Applying Attention-Based LSTM Neural Networks for Personality Prediction in an Online Dating Application

Pavandeep Sidhu

BSc Artificial Intelligence and Computer Science

Supervised by Mohammed Bahja

10th April 2020

School of Computer Science

University of Birmingham

United Kingdom

Abstract

Online dating services use a wide variety of techniques to recommend matches to an individual, however, personality is often underutilised. To understand if personality could be used to pair users, a bidirectional long short-term memory (LSTM) neural network with an attention mechanism was developed to predict a user's personality. The similarity of two user's personalities was used as the deciding factor in the matching algorithm. A cross-platform iOS and Android app was created for users to talk to a chatbot, named Aida, and manage their matches. Aida asks common dating questions throughout the day to collect data for personality prediction. The neural network was trained on the Essays dataset and outperformed the current state of the art results on four of the Big Five personality traits, by an average of 3.22%. The BLSTM performed with the same accuracy on unseen data that was collected from a questionnaire, asking participants to answer several of Aida's questions. Overall, the application produced a mixed set of results during user testing. Some users were not comfortable giving answers about the personal information to a chatbot, however, others enjoyed the simplicity of the app and having personal conversations with Aida.

Contents

1 Introduction

1.1 Online Dating

Since the rise of the modern internet, the demand for free dating services has created a large industry for online dating. In 2019, the dating industry was worth 1.52 billion GBP and is increasing at a substantial rate (Statista 2020), with 20% of committed relationships begin online (eharmony 2015).

There is a big responsibility put on online dating services due to the increasing number of people finding partners on the internet. Currently, popular dating services focus on gamification, rather than improving the quality of matches (Rocha & Fakultet 2018). This leads to biases in dating services, favouring a small portion of users and adversely affects minority users (Chaney et al. 2018). Furthermore, there is often too much choice in matches (Iyengar & Lepper 2000), with the majority of user's time spent finding potential partners.

Personality plays an important role in predicting the quality of a relationship (Robins et al. 2000, Jarrett 2018). however popular dating systems do not always use it as a factor for matching. This research aims to show matches can be generated with personality similarity as the deciding factor.

1.2 Natural Language Processing

Within the area of Linguistics and Computer Science, Natural Language Processing (NLP) is the study into using computers to read, analyse and derive meaning from human language. There have been vast improvements in the field of NLP through increased computational power and deep learning with neural networks.

To achieve the goal of matching users based on personality, automatic personality prediction is used. This is the task of identifying a person's personality given a piece of text they provided. Automatic personality prediction from language is one of many text classification problems within NLP. Predicting personality from text is more difficult than other sources, such as videos or audio (Yang & Glaser 2017). That is not to say the limits of predicting personality through text have been reached.

Personality can be quantified using the Big Five personality traits which describes a user's personality along five dimensions: Agreeableness, Conscientiousness, Extroversion, Openness and Neuroticism.

1.3 Aims

The three main aims of my research are as follows:

1. Use an NLP based approach to create a matching algorithm for a dating application, where personality is used as a means of matching users
2. Design a chatbot that is capable of collecting data from an individual to later analyse their personality
3. Build a user interface to interact with the chatbot and talk to matches

2 Background

2.1 Dating Algorithms

It is important to recognise that no perfect dating algorithm exists due to the complexities and differences between people and cultures. This is best known in mathematics as the Stable Marriage Problem (Shi et al. 2018). Compromises must be made when deciding on the factors included in a dating algorithm.

2.1.1 Profile Matching

A profile matching algorithm surveys users to learn about their preferences for a partner such as height, age or interests. If two users have characteristics they both liked in each other, they will be matched.

Match.com and eHarmony announced they used preference matching to pair users (Kessler 2011, Knapton 2017). Both platforms discovered issues when a user's preferences and activity on the service did not match. For example, 57% of women on the website who say a partner's desire to have children is a "must-have", messaged partners who did not fit the criteria. Hence, their recommendation algorithms adjust for criteria that can be compromised, such as recommending partners with no desire to have children.

2.1.2 Collaborative Filtering

Many systems use collaborative filtering for recommending products on e-commerce websites, movies on streaming websites or people to follow on social media. Within the area of dating, it is used for recommending matches to other users.

Collaborative filtering is the process of filtering items using the opinions of other people (Schafer 2007). It is based around the assumption that similar people like similar things. A range of machine learning methods can be used for collaborative filtering, including clustering algorithms, matrix factorization (Mustafa et al. 2017) and, more recently, neural networks (He et al. 2017).

This technique has been used in dating algorithms with success (Krzywicki et al. 2015, Brozovsky & Petricek 2007). However, there is evidence to suggest that collaborative filtering homogenises user behaviour (Chaney et al. 2018) and reduces the utility of the algorithm. This kind of algorithm can become problematic due to popularity bias. Users with lots in common with others will always be recommended, leading to a small number of popular individuals. Alongside this, early users of the system have a large impact on future decisions made by the recommendation system. If an early user dismisses a profile who has a certain attribute, a similar individual would not be recommended the profile since it is assumed they will also dislike the profile.

2.1.3 Elo Score

Popular mobile dating app, Tinder works by displaying a stack of profiles and asking users to swipe right to show interest or swipe left to dismiss the profile. If two users both swipe right for each other, they match and can talk within the app.

The algorithm for sorting profiles is Tinder's most important aspect. Although many dating services do not disclose their algorithms, Tinder revealed they previously used an Elo rating system (Tinder 2019). This is commonly used to rate the skills of chess players (Tiffany 2019) but has been implemented in many other areas, including sports, video games and board games. For Tinder, Elo assigns users with a score, the higher the score the more they are perceived as desirable. An individual's score increases when other users swipe right on their profile and decreases if swiped left. The effect on an individual's score is more substantial if the user swiping has more points. This can be seen in the mathematical representation of the Elo rating system (Aldous 2017).

Each user i is given an initial rating, a real number y_i . When user i comes across the profile of user j, the ratings of both players are updated using a function Υ:

$$\begin{aligned} \text{if } i \text{ beats } j \text{ then } y_i \to y_i + \Upsilon(y_i - y_j) \text{ and } y_j - \Upsilon(y_i - y_j) \\ \text{if } i \text{ loses to } j \text{ then } y_i \to y_i - \Upsilon(y_j - y_i) \text{ and } y_j + \Upsilon(y_j - y_i) \end{aligned} \tag{1}$$

2.2 Chatbots

Chatbots are not very common within the dating industry. Match.com released a dating coach chatbot named Lara as a companion for their dating service (Match 2018). Lara provides the user with daily matches, dating tips and date suggestions. Besides Lara, no dating platform has used a chatbot as an integral part of their system to analyse users.

Although the chatbot is a means of collecting user information for personality recognition, it is not the area of focus for this report. A brief overview of the types of chatbots (Jwala et al. 2019) is provided to help understand the current state of chatbots and its research.

2.2.1 Rule-Based Chatbots

The most common chatbots use a rule-based approach where a conversation follows a pre-defined flow. They work best for chatbots that have a specific goal in mind. Techniques such as pattern matching are used to build such chatbots. This makes them very easy to build, however, limited in terms of user experience. Artificial Intelligence Markup Language (AIML) is a language based on XML used to write rules for a chatbot

to follow (Das Graças et al. 2013).

The user experience is much more controlled using rule-based chatbots as they are predictable and easy to build. Problems occur when a user requires information or asks a query that does not follow the conversational flow. For repetitive tasks, rule-based chatbots work well but more complex requirements require more advanced methods.

2.2.2 Generative Chatbots

Generative chatbots use machine learning models to have human-like conversations, as opposed to following a fixed conversational flow. Answers are generated rather than coming from a fixed set of predefined responses. Generative chatbot models take a user query word by word and so are more likely to fail due to user spelling and grammar errors. This means that they must be trained with much more precision. A popular example of such a chatbot is Microsoft Tay (Liu 2017).

2.2.3 Retrieval-Based Chatbot

A retrieval-based model is a hybrid of rule-based and generative chatbots (). A user's query and context are matched to an intent which is predefined by the chatbot. The chatbot can then choose how to handle the intent from a set of responses. If a user asks a chatbot "what's the weather today?", the chatbot would classify the input as a weather intent and then respond.

This kind of model is more predictable and less complex than a generative chatbot. They are also more flexible to user input then rule-based chatbots. For this reason, most consumer chatbots tend to be retrieval-based.

2.3 Predicting Personality

Personality plays a large role in determining the success of a relationship (Donnellan et al. 2004, Robins et al. 2000). Specifically, having a similar personality to your partner can predict happiness within the relationship (Laubu et al. 2016, Wang et al. 2018).

2.3.1 Big Five Personality Model

The most popular and well researched universal model for personality within psychology is the Big Five personality traits. The model uses five independent dimensions for defining the personality of a human.

