LoveBERT
Leveraging Language Models to Improve Matchmaking at Breeze

Ricardo Jongerius | October 2022

Delft University of Technology

TU Delft

LoveBERT

Leveraging Language Models to Improve Matchmaking at Breeze

by

Ricardo Jongerius

to obtain the degree of Master of Science
at the *Delft University of Technology*,
to be defended publicly on Wednesday October 12, 2022 at 15:30.

Student number:	4379500	
Project duration:	November 15, 2022 – October 12, 2022	
Thesis committee:	Prof. dr. ir. G.J.P.M. Houben,	TU Delft, chair
	Dr. J. Yang,	TU Delft, supervisor
	Dr. ir. S.E. Verwer,	TU Delft
	D. Alkemade MSc,	Breeze, supervisor

Abstract

Online dating has become the most popular method of finding potential romantic partners. At the core of these platforms, there is a reciprocal recommender system which recommends users to other users on the platform. Breeze is an example of such a dating app, serving its users potential romantic partners every day in the hopes of sending them on a date. Current approaches to matchmaking depend exclusively on interaction data and static features such as age, height and location. There is more information present on the user profiles, however, in the form of bios and answers to open questions. Given the state-of-the-art performance of pre-trained language models in a multitude of general language understanding tasks, these free-text profile sections present a source of untapped potential.

Our work combines the potential of these free-text profile sections and the state-of-the-art performance of recent language models to create a new approach for matchmaking, or reciprocal recommender systems in general. In this work, the goal is to find out how textual data from user profiles can be leveraged to make good suggestions in a reciprocal recommender system, and how language models can be used for this. Furthermore, this work also explores whether the cold-start problem can be alleviated using this approach. We also perform user research to find out how Breeze users make their decisions on suggestions served by the platform. We introduce LoveBERT, a model to serve text-based recommendations, harnessing the power of several language models fine-tuned on different free-text user profile sections.

Our results show that while LoveBERT is better at predicting unidirectional likes than traditional recommender system approaches, it does not outperform them when considering bidirectional matches. Furthermore, we show that while LoveBERT is not able to circumvent the cold-start problem, it is more robust, losing less performance than traditional techniques. Lastly, we show that especially the relation between different profile sections is an effective predictor for matches.

Preface

It all started out as a joke on vacation between Thomas Oomens, one of the founders of Breeze, and myself. "Why don't you just do your thesis at my company?" he said. I knew I wanted to pursue a graduation project in the NLP domain, so I didn't see it happening at first. What textual data does a dating app even have? Especially one where you can't even chat with the others!

But when we thought about it more, we realised that the user profiles do contain some text: the open questions. Not to mention the bios, which were on the roadmap to be added soon. Slowly, the joke actually began to become reality. After getting Daan involved, we settled on an interesting and novel idea and the title of this thesis: Leveraging Language Models to Improve Matchmaking at Breeze. It has felt amazing to work on a project with the potential to spark actual real world romance.

I didn't get to this point by myself, this project would not have been possible without my supervisor Jie. Thank you, Jie, for your guidance and expertise throughout the project. You always kept me focused on the scientific side of the project, which can be difficult when working in a fast-paced start-up.

I also thank Daan, who supervised me from Breeze's perspective. I never expected you to be part of my thesis committee back when we were in high school together, but here we are. Thanks for your time and trust, and for the freedom I got to always satisfy the academic requirements of the project. My appreciation extends to the rest of Breeze as well. The unforgettable workation in my third week was an amazing introduction to everyone, and it's been a pleasure to come to the office and work with you all ever since.

I'd also like to extend my thanks to the two other members of my committee, Geert-Jan Houben and Sicco Verwer, for taking the time to read my thesis and attend the defence.

Finally, I appreciate all my friends who spent time with me, and kept me sane while graduating. Thank you Oomens in particular, for making the joke that started it all.

Ricardo Jongerius
Rotterdam, October 2022

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

Abbreviation	Definition
BERT	Bidirectional Encoder Representations from Transformers
CB	Content-based
CBoW	Continuous Bag of Words
CF	Collaborative Filtering
GPT	Generative Pre-trained Transformer
LM	Language Model
MCC	Matthews Correlation Coefficient
NLP	Natural Language Processing
RS	Recommender System
RSS	Reciprocal Recommender System

Chapter 1

Introduction

Over the last 20 years, online dating sites and apps have quickly become one of the most popular platforms where people find and look for potential romantic partners [56]. The online domain is able to offer unprecedented levels of access to possible matches that are simply not attainable through traditional means. In 2013 in the United States, online platforms have surpassed meeting through friends as the most popular way heterosexual couples meet, with 39% of couples having met in an online setting in 2017 [57].

At the core of most online dating platforms lies a recommender system (RS). Traditionally, RSs have been built with the goal of providing a user with a set of recommended items that the system predicts that the user will like. With the enormous increase in content on the internet over the last decade, especially driven by the social web with its user-generated content, RSs have become ubiquitous. Therefore, the recommended items can be virtually anything: movies, books, songs, apps, websites, holiday destinations, people and e-learning material are all recommended by RSs on a daily basis [5]. It is important to note that RSs produce *personalised* recommendations. In other words, each user of the system receives a different list of recommendations based on their preferences or behaviour. The recommendations of a RS can be based on either explicit information, typically users' ratings, or implicit information like the past behaviour of users, such as previously watched videos or listened to songs. Users' demographic data like age, nationality and gender can also be leveraged to provide better suggestions [3, 59].

The RSs powering the matchmaking algorithms of all popular dating apps (e.g. Tinder, Badoo, Bumble) include certain subtleties which sets them apart from other RSs. Namely, in these: (i) the users are now the object being recommended themselves, and (ii) not only should suggestions be of interest to the user, but the suggested users should also have an interest in those they are suggested to. That is, successful recommendations only occur when both people like each other, or *reciprocate*. Pizzato et al. defined this family of RSs in 2010 as Reciprocal Recommender Systems (RRSs) [49].

For example, let us imagine two people, Alice and Bob, who are actively looking for a romantic partner and have signed up for a dating app. In a traditional user-to-user RS, say Twitter's recommendation list of people to follow, Bob might be suggested to Alice as an interesting profile to follow. In this scenario, it does not matter what Bob thinks of Alice, just whether Alice thinks Bob's Twitter profile is interesting enough to follow. In contrast, in the setting of our dating app (i.e. a RRS), if Bob is recommended to Alice and she likes his profile, this is not enough to be considered a successful match. An additional constraint is required, namely, Bob should also deem Alice's profile interesting enough to like her back. Both of their preferences should be taken into account, as it does not make sense to recommend profiles to Alice which are unlikely to like her back.

The extra constraint of reciprocity adds another layer of complexity to a RRS when compared to a traditional RS. Therefore, aggregation strategies to combine the level of reciprocity or mutual compatibility between two users based on unilateral preferences is an active area of research [24, 46, 72]. Other challenges especially prevalent in the RRS family include: the prevention of biased recommendations following (severe) popularity imbalance between different users [24], the lack of available datasets to stimulate more research as a result of privacy concerns, and data sparsity and the cold-start problem [20]. The cold-start problem is particularly troublesome for a RRSs such as an online dating platform. It concerns the issue where a RS cannot reliably draw inferences for a user when it has not yet learned enough information about what they like yet. Therefore, a matchmaking algorithm which can infer information based on the explicit information in user profiles is desirable, rather than one that requires usage data in the form of ratings. Some other fields besides online dating in which RRSs are relevant include recruitment, online learning environments, mentoring, skill-sharing and social media platforms on which reciprocity can produce better matching between people [36, 37, 44, 47].

