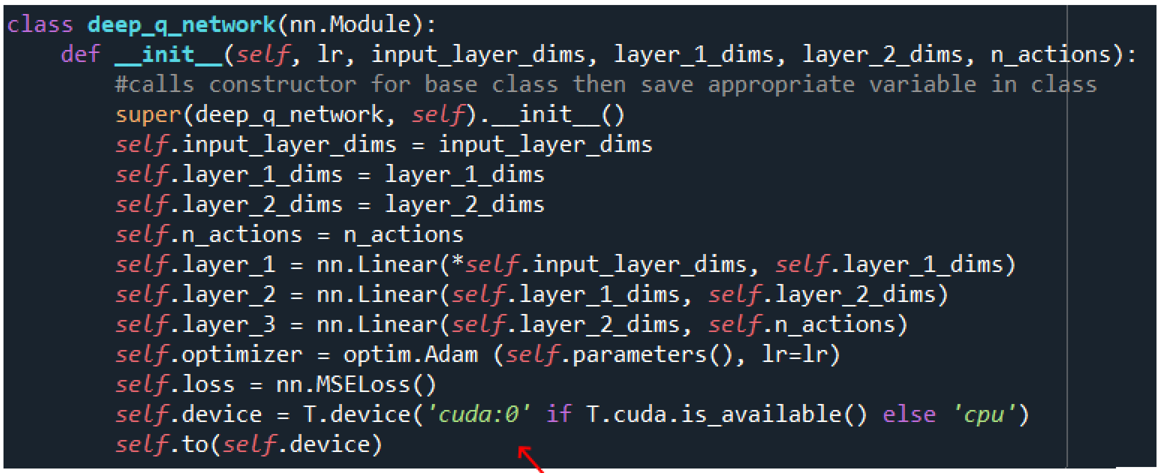


```python
class deep_q_network(nn.Module):
    def __init__(self, lr, input_layer_dims, layer_1_dims, layer_2_dims, n_actions):
        #calls constructor for base class then save appropriate variable in class
        super(deep_q_network, self).__init__()
        self.input_layer_dims = input_layer_dims
        self.layer_1_dims = layer_1_dims
        self.layer_2_dims = layer_2_dims
        self.n_actions = n_actions
        self.layer_1 = nn.Linear(*self.input_layer_dims, self.layer_1_dims)
        self.layer_2 = nn.Linear(self.layer_1_dims, self.layer_2_dims)
        self.layer_3 = nn.Linear(self.layer_2_dims, self.n_actions)
        self.optimizer = optim.Adam (self.parameters(), lr=lr)
        self.loss = nn.MSELoss()
        self.device = T.device('cuda:0' if T.cuda.is_available() else 'cpu')
        self.to(self.device)
```

Take the number of actions as the output layer size because the NN is a predictor of the next action of the DQN

Using Adam optimizer

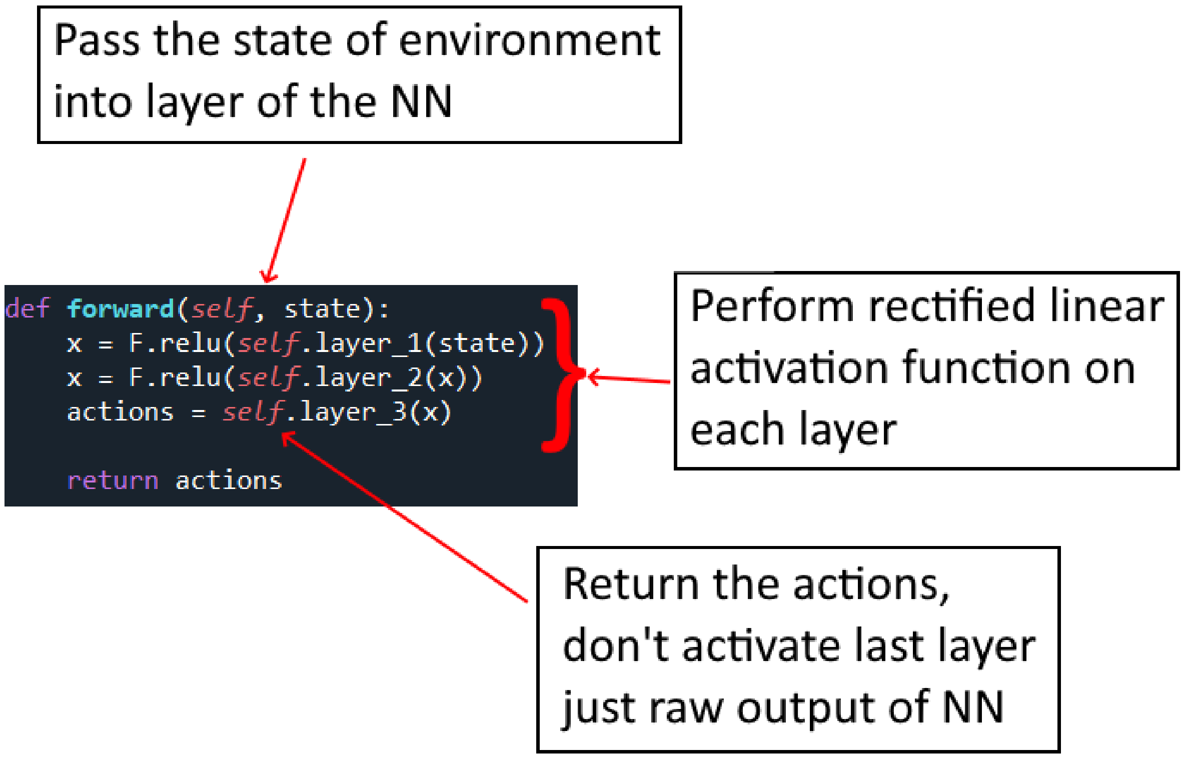
Mean squared error loss



```python
class deep_q_network(nn.Module):
    def __init__(self, lr, input_layer_dims, layer_1_dims, layer_2_dims, n_actions):
        #calls constructor for base class then save appropriate variable in class
        super(deep_q_network, self).__init__()
        self.input_layer_dims = input_layer_dims
        self.layer_1_dims = layer_1_dims
        self.layer_2_dims = layer_2_dims
        self.n_actions = n_actions
        self.layer_1 = nn.Linear(*self.input_layer_dims, self.layer_1_dims)
        self.layer_2 = nn.Linear(self.layer_1_dims, self.layer_2_dims)
        self.layer_3 = nn.Linear(self.layer_2_dims, self.n_actions)
        self.optimizer = optim.Adam (self.parameters(), lr=lr)
        self.loss = nn.MSELoss()
        self.device = T.device('cuda:0' if T.cuda.is_available() else 'cpu')
        self.to(self.device)
```

If the computer has a cuda-compatible GPU then use that as the device, this can dratically improve performance.

While backpropagation is handled by PyTorch, the neural networks forward pass is done manually.



```
def forward(self, state):
    x = F.relu(self.layer_1(state))
    x = F.relu(self.layer_2(x))
    actions = self.layer_3(x)

    return actions
```

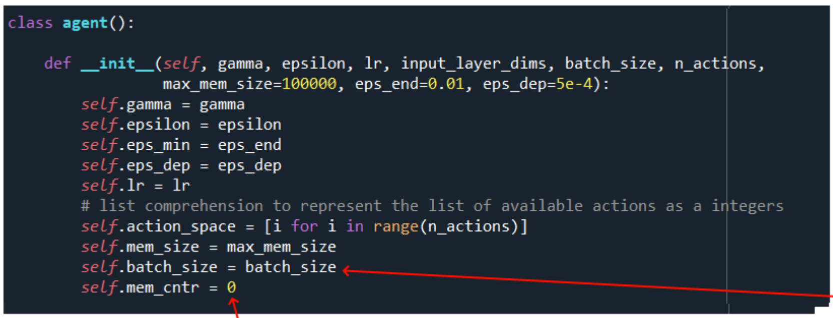
Pass the state of environment into layer of the NN
Perform rectified linear activation function on each layer
Return the actions, don't activate last layer just raw output of NN

Now onto agent class:

The agent has a set of hyperparameters that determine the various factors of q-learning.

Initialising the agent class

Hyperparameters as defined in research section:
- Gamma: Determines the weighting of future rewards
- Epsilon: Deprecating solution to explore vs exploit (ratio of how often agent spends exploring vs using best known action)
- eps_min: The minimum value epsilong can reach; at this point is has finished learning
- eps_dep: Rate at which epsilon deprecates
- lr: learning rate

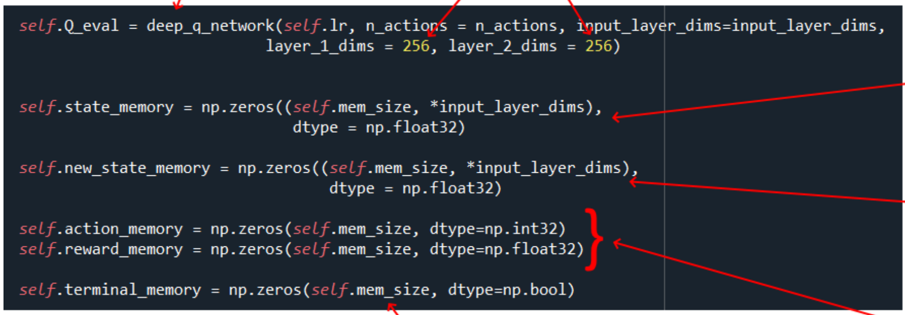


```
class agent():

    def __init__(self, gamma, epsilon, lr, input_layer_dims, batch_size, n_actions,
                 max_mem_size=100000, eps_end=0.01, eps_dep=5e-4):
        self.gamma = gamma
        self.epsilon = epsilon
        self.eps_min = eps_end
        self.eps_dep = eps_dep
        self.lr = lr
        # list comprehension to represent the list of available actions as a integers
        self.action_space = [i for i in range(n_actions)]
        self.mem_size = max_mem_size
        self.batch_size = batch_size
        self.mem_cntr = 0
```

Batch size is the size each batch of memories is that agent learns from

Signifies first available memory for that when it start to overwrite

Initialise evaluation network: dimensions of each layer can be tested at different values



```
self.Q_eval = deep_q_network(self.lr, n_actions = n_actions, input_layer_dims=input_layer_dims,
                             layer_1_dims = 256, layer_2_dims = 256)


self.state_memory = np.zeros((self.mem_size, *input_layer_dims),
                             dtype = np.float32)

self.new_state_memory = np.zeros((self.mem_size, *input_layer_dims),
                                 dtype = np.float32)

self.action_memory = np.zeros(self.mem_size, dtype=np.int32)
self.reward_memory = np.zeros(self.mem_size, dtype=np.float32)

self.terminal_memory = np.zeros(self.mem_size, dtype=np.bool)
```

Storing the memory of the current state as a numpy array. PyTorch enforces type checking so memory must be stored as float32 to avoid losses during precision transfers.