These are openness, conscientiousness, extroversion, agreeableness and neuroticism.

The Big Five traits can be described as follows (Donnellan et al. n.d.):

- Openness (as opposed to closedness to experience): curious, imaginative, artistic, wide interests, excitable, unconventional
- Conscientiousness (as opposed to lack of direction): efficient, organised, not careless, thorough, not lazy, not impulsive
- Extroversion (as opposed to introversion): sociable, forceful, energetic, adventurous, enthusiastic, outgoing
- Agreeableness (as opposed to antagonism): forgiving, not demanding, warm, not stubborn, not showoff, sympathetic
- Neuroticism (as opposed to emotional stability): tense, irritable, not contented, shy, moody, not self-confident

The Big Five model is widely used within personality classification and will be the model used to classify users for the dating application.

2.3.2 Essays Dataset

Most studies into personality prediction use the Essays dataset (Pennebaker & King 1999). This dataset consists of 2468 essays written by psychology students. The students were asked to write anything for 20 minutes and complete a Big Five Inventory questionnaire. The questionnaire required students to rank themselves on a five-point scale based on a list of descriptions. The final dataset consists of essays paired with each of the Big Five personality traits. Each of the traits is either labelled "y" to indicate the student has a particular trait or "n" to indicate not having the trait. Language dimensions were shown to correlate with the five personality traits, indicating that personality traits could be inferred from text.

2.3.3 Linguistic Inquiry and Word Count

Linguistic Inquiry and Word Count (LIWC), is a commonly used program for text analysis. The program counts the number of words that are present that fit linguistic markers (Conglomerates 2015). For example, the word 'cried' fits into the categories of sadness, negative emotion, overall affect, verb and past focus. Given the number of words in each category, percentages can be found to measure how prominent a piece of text fits a category.

Using LIWC, correlations can be found between these linguistic markers and personality traits. LIWC has been used to build models for the Essays dataset (Mairesse et al. 2007, Yarkoni 2010, Tighe et al. 2016) using techniques such as support vector machines, naive Bayes, decision trees and linear regression.

2.3.4 Word Embeddings

Word embeddings are a popular way of representing words in a document as a list of multidimensional vectors (Mikolov et al. 2013). A vector for a given word can capture the semantic meaning of a word. Words with similar semantic meanings occupy the same spacial positions. For example, the vectors for the words 'car' and 'bus' may have similar vectors but 'car' and 'elephant' would have very different vectors. Similar to LIWC, capturing the semantic meaning of a sentence using word embeddings extracts linguistic markers. The vectors can be used for further analysis using a machine learning model.

The most recent advancement for generating word embeddings is BERT (Bidirectional Encoder Representations from Transformer) (Devlin et al. 2019). RoBERTa, a variant of BERT, was also created with better performance by expanding the size of the neural network (Liu et al. 2019). BERT and RoBERTa's workings are explained further on in the background section.

2.3.5 Neural Networks

Neural networks are machine learning models heavily inspired by biological neurons in the brain. They are often useful for their ability to learn non-linear patterns in data. They take input and after calculations, within their hidden layers, return an output. For example, for personality classification, the input could be a list of word embeddings and the output a probability of a personality trait being present.

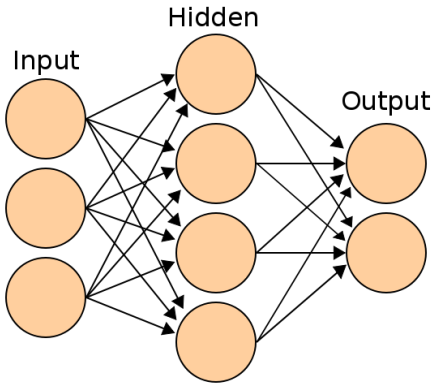


Figure 1: Feedforward neural network

A neural network comprises of nodes arranged in layers (Figure 1). Each node takes in a set of numerical inputs and had an equally sized set of weights. The dot product of the input and weights is added to a bias. Finally, the number is applied to an activation function. The activation function allows the neural network to learn non-linear functions from the input data. A simple example is a binary step activation function, where a value greater than a threshold should output one otherwise zero. More complex activation functions, such as sigmoid (Figure 2), are useful for binary classification and convert any number to a real value from 0 to 1.

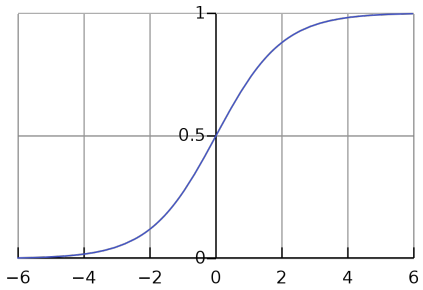


Figure 2: Sigmoid activation function

The weights and biases of the neural network are learned during the training phase of a neural network to give the desired output. When a neural network is initialised, the weights and biases are usually randomly generated.

To train a neural network, a dataset is needed usually containing data and labels. For a text classification problem, the data will contain a list of texts and labels for each piece of text such as personality trait classifications. Within the area of NLP, it's common for text to be converted to word embeddings. The generated word embeddings for each record in the dataset are inputted into the neural network one by one and the output is checked against the true value from the dataset.

A loss function is used to measure the difference in the neural network's output and the true value from the dataset. The loss function returns a value, the lower the loss value the better the model performed.

Given the loss, the model's weights must be tweaked to improve its performance. A mechanism known as backpropagation uses the chain rule to find the derivative of the loss function with respect to a node's weights. The derivatives are gradients of the weights. Using the gradients, an optimiser's goal is to adjust the weights using a technique known as gradient descent where the weights change to reduce the gradient to the global minimum. This is where the loss is at its lowest possible value with respect to the weight.

A learning rate is set for the optimizer and is used to specify the rate at which weights should be updated. A small learning rate will cause training to be too long but a high learning rate may cause the model to overshoot the global minimum. The learning rate of the optimizer must be tweaked for optimal training.

Calculating the loss, the gradients and then optimising the model is usually slow when done one by one for each record in the dataset. To improve training speed, a batch of records from the dataset is usually trained at one time. The loss is totalled for each record and the optimizer updates the model after the batch.

Neural networks are usually trained several epochs, where an epoch is one iteration of the dataset being fed through the model. The dataset is commonly split into two sets, a training and test split. During training, the model learns using the training set but never sees the test set. The test set is used to evaluate the model on unseen data. During testing the model's weights are not updated so no training occurs. The model may perform well on the training set and poorly on the test data, a problem known as overfitting. Splitting the data into two sets helps to understand if the model is generalised enough for any input or is only specifically preferment on the training set.

2.3.6 Convolutional Neural Networks

Convolutional neural networks (CNNs) are popular within computer vision (Krizhevsky et al. 2012) however they have benefits within NLP too. CNNs are neural networks that have at least one convolutional layer. This layer is one where nodes near each other in the last layer are connected to one node in the convolutional layer. For NLP, this is the equivalent of words near each other being fed into one shared node. This technique has the benefit of only taking important features from the set of words and disregarding unnecessary information. Additionally, CNNs commonly have pooling layers which have several nodes as input and, most often, return the maximum or average value. The pooling layers reduce the complexity of the model and similar to the convolutional layer, preventing overfitting since it generalises features.

CNNs have been well researched for personality prediction. Personality traits have been classified on the Essays dataset (Jiang 2018, Majumder et al. 2017, Mohammad & Kiritchenko 2014), outperforming previous results with LIWC.

2.3.7 Long Short-Term Memory Neural Networks

A recurrent neural network (RNN) is a type of neural network that contains loops. This is where the output of a node is used as input for a node earlier on in the network (Figure 3). RNNs are beneficial for sequenced data, such as sentences, where each item in the sequence has a better understanding when provided the items before it.