Building and evaluating RRS models requires a sufficient amount of high-quality data (i.e. data of both user-system and user-user interactions, user-related information, etc.). The vast majority of evaluations of RRSs, specifically those in the online dating realm, rely on corporate data. For that reason, this thesis will be done in the context of the Breeze dating app[1]. Breeze sets itself apart from other dating apps by not allowing its users to chat at all. Once two people match, they immediately go on their first physical date. As a result, Breeze users cannot endlessly swipe through an infinite list of recommendations, as there is only limited availability for the number of dates which can be planned. Instead, they are only presented with a few profiles each day. Accordingly, Breeze's matchmaking algorithm is paramount, even more so compared to

[1]`https://breeze.social/`

other dating apps. Unfortunately, the limited number of suggestions also exacerbates the cold-start problem, as the system learns the user's preference at a much slower pace.

One source of untapped potential are the free-text portions of user profiles, which consist of personal bios, and open-ended questions which the users can answer. This section of the profile allows users to describe themselves and what they like in their own words, but also provide an indication of what they are looking for in a partner. Answers can range from a few words to several sentences, and represent a source of explicit information given by the user. Fiore et al. found that the free-text components of a user profile are important factors in determining its attractiveness [17, 18]. Tyson et al. showed that profiles on Tinder without bios got significantly fewer matches than profiles with one [64], which also indicates users utilise the information present in free-text portions of user profiles. It is not difficult to see that there can be invaluable information present in these, currently unused, free-text profile sections that say a great deal about the person's interests and traits. Additionally, using this information presents an opportunity to learn about the user before they have rated the first suggested profiles, potentially alleviating the cold-start problem.

Recently, the field of Natural Language Processing (NLP) has been revolutionised by self-supervised pre-trained language models. This approach first uses self-supervised pre-training of a neural model on a large varied corpus of unlabelled text. This allows the model to extract semantic information and relationships from general text data. Afterwards, these language models can be fine-tuned for specific NLP tasks or particular domains. The most famous language model is BERT [15], which achieved state-of-the-art results when it was initially released. The introduction of BERT spurred a wave of research on pre-trained language models. Its derivatives, such as RoBERTa [38], still perform extremely well in several general language understanding tasks[2]. None of these models have been applied to RRSs before in literature, though. This work aims to fill that gap.

In our research, we intend to extend existing approaches to RRSs with pre-trained language models using semantic information present in the free-text portions of user profiles. The proposed approach will be tested on real-world data. Altogether, the research questions we aim to answer in this work are as follows:

- **RQ1:** How can textual data from profiles be leveraged to suggest attractive user profiles in a reciprocal recommender system?
  - **RQ1.1**: Is there a difference in the effectiveness of the utilisation of the various kinds of free-text data (e.g. bios, open answers)?

[2]The General Language Understanding Evaluation leaderboard is full of pre-trained language models: `https://super.gluebenchmark.com/leaderboard`

- **RQ2:** How can language models be used in a reciprocal recommender system, and which technique performs best?
  - **RQ2.1**: Can a language model help circumvent the cold-start problem?

The main contribution of this work is twofold:

- User research to discover what Breeze users generally look for in suggested user profiles, what they deem important, and how they judge free-text profile sections. Subsequently, these findings were used to inspire the design of:
- LoveBERT, a novel, generalisable model build with fine-tuned language model components designed for reciprocal matchmaking systems.

The rest of this thesis is structured as follows. First, we discuss the background and related work in Chapter 2. Then in Chapter 3, we discuss the user research performed and introduce our novel approach to matchmaking at Breeze. In Chapter 4, we describe the setup of our experiments and analyse our results. Chapter 5 contains a discussion of the limitations of our work, and describes some avenues we explored that garnered no meaningful results. We also discuss the practical implications of this work. Finally, we conclude the thesis in Chapter 6 and provide some recommendations for future work.

Chapter 2

Background

In this chapter, we provide background information and an overview of relevant related works. First, we take a look at recommender systems and describe some of the most widely used approaches. Next, we zoom in on the reciprocal subset of these systems. Finally, we discuss literature on extracting semantic meaning from short pieces of text, and discuss related works.

2.1. Recommender Systems

In this section we provide an overview of the fundamentals of Recommender Systems (RSs). RSs rose in popularity primarily throughout the advent of web 2.0. With the massive amount of user-generated content available, users can easily be overwhelmed. RSs are the solution presented to the problem of such information overload [50]. The goal of a RS is to produce a list of recommendations to be presented to a user, where these recommendations are useful to the user for the accomplishment of a certain task [60]. The task can be anything, from the navigation to relevant web pages [35], browsing a catalogue of items to buy [9, 68], explore learning resources [14] or find people to socialise or collaborate with [31, 49, 55, 62]. In essence, the goal of a RS is to predict user-item preferences (i.e. how much a user likes some previously unseen item) and recommend the items which are probably liked most by the user.

Palomares et al. formally defined in [47] an item-to-user RS as follows:

Definition 2.1.1 (Item-to-user RS [47])**.** Given a user $x \in U$, a recommender $\mathcal{R}(x)$ is a system that recommends a list of items $R \subset I$ such that the (predicted) degree of preference $p_{x,i}$ by x towards every item $i \in R$ is stronger than the preference degree by x towards any item $i\prime \notin R$:

$$\mathcal{R}_I(x) = \left\{ i : p_{x,i} > p_{x,i'}, \forall i \in R, \forall i' \notin R \right\} \tag{2.1}$$

with R being the list of recommended items for x.

Items are not the only things which can be recommended, though. Especially since the rise of social media, there is a plethora of platforms which connect people in different ways. To accommodate user to user recommendation, we extend definition 2.1.1 to formally define unidirectional user-to-user RSs.

Definition 2.1.2 (User-to-user RS [47])**.** Given a user $x \in U$, an unidirectional user-to-user $\mathcal{R}_U(x)$ is a system that recommends a list of users $y \in R \subset U$ such that the (predicted) degree of preference $p_{x,y}$ by x for every $y \in R$ is stronger than the preference degree by x towards any other user $y\prime \notin R$, and also $y \neq x$ for every $y \in R$:

$$\mathcal{R}_U(x) = \left\{y : p_{x,y} > p_{x,y'}, \forall y \in R, \forall y' \notin R, y \neq x\right\} \tag{2.2}$$

with R being the list of recommended items for x.

In order to generate the list of recommendations, generally, the RS has to learn something about the user's behaviour. However, when a new user joins the system, the RS cannot know anything about their preferences yet. This is known as the cold-start problem [33] which will be expanded upon in Section 2.1.2.

The context of this work is the online dating app Breeze. In this app, users are presented with a list of recommendations of people who they might want to date every day. However, a date does not get planned immediately when such a recommendation is liked. It is important that both sides like each other. There must be mutual interest.

2.1.1. Reciprocal Recommender Systems

RSs in which users are suggested to other users, but a successful match only occurs when *both* parties of the recommendation like each other are called Reciprocal Recommender Systems (RRSs). This term was coined by Pizzato et al. in 2010[49]. Their work is widely seen as the foundational work specifically focusing on RRSs. An example of a scenario where a RRS would be applicable is an employment recommender system in which potential employees are looking for job positions, offered by employers in need of suitable employees. Both of these parties need to be satisfied before a recommendation can be seen as successful. Other examples include recruitment systems [21, 36, 37], skill-sharing platforms [44] and online learning environments. However, the vast majority of RRS research, as well as this work, addresses a specific domain in which RRSs are imperative: online dating platforms [23, 24, 30, 49, 55, 62, 72, 73].

There are several factors that differentiate a RRS from a traditional RS. An overview of the largest differences is provided in Table 2.1. To further illustrate these differences, we build upon our example of Alice and Bob from Chapter 1. As said before, reciprocity

Traditional Recommender System	**Reciprocal Recommender System**
Success is determined solely by the user receiving recommendations.	Success is determined by both sides of the recommendation.
Successful recommendations are likely to retain users for more recommendations.	Successful recommendations may cause users to leave the system.
Users build rich implicit preference profiles through usage history.	Users can leave the system after a short time, and often do not build a rich history of implicit information. Cold-start problem is therefore particularly critical.
Users tend not to provide detailed explicit user profiles.	Users expect to provide detailed user profiles.
Same recommendation can generally be made to many users.	Due to limited availability, one user cannot be recommended to a large number of other users.
It is allowed that some items are never recommended.	It is important all users are part of recommendations, or they might leave the system dissatisfied.
Small portion of recommendations can be of poor quality.	Poor recommendations should be avoided, because users may suffer if they are rejected repeatedly.
Users expect a large number of decent recommendations. Quantity is more important.	Users expect a few high-quality recommendations. Quality is more important.