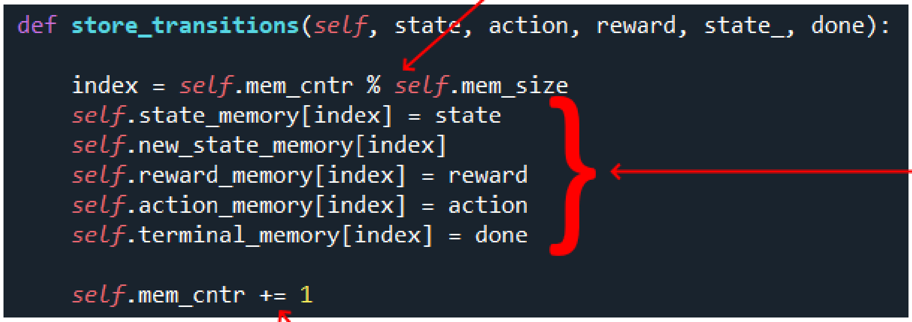
This stores the memory of the next step

These two store in the memory the action taken and the reward received.

This value is always 0, if it is encountered the Q-learning agent has reached the end-point of its environment.

Function to store the transitional data in the agent class:

Position of the agents first unoccupied memory so that once it has reached a certain memory it loops back round and starts overwriting old memories with new memories

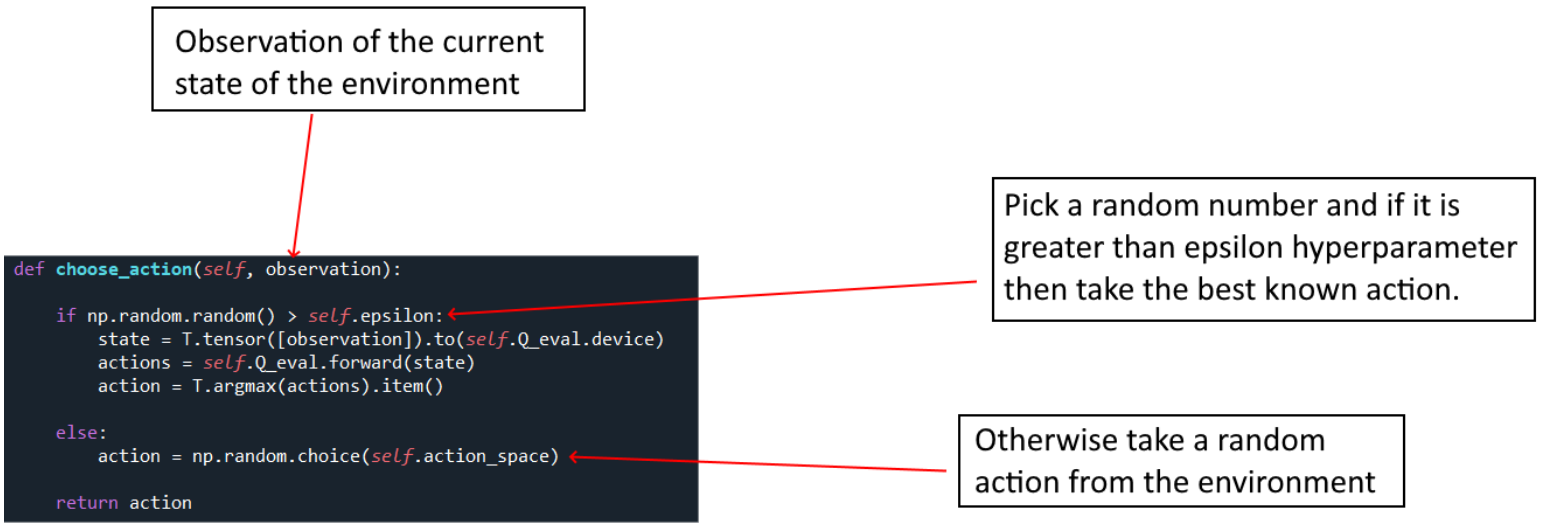


```
def store_transitions(self, state, action, reward, state_, done):

    index = self.mem_cntr % self.mem_size
    self.state_memory[index] = state
    self.new_state_memory[index]
    self.reward_memory[index] = reward
    self.action_memory[index] = action
    self.terminal_memory[index] = done

    self.mem_cntr += 1
```

Store memories in variables initialised above

Increment memory counter

<u>Function to choose the next action</u>:

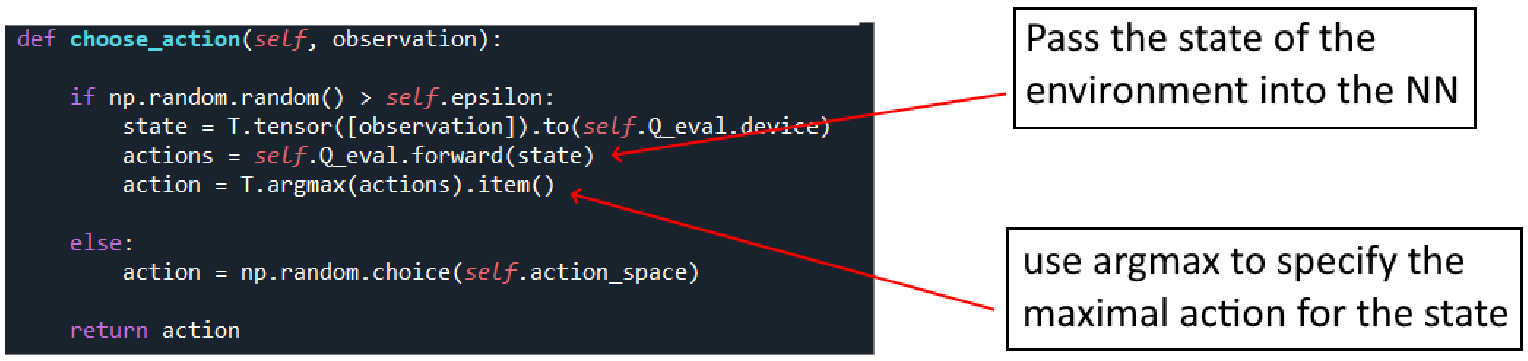


Observation of the current state of the environment

```python
def choose_action(self, observation):

    if np.random.random() > self.epsilon:
        state = T.tensor([observation]).to(self.Q_eval.device)
        actions = self.Q_eval.forward(state)
        action = T.argmax(actions).item()

    else:
        action = np.random.choice(self.action_space)

    return action
```

Pick a random number and if it is greater than epsilon hyperparameter then take the best known action.

Otherwise take a random action from the environment

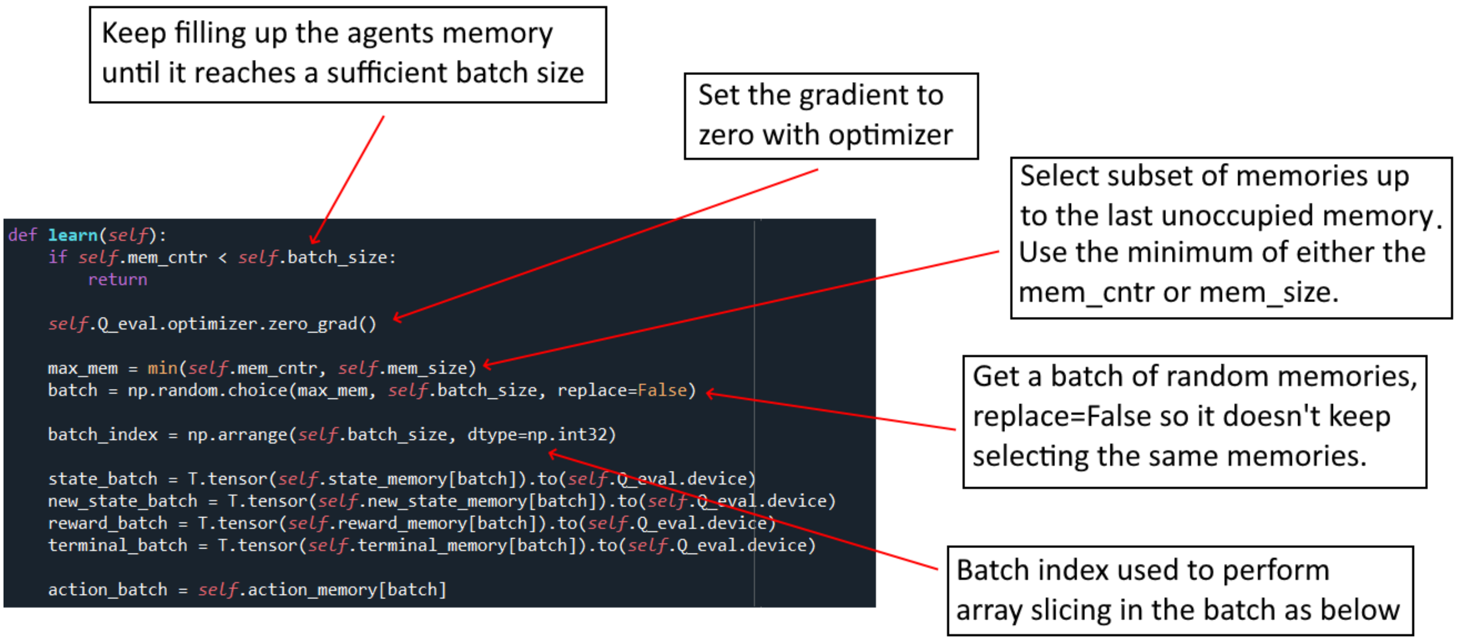


```python
def choose_action(self, observation):

    if np.random.random() > self.epsilon:
        state = T.tensor([observation]).to(self.Q_eval.device)
        actions = self.Q_eval.forward(state)
        action = T.argmax(actions).item()

    else:
        action = np.random.choice(self.action_space)

    return action
```