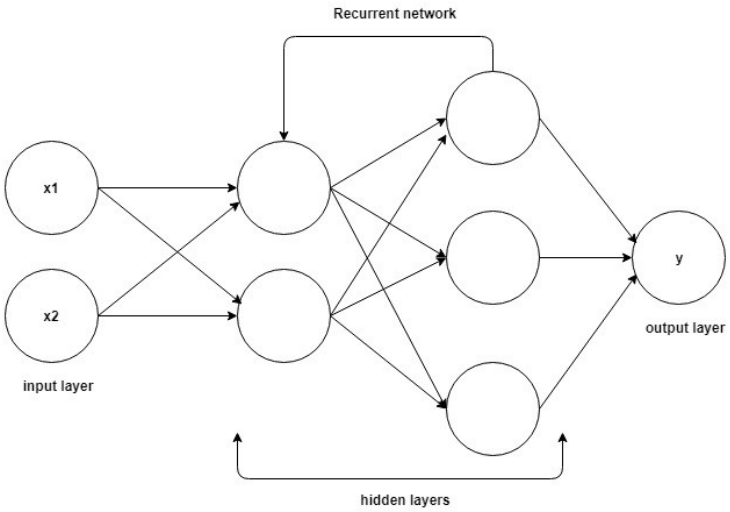


Figure 3: Recurrent neural network

RNNs, in theory, can use past context in a sequence to help understand its current item. In practice, when important features are further back in the sequence (long-term dependencies), the RNN is unable to learn (Bengio et al. 1994). Long Short-Term Memory neural networks (LSTMs) are specially designed to learn long-term dependencies more effectively (Hochreiter & Schmidhuber 1997). Within an LSTM layer, a cell state is shared between nodes that can store context. LSTMs are arranged like a chain in a layer where the starting cell state and the output from the first node in the chain are passed down to the last node. This is equivalent to reading a sentence from start to finish, whilst maintaining the context of the previous words.

Each LSTM node has three essential gates. The first is a forget gate, which decides whether to remove the cell state if the context is not important. The second, an input gate, decides if the cell state should be modified and updates it if required. The final gate is the output gate which uses the input and the updated cell state to generate an output. The output and cell state are passed to the next node. This is repeated until all nodes in the layer have calculated their output. The last node outputs a context vector that can be used for classification later in the network.

A standard LSTM builds the context vector by reading a sentence from start to finish. A bidirectional LSTM (BLSTM) builds a second context vector by reading the sentence in reverse, to improve semantic understanding. To do so, an extra sub-layer in the LSTM is added where the LSTM starts at the end of the input. The final context vector outputted by the BLSTM is the concatenation of the forward and reverse context vector. This technique is beneficial in situations where the context of a word is derived after its position in the sentence. For example, the word “bank” in “money bank” receives its meaning from the previous word but for “bank of the river” it receives its meaning after its place in the sentence.

LSTMs still struggle with long-term dependencies although to a lesser extent than RNNs (Bahdanau et al.

2015). This is because storing the whole output of the LSTM layer into a context vector leads to important information being lost. This is especially problematic in longer sentences. One solution, originally proposed for text translation, is to use an attention mechanism (Bahdanau et al. 2015). Rather than each node in an LSTM layer passing it's output to an adjacent node, it also passes its output to an attention layer. Hence, the attention layer receives the output of all the nodes in the LSTM layer rather than just the context vector. The attention layer decides what information is considered important by calculating a probability for each LSTM node's output using a softmax activation function. The probability is multiplied by the LSTM layer's output to decrease the values of unimportant inputs.

In comparison to CNNs, LSTM networks have not been as prominent in personality prediction. A BLSTM performed marginally better with certain traits on the essays dataset (Jiang 2018) however adding an attention mechanism to the BLSTM improved results on all traits, outperforming CNN models with and without attention.

2.3.8 BERT and RoBERTa

BERT (Bidirectional Encoder Representations from Transformers) is a neural network by Google, based on the Transformer architecture (Devlin et al. 2019). BERT has become very popular due to its ability to be applied to many NLP tasks with state of the art results, particularly text classification tasks. BERT is made up of 12 encoders that each have a self-attention layer. As opposed to an LSTM that reads input from start to finish, BERT reads the entire input sequence at once and learns the context of a word from its surrounding words using the self-attention layer.

Two tasks were used to train BERT. These were Masked Language Modeling (MLM) and Next Sentence Prediction. During MLM, 15% of the words in a sequence were replaced with the token [MASK] or a random word. This model was required to predict the replaced words. Next Sentence Prediction trained BERT by providing two sentences and asked if the second sentence would be the next sentence to the first sentence.

Similar to BERT, RoBERTa is another neural network but has more parameters so can produce larger word embeddings with more contextual information for each word (Liu et al. 2019). RoBERTa generally performs better than BERT on most tasks, however, due to its larger size, requires more computational power and memory.

BERT and RoBERTa are language models and can be used for two reasons. The first is to generate word embeddings to use as input for another model. The second is for fine-tuning the model to learn a text classification task. This involves adding an untrained layer to the end of the model and training the whole model for some epochs.

The most recent advancement within personality classification on the Essays dataset takes a pre-trained RoBERTa model and fine-tunes the output by adding an untrained layer of neurons to the end of the network (Jiang et al. 2019). By training the model on the Essays dataset, RoBERTa was able to perform better on all personality traits besides conscientiousness.

3 Software Engineering Methodology

The software engineering portion was not the main focus of the project. The system was built as a means of collecting data for personality prediction and presenting the matching algorithm. As a result, decisions made during the process of building the service were designed to speed up development to allow more time to work on the personality prediction model.

3.1 Requirements

To understand what the system should provide for the user, a list of functional and non-functional requirements were made.

3.1.1 Functional Requirements

- When the app first launches, the user should be able to create an account using their Google account. Once an account is created, they must be able to sign in again afterwards
- If an account is new, the user should go through an onboarding process, where the functionality of the app is explained. The user should also be able to insert details for their dating profile such as their first name, their gender and a profile picture
- The user should be able to interact with a chatbot and see their chat history
- A list of matches should be displayed including their profile picture and their name
- The user should have the option to send messages to their matches
- The user should be able to see their profile including their name, profile picture and their inferred personality traits
- The user should be able to sign out of the app
- A notification should be sent when a match is generated

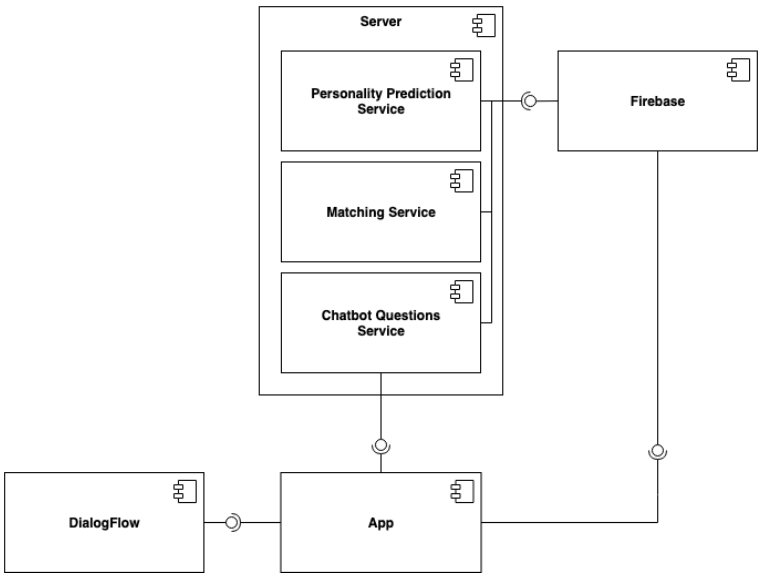


Figure 4: Component diagram for the dating service.

3.1.2 Non-Functional Requirements

- When using the app, the user should not experience any stutters or slowness.
- The app should save data offline and notify the user if they are not online if they try to carry out an action that requires connectivity.
- All network requests should take a reasonable amount of time.
- The app should be secure and only access resources using HTTPS.
- The user should be able to open the app and start talking to the chatbot instantly.
- The app should never crash when in use unless it is because of uncontrollable factors such as an unavoidable fault with the operating system.

3.2 Overview

The main components of the system include a mobile app for front-end, a server for scheduling background tasks and a database for storing user data (Figure 4).

3.3 Mobile App

Compared to other solutions, a mobile app was decided as the best option for delivering the service. If questions were to be asked throughout the day, a user is more likely to answer through an app as opposed

to a website or other medium. Furthermore, being able to message matches at any time is important.

To match the service provided by other dating apps, the system had to support both iOS and Android devices. Building two separate apps for both platforms is a task in itself and not feasible given the time frame for this project. A common solution to support both platforms is to use a web app that wraps a website into a native app. Though a valid solution, the final solution can be slow and not provide the same native-feeling user experience an individual would expect from a mobile app. The best solution was to use React Native, a JavaScript framework for building cross-platform native apps. React Native allows the developer to interface with the native APIs of iOS and Android, using JavaScript, to build a native app. React Native allowed for fast development whilst maintaining the quality and performance expected by a native app. An overview of the components of the app is available in Appendix A and a preview of the app in Appendix B.

3.4 Chatbot

The majority of the user's attention in the app is with the chatbot, named Aida, who's goal is to learn about the user and provide matches. As the main focus of this research is on personality prediction, Aida is predominantly rule-based chatbot. When the user first signs in to the app, Aida introduces herself and follows a predefined flow to collect user details to create their profile. This includes the user's name, age, gender and profile picture. After, Aida explains how the app works and what the user can expect.