Table 2.1: Major distinctions between traditional and reciprocal RSs.
Builds on the work of Pizzato et al. [49]

is a factor in RRSs. Therefore, success is determined by both sides of the recommendation which makes success more rare. However, this also means that a successful recommendation may cause users to leave the system. For example, consider a traditional RS like Netflix's movie recommendation. When Alice gets very good suggestions for content to watch, she will be impressed and keep coming back for more apt recommendations. On the other hand, in a RRS like Breeze, when Alice and Bob are perfectly matched they will go on a date, start a serious relationship, and live happily ever after. Alice and Bob will likely never return to the system. This is also the case for other RRSs such as job-seeking platforms. This is not an absolute truth, as some people will be in need of several jobs or seek multiple partners.

However, a consequence of this characteristic is that RRSs should consider that the accumulation of *implicit* information might end abruptly. Another reason Alice or Bob might leave the system is because they are dissatisfied with the system's performance.

The key takeaway is that it can be very difficult for the system to distinguish between these two cases. At any rate, the cold-start problem, which will be explained further in the next section, is especially acute for RRSs. Luckily, new users, like Alice and Bob, are much more inclined to provide a RRS with a lot of *explicit* information. They are expected to complete a detailed user profile containing information about themselves and their ideal partners. For example, by answering closed questions about attributes such as gender, age, sexual preference and location. Generally, Alice and Bob will also provide more complex data such as photos and free-text descriptions of themselves called 'bios'. These kinds of elaborate users profiles are rarely present in traditional RSs such as movie recommenders.

The next big difference concerns overwhelming people with options. In a movie RS, it is no problem for the same movie to be recommended to several dozen, or even thousands, of people. However, in a RRS such as Breeze, the same user, say Alice, should not be recommended to a large number of people. First of all, she can only select a handful of people to date in a reasonable time frame. Additionally, say Alice was recommended to Bob, Charles, Daniel, Edgar and Frans, who were all great matches. When that results in five matches in a short time span, Alice could easily become overwhelmed and stop responding to all of them. Conversely, if Alice is never recommended to anyone at all, she will definitely experience the platform negatively, as her efforts in finding a match would be fruitless. Yet, if a certain movie is never recommended, it would generally be fine. Therefore, balance is key in the number of recommendations in RRSs.

Another important distinction lies in the quality of the recommendations. In traditional RSs, a user can easily skip over a few bad recommendations and give no second thought to them. However, this is different in RRSs due to the aforementioned limit on the number of recommendations. Specifically, when Alice receives a set of poor recommendations which would never reciprocate, this can be very discouraging and upsetting. For this reason, quality is more important to RRSs when compared to traditional RSs.

Given all these characteristics, let us extend the definitions from section 2.1 and formally define a RRS.

Definition 2.1.3 (Reciprocal Recommender System)**.** Given two users $x, y \in U$, $x \neq y$, let x be the subject user who uses the system to receive recommendations, and let y be an object user who is susceptible to being recommended to x where $y \in R \subset U$. A reciprocal recommender, denoted by $\mathcal{RR}_U(x)$, is a system which combines two unidirectional user-to-user recommenders (see Definition 2.1.2) $\mathcal{R}_U(x)$ and $\mathcal{R}_U(y)$, with the goal of simultaneously satisfying the interests of both x and any $y \in \mathcal{RR}_U(x)$.

$$\begin{aligned}\mathcal{RR}_U(x) &= \{y : y \in \mathcal{R}_U(x) \text{ and } x : x \in \mathcal{R}_U(y)\} \\ &= \{y : p_{x\leftrightarrow y} > p_{x\leftrightarrow y'}, \forall y \in R, \forall y' \notin R, y \neq x\}\end{aligned} \tag{2.3}$$

where $p_{x\leftrightarrow y}$ is the measure of mutual interest or compatibility (reciprocity) between users x and y, obtained using some aggregation function ϕ, i.e. $p_{x\leftrightarrow y} = \phi(p_{x,y}, p_{y,x})$.

2.1.2. The Cold-start Problem

The cold start problem is a potential problem that affects RSs. It relates to the sparsity of information available to the recommendation algorithm, making inference difficult [33]. As mentioned before, this problem is especially serious in RRSs, because users will leave the system more easily [49]. Generally, the cold-start problem is divided in three different cases: new community, new item and new user [6]. The new user variant is particularly relevant in our case. When a new user joins a system such as Breeze, they have not liked any profiles yet. As a result, they cannot receive any personalised recommendations based on their past behaviour; there is simply no data available. In other words, there is no *implicit* information to go on. Even after the initial few ratings, the data is generally too sparse to make Collaborative Filtering approaches reliable, because as section 2.1.3 will explain, those rely on the presence of such implicit information. Therefore, initially, it would be better if recommendations were based on *explicit* information present in the user profiles. Such approaches tend to be Content-based. Section 2.1.3 will elaborate on these different approaches. Our approach also works like this, as it leverages static information from user profiles. We describe our approach in more detail in section 3.2.2.

2.1.3. Taxonomy of Recommender System Approaches

In most literature, (R)RSs are divided into three main categories: collaborative filtering, content-based filtering and hybrid approaches [5, 47, 50]. To provide a baseline understanding of each of these main families of algorithms within the scope of RSs, we provide an outline of their basic principles and core ideas.

Collaborative filtering

Collaborative Filtering (CF) approaches are the most influential and widely used recommendation technique [19, 42, 50]. CF approaches generate recommendations based on similarities between user behaviour. It is based on the basic assumption that people who had similar preferences in the past will also prefer similar things in the future [50]. CF approaches determine the relationships between different users, and the interdependencies between recommended objects, be it items or people [19]. At its core, this is achieved through the use of *implicit* preference information gathered by the system, i.e. active user ratings and other behavioural data. This can be represented as a matrix in which each cell represents the user rating of a particular item or other user. In general, CF approaches rely heavily on the availability of sufficient behavioural user data. Their performance depends on adequate rating information, as this is the information used to generate the recommendations. Therefore, it is particularly affected by the cold-start problem [6, 19]. CF is also sometimes aptly called a people-to-people correlation [50].

Resnick et al. [54] developed the first CF approach for a RS, called GroupLens. It was built to recommend articles to 'Netnews' clients. It produced recommendations using the rating server, which predicted scores based on the heuristic that people who rated similarly in the past will probably give the same rating again.

More recently, CF has also successfully been applied to RRSs. Cai et al. [8] were the first to capture the reciprocal role of user interactions within a social network. They formulate a neighbour-based CF approach enabling people-to-people recommendation, called SocialCollab. In their model, users can be similar to others in two ways: having similar 'taste' in other users of the system, or having similar 'attractiveness' from other users who liked them. More concretely, if two users x and y both like several users $z_1, z_2, ..., z_n$, then x and y have similar taste. Likewise, if two users x and y are liked by several other users $z_1, z_2, ..., z_n$, then x and y have similar attractiveness. Accordingly, y should be recommended to x when y likes people with similar attractiveness to x, and when people with similar taste to y like x.

Xia et al. [72] introduce similarity measures, similar to taste and attractiveness, that capture the unique characteristics of the online dating network. They propose a generalised RRS that aims to match people with mutual interest. The general procedure consists of two steps: first preference fusion, where the reciprocal preference $p_{x \leftrightarrow y}$ is calculated using the harmonic mean; then filter for and recommend top-k users with the highest reciprocal preference. This approach is relatively simple to implement and understand, but it bears a high computational and temporal cost on large datasets.