Pass the state of the environment into the NN

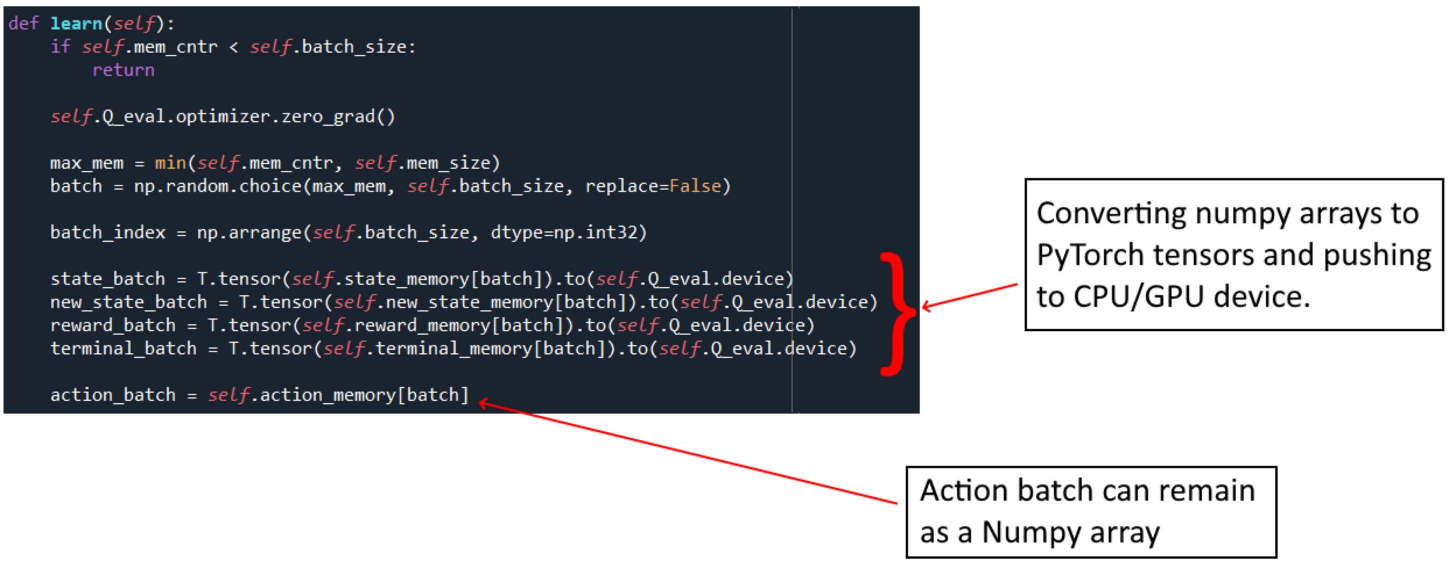
use argmax to specify the maximal action for the state

Function to handle the neural nets learning:


Keep filling up the agents memory until it reaches a sufficient batch size
Set the gradient to zero with optimizer
Select subset of memories up to the last unoccupied memory. Use the minimum of either the mem_cntr or mem_size.
Get a batch of random memories, replace=False so it doesn't keep selecting the same memories.
Batch index used to perform array slicing in the batch as below

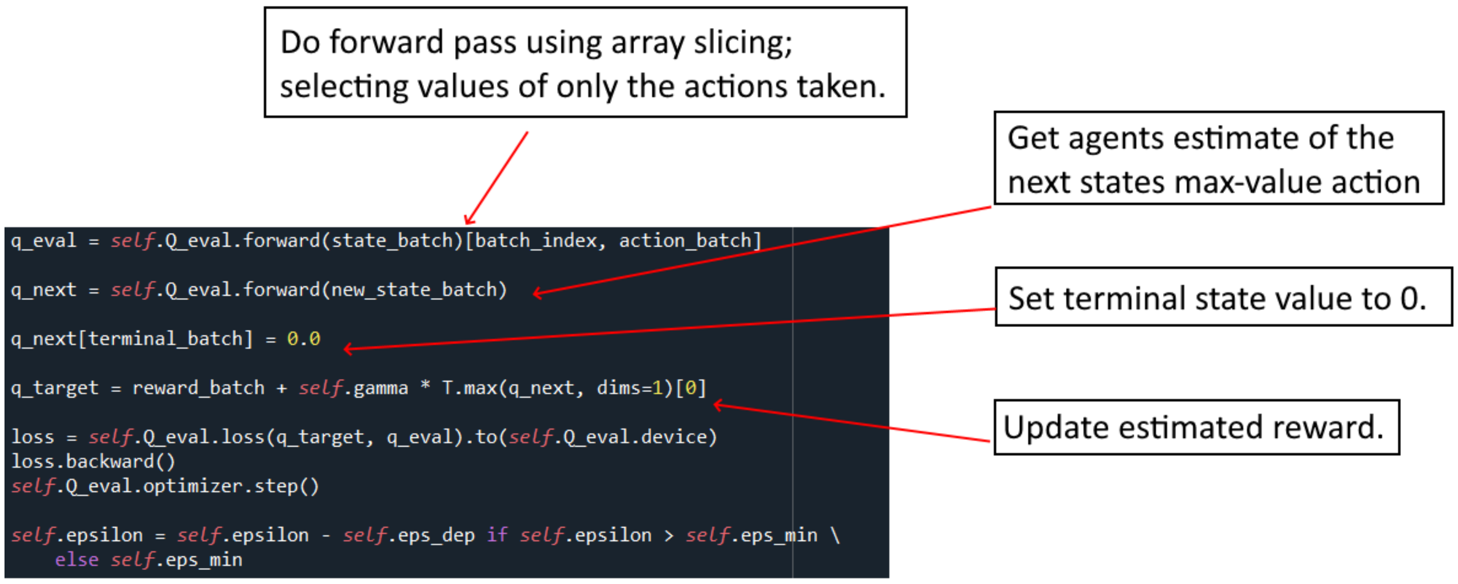
```
def learn(self):
    if self.mem_cntr < self.batch_size:
        return

    self.Q_eval.optimizer.zero_grad()

    max_mem = min(self.mem_cntr, self.mem_size)
    batch = np.random.choice(max_mem, self.batch_size, replace=False)

    batch_index = np.arrange(self.batch_size, dtype=np.int32)

    state_batch = T.tensor(self.state_memory[batch]).to(self.Q_eval.device)
    new_state_batch = T.tensor(self.new_state_memory[batch]).to(self.Q_eval.device)
    reward_batch = T.tensor(self.reward_memory[batch]).to(self.Q_eval.device)
    terminal_batch = T.tensor(self.terminal_memory[batch]).to(self.Q_eval.device)

    action_batch = self.action_memory[batch]
```

First initialise batches for learning.


Converting numpy arrays to PyTorch tensors and pushing to CPU/GPU device.
Action batch can remain as a Numpy array

```
def learn(self):
    if self.mem_cntr < self.batch_size:
        return

    self.Q_eval.optimizer.zero_grad()

    max_mem = min(self.mem_cntr, self.mem_size)
    batch = np.random.choice(max_mem, self.batch_size, replace=False)

    batch_index = np.arrange(self.batch_size, dtype=np.int32)

    state_batch = T.tensor(self.state_memory[batch]).to(self.Q_eval.device)
    new_state_batch = T.tensor(self.new_state_memory[batch]).to(self.Q_eval.device)
    reward_batch = T.tensor(self.reward_memory[batch]).to(self.Q_eval.device)
    terminal_batch = T.tensor(self.terminal_memory[batch]).to(self.Q_eval.device)

    action_batch = self.action_memory[batch]
```

Now perform feed-forward to get relevant parameters for loss function. Moving the agents estimate for the value of the current state toward the maximal value of the next state: make it tend toward selecting maximal actions. If using a target network it would be done here.



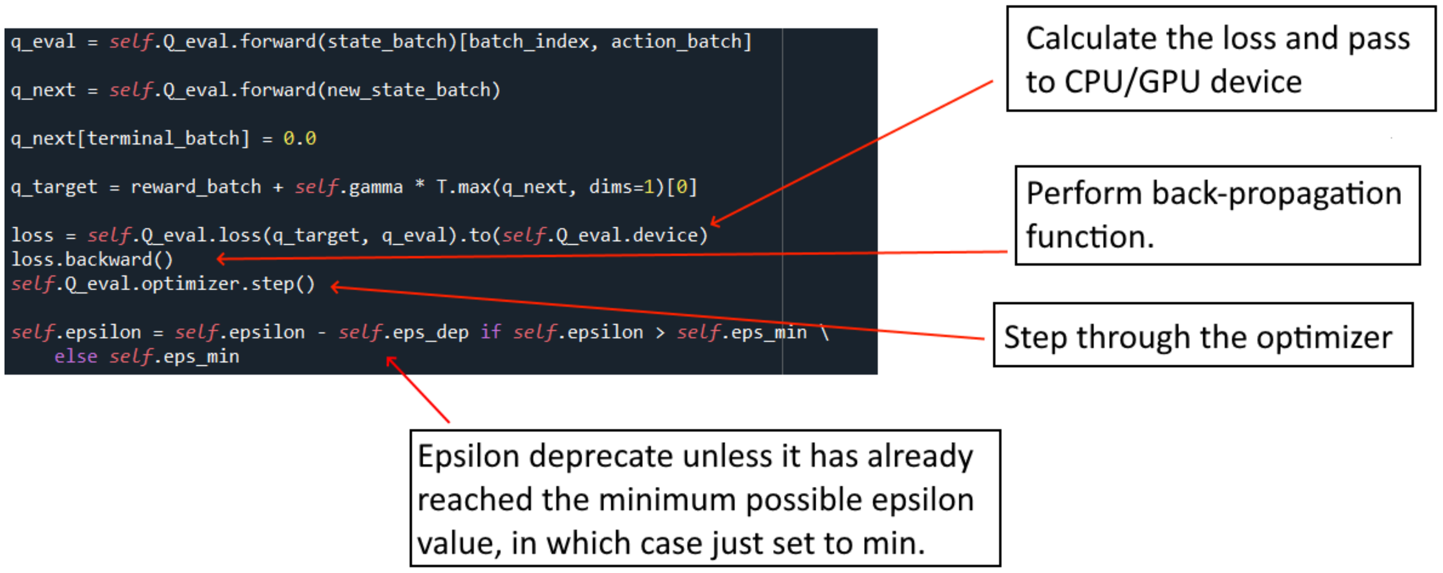
Do forward pass using array slicing; selecting values of only the actions taken.

Get agents estimate of the next states max-value action

Set terminal state value to 0.

Update estimated reward.

```
q_eval = self.Q_eval.forward(state_batch)[batch_index, action_batch]

q_next = self.Q_eval.forward(new_state_batch)

q_next[terminal_batch] = 0.0

q_target = reward_batch + self.gamma * T.max(q_next, dims=1)[0]

loss = self.Q_eval.loss(q_target, q_eval).to(self.Q_eval.device)
loss.backward()
self.Q_eval.optimizer.step()

self.epsilon = self.epsilon - self.eps_dep if self.epsilon > self.eps_min \
    else self.eps_min
```



```
q_eval = self.Q_eval.forward(state_batch)[batch_index, action_batch]

q_next = self.Q_eval.forward(new_state_batch)

q_next[terminal_batch] = 0.0

q_target = reward_batch + self.gamma * T.max(q_next, dims=1)[0]

loss = self.Q_eval.loss(q_target, q_eval).to(self.Q_eval.device)
loss.backward()
self.Q_eval.optimizer.step()

self.epsilon = self.epsilon - self.eps_dep if self.epsilon > self.eps_min \
    else self.eps_min
```

Calculate the loss and pass to CPU/GPU device

Perform back-propagation function.

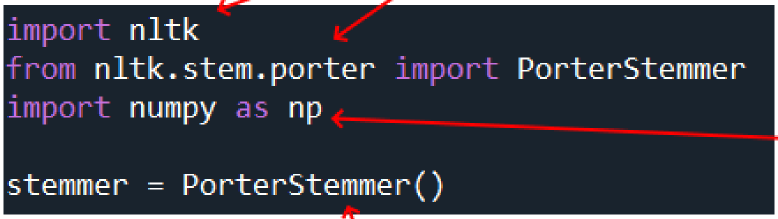
Step through the optimizer

Epsilon deprecate unless it has already reached the minimum possible epsilon value, in which case just set to min.