Once onboarding has been completed, questions are sent to the user periodically throughout the day within a random range of 30 minutes to 5 hours. These questions come from a curated list of common first date questions (World n.d.). They are presented to users as practice questions for when they eventually match with other people, for example, Aida could ask "What would be your dream job?" or "Tell me about your hobbies". More examples can be found in Appendix C.

To improve the user experience when talking to Aida, Dialogflow, an API service for creating retrieval-based chatbots handles small talk. This means Aida can respond to messages such as "How are you?", "You're so sweet" or "I'm going to bed". Dialogflow matches these messages to an intent and returns an answer from a list of predefined responses.

3.5 Back-End Server

The time spent building the back-end server was minimized by using a mobile development platform, Firebase, to store user data. Firebase stores user data within a NoSQL database and also stores photos. The mobile app interacts with Firebase using an API.

A server setup using Flask, a Python web framework, handles tasks that Firebase cannot manage. The Python server takes care of the following tasks:

1. Scheduling questions to send to users
2. Scheduling personality prediction based on the user's conversations with the chatbot
3. Scheduling the matching algorithm to pair users together

Scheduled jobs are saved to a Redis database. Redis is a key-value database that stores data within memory rather than saving data to disk, so saving and reading data is fast and scalable compared to a traditional database system. Moreover, in the case of a system outage where memory is lost, jobs can be rescheduled again without an issue.

The match finding algorithm runs per user every 8 to 24 hours. Personality prediction is carried out every 24 hours for each user. This allows for predictions to always be based on the latest data received from the user and to result in higher quality matches. These values are arbitrary and should be adjusted in a production-level application, with respect to user behaviour.

Personality is only calculated if the user has answered enough questions with Aida. The minimum number of messages required is set to 10. The average number of words in the Essays dataset is 654. If a user answers 10 questions with 65 words, it matches to one essay, enough data to generate personality accurately.

An admin web system to search users, matches and test all the scheduling utilities is available too. A preview of the website is available in Appendix D. This was built with React, a web framework. It was useful for debugging database calls and testing functionality.

4 Data Science Methodology

4.1 Personality Prediction

4.1.1 Dataset Split

The dataset will be split with 10-fold stratified cross-validation (Kohavi 1995). Cross-validation randomly splits the dataset into 10 folds and trains the model on 9 of the folds. The last fold is used for testing. This is usually done 10 times on all possible combinations of folds and the average results are used. For neural networks, training takes a lot of time so only one combination of folds is used, greatly speeding up training.

Cross-validation does not guarantee that each fold equally represents all the classes in the dataset due to its random nature. For personality prediction, this means having an even split of 'y' and 'n's for a personality trait. To prevent this problem stratified cross-validation ensures each fold has an even proportion of classifications.

Using cross-validation over a traditional train-test split allows the model to learn the whole dataset. This is especially useful for smaller datasets. Furthermore, it migrates the effect of overfitting, a common issue with neural networks (Zaremba et al. 2015). A constant seed value is used for randomisation of the dataset, so when training and testing the model the same splits are generated.

4.1.2 Model

Five models were produced for each of the Big Five personality traits. A BLSTM with an attention mechanism was the model chosen for the task (Figure 5). Its ability to derive context from both directions in a sentence makes it a good option. The attention mechanism ensures the model can learn long-term dependencies and select the most important words for classifying personality traits (Jiang 2018). Previous models use word2vec (Majumder et al. 2017) and fastText (Jiang 2018) to generate word embeddings. No research has been done on using context-sensitive word embeddings which provide more information. As a result, BERT and RoBERTa word embeddings will both be tested with the model.

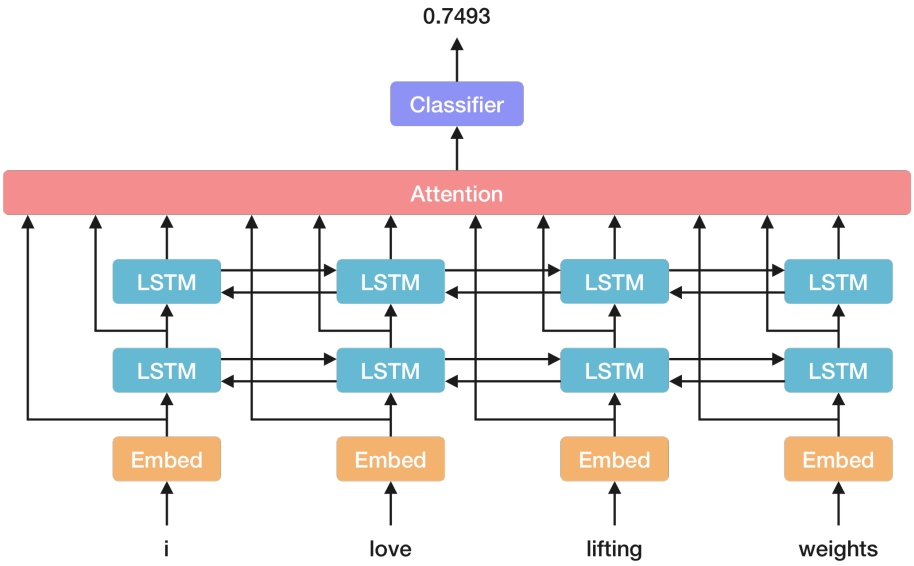


Figure 5: Bidirectional LSTM with an attention layer.

The first layer of the neural network converts each word of a sentence to word embeddings, using either BERT or RoBERTa. The LSTM layer captures the semantics of the sentence from start to end and vice

versa. A second LSTM increases model complexity to prevent overfitting the training data. The embeddings and outputs of both LSTM layers are all sent to the attention layer. The attention layer selects the most important information for classification. The final layer is only made up of one node, that takes the output from the attention layer and returns a probability from 0 to 1 using a sigmoid activation function. If the probability is greater than or equal to 0.5, the trait is present. Otherwise, the trait is not present.

As predicting personality traits is a binary classification problem, binary cross-entropy was the chosen loss function. This loss function quantifies the difference between two probability distributions, in this case, the predicted values from the model and the true values from the dataset.

The optimizer used was Adaptive Moment Estimation (Adam) (Kingma & Ba 2014). Adam uses individual adaptive learning rates for each parameter in the network. Using individual learning rates works well in situations with sparse gradients. This is when a neural network does not receive enough information to tune its weights. Sparse gradients are common within NLP, hence why Adam was the optimizer of choice for the model.

LSTMs are prone to overfitting (Zaremba et al. 2015). This refers to the model performing well on the training set but poorly on the validation set. To prevent overfitting, the model should learn generalised features that occur in both the training and validation set, not just the training set. To do so, several techniques can be used such as dropout and weight decay. Dropout refers to randomly ignoring nodes in a neural network to prevent prevents co-adaptation. This is where one node may be trained to correct for the mistakes of another node. Co-adaptations do not generalise on unseen data and lead to overfitting, however, dropout accounts for this problem. Within the model, dropout is added to the first LSTM layer. A dropout of 0.2 means 20% of the nodes within the first LSTM layer are ignored. Generally, large values for weights indicates overfitting. Weight decay penalises nodes from having high weights. It does this by subtracting a proportion of the weight when updating the weight of a node. As a result, weights do not become highly positive or negative values.

To speed up training, early stopping, a technique to stop training when no learning occurs was used. If the validation loss does not decrease over a set number of epochs, training is stopped. Further training is unlikely to find a better solution.

4.2 Matching Algorithm

Personality similarity was chosen as the deciding factor for matching users. This is because similarity can predict happiness in a relationship (Laubu et al. 2016, Wang et al. 2018). It is calculated by the number of Big Five personality traits two users had in common, measured on a scale of 0 to 1. The higher the value, the more similarities between a pair of users. A step by step process of finding a match for an individual is

as follows:

1. Ensure the individual has a minimum of 10 questions answered and their personality results have been generated
2. Find potential matches within a 10km range that the individual has not yet matched
3. Calculate personality similarity of the individual with all nearby users
4. Sort nearby users in order of similarity
5. Select the most similar user as the match

5 Experiment

5.1 Binary Classification on Essays

5.1.1 Preprocessing

The Essays dataset, containing essays tagged with the authors Big Five personality traits, was fetched and parsed from a CSV (comma-separated values) file. One essay was removed from the dataset that only contained the text "Err:508", leaving a remaining total of 2467 essays. No further filtering was done on the essays as they all contained at least one emotionally charged word (Majumder et al. 2017). If any neutral sentences were in an essay, the attention mechanism should filter them during training.

In an essay, personality traits were labelled as 'y' for a trait being present and 'n' if not. These values were replaced with 1.0 for a trait being present and 0.0 otherwise. Converting the tags to numbers allowed the model's output to be compared to the dataset to calculate the loss.