Never et al. [45] introduce a model-based CF method for RRSs that determines latent user attributes. The basic idea of this approach is that there are two preference matrices: female-to-male preferences and male-to-female preferences. Subsequently, a latent factor model is trained for each matrix through matrix factorisation. Finally, the preference $p_{x \leftrightarrow y}$ is estimated by taking the resulting dot product of the two latent vectors. It boasts similarly promising performance to the approach by Xia et al. [72], but with better efficiency, allowing for real-time recommendations even on large datasets.

Content-based filtering

At its core, Content-Based (CB) recommendation is based on item descriptions or user profiles containing characteristics. In traditional user-to-item systems, CB approaches generate recommendations based on previously liked, bought or watched items [19, 50]. User models are generated from previously picked items by characterising users according to the item attributes. Items are then recommended to the user based on the items that are similar to the items they have liked before [42]. In the reciprocal setting, this is done using similarity between users. Therefore, CB approaches largely rely on *explicit* information from user profiles. Examples include profile bios, tags, specific preferences like age and location ranges, or sexuality and gender. User profiles (or item descriptions) can be analysed in order to establish a similarity between the objects [5].

There are some clear advantages to CB approaches. Firstly, CB methods tend to be very explainable. They can provide a clear explanation as to why a certain item or user was recommended [16], as the logic behind their recommendations is based on specific attributes of the user (or item). Furthermore, they are less susceptible to the cold-start problem, as most of the inference is based on *explicit* information from user profiles [19]. They are also able to match users with peculiar or extremely specific interests. There are also downsides, however. CB approaches can run into scalability issues, as all users in the system have to be examined. Although in the reciprocal setting, this can be greatly alleviated by the explicit preferences regarding location and range. There can also be a lack of serendipity in the system [19], where users end up being recommended things from a filter bubble of extremely similar things. However, this is much more severe in an item-to-user system than in a RRS.

RECON, the first algorithm for a RRS, introduced by Pizzato et al. [49] used a CB approach. Their approach is based on the set of attributes of user profiles from users who messaged each other. RECON is built on two key ideas: (i) building user preference models, and (ii) calculating compatibility with unknown users. Concretely, users preferences towards attributes from other users (e.g. height, non-smoker, eye colour, etc.) are predicted based on that user's interaction history with others, e.g. messages exchanged with users who exhibited specific attributes [49]. Unidirectional preferences are subsequently aggregated into a reciprocal score $p_{x\leftrightarrow y}$ using the harmonic mean.

Liu et al. introduced an employer-graduate oriented RRS in [36] and [37]. One focused on the employer's point of view, while the other focused more on the graduate perspective. In these models, employers were modelled as a set of recently hired graduates, and vice versa. Both these models were based on profile-to-profile similarity, where the most similar profiles to positive samples were recommended next.

Hybrid approaches

The hybrid recommendation approach, as the name suggests, aims to combine several different RS methods in order to provide better recommendations than using a single technique. Hybrid RSs are generally used to either leverage the power of several data sources, or to increase the performance of existing RSs within a particular data modality [32]. The main goal of a hybrid approach is to mitigate the weakness of several paradigms (e.g. CF and CB), while capitalising on their complementary strength in order to build a more robust system [19, 42, 50]. This is achieved by combining key components from, say, CF and CB techniques into a single approach. There are various ways to design such a hybrid approach, often resulting in increased complexity of the system [19].

Akehurst et al. [1] introduced CCR, the first RRS that integrates CB and CF. It combines the distance metrics from both CB and CF in order to determine interaction groups (i.e. a list of potential good recommendations) for all users. The CB part is activated first, which takes into account the distance between users based on profile attributes. Inside this group of similar users, the CF component is applied that picks out similar users who like and are liked by similar people.

The work of Rodiguez et al. [55] combines a CB and knowledge-based approach in their BlindDate recommender. In order to identify potential matches between users, BlindDate utilises DBPedia[1] repositories to obtain information that is used to enrich an ontology model. In this system, a similarity matrix is built from a multi-graph conceptual model. The knowledge integration contributes to a higher precision than non-hybrid baselines, though the specific and complex nature of the model limits its generalisability.

Hong et al. [21] introduced a hybrid RRS for recruitment. In this system, users are grouped based on their activity level by a clustering algorithm. Next, the cluster the user belongs to dictates which of several different filtering approaches is applied. Users who have logged sufficient activity in the system can rely on a CF process, while more passive users were served recommendations based on a CB approach.

2.2. Attractiveness in Social Sciences

The vast majority of the previously described approaches all assume that the optimal candidates to suggest are those most similar to the querier. The idea seems to stem from more classical RSs, in which it makes sense to recommend similar items to those the user is actively engaging with. However, we can ask whether this belief holds up for the online dating domain. Is romantic mate selection in the real world based on similarity?

[1] `https://www.dbpedia.org/`

There has been some research surrounding this question in the social sciences.

Klohnen and Luo [25] were one of the first to show that self similarity, the similarity of one's partner to themselves, is indeed a predictor of attraction. This research is from before online dating really took off, showing that even before a constant stream of easily digestible user profiles, people typically tried to find similar mates.

More recently, Hudson and Fraley show that in dating in general, people are mostly looking for similarities [22]. They found that anxious individuals tend to be more satisfied with highly similar partners, while avoidant individuals appear to be more satisfied with moderate levels of similarity. But in all cases similarity between mates was desired.

Launay and Dunbar concluded the same, showing that having more shared traits leads to linearly increasing ratings of partner likeability [27]. Moreover, they narrowed it down to a few specific traits which were especially important, with musical taste, ethical and political views, ethnicity and religion topping that list.

On the other hand, there is also research that shows no evidence for the similarity principle [39]. Although, this research was based on the results of high volume speed-dating only, meaning it does not necessarily represent a real-life scenario.

2.3. Semantic Information Extraction from Text

The science of extracting semantic information from textual data has changed dramatically over the past decade. As the preferred name of this field gradually shifted to Natural Language Processing (NLP), the methodology used has changed significantly as well. The field has moved away from simpler probabilistic language models to self-supervised pre-trained language models as a basis for a wide range of general language understanding tasks aiming to extract different valuable insights from raw text.

One of the most famous language models is BERT [15], developed by Google researchers. BERTs model architecture is a multi-layer bidirectional Transformer encoder based on the original implementation described in Vaswani et al. [65]. BERT's release together with its impressive results kickstarted a new wave of research on pre-trained language models, paving the way for auto-regressive models like GPT-3 [7], transfer learning models like T5 [51], and models which fuse these methods like BART [29] and ERNIE 3.0 [61].

One thing that all these works show is that language models are very powerful. They are extremely effective at extracting semantic information from texts, oftentimes even beating human performance[2] Therefore, a system which has text data available that can

[2]For example, in several different tasks of the General Language Understanding Evaluation set: `https://super.gluebenchmark.com/leaderboard`

be analysed presents an opportunity to leverage said text data to improve the system's performance. The Breeze dating app is such a system, where users can write a short bio about themselves, and answer several open-ended questions freely. Furthermore, there are several other RSs which have successfully utilised NLP techniques in order to improve recommendations, which indicates there is ample opportunity to incorporate NLP techniques in the Breeze matchmaking process.

2.4. Related work on NLP techniques in Recommender Systems

There are several works which have attempted to incorporate different NLP techniques to recommendation algorithms. We provide a brief description of the most relevant ones, and explain how they relate to our work.

Musto et al. [43] extended a CB approach for an item-to-user RS by using word embeddings learned from Wikipedia articles. This approach links each item to its corresponding Wikipedia page, and builds an item embedding by taking the average of all word embeddings from the (pre-processed) Wikipedia page. Different techniques were used to retrieve word embeddings, with Word2Vec [12] performing best. Then, user profiles were created for each user by taking the average of all their previously liked items. The system's recommendation list was generated by taking the item embeddings closest to a user profile. The authors evaluated their system on two datasets: one containing movies and the other containing books. Their results were in line with state-of-the-art CF approaches, which usually achieved higher performance than CB approaches. A downside of this approach is that it still requires a lot of historical data to build reliable user profiles, and therefore does not solve the cold start problem. The naive averaging of embeddings also allows for noisy results.