Now to build the learning environment for Q-learning agent. This is the simulated version of Twitter using essentially the NLU part of a chatbot trained on a dataset of 1.6 million sentiment analysed tweets.

Firstly I have made an nltk_utils file:

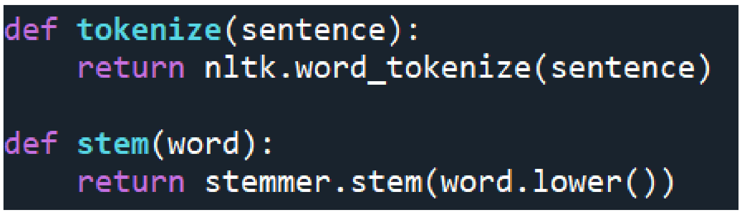
Natural language toolkit contains pre-built functions for data pre-processing as researched above.



```python
import nltk
from nltk.stem.porter import PorterStemmer
import numpy as np

stemmer = PorterStemmer()
```

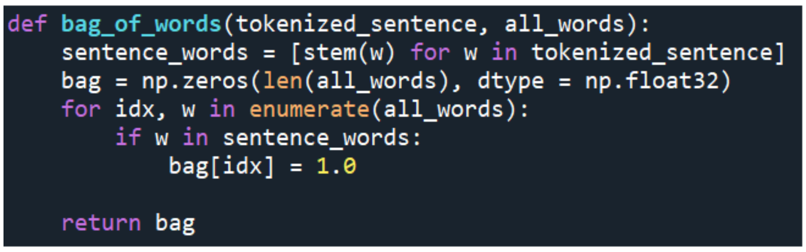
Numpy for data manipulation

Stemmer for the stemming part of NLU pre-processing. I have chosen PorterStemmer although there are others.



```python
def tokenize(sentence):
    return nltk.word_tokenize(sentence)

def stem(word):
    return stemmer.stem(word.lower())
```

Built functions to tokenize and stem sentences as outlined in research.

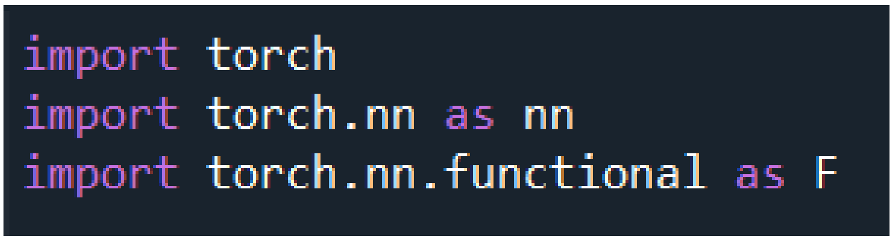


```python
def bag_of_words(tokenized_sentence, all_words):
    sentence_words = [stem(w) for w in tokenized_sentence]
    bag = np.zeros(len(all_words), dtype = np.float32)
    for idx, w in enumerate(all_words):
        if w in sentence_words:
            bag[idx] = 1.0

    return bag
```

The nltk Python library doesn't have a bag of words function so here is one made from scratch

Secondly to initialise the neural network:

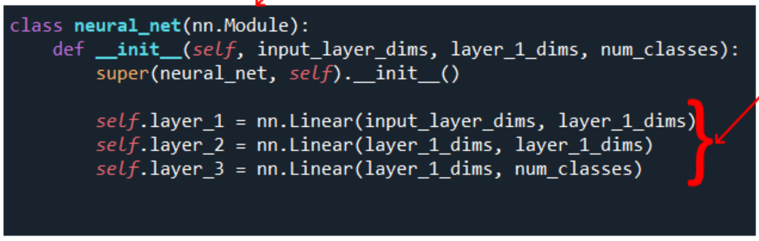
Again start by making the same imports but without numpy.



```python
import torch
import torch.nn as nn
import torch.nn.functional as F
```

At this point the neural network initialisation is much simpler as some of the functionality is to be put in other files.

Again initialise the neural network class by extending the basic PyTorch nn.Module object



```python
class neural_net(nn.Module):
    def __init__(self, input_layer_dims, layer_1_dims, num_classes):
        super(neural_net, self).__init__()

        self.layer_1 = nn.Linear(input_layer_dims, layer_1_dims)
        self.layer_2 = nn.Linear(layer_1_dims, layer_1_dims)
        self.layer_3 = nn.Linear(layer_1_dims, num_classes)
```

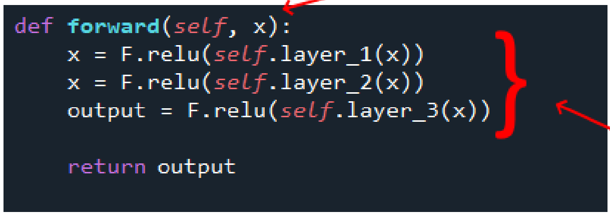
Initialising layers of neural network

Slightly improved efficiency with standard compared to Q-learning agent with standard Pythonic practices.

Does not need a loss function or optimizer here.

The forward pass is almost exactly the same as that of the Q-learning agent

Here instead of passing the state of the environment as with the Q-learning agent it simply handles x - the input values



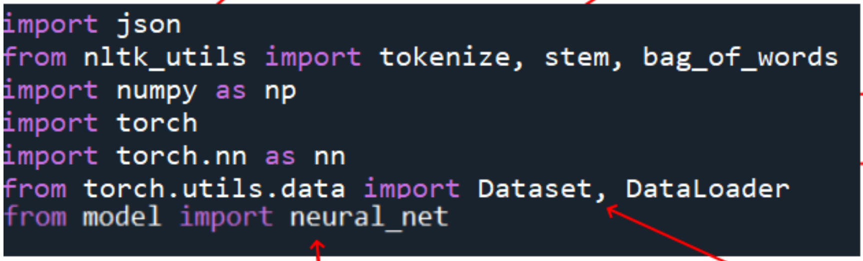
```python
def forward(self, x):
    x = F.relu(self.layer_1(x))
    x = F.relu(self.layer_2(x))
    output = F.relu(self.layer_3(x))

    return output
```

The rest of the forward pass functionality is the same, however output is used as instead of actions.

<u>Now constructing a new Python file to train the model</u>:

Importing json as this will help with inputting the text data

Import the previously built nltk_utils functions.



```python
import json
from nltk_utils import tokenize, stem, bag_of_words
import numpy as np
import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader
from model import neural_net
```

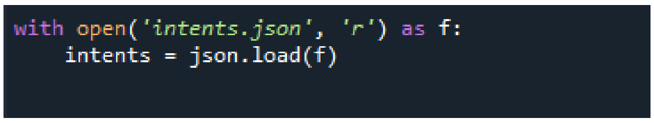
Numpy for data manipulation

Torch imports for the loss and optimisation which has been moved out of the model file

Import neural net model that has been initialised above.

PyTorch Dataset and Dataloader modules are used for manipulating text imports effectively.

I had planned for the NLG to be a set of sentences that covered a wide variety of emotions such that the RLA could choose from them and learn from them, however as the RLA reward system was not completed I left it as a basic list of intents.

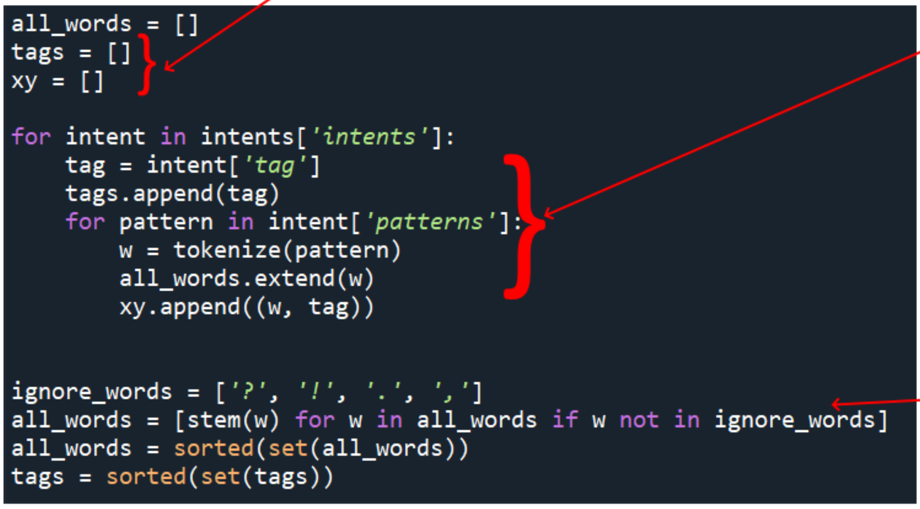


```python
with open('intents.json', 'r') as f:
    intents = json.load(f)
```

Here I have shown how the NLG would be had it been properly implemented.

Initialise lists to collect patterns and their tags

Go through each intent and
- add its tag to the list of tags
- go through list of patterns associated with tag and tokenize the sentence



```python
all_words = []
tags = []
xy = []

for intent in intents['intents']:
    tag = intent['tag']
    tags.append(tag)
    for pattern in intent['patterns']:
        w = tokenize(pattern)
        all_words.extend(w)
        xy.append((w, tag))


ignore_words = ['?', '!', '.', ',']
all_words = [stem(w) for w in all_words if w not in ignore_words]
all_words = sorted(set(all_words))
tags = sorted(set(tags))
```

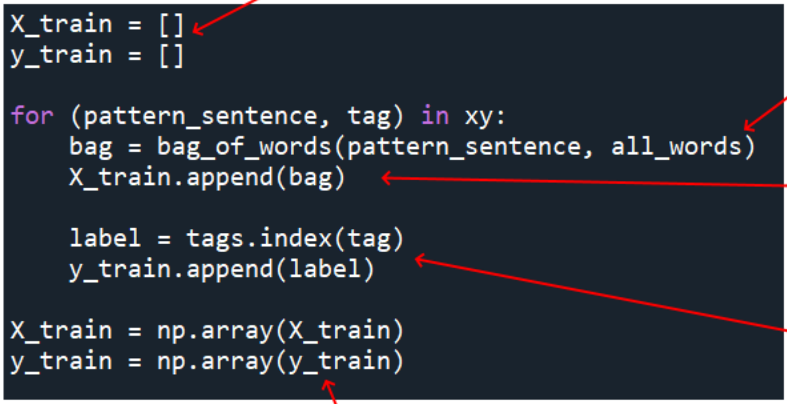
Remove punctuation and stem all of the words in the sentence

Initialise two lists for training

Run bag of words function on all the words for each pattern stored.