Essays were converted to lowercase text to prevent the model from processing the same words capitalised differently as two different words. For example, one essay containing mostly capitalised words such as "HE WAS A REALLY COOL GUY" would be processed differently if it was lowercase. Maintaining consistency of capitalisation should improve accuracy.

The common practice for pre-processing datasets for neural network usually includes removing stop words. These are common words such as "and", "is" and "the". This is detrimental to the performance of models like LSTMs where stop words are required to understand the semantic meaning of a sentence (Saif et al. 2014). Hence, stop words were not removed from the essays.

Generating word embeddings with BERT and RoBERTa is a time-consuming process due to the complexity and size of their models. This worsens when training is repeated a number of epochs and embeddings must be regenerated. To improve training speed, word embeddings are generated once beforehand and used for every epoch. Word embeddings were truncated to 400 tokens before training so the embeddings could fit within memory. BERT word embeddings were tested first and RoBERTa word embeddings were tested second.

5.1.2 Classification

For classification, the attention-based BLSTM model was implemented and trained using either BERT or RoBERTa word embeddings. Five models were trained for each of the Big Five personality traits and validated using stratified 10-fold cross-validation. The hyperparameters of the model were manually fine-tuned. This involves setting initial hyperparameters, checking results and changing the hyperparameters to reduce the validation loss and improve accuracy. The hyperparameters included hidden dimension size, learning rate, weight decay and dropout before the first LSTM layer.

All models had a fixed seed so that the weights were not randomly set during model initialisation. This allowed for comparisons to be made when fine-tuning the hyperparameters. Without setting a fixed seed, hyperparameter changes become harder to interpret because results are varied when the initial weights are always different.

To understand how hyperparameters affect the model's performance, the training results were visualised using a parallel coordinate plot, as seen in Appendix E. Using this graph, it was clear that the model was overfitting on the training data. Tuning dropout and weight decay significantly improved the performance of the model on the validation data (Figure 6).

The best hyperparameters to achieve the best model performance were the same for all models, as shown in Table 1.

Dropout	Weight Decay	Learning Rate	Hidden Dimension Size
0.2	0.001	0.00001	192

Table 1: The hyperparameters that produced the best results.

Once the best hyperparameters were found. The fixed seed was removed so model weights were randomly set. Each model was trained 10 times to find the best result and to understand the standard deviation. The results of the Essays dataset can be seen in Table 2. The state of the art results for agreeableness, extroversion, openness and neuroticism refers to a fine-tuned RoBERTa model (Jiang et al. 2019). The state

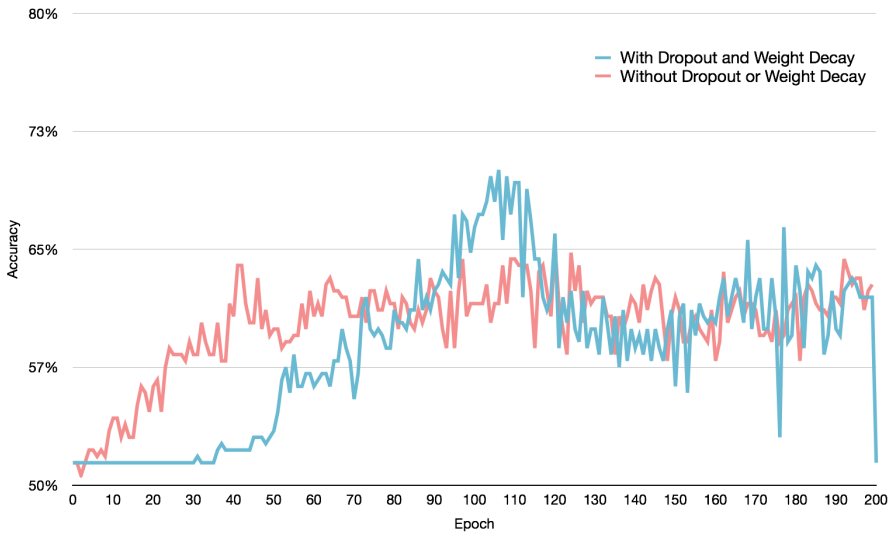


Figure 6: Graph of training extroversion over 200 epochs

of the art for conscientiousness refers to an attention-based CNN (Jiang 2018). The attention-based BLSTM with RoBERTa word embeddings outperformed all the previous state of the art scores when RoBERTa embeddings were used.

	Agreeableness	Conscientiousness	Extroversion	Openness	Neuroticism	Average
BERT	55.87	57.89	66.80	67.21	62.75	62.10
RoBERTa	59.51	61.94	70.04	68.83	63.16	64.70
Current Best	59.72	60.13	60.62	65.86	61.07	61.48
Improvement	-0.21	1.81	9.42	2.97	2.09	3.22

Table 2: Accuracy of the models on the Essays dataset.

6 Evaluation

Two methods were used to evaluate the success of the system. The first was user acceptance testing. The app was given to users to see if the initial requirements were met and if they found any unforeseen benefits or problems. The second was to test the accuracy of the personality model on unseen data.

6.1 User Acceptance Testing

As the goal of this research was to build a dating service, user acceptance testing was an important metric for evaluation. A copy of the app was shown to participants along with a questionnaire for them to fill in.

A full copy of the questionnaire can be found in Appendix F. The questionnaire guides the user through the app. This includes signing in, creating a dating profile, answering the chatbot's common dating questions, generating a match and messaging their matches. Participants were given actions to move through the app and were asked if they were able to successfully complete the action. These actions were designed to test the requirements of the system. To allow for the discussion to not be limited to a strict questionnaire, discussions were held with participants about their thoughts and added to the end of the form.

Five participants were involved with user acceptance testing, three of which interacted with the app in person. The other participants used the app remotely and gave their responses over video call due to the events of the COVID-19 outbreak.

Participants had no major problems with many of the features of the app as they all worked as expected. No crashes were experienced however two participants experienced delays in messages from the chatbot. This problem was due to a drop in internet connectivity, however, there was no response from the app, with regards to the issue. This confused one participant who believed the app was broken and that they did not do an action correctly.

The remaining feedback from the app was beyond the software and focused on the user experience. Several participants enjoyed the process of answering dating questions asked by the chatbot because it felt personal and more thought-provoking than regular questions asked by dating apps. The simplicity of the user interface was also popular, proving it to be intuitive and fun to use.

Besides the positive feedback, some criticism was given. One participant said they had difficulty talking to a chatbot and would only give responses of only a few words. Others suggested that the chatbot did not explain the functionality of the app very well. Lastly, one participant said Aida had too much control over what matches were provided. They would have preferred to manually select matches and use their personality, to help filter some of the users.

6.2 Personality Model Evaluation

The performance of the personality model is vital when determining the success of the app. Although the accuracy of the model on the dataset indicates some level of success, real-world performance is more important. To test the model, a second questionnaire collected data from participants. The questionnaire was split into two sections. The first section asked participants to fill out a Big Five personality test (Truity 2018). The test requires participants to rate statements according to how well it describes themselves. The results of the test were recorded in the questionnaire. The second section asked participants to answer some example questions from Aida's question bank. A website was created so answers could be submitted anonymously to the back-end server, for personality prediction. The data was not saved, to ensure the

integrity of answers. The Essays dataset has an average of 654 words per essay. Given this data, the website had 8 questions with a word count prompting participants to write 80 words for each answer. The total recommended words added up to 640 words, similar to one essay. The participants recorded their personality results from the website in the questionnaire.

11 participants were involved with the personality questionnaire. The results are presented in Table 3.

	Agreeableness	Conscientiousness	Extroversion	Openness	Neuroticism	Average
Questionnaire	72.73	45.45	45.45	90.91	72.73	65.45
Essays	59.51	61.94	70.04	68.83	63.16	64.70

Table 3: Accuracy of the models on unseen data from the questionnaire in comparison to the Essays dataset.

7 Discussion

7.1 Personality Prediction

The results show that using an attention-based BLSTM is a well-suited model for the task of personality prediction. It outperformed the state of the art fine-tuned RoBERTa model on extroversion, openness and neuroticism and the state of the art attention-based CNN on conscientiousness. Agreeableness was the only trait where the model performed worse.

Manually tuning hyperparameters may have introduced human error and a hyperparameter optimization algorithm such as a grid search may have selected hyperparameters that led to better model performance, however, this was difficult to do within the scope of the project due to a lack of resources.

Besides the model's accuracy on the Essays dataset, the results of the model from the questionnaire showed that the model is able to identify personality traits from unseen data. The average accuracy on the Essays dataset and questionnaire answers were almost identical, with a difference less than 1%. The results of this questionnaire demonstrate that the model is generalised and will perform well given enough emotionally charged text, such as the questions asked by the chatbot. Although the model performs very well compared to other personality prediction models, it's accuracy is still low and in a production dating application, may hinder the quality of matches.