Wang et al. [67] introduced a CB RS to suggest the ideal conference or journal for computer science publications. It works with the term-category principle, where categories can be distinguished by specific important terms. In this context, a conference or journal is a category, and important terms appear in abstracts from the publications. The system uses the chi-square statistic to measure dependence between the terms and each category, in order to determine the most distinctive and defining terms for each category. These terms are subsequently used to build feature vectors for each category. A recommendation for the closest category for a publication is then found based on the tf-idf measure of the terms in its abstract. This biggest disadvantage of this work is the need for predefined categories, which makes it difficult to generalise to different domains such as this thesis. Additionally, it relies on much longer texts than the generally short user profile bios. Therefore, this approach does not translate well to the online dating domain.

CoupleNet, introduced by Tay et al. [62], is a hierarchical recurrent model utilising multi-layered attentions at different hierarchical levels, which recommends two Twitter users to explore a romantic relationship. The recommendations are based on text data in large social media platforms like Twitter. Tweet embeddings are generated for all tweets from a user using word embeddings and attention layers. A coupled attention layer between users using their respective tweet embeddings is subsequently used to calculate user embeddings, which are then compared using cosine distance. Finally, nearby users are recommended as potential mates. While CoupleNet significantly outperformed all baselines in terms of precision, the positive labels used for training and tested were inferred from the type of language used in tweet (e.g. "I love you @user1234" is labelled as people in a relationship). Such data is difficult to confirm, and might not represent the ground truth. Although this architecture is very interesting, it requires a lot of data (i.e. tweets) per person to build a representative user profile. It is therefore quite difficult to implement on the comparatively limited user profiles in dating apps such as Breeze.

In [69], Wang et al. introduce a sentiment-enhanced RS for movie recommendation. Their system builds on a relatively straightforward CF approach, but incorporates review data in the pipeline. The core idea is that the recommendation rank of a movie for user x is influenced by positive or negative sentiment in movie reviews made by similar users to x. Such an approach is clearly geared towards item-to-user RSs, which means it is non-trivial to incorporate it in a dating app such as Breeze. One could incorporate the sentiment of the profile bios in the matching algorithm, although this would add little information due to the binary distinction (i.e. positive or negative). An interesting approach could be to introduce an (anonymous) review functionality after users have matched and been on a date, but this is out of scope for this thesis.

In the context of skill-sharing platforms, Neve et al. [44] presented a hybrid approach to reciprocal recommendation. It describes a RRS that facilitates the matching of users in a recipe sharing service called CookPad.[3] On CookPad, users can connect with each other, public recipes can be shared, and users can indicate content preference indicators. The basis of the algorithm uses the Jaccard Index[4] to calculate user similarity based on liked recipe overlap. However, the Jaccard Index only accounts for co-occurrences of exactly the same item, while there are many recipes on the platform which are very similar, but not identical. For example, Alice might like a potato omelette recipe, with Bob liking a Spanish potato omelette recipe. In order to also incorporate such recipes, the system uses word embeddings to detect similar (but non-

[3] `https://cookpad.com/`

[4] The Jaccard Index is defined as $J(A,B) = \frac{|A \cap B|}{|A \cup B|}$

identical) recipes. This approach works well with the inherent amount of overlapping words in similar recipes (especially because of lists of identical ingredients), but this performance does not necessarily transfer to the user profile domain due to their varied nature. We need a more sophisticated comparison method than comparing all word embeddings with each other, like a language model such as BERT [15].

One example of an item-based CF approach utilising BERT was introduced by Wang et al. [68]. Their work is set in the context of Ebay, where items to be purchased next are recommended to users. It works by using the next sentence prediction component of BERT, by feeding it the seed item as sentence A, and target item as sentence B. Positive samples are defined as items purchased within a single user session, and negative samples are randomly sampled. The BERT model then predicts a next item to offer the user based on an input. This approach is interesting, but difficult to apply to the online dating setting. There are no clearly defined user sessions in a dating app, so it would be difficult to extract positive training samples.

Quite recently, Malkiel et al. [40] introduced RecoBERT: a BERT-based approach for learning catalogue-specialised language models for text-based item recommendation. They train a BERT-like language model using title description pairs as the input, and create feature vectors based on the embedding sequence output of BERT. Through this procedure, the model can score similarities between pairs of items, without the need for item similarity labels. The final recommendations are generated by incorporating four cosine similarity scores (all combinations of seed title and description, and candidate title and description). RecoBERT boasts state-of-the-art performance, and can infer text-based item-to-item similarities more accurately than other techniques. One obstacle in translating this approach to the online dating setting is that RecoBERT requires two separate information points (title *and* description), as it also utilises the relationship between those two. Such information is not present in Breeze data.

However, we can still use the general idea of fine-tuning a language model (LM) using free-text data belonging to the objects in the catalogue. To that end, we introduce LoveBERT: the matchmaking algorithm leveraging LMs. LoveBERT is built up from several LMs, similar to RecoBERT. However, the LoveBERT components are all fine-tuned on specific profile sections of users. For this, we use historical data of suggestions made where likes are counted as positive samples, and the profile sections of both the querier and candidate profile are the input of the LM. LoveBERT is extensible by design, and works on any user-to-user RRSs with discernible free-text profile sections. We elaborate further on the design of LoveBERT in section 3.2.2.

Chapter 3

Building a Matchmaking Algorithm at Breeze

The goal of our study is to build and evaluate a new matchmaking algorithm for Breeze which exploits the text data present in user profiles. In essence, this translates to the NLP task of producing a label, i.e. whether two people are a match or not, based on some textual input, i.e. the user profile texts. As discussed in Chapter 2, Language Models (LMs) deliver state-of-the-art performance in such NLP tasks, and are therefore the most promising avenue to explore. For that reason, our algorithm should utilise a LM that takes a user's bio and answers to questions as an input, and produce a list of recommendations of other users consisting of potential romantic partners based on their 'match score'. In this section, we describe the design process of our proposed matchmaking approach. Additionally, we introduce LoveBERT: our novel LM designed specifically for RRSs in online dating. First of all, we describe the survey designed to better understand the behaviour and motivations of users. Then, we characterise the data and the different ways it is used. Lastly, we present the final design of LoveBERT.

3.1. User Research

Before we design our LM, it should first be understood how users make their decision. What do they deem important when rating profiles? How do they judge bios and the answers to open questions? What should be in them? To this end, we designed a survey which was sent to a portion of active users of Breeze.

3.1.1. Methodology

The survey design is primarily based on information from 'Introduction to Survey Quality' by Biemer and Lyberg [4] and the Pew Research Center [71], which has helped with the phrasing of questions, order and scope of answers, and how to approach participants. The survey was held online as a Google Form. The full survey, including all answers and results, has been published online[1].

Question design

The survey was divided into three sections: questions about user profiles in general, questions about bios specifically, and questions about open answers. The goal of the profile questions was to determine the importance of each user profile section, and whether they have to be similar to one's own profile. For this, a Likert-type scale [34] was used as this is the most widely used approach to scaling responses in scientific surveys. For the bio and open answer questions, the main goal was to establish what 'kind' of bio or answers people want to see on other profiles, as well as what they have on their own profiles. Additionally, people were asked whether they have a bio at all (as this is not a required profile section), including their reasoning for one or not. For questions where it made sense, answer options were randomly shuffled for each participant, so as to avoid bias for a certain option. There were several open questions to allow people to give specific reasoning for some of their motivations and desires.

Participants

The link to the survey was sent to a subset of active users of Breeze through the in-app help desk. The subset was created by randomly sampling users active in the last month, with the only constraint being that they had not already been actively contacted by Breeze recently for a different kind of input. Additionally, the link to the survey was included in one of the marketing mails sent to people who have chosen to receive those. In total, it was sent to roughly 800 people, which represented around 5% of the weekly active userbase at the time. Participants had to actively opt-in to participate in the survey by clicking the link and completing it. In the end, there were 46 respondents who completed the survey. There was no direct incentive offered apart from the opportunity to help improve Breeze's matchmaking.