Add these to training data

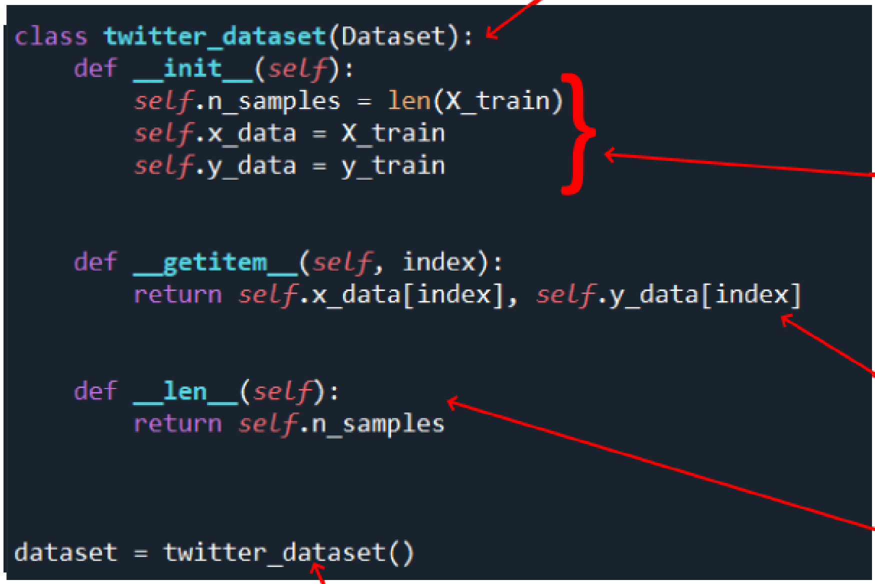
Use the index of each word as the label. This can be done in a one-hot fashion but as PyTorch does not require this I have not done so.



```python
X_train = []
y_train = []

for (pattern_sentence, tag) in xy:
    bag = bag_of_words(pattern_sentence, all_words)
    X_train.append(bag)

    label = tags.index(tag)
    y_train.append(label)

X_train = np.array(X_train)
y_train = np.array(y_train)
```

Convert to numpy arrays

Initialising dataset to take on the Twitter dataset as shown above - inherit Dataset module.



```python
class twitter_dataset(Dataset):
    def __init__(self):
        self.n_samples = len(X_train)
        self.x_data = X_train
        self.y_data = y_train


    def __getitem__(self, index):
        return self.x_data[index], self.y_data[index]


    def __len__(self):
        return self.n_samples



dataset = twitter_dataset()
```

Give dataset attributes x_data and y_data set to respective training arrays.

Set up two functions to allow for accessing the dataset by index

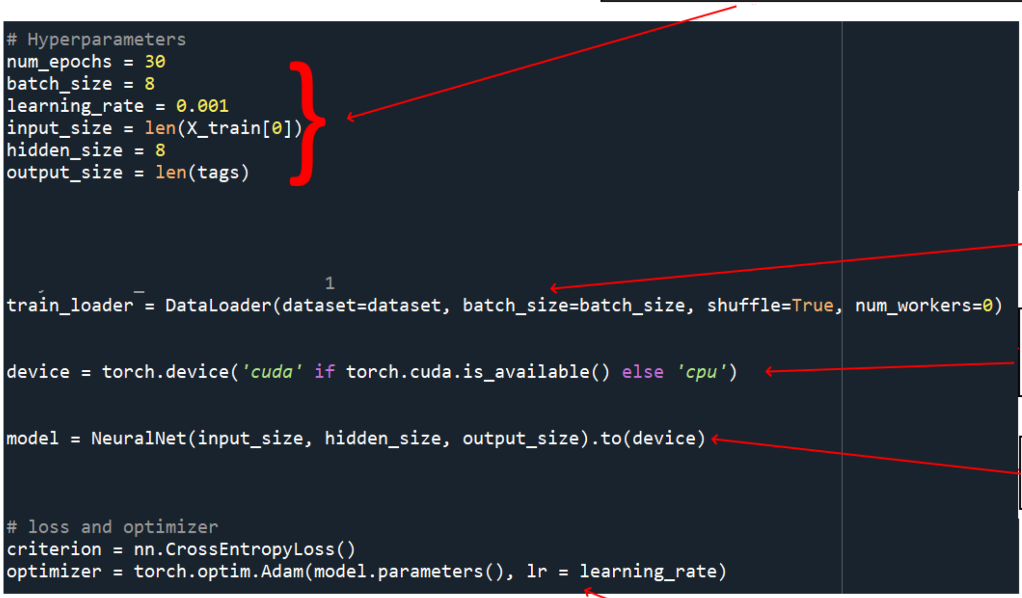
Function to return length

Create dataset

Hyperparameters of neural net:
- num_epochs determines the amount of cycles through the entire dataset

batch size, learning rate have been shown before

The sizes of each layer have been placed outside so they can be more easily changed when training a new dataset.



```python
# Hyperparameters
num_epochs = 30
batch_size = 8
learning_rate = 0.001
input_size = len(X_train[0])
hidden_size = 8
output_size = len(tags)



train_loader = DataLoader(dataset=dataset, batch_size=batch_size, shuffle=True, num_workers=0)


device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')


model = NeuralNet(input_size, hidden_size, output_size).to(device)


# loss and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr = learning_rate)
```

Initialise training loader

Again use a cuda compatible GPU if available

Make a new neural net model and push to CPU/GPU device

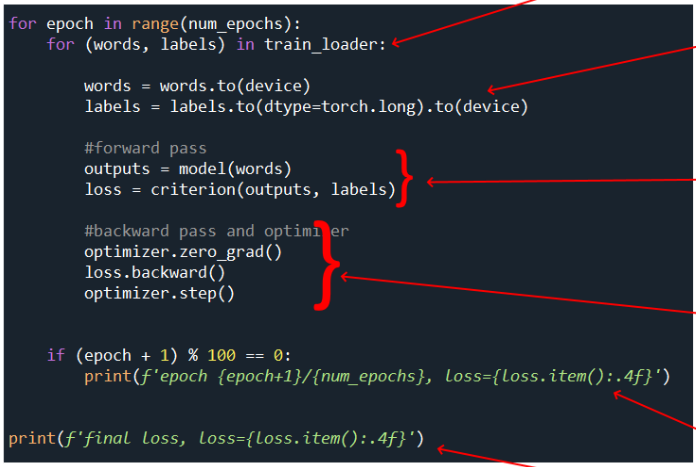
Using PyTorch's built in Cross entropy loss and again using Adam optimizer

<u>Now for the training loop</u>:

Over every epoch:

Get each word from training loader

Push words and labels to CPU/GPU device



```python
for epoch in range(num_epochs):
    for (words, labels) in train_loader:

        words = words.to(device)
        labels = labels.to(dtype=torch.long).to(device)

        #forward pass
        outputs = model(words)
        loss = criterion(outputs, labels)

        #backward pass and optimizer
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()


    if (epoch + 1) % 100 == 0:
        print(f'epoch {epoch+1}/{num_epochs}, loss={loss.item():.4f}')


print(f'final loss, loss={loss.item():.4f}')
```

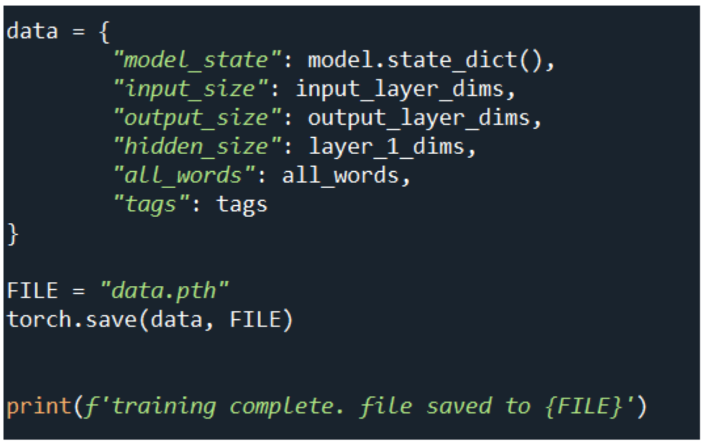
Perform forward pass using model neural network

Calculate the loss

Perform backward pass
- Empty gradients
- Pass the loss backward through each layer
- Step the optimizer

Print training details every 100 epochs

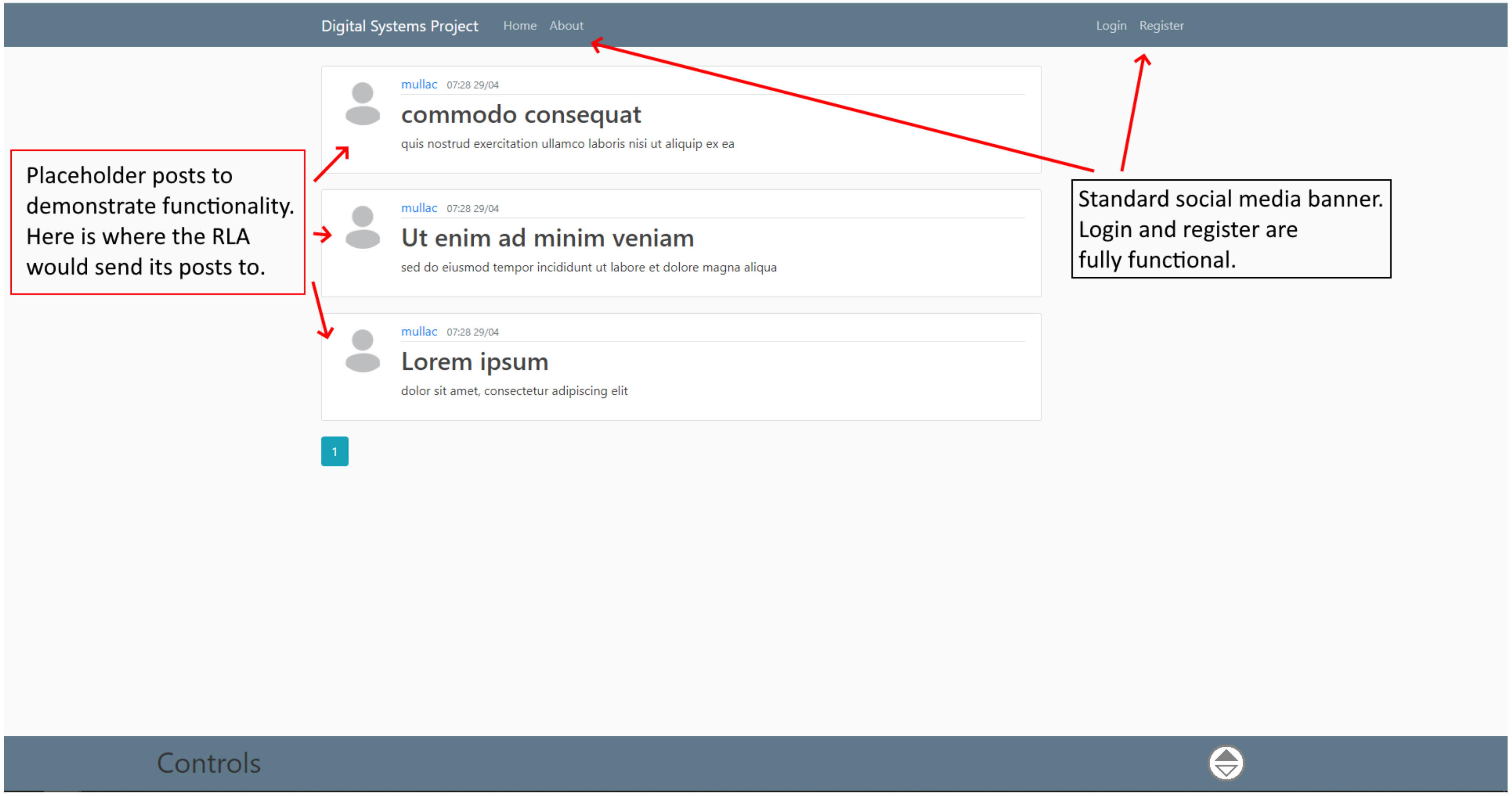
Print final data



```python
data = {
        "model_state": model.state_dict(),
        "input_size": input_layer_dims,
        "output_size": output_layer_dims,
        "hidden_size": layer_1_dims,
        "all_words": all_words,
        "tags": tags
}

FILE = "data.pth"
torch.save(data, FILE)


print(f'training complete. file saved to {FILE}')
```

Write the completed training data to a saved file.