7.2 User Feedback

The feedback from the user acceptance testing shows that the use of a chatbot has benefits and drawbacks. Using a chatbot to find matches was an enjoyable and personal experience that other dating services do not provide. The simplicity of the app was also popular and is mostly due to the chatbot leading most of the user's journey through the app. One participant's difficulty with talking to the chatbot would be problematic for the personality prediction model. Without enough quality data, the user's inferred personality would be inaccurate and the matching algorithm would be less successful.

Although the user interface was very popular, the issue where the internet connection dropped while the participants were talking to the chatbot was overlooked. If the user is expecting a reply from the chatbot and there is no internet connectivity, the chatbot waits until there is connectivity to respond without notifying the user to wait. A better solution would be for the chatbot to say they cannot connect to the internet and to try again later. The poor explanation of the app by the chatbot is another problem that can be fixed by rewriting the onboarding flow to be clearer.

8 Conclusion

Personality has been shown to work as a deciding factor for a dating application, however, the full extent of the system's success is not fully possible to understand until it is put in real-world practice. The attention-based BLSTM with RoBERTa word embeddings has proven to be a good performer for automatic personality prediction. It achieved higher scores on four out of five personality traits than the state of the art on the Essays dataset (Jiang et al. 2019). The model had an average improvement of 3.22%. Performance on real world data was just as accurate, with a difference of less than 1%.

The use of a chatbot to collect data from users has shown some promise. The chatbot gives the opportunity for users to have a more engaging experience for finding matches, however, limited conversational skills can deteriorate the user experience. For a user to be able to feel comfortable talking to a chatbot, a more complex solution should be implemented.

8.1 Future Work

There are many improvements that could be made to the dating service produced in this research. Firstly, a neural network trained on audio or video data would greatly improve personality prediction (Yang & Glaser 2017). The chatbot could be reshaped to allow for this kind of input. Secondly, the chatbot's capabilities could be greatly improved by using a generative chatbot model trained on a large dataset so conversations

could be held with the user. This would likely improve the quality and quantity of data collected from the user for personality analysis. Thirdly, the compatibility of two people is more complex than the similarity of their personalities (Jarrett 2018). More thought into other factors such as user interests and activity should be considered when matching users to improve the quality of the algorithm.

References

Aldous, D. (2017), 'Elo ratings and the sports model: A neglected topic in applied probability?', *Statistical Science* **32**, 616–629.

Bahdanau, D., Cho, K. & Bengio, Y. (2015), 'Neural machine translation by jointly learning to align and translate'.

Bengio, Y., Simard, P. & Frasconi, P. (1994), 'Learning long-term dependencies with gradient descent is difficult', *IEEE Transactions on Neural Networks* **5**, 157–166.

Brozovsky, L. & Petricek, V. (2007), 'Recommender system for online dating service', *arXiv:cs/0703042* .

Chaney, A. J. B., Stewart, B. M. & Engelhardt, B. E. (2018), 'How algorithmic confounding in recommendation systems increases homogeneity and decreases utility', *Proceedings of the 12th ACM Conference on Recommender Systems - RecSys '18* .

Conglomerates, P. (2015), 'Liwc 2015: How it works — liwc'.

Das Graças, M., Marietto, B., Varago De Aguiar, R., De Oliveira Barbosa, G., Tanaka Botelho, W., Pimentel, E., Robson, França, S. & Lúcia Da Silva, V. (2013), 'Artificial intelligence markup language: A brief tutorial'.

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2019), 'Bert: Pre-training of deep bidirectional transformers for language understanding'.

Donnellan, M. B., Conger, R. D. & Bryant, C. M. (2004), 'The big five and enduring marriages', *Journal of Research in Personality* **38**, 481–504.

Donnellan, M. B., Conger, R. D. & Bryant, C. M. (n.d.), 'Big five inventory (bfi)'.

eharmony (2015), '10 online dating statistics you should know (u.s.)'.

He, X., Liao, L., Zhang, H., Nie, L., Hu, X. & Chua, T.-S. (2017), 'Neural collaborative filtering', *Proceedings of the 26th International Conference on World Wide Web - WWW '17* .

Hochreiter, S. & Schmidhuber, J. (1997), 'Long short-term memory', *Neural Computation* **9**, 1–42.

Iyengar, S. S. & Lepper, M. R. (2000), 'When choice is demotivating: Can one desire too much of a good thing?', *Journal of Personality and Social Psychology* **79**, 995–1006.

Jarrett, C. (2018), 'Is it better to be like your partner?'.

Jiang, H. (2018), 'Automatic personality prediction with attention-based neural networks'.

Jiang, H., Zhang, X. & Choi, J. (2019), 'Automatic text-based personality recognition on monologues and multiparty dialogues using attentive networks and contextual embeddings'.

Jwala, K., Sirisha, G. & Padma Raju, G. (2019), 'Developing a chatbot using machine learning', *International Journal of Recent Technology and Engineering (IJRTE)* **8**.

Kessler, S. (2011), 'The love equation: How match.com calculates your ideal mate'.

Kingma, D. P. & Ba, J. (2014), 'Adam: A method for stochastic optimization'.

Knapton, S. (2017), 'Secret of eharmony algorithm is revealed....', *The Telegraph* .

Kohavi, R. (1995), 'A study of cross-validation and bootstrap for accuracy estimation and model selection'.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), 'Imagenet classification with deep convolutional neural networks'.

Krzywicki, A., Wobcke, W., Kim, Y., Cai, X., Bain, M., Mahidadia, A. & Compton, P. (2015), 'Collaborative filtering for people-to-people recommendation in online dating: Data analysis and user trial', *International Journal of Human-Computer Studies* **76**, 50–66.

Laubu, C., Dechaume-Moncharmont, F.-X., Motreuil, S. & Schweitzer, C. (2016), 'Mismatched partners that achieve postpairing behavioral similarity improve their reproductive success', *Science Advances* **2**, e1501013.

Liu, Y. (2017), 'The accountability of ai — case study: Microsoft's tay experiment'.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V. & Allen, P. (2019), 'Roberta: A robustly optimized bert pretraining approach'.

Mairesse, F., Walker, M. A., Mehl, M. R. & Moore, R. K. (2007), 'Using linguistic cues for the automatic recognition of personality in conversation and text', *Journal of Artificial Intelligence Research* **30**, 457–500.

Majumder, N., Poria, S., Gelbukh, A. & Cambria, E. (2017), 'Deep learning-based document modeling for personality detection from text', *IEEE Intelligent Systems* **32**, 74–79.

Match (2018), 'Match launches uk's first ai dating chatbot, lara, on the google assistant - uk dating - match'.

Mikolov, T., Chen, K., Corrado, G. & Dean, J. (2013), 'Efficient estimation of word representations in vector space'.

Mohammad, S. M. & Kiritchenko, S. (2014), 'Using hashtags to capture fine emotion categories from tweets', *Computational Intelligence* **31**, 301–326.

Mustafa, N., Ibrahim, A. O., Ahmed, A. & Abdullah, A. (2017), 'Collaborative filtering: Techniques and applications'.

Pennebaker, J. W. & King, L. A. (1999), 'Linguistic styles: Language use as an individual difference.', *Journal of Personality and Social Psychology* **77**, 1296–1312.

Robins, R., Caspi, A. & Moffitt, T. (2000), 'Two personalities, one relationship: Both partners' personality traits shape the quality of their relationship', *Journal of personality and social psychology* **79**, 251–9.

Rocha, P. & Fakultet, D. (2018), 'Gamification of love: a case study of tinder in oslo'.

Saif, H., Fernandez, M., He, Y. & Alani, H. (2014), 'On stopwords, filtering and data sparsity for sentiment analysis of twitter'.

Schafer, B. (2007), 'Collaborative filtering recommender systems'.

Shi, G.-Y., Kong, Y.-X., Chen, B.-L., Yuan, G.-H. & Wu, R.-J. (2018), 'Instability in stable marriage problem: Matching unequally numbered men and women'.

Statista (2020), 'Online dating - worldwide — statista market forecast'.

Tiffany, K. (2019), 'The tinder algorithm, explained'.

Tighe, E., Ureta, J., Pollo, B. A., Cheng, C. & Bulos, R. (2016), 'Personality trait classification of essays with the application of feature reduction'.

Tinder (2019), 'Powering tinder® — the method behind our matching'.

Truity (2018), 'The big five personality test'.

Wang, S., Kim, K. & Boerner, K. (2018), 'Personality similarity and marital quality among couples in later life', *Personal Relationships* **25**, 565–580.