The opt-in nature of the survey resulted in a set of respondents based on convenience sampling [4], a form of non-probability sampling usually inferior to e.g. random sampling. However, Vehovar et al. present numerous examples of successful and cost-effective implementations where it works in practice [66], especially when the survey does not concern a divisive issue, so we accept this outcome. Furthermore, figure 3.1 shows the age distributions of the survey participants, as well as the overall userbase age distribution. The respondents appear to accurately portray the overall userbase.

[1]`https://bit.ly/3Q9qkoi`

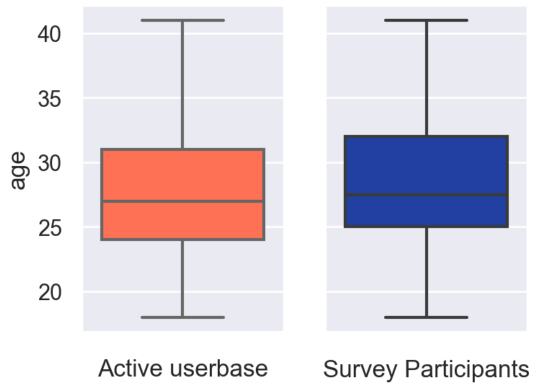


Figure 3.1: Age distribution of overall userbase and of the survey participants. Outliers are hidden for visual clarity.

3.1.2. Key Results

The survey results let to several key insights about user profile data. The first question asked, for each section of a user profile, how important it is in making the decision on whether or not to like the suggested profile.

As can be seen in figure 3.2, the pictures are overwhelmingly the most important aspect. This was to be expected, as looks are the first thing people look at on online dating apps [17, 18], and also the biggest part of the profiles visually. In fact, it is the only thing users see if they do not click on a profile to investigate it further. Second place goes to the interests tags, which also makes sense, as people tend to want their potential partner to have similar interests and hobbies, as we show in section 2.2. After that come the personal attributes, which are things such as age, living location and height. These attributes can be limited by the users in the app. While suggestions are never made outside the limits set by people, these factors can still play a significant role in deciding whether or not to date people. For example, if Bob's age range is set to 20-26, a profile of a 28 year old will never be suggested. However, Bob may still prefer someone in the 22-24 range over younger or older people. The free-text portions of the profiles achieve the lowest importance rating, with the bio being deemed slightly less important than the open answers. This can partly be explained by the fact that the bio is a relatively recent feature (since late December 2021), so not nearly all profiles have one.

Although people seem to not care about the bio that much, once we consider the data, it interestingly shows that having a bio significantly improves the match rate. Overall, the match rate of suggestions is 0.66%. When we consider suggestions in which one of the two parties has a bio, the match rate increases to 0.76%. Moreover, the match

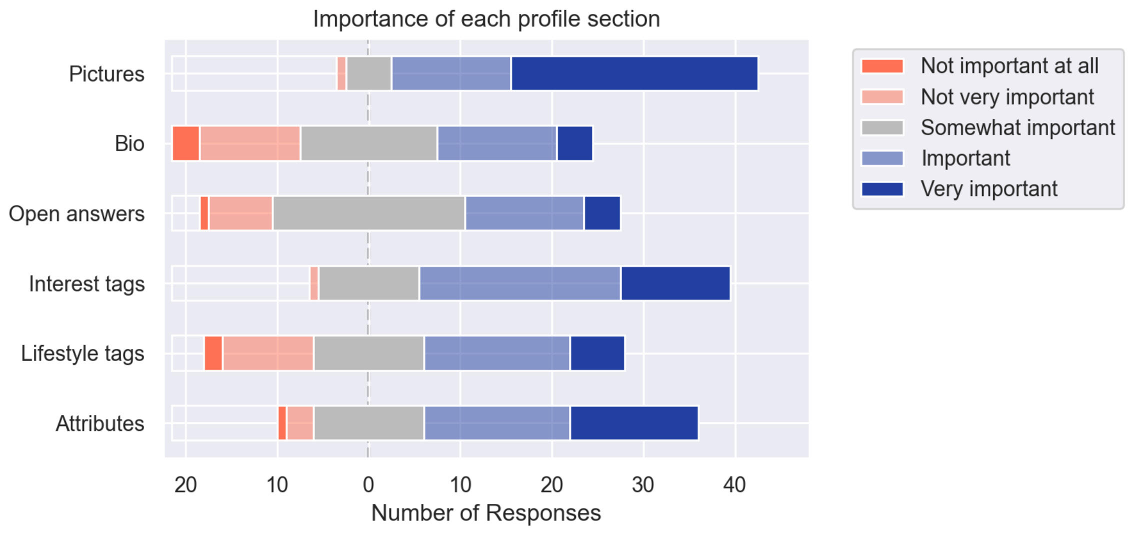


Figure 3.2: Likert scale of the importance of each different section of a user profile.

Suggestions	**# of suggestions**	**# of matches**	**match %**
All	5,314,152	35,110	0.660%
One user with bio	1,509,041	11,437	0.758%
Both users with bio	113,197	1,232	1.088%

Table 3.1: Overview of match percentages for suggestions with and without bios

rate of suggestions where both users have a bio is 1.09%, an increase of roughly 65%. An overview of this is given in table 3.1. Fiore et al. showed that the bio is a very important aspect of user profiles [18], and increases desirability [17], and our data confirms this, even though people may not actively realise it.

The second question asked whether people mainly look for similarities or differences in suggested user profiles. As mentioned in section 2.2, Hudson et al. show that in dating in general, people are mostly looking for similarities [22]. Launay and Dunbar concluded the same, but narrowed it down to a few specific traits which were especially important (i.e. musical taste, political/ethical views, hobbies and preferred jokes) [27]. Although, there is also research that shows no evidence for the similarity principle [39]. Our results appear to mimic that of the previous research. We found that people do indeed mainly look for similarities in suggested profiles, with 28 of the respondents (60.9%) indicating they look for similarities, and only 2 people (4.3%) actively looking for differences. The remaining 34.8% had no preference. Therefore, we conclude that if people are looking for anything, they overwhelmingly look for similarities.

Bio

The following few questions all pertained to the bio. Together with the Breeze team, a list of the different types of bio was devised, and users were asked which kind of bios they preferred. More than one type of bio could be selected, so people did not have to

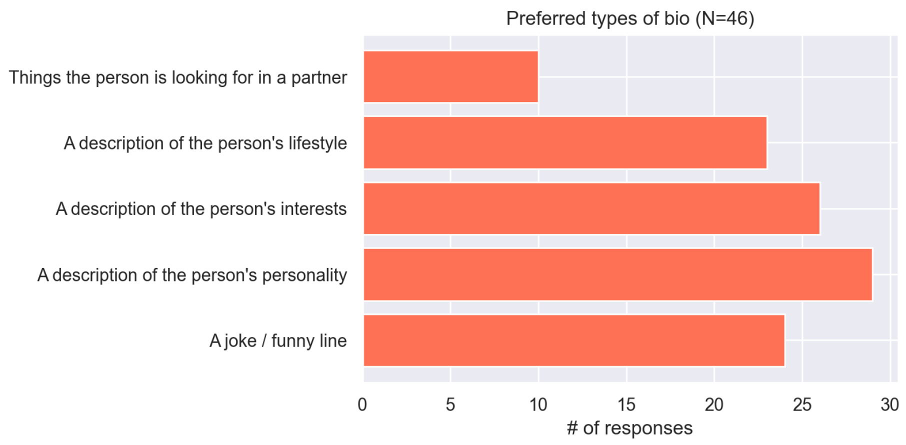


Figure 3.3: Bar chart of the different types of bio, showing how many people like each type.

choose between their preferences. Figure 3.3 provides an overview of the different types of bios. It does not show a clear 'winner' in terms of what bio is universally liked most, however there is a clear loser: a bio describing what the user is looking for in a partner. From this, we conclude that in general, a bio should describe the user. This is further confirmed by answers to an open question from the survey which asked, *"What are you looking for in the bio on suggested profiles?"*. Some answers include:

> "Something fun and that someone had put in a little effort to show what she likes/does/cares about etc."
> "Something that shows the person's humor and general 'vibe'."
> "Something that tells about themselves. Could be funny, serieus *[sic]*, personal or something they like doing."