This is the GUI that I built which would allow to interact with the RLA



mullac 07:28 29/04

commodo consequat

quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea

mullac 07:28 29/04

Ut enim ad minim veniam

sed do eiusmod tempor incididunt ut labore et dolore magna aliqua

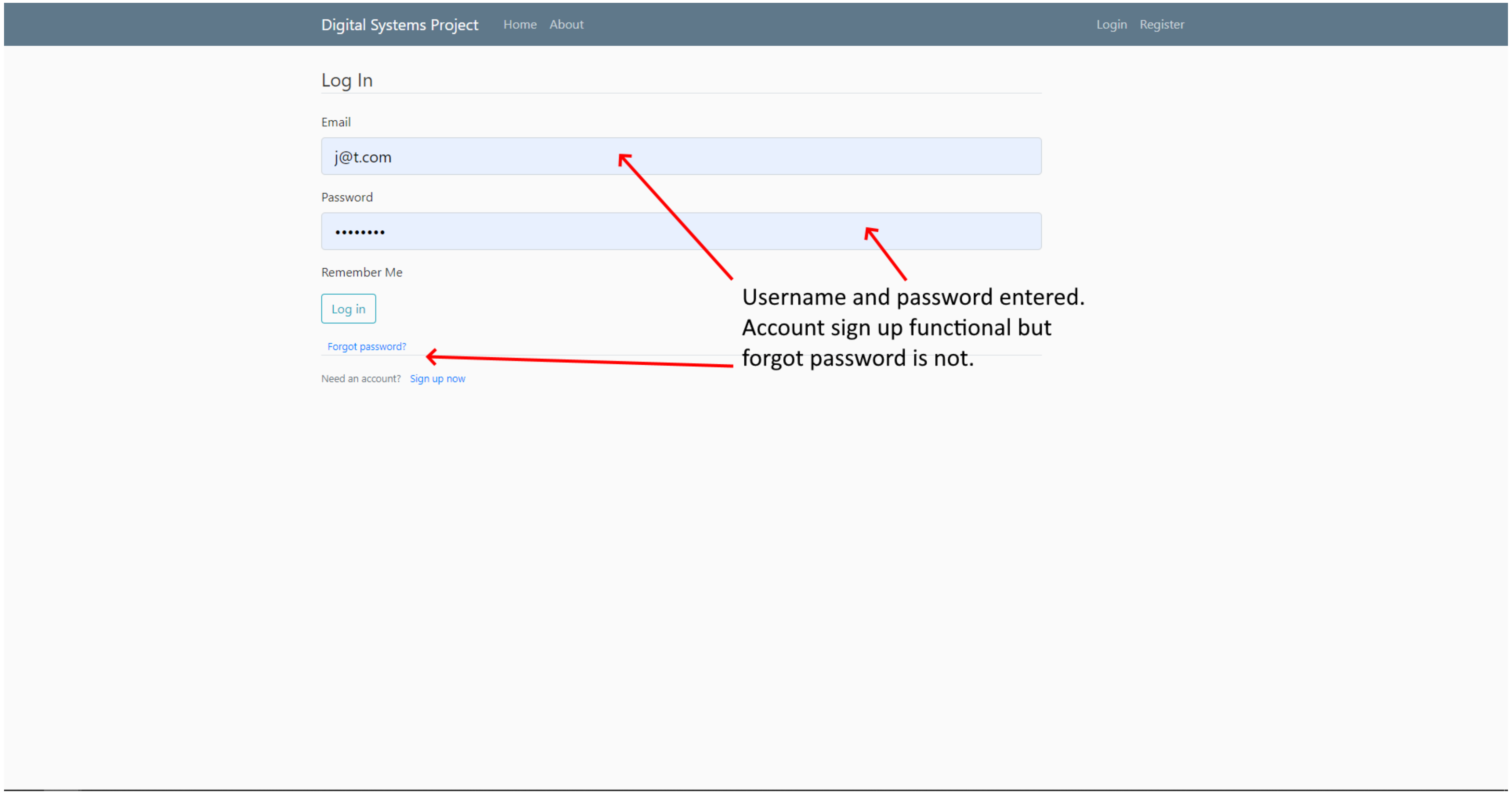
mullac 07:28 29/04

Lorem ipsum

dolor sit amet, consectetur adipiscing elit

Placeholder posts to demonstrate functionality. Here is where the RLA would send its posts to.

Standard social media banner. Login and register are fully functional.



Digital Systems Project Home About Login Register

Log In

Email

j@t.com

Password

••••••••

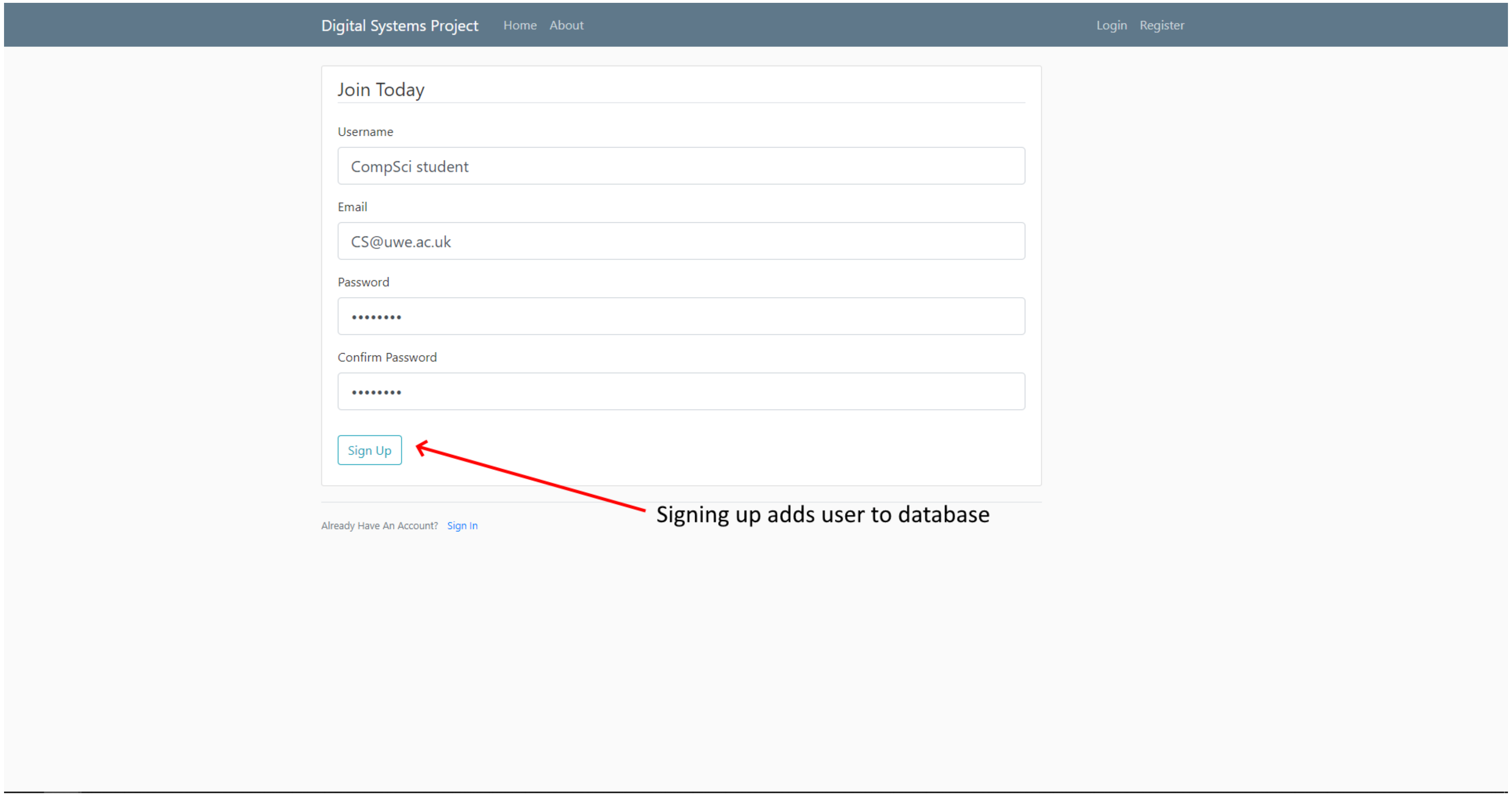
Remember Me

Log in

Forgot password?

Need an account? Sign up now

Username and password entered. Account sign up functional but forgot password is not.