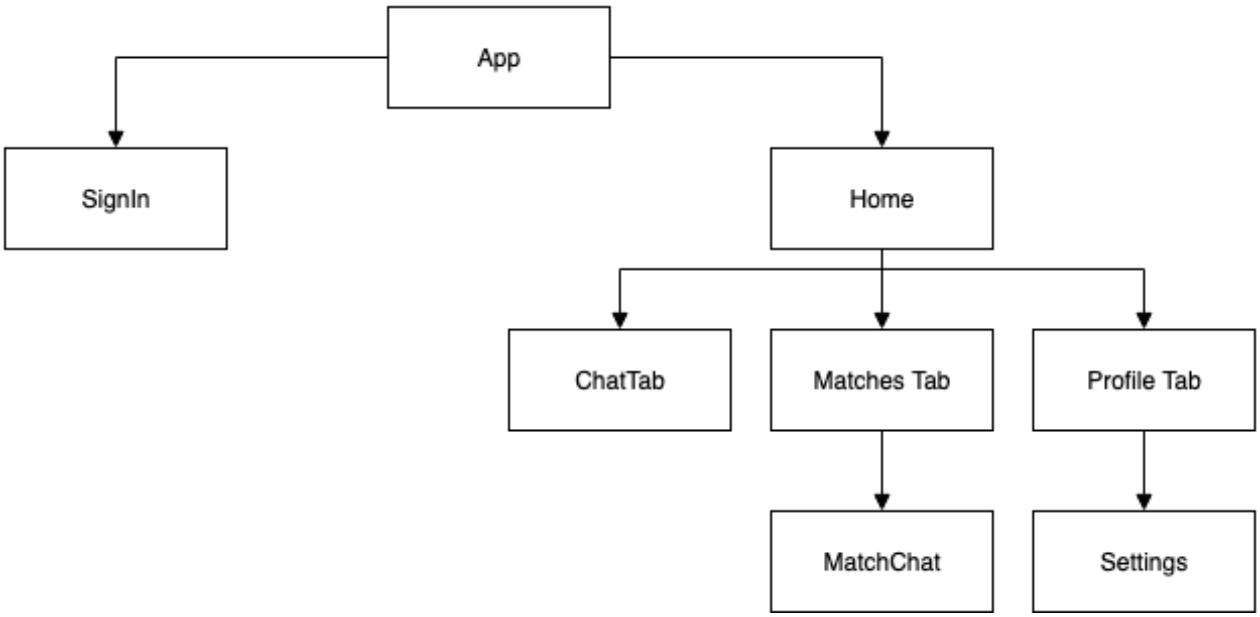
World, C. S. (n.d.), '160 first date questions - the only list you'll need.'.

Yang, K. & Glaser, N. (2017), 'Prediction of personality first impressions with deep bimodal lstm'.

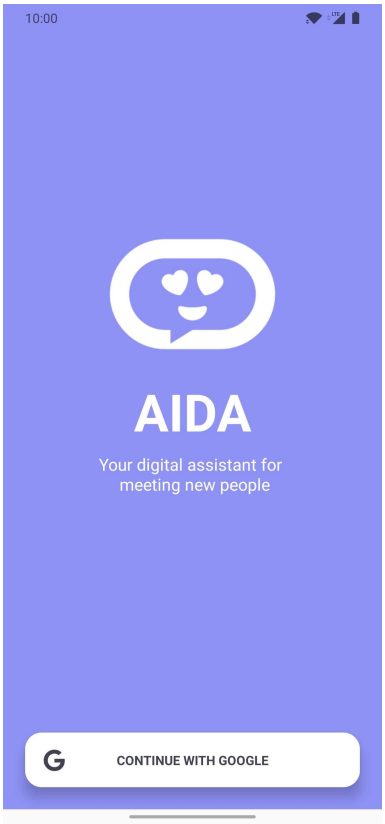
Yarkoni, T. (2010), 'Personality in 100,000 words: A large-scale analysis of personality and word use among bloggers', *Journal of Research in Personality* **44**, 363–373.

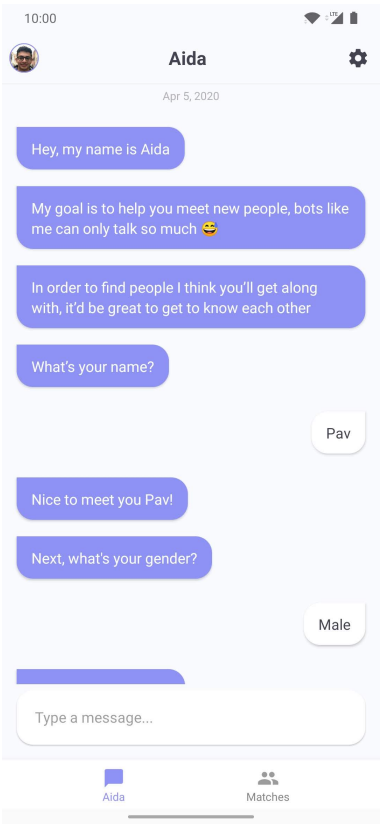
Zaremba, W., Sutskever, I., Vinyals, O. & Brain, G. (2015), 'Recurrent neural network regularization'.