The full list of responses can be found in the online publication of the results.[2]

Although there seems to be no dominantly preferred type of bio, preference is not equally distributed among users. There are several correlations between the prefer-

[2] `https://bit.ly/3Q9qkoi`

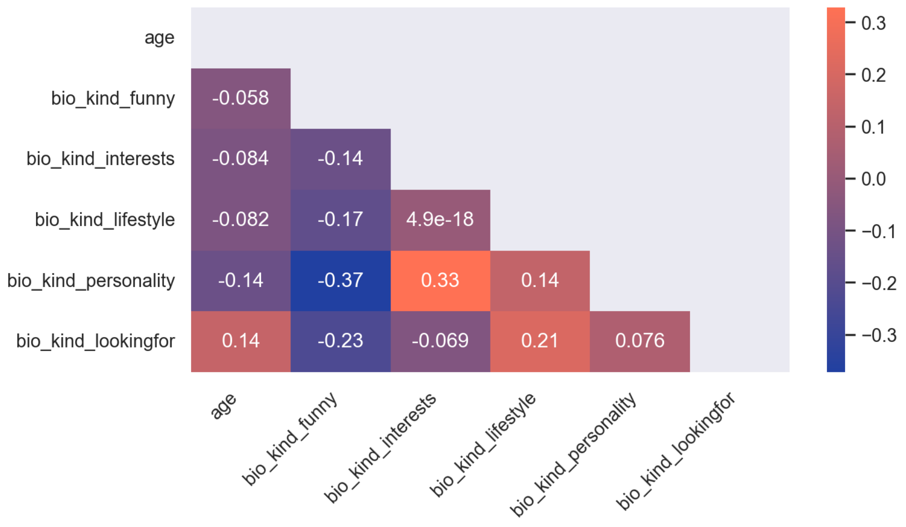


Figure 3.4: Correlation heatmap of the relationship between the different types of bio.

ences of users, as shown in figure 3.4. First of all, a funny bio has a relatively strong negative correlation with all other types of bio. This indicates that people looking for a funny bio tend to be less interested in actual information about the person. Inversely, people who are looking for the others' interests, lifestyle, and particularly personality do not prefer a joke or funny bio. Another relatively strong correlation can be found between a personality bio and an interests bio. This combo is further evidence that people want descriptive bios, characterising the person. A lifestyle bio also correlates with personality, probably because one's lifestyle can also paint a picture of what they are like. Lifestyle bios also correlate with a 'what are you looking for'-bio. This makes sense, as describing what you are looking for also gives some hint as to what your lifestyle might be.

Participants were also inquired about the length of bios. Figure 3.5 shows that the length of a bio does not seem to matter too much. When it does matter, opinions are divided roughly equally between the two options. Therefore, the bio length is difficult to meaningfully employ in a matchmaking approach.

As the bio is an optional, relatively recently introduced profile section, not every user has one. Of the surveyed people, 18 people (39.1%) had a bio at the time of filling in the survey, which is roughly in line with the ratio of the total active userbase (~35%). People with a bio tend to be older than those without one, as can be seen in figure 3.6, which also holds up for the overall active userbase. This again confirms that the survey participants are a representative sample of active users.

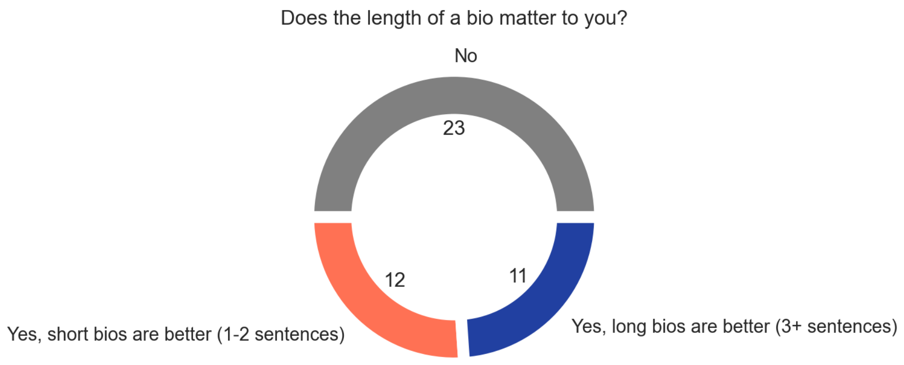


Figure 3.5: Opinion on the length of a bio, and whether it matters or not.

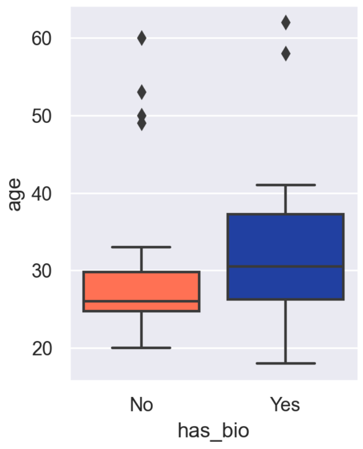


Figure 3.6: Age distribution of participant user profiles with and without a bio.

The participants who have a bio were also asked what type of bio they have themselves. Multiple options could be chosen, as multiple can be applicable. Most people provide a description of their interests in their bio. A complete overview of the responses is given in figure 3.7. People seem to want to describe the things they like. It makes sense that people want to write what they are passionate about, and coincidentally is also what others are generally looking for in a bio. This may be one of the reasons why profiles with a bio have a significantly higher match ratio, as seen in table 3.1. A larger fraction of the participants, as well as the active userbase, does not have a bio on the profile. These people were presented with a different section of the survey asking about their lack of bio. The predominant reasons people do not have a bio are

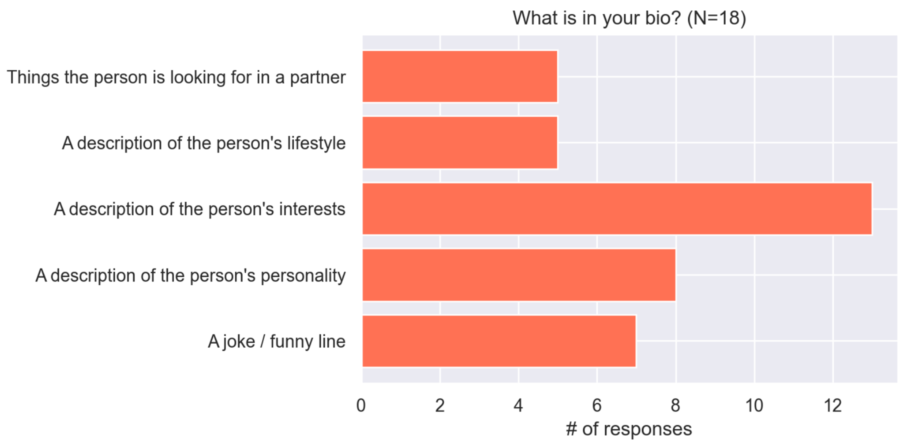


Figure 3.7: Bar chart of the different types of bio, showing how many of each type is present in people's own bios.

because they simply do not know what to put in it (53.6%), or think it is too much effort (17.9%). The open questions provide a specific prompt that the user can answer directly, while the bio is completely up to them. Therefore, it does require more creativity to fill it in. Another often-mentioned reason was that some users think it makes the profile too serious, especially by people under 30. Some examples of what people answered include:

> "It is really difficult to think of what to write in your bio. Maybe need more hints"
> "I feel like it makes my profile too serious and I don't know what I would write there."
> "Because I analyse other people's bios closely, I've put too much pressure on writing my own bio! :') But I wish to add one soon."