Join Today

Username

CompSci student

Email

CS@uwe.ac.uk

Password

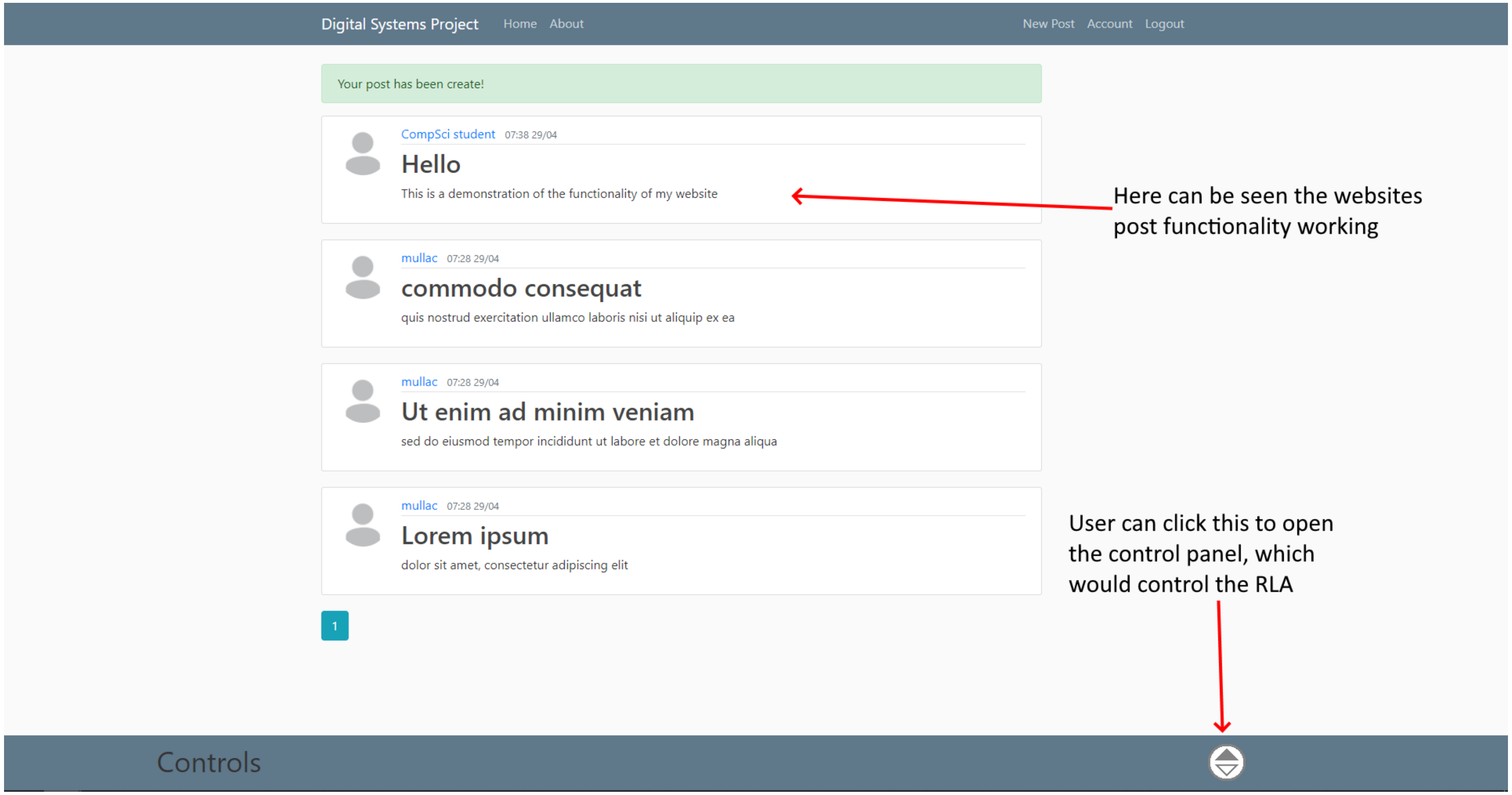
••••••••

Confirm Password

••••••••

Signing up adds user to database

Already Have An Account? Sign In



Digital Systems Project Home About New Post Account Logout

Your post has been create!

CompSci student 07:38 29/04

Hello

This is a demonstration of the functionality of my website

mullac 07:28 29/04

commodo consequat

quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea

mullac 07:28 29/04

Ut enim ad minim veniam

sed do eiusmod tempor incididunt ut labore et dolore magna aliqua

mullac 07:28 29/04

Lorem ipsum

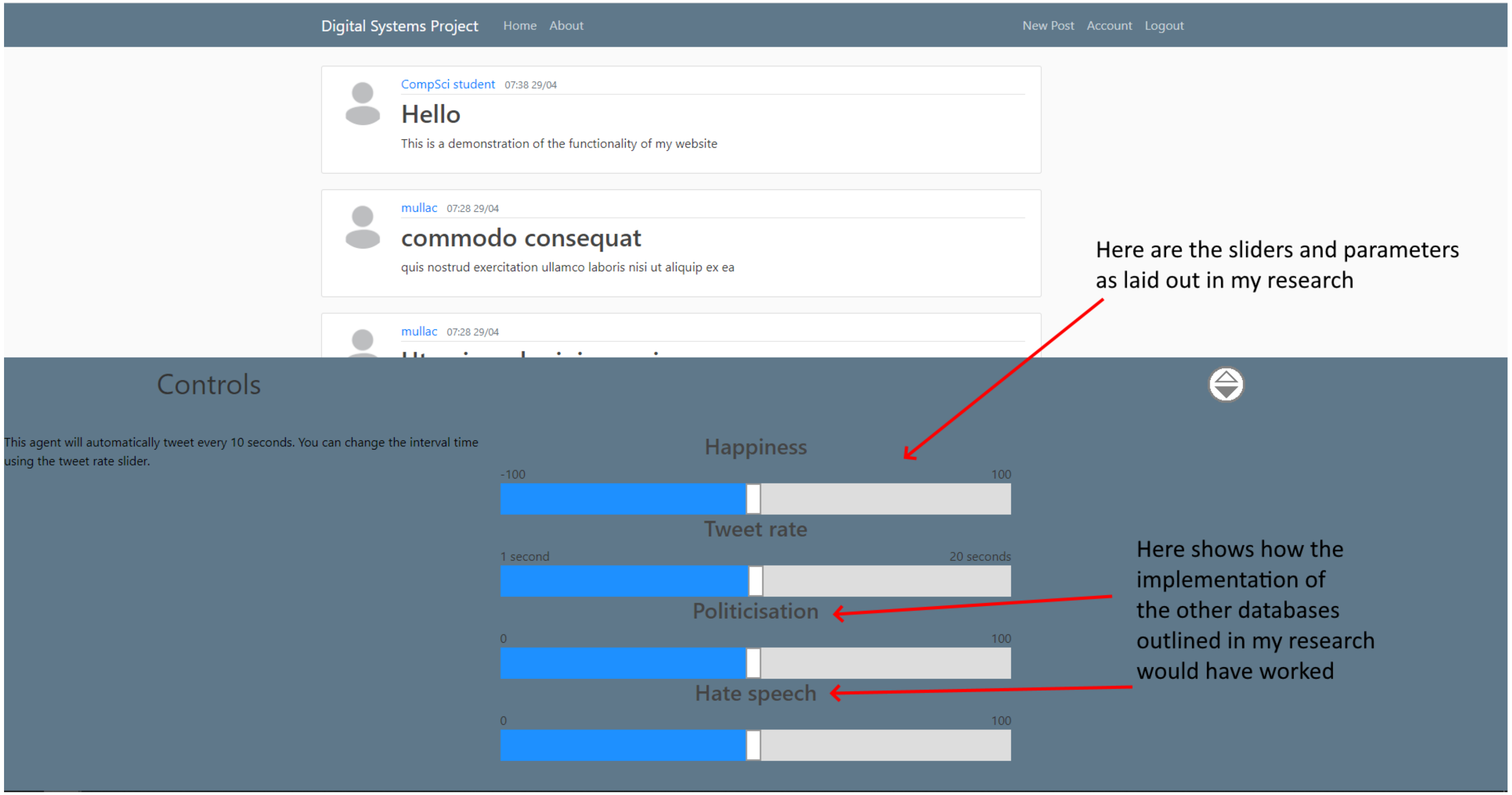
dolor sit amet, consectetur adipiscing elit

1

Controls

Here can be seen the websites post functionality working

User can click this to open the control panel, which would control the RLA



CompSci student 07:38 29/04

Hello

This is a demonstration of the functionality of my website

mullac 07:28 29/04

commodo consequat

quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea

mullac 07:28 29/04

Controls

This agent will automatically tweet every 10 seconds. You can change the interval time using the tweet rate slider.

Here are the sliders and parameters as laid out in my research

Here shows how the implementation of the other databases outlined in my research would have worked

Project evaluation

Based on the functional and non-functional requirements laid out before the project breakdown, and these further descriptive requirements:

- The system must show how, based on my research, people can be manipulated by what they read on Twitter, and how this can be automated using machine learning.
- The first step is to show a brief explanation of how neural networks learn and, through the GUI, allow them to start the learning process of the NLU.
- Next will be a brief explanation of how reinforcement learning works, again through the GUI.
- Then the user must be allowed to interact with this system and see how different parameters affect the performance of the system.

My evaluation is as follows:
Some of the functional requirements I set out have been met however due to time constraints the majority of them are not quite complete but would be with slight tweaks to the system: The knowledge base accurately learns how to analyse sentiment using the dataset of tweets; the user can interact with a simulated version of Twitter and see where they would be able to change parameters; and the user can see how the tweets would be presented. Each part of the system - the RLA, NLU and GUI - are at roughly 80% functionality.

- The RLA is fully built and works given simple instructions.
- The NLU has learnt from the database in line with my research
- The GUI is clear and working to a high degree

However the issues faced are to do with the entire system not quite coming together. The single major problem is that the RLA and NLU do not communicate properly as I didn't manage to construct a proper reward system.
These two following more minor issues would not take too long to implement but can't really be completed without the implementation of a reward system:

- The RLA doesn't automatically generate/post tweets.
- The NLG functionality of the RLA is non-functioning

Also, the GUI doesn't send the users input to the RLA parameters, however this is because my initial plan involved having the NLU trained on multiple databases as shown in my research, which I decided not to do as it would have been too complicated. However the sliders have been implemented on the GUI so the user can see what options they would have if viewed in tandem with my research.

The non-functional requirements are mostly met: the system does run on Windows, it has not been tested on Mac OS or Linux as I have not had access to either; The GUI is responsive to user input and is clear and simple to use; The system has been designed such that further functionality can be added; The code is efficient and would run at scale. Those that are not met are because of the aforementioned issues: if the RLA reward system was functioning then the system would be able to learn from other datasets; it may require substantial changes to deploy into the real world, but only because I decided to use a simple dataset - if it was trained on multiple datasets then it might be more ready to run based on real tweets.

Overall I think the system was reasonably successful and is only held back from being very successful by one major issue. My main criticism of my work is the time constraints, mostly due to my having spent more time than expected writing up the report section. Although I would have liked to see the practical project completed, I think this is a fair trade because I believe my report section has some very good research and insight and is very applicable to the attempted practical project.

Conclusion

This research shows that, as social media has become a much more vital and interconnected part of society - whether it be through its hybridisation with traditional media or it politicisation during the presidency of Donald Trump - immoral marketing firms and government espionage teams alike have seized this opportunity and exploited the personal, private data about individuals that the power of social media has gathered. This exploitation has essentially gone unchecked and people remain unaware, uninformed or uncaring. The remedy presented is again through the power of social media, but instead to strive for a deeper connection, solidarity and unity so that this nationalism and division can be overcome, and proper international regulation can be put in place giving everyone the privacy they deserve. This project was intended to educate people in the power and effectiveness of machine learning and how it is used to divide, and while it is incomplete, the research I have conducted clearly shows that a completed system would function as presented.

References

Smythe, D. (1977) Communications: Blindspot of Western Marxism. *Canadian Journal of Political and Social Theory/Revue.* 1 (3). [Accessed 12/02/2021].

Fischer, E., Fuchs, C. (2015). Audience Labour on Social Media: Learning from Sponsored Stories. *Reconsidering Value and Labour in the Digital Age.* pp. 115-132.

Yu-Qian, Z., Houn-Gee, C. (2015) Social media and human need satisfaction: Implications for social media marketing. *Business Horizons* [online]. 58 (3), pp 335-345.

Sheldon, K. M., Elliot, A. J., Kim, Y., Kasser, T. (2001) What is satisfying about satisfying events? Testing 10 candidate psychological needs. *Journal of Personality and Social Psychology* [online]. 80 (2), pp 325-339.