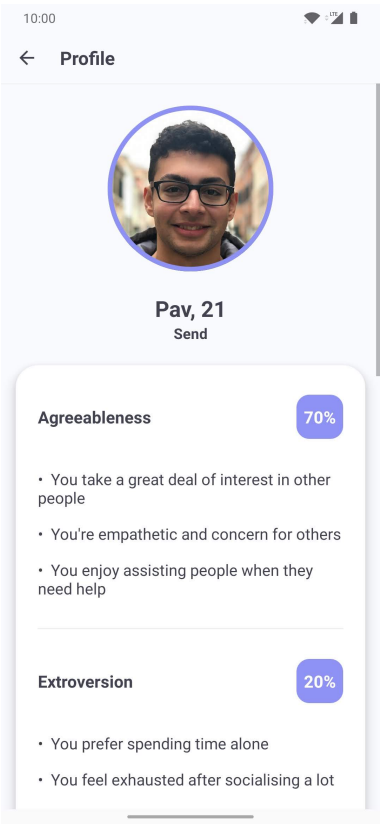
Appendix A App Component Diagram

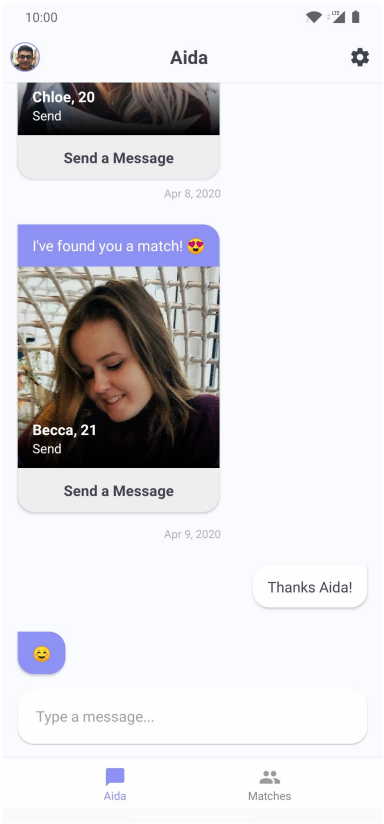


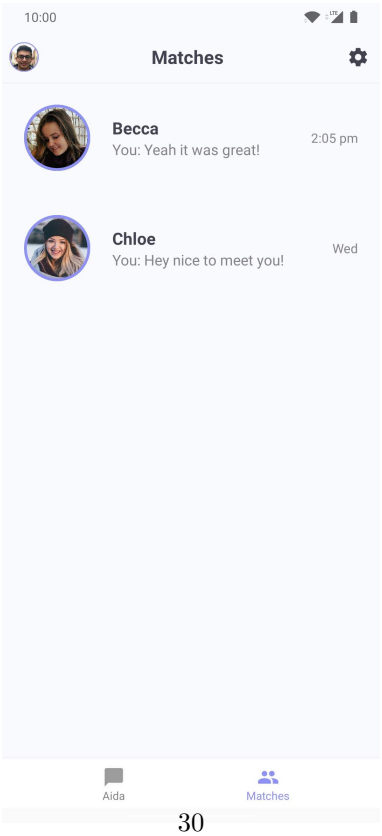
Appendix B Mobile App User Interface

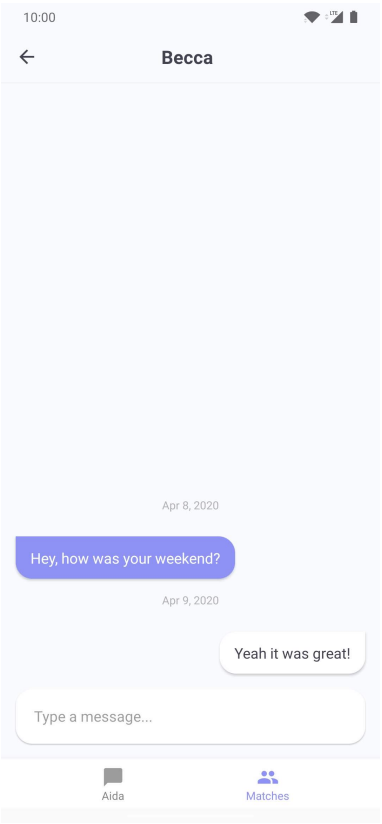








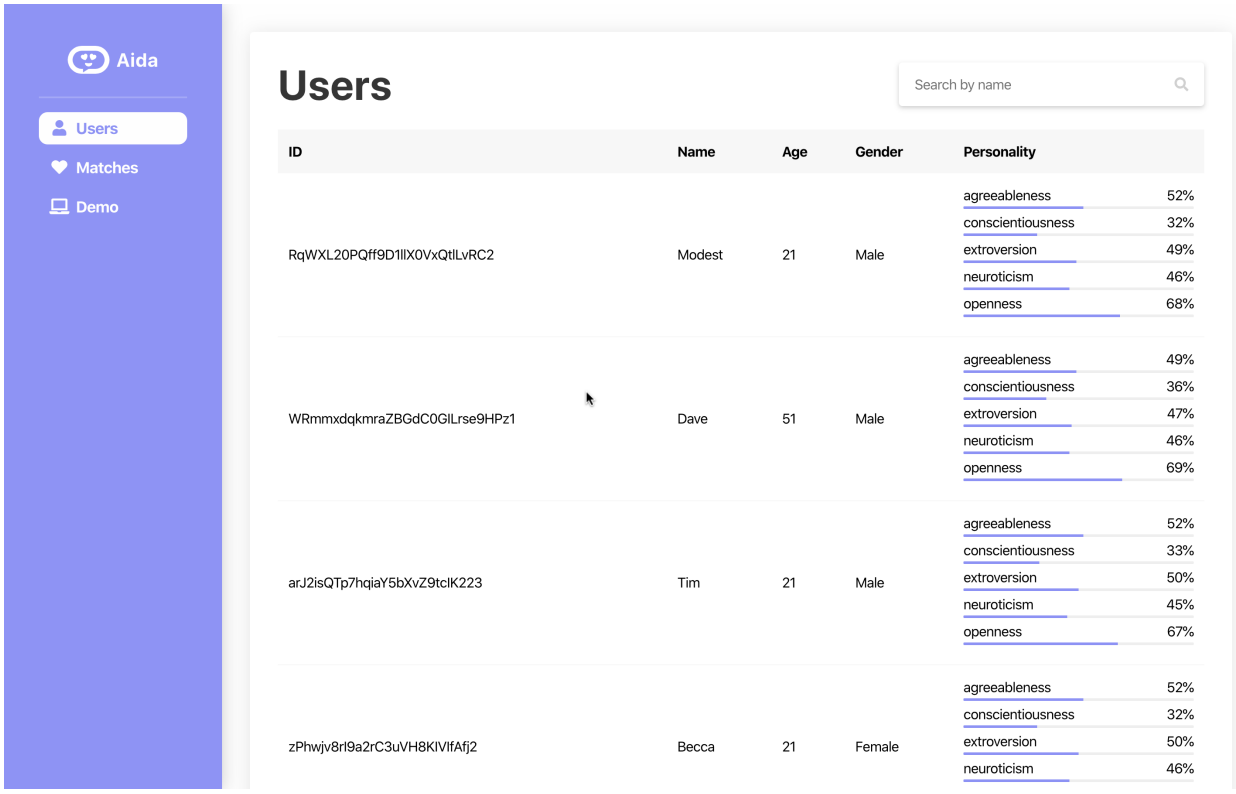




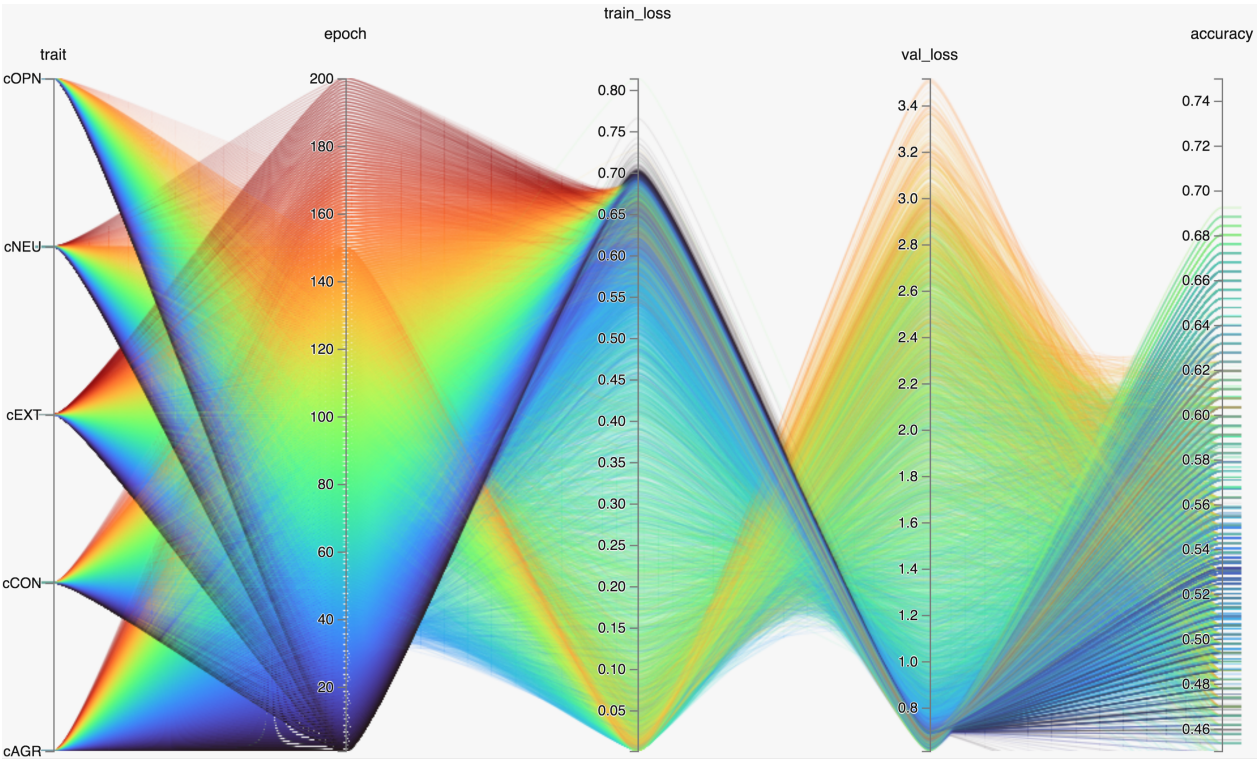
Appendix C Example Dating Questions

1. What's your most favourite place on Earth?
2. What small things brighten up your day when they happen?
3. What were some of the turning points in your life?
4. What habit do you wish you could start?
5. What's the worst or best job you've had?
6. If you unexpectedly won £10,000, what would you spend it on?
7. What's the hardest you've worked for something?
8. If you had the power to change one law, what law would you change?
9. What's something you're interested in that most people wouldn't expect?
10. Where did you take family vacations to when you were younger?

Appendix D Admin Back-End Website



Appendix E Essays Training Graph



Appendix F User Evaluation Questionnaire

Aida User Evaluation

Aida is a dating app where personality is used to matching people. Aida learns about your personality by chatting with you. The purpose of this questionnaire is to understand if you can learn and navigate your way around the app.

* Required

1. What's your gender? *

   Mark only one oval.

   - ◯ Female
   - ◯ Male
   - ◯ Prefer not to say
   - ◯ Other: _____

2. How old are you? *

   _____

Using the App

3. Please try sign into the app, did you manage to sign in successfully? *

   Mark only one oval.

   - ◯ Yes
   - ◯ No

4. Were you able to setup your profile with Aida? Your profile includes your name, age, gender and a photo. *

   Mark only one oval.

   - ◯ Yes
   - ◯ No

5. After setting up your account with Aida, did she clearly describe how the app worked? Please explain your answer if you can. *

6. Did Aida ask you questions about yourself? *

Mark only one oval.

- Yes
- No

7. Did you match with another user? *

Mark only one oval.

- Yes
- No

8. If you matched with another user, were you able to message them? *

Mark only one oval.

- Yes
- No

9. Did you experience any crashes, stutters, or delays? If so, please explain what happened. *

10. Was there anything you didn't understand regarding the app? *

11. What did you like about the app?

12. Anything else you would like to add? *