Chaffey, D. (2021) Global social media research summary 2021. *Smart Insights* [online]. [Accessed 12/02/2021].

The World Bank (2018) *Decline of Global Extreme Poverty Continues but Has Slowed: World Bank* [online]. Washington, D.C., US. Available from: https://www.worldbank.org/en/news/press-release/2018/09/19/decline-of-global-extreme-poverty-continues-but-has-slowed-world-bank

Carr, S. (2021) How Many Ads Do We See A Day In 2021? PPC Protect [blog]. 15 Feb. Available from https://ppcprotect.com/how-many-ads-do-we-see-a-day/ [Accessed 12/03/2021].

Bozios, T. Lekakos, G. Skoularidou, V. Chorianopoulos, K. (2001). Advanced Techniques for Personalized Advertising in a Digital TV Environment: The iMEDIA System. In: Conference: eBusiness and eWork Conference. Athens, Greece, January 2001.

Heath, R. (2007) How do we predict advertising attention and engagement? *School of Management Working Paper Series.* [online] [Accessed 21/03/2021].

Fisher, E, Fuchs C (2015) *Reconsidering Value and Labour in the Digital Age* [online]. London: Palgrave Macmillan. [Accessed 12/02/2021].

Dragliev, D. (2019) How to Analyze Your Facebook Ad Performance: 9 Ways. Social media examiner [blog]. 11 November. Available from https://www.socialmediaexaminer.com/how-to-analyze-facebook-ad-performance-9-ways/ [Accessed 12/03/2021].

GMI Blogger (2017) Scroll-speed - An Effective KPI For Measuring Content Consumption. Website Design and Development [blog]. 21 June. Available from https://www.globalmediainsight.com/blog/scroll-speed-effective-kpi/ [Accessed 12/03/2021].

YouTube support (2021) [online] California: Google. Available from: https://support.google.com/youtube/answer/2375431?hl=en-GB [Accessed 12/03/2021].

Appel, G. Grewal, L. Hadi, R. (2020) The future of social media in marketing. *Journal of the Academy of Marketing Science volume* [online]. 48, pp 79-95. [Accessed 14/03/2021].

McFarlane, G (2020) How Facebook, Twitter, Social Media Make Money From You. *Investopedia* [online]. [Accessed 12/02/2021].

Watson, Z. (2016) How to influence customer need recognition. Technology Advice [blog]. 7 July. Available from https://technologyadvice.com/blog/marketing/how-to-influence-customer-need-recognition/#:~:text=Need%20recognition%20occurs%20when%20someone,begin%20searching%20for%20an%20answer [Accessed 19/03/2021].

Dwivedi, Y. K., Ismagilova, E., Hughes, L.D., Carlson, J., Filierie, R., Jacobson, J., Jain, V., Karjaluoto, H., Kefi, H., Krishen, A.S., Kumar, V., Rahman, M.M., Raman, R., Raschnabel, P.A., Rowley, J., Salo, J., Tran, G.A., Wang, Y. (2020) Setting the future of digital and social media marketing research: Perspectives and research propositions. *International Journal of Information Management*. [Accessed 16/03/2021].

Isaak, J., Hanna, M.J. (2018) User Data Privacy: Facebook, Cambridge Analytica, and Privacy Protection. *IEEE* [online]. 51 (8), pp 56-59 [Accessed 02/04/2021].

Venturini, T., Rogers, R. (2019) “API-Based Research” or How can Digital Sociology and Journalism Studies Learn from the Facebook and Cambridge Analytica Data Breach. *Digital Journalism* [online]. 7 (4), pp 532-540 [Accessed 02/04/2021].

Alsunni, A.A., Latif, R. (2021) Higher emotional investment in social media is related to anxiety and depression in university students. *Journal of Taibah University Medical Sciences* [online]. 16 (2), pp 247-252 [Accessed 04/04/2021].

Zigmond, A.S., Snaith, R.P. (1983) The Hospital Anxiety and Depression Scale. *Acta Psychiatrica Scandinavica* [online]. 67 (6), pp 361-370. [Accessed 10/04/2021].

Wells, C. Shah, D. Lukito, J. (2020) Trump, Twitter, and news media responsiveness: A media systems approach. *New Media & Society* [online]. [Accessed 11/04/2021].

Edrington, C.L., Lee, N. (2018) Tweeting a Social Movement: Black Lives Matter and its use of Twitter to Share Information, Build Community, and Promote Action. *Journal of Public Interest Communication* [online]. 2 (2). [Accessed 10/04/2021].

Ekstrom, M., Shehata, A. (2016) Social media, porous boundaries, and the development of online political engagement among young citizens. *New Media & Society* [online]. 20 (2), pp 740-759. [Accessed 01/04/2021].

Mueller, R.S. (2019) Report On The Investigation Into Russian Interference In The 2016 Presidential Election. *US Department of Justice.*

Lee, D. (2018) The tactics of a Russian troll farm. *BBC* [online]. 16 February. Available from: https://www.bbc.co.uk/news/technology-43093390 [Accessed 02/02/2021].

Ellehuss, R., Ruy, D. (2020) Did Russia Influence Brexit? Center For Strategic & International Studies [blog]. 21 July. Available from: https://www.csis.org/blogs/brexit-bits-bobs-and-blogs/did-russia-influence-brexit [Accessed 05/02/2021].

Keating, A. Melis, G. (2017) Social media and youth political engagement: Preaching to the converted or providing a new voice for youth? *The British Journal of Politics and International Relations* [online]. 19 (4) pp 877-894. [Accessed 24/03/2021].

Rodriguez-Pose, A. (2018) The revenge of the places that don't matter (and what to do about it). *Cambridge Journal of Regions, Economy and Society* [online]. 11 (1) pp 189-209. [Accessed 24/03/2021].

Dorling, D. (2019) Rule Britannia: Brexit is the last gasp of empire. LSE [blog]. 20 February. Available from: https://blogs.lse.ac.uk/brexit/2019/02/20/misrule-britannia-brexit-is-the-last-gasp-of-empire/ [Accessed 28/03/2021]

Lichter, D.T., Ziliak, J.P. (2017) The Rural-Urban Interface: New Patterns of Spatial Interdependence and Inequality in America.
The ANNALS of the American Academy of Political and Social Science [online]. 672 (1), pp 6-25. [Accessed 28/03/2021].

Abreu, M. Oner, O. (2020) Disentangling the Brexit vote: The role of economic, social and cultural contexts in explaining the UK's EU referendum vote. *Environment and Planning A* [online] 52 (7), pp 1434-1456 [Accessed 29/03/2021].

Gellwitzki, C.N.L., Houde, A.M. (2021) Emotional Politicisation of the European Union: Towards a New Framework of Analysis. *Party Politics* [online] 8 (4), pp 389-403 [Accessed 30/03/2021].

Sandoval, M. (2016) The Hands and Brains of Digital Culture: Arguments for an Inclusive Approach to Cultural Labour. *Reconsidering Value and Labour in the Digital Age* [online] pp 42-59 [Accessed 30/03/2021].

McDonald, D.D. (2010) *Natural Language Generation* [online]. New York, US: Marcel Dekker [Accessed 30/03/2021].

Ratnaparkhi, A. (2002) Trainable Methods for Surface Natural Language Generation. 1st Meeting of the North American Chapter of the Association for Computational Linguistics. [Accessed 30/03/2021].

Gatt, A. Krahmer, E. (2017) Survey of the State of the Art in Natural Language Generation: Core tasks, applications and evaluation. *Journal of AI Research* [online] 61, pp 75-170. [Accessed 08/04/2021].

Duboue, P.A., McKeown, K. (2003) Statistical acquisition of content selection rules for natural language generation. Proceedings of the 2003 conference on Empirical methods in natural language processing.

Neethu, M.S., Rajasree, R. (2013) 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT). Tiruchengode, India, 4-6 July.

Martinez-Camara. E., Martin-Valdivia, M.T., Urena-Lopez, L.A., Montejo-Raez, A. (2014) Sentiment analysis in Twitter. *Natural Language Engineering* [online] 20 (01), pp 1-28. [Accessed 04/04/2021].

Davidov, D., Tsur, O., Rappoport, A. (2010) Enhanced Sentiment Learning Using Twitter Hashtags and Smileys. 23rd International Conference on Computational Linguistics. Beijing, China, August. Beijing China: Coling 2010.

Zou, L. Lam, N.S.N., Cai, H. Qiang, Y. (2018) Mining Twitter Data for Improved Understanding of Disaster Resilience. *Annals of the American Association of Geographers* [online]. 108 (5), pp 1422-1441 [Accessed 08/04/2021].

Bellman, R. (1956) Dynamic Programming and Lagrange Multipliers. *Applied Mathematical Modelling* [online]. 33 (3), pp 1457-1478 [Accessed 06/04/2021].

Burke, C.J., Rosenblatt, M. (1958) A Markovian Function of a Markov Chain. *The Annals of Mathematical Statistics* [online]. 29 (4), pp 1112-1122 [Accessed 27/03/2021].

Watkins, C.J.C.H., Dayan, P. (1992) Q-Learning. *Machine Learning* [online]. 8 pp 279-292 [Accessed 01/04/2021].

Gaon, M., Brafman, R.I. (1995) Reinforcement Learning with Non-Markovian Rewards. *Artificial Intelligence* [online]. 73 (1-2), pp 271-306 [Accessed 05/04/2021].

Sutton, R.S., Barto, A.G. (2018) *Reinforcement Learning* [online]. *Adaptive Computation and Machine Learning* [online]. Second edition. Cambridge, Massachusetts: MIT Press. [Accessed 06/04/2021].

Brownlee, J. (2020) *Deep Learning for Time Series Forecasting* [online] Edition 1.9. Machine Learning Mastery. [Accessed 06/04/2021].

Wang, M. (2020) Deep Q-Learning Tutorial: minDQN. Towards Data Science [blog]. 18 November. Available from: https://towardsdatascience.com/deep-q-learning-tutorial-mindqn-2a4c855abffc#:~:text=Critically%2C%20Deep%20Q%2DLearning%20replaces,process%20uses%202%20neural%20networks [Accessed 11/04/2021].

Nabi, J. (2018) Machine Learning — Text Processing. Towards Data Science [blog]. 13 September. Available from: https://towardsdatascience.com/machine-learning-text-processing-1d5a2d638958 [Accessed 14/04/2021].

KazAnova M.M. (2017) Sentiment140 dataset with 1.6 million tweets, *Kaggle* [online]. Available from: https://www.kaggle.com/kazanova/sentiment140 [Accessed 25/11/2020].

Kumar, C. (2020) Twitter and Reddit Sentimental analysis Dataset, *Kaggle* [online]. Available from: https://www.kaggle.com/cosmos98/twitter-and-reddit-sentimental-analysis-dataset?select=Twitter_Data.csv [Accessed 08/04/2021].

Toosi, A. (2019) Twitter Sentiment Analysis, *Kaggle* [online]. Available from: https://www.kaggle.com/arkhoshghalb/twitter-sentiment-analysis-hatred-speech?select=train.csv [Accessed 08/04/2021].