Again, the full list of responses can be found in the online publication of the results.[3] As the first response suggests, it might be worthwhile to provide some form of guidance when editing a profile suggesting what to put in your bio, for example using the data from figure 3.3.

Open questions

The final section of the survey asked about the open questions people can select and answer on their profiles. Users can select any number of the predetermined twenty-one open questions present in the app to answer freely, which will subsequently be

[3] https://bit.ly/3Q9qkoi

shown on their profile. Participants were asked to describe what they are looking for in open answers in their own words, and pick from some predetermined 'types' of answers again. As can be seen in figure 3.8, an overwhelming 89.1% of people want the open

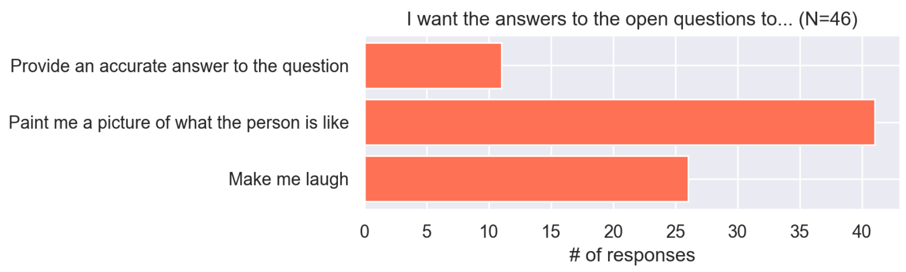


Figure 3.8: Bar chart of the different types of open answers showing how many people prefer each type.

answers to paint a picture of what the person is like. A little over half of people are also looking for humour in the answers, again positively correlated with younger users. Participants were also asked directly whether or not the answers should be serious, funny, or a mix of both. In this case, 73.9% of people wanted a mix of both. So answers should still provide some insights about the user, and not just be a joke.

Another question asked whether participants look for answers similar to their own answer to the question. This does not appear to be a useful factor however, as for 69.6% of people, it depends on the question, and a further 10.9% does not care. Finally, they were asked what the ideal number of open questions answered on a user profile is, and how many they have on their own profile. Figure 3.9 shows that 2 to 3 questions appears to be a good amount in general.

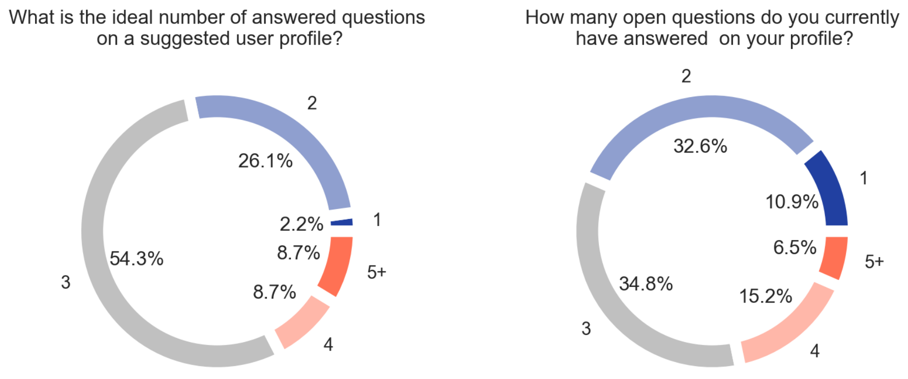


Figure 3.9: Responses regarding number of open questions on a user profile. Left shows the ideal number to include on a profile. Right shows how many answers the respondents have on their own profile.

RQ1 Conclusion

RQ1 asks how textual data from profiles can be leveraged to find similar user profiles in a reciprocal recommender system. We observe that there are several ways to do this. Firstly, as the survey showed, people who describe their own interests and hobbies are responsive to bios and open answers which describe the others' interests and hobbies. This also presents potential in the link between different profile sections, rather than only comparing the same section.

Crucially, we observe from the survey results that people are indeed overwhelmingly looking for similarities in the suggested user profiles on Breeze. As research also backs up that similarities tend to attract, we conclude that attempting to match two similar profiles is the correct approach in a matchmaking algorithm. Considering people are looking for similarities, we can feed the bio and open answers into a LM which can find similarities inside these profile sections. The most similar profiles are subsequently the best candidates to suggest.

Another observation is that the, generally older, users with serious bios are more interested in others who also have a more serious bio. Conversely, the same hold for mainly younger users with a funny or jokey bio. Therefore, we could use some form of humour detection to give each profile a 'humour index', and match people with similar scores.

3.2. Matching Two Lovebirds

Now that we have an idea of what the users are looking for, we present the final design of our matchmaking algorithm. The procedure of matching two users together is divided into two parts. First, we discuss our approach of finding the most similar users, which is increasingly important as the size of the system (i.e. userbase) grows. Second, we provide an outline of the final design of the LM at the core of our matching algorithm.

3.2.1. Finding Similar Plumage

Our goal is to find similar user profiles based on their bio and open answers, resulting in a fully content-based (CB) approach. To achieve this goal, we will create profile embeddings which are, in essence, a neural vector representation of all the textual data of a user profile. Such embeddings are ubiquitous in the field of NLP nowadays, so it should be a familiar data structure to most. The cosine distance between two profile embeddings can subsequently be used to determine how well it mirrors user judgement of similarity, based on the work of Le and Mikolov [28]. In our case, the profile embeddings will be composed of two components: (i) a bio embedding, and (ii) an open answers embedding. However, it is important to note that in practice, any text-based profile section can be incorporated as a component of the profile embedding.

One naive approach would be to simply take the average of all the individual word embeddings obtained by something like *word2vec* [41] or *GloVe* [48]. This approach is sometimes called Continuous Bag of Words (CBoW). However, CBoW is a suboptimal approach for several reasons. The order of the words in completely disregarded, it does not take the context of a word into account, and two sentences with an opposite meaning can result in the exact same embedding (e.g. "The **attractive girl** asked out the **shy boy** " and "The **shy boy** asked out the **attractive girl**"). Basically, the semantic information present in all the sentences would be lost, while that is very valuable information. Musto et al. also show it allows for noisy results in their item-to-user RS [43].

Instead, we want to use the semantic information present in the profile text and use a context-aware approach. For this, we need transformers [65]; more specifically, we need BERT [15]. As mentioned in section 2.3, BERT-based LMs achieve state-of-the-art performance on virtually all NLP tasks, including sentence classification, sentence-pair regression, and Semantic Textual Similarity (STS) tasks, which are especially relevant in our context. For sentence-pair regression tasks, which our problem approximately is, a basic BERT model uses cross-encoder networks where a sentence pair is passed to the internal transformer network and a target value is predicted. Crucially, however, a cross-encoder does not produce a sentence embedding, and cannot handle a single sentence as input. Rather, a cross-encoder infers the output for each possible pair of sentences separately.

Unfortunately, this setup can be unsuitable for some pair regression tasks, because the amount of possible combinations makes computation expensive ($O(n^2)$). Finding the sentence pair with the highest similarity in a collection of $n = 10,000$ sentences would require $n \cdot (n-1)/2 = 49,995,000$ inference computations, which would take about 65 hours even on a modern Nvidia V100 GPU [52]. At the moment, these computations would not be an issue yet for Breeze, as they provide ~100,000 suggestions a day. However, scalability is an important factor to keep in mind when designing a matchmaking system for a fast-growing start-up.

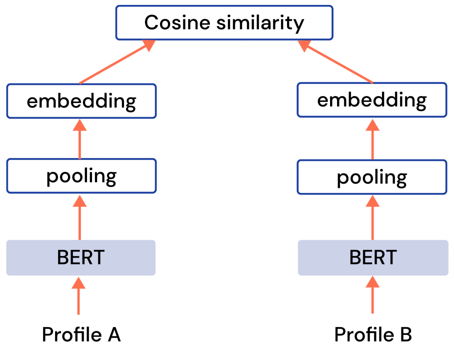


Figure 3.10: Siamese network structure in SBERT with a regression objective function. The two BERT networks have tied weights.

Therefore, we want an approach which derives fixed-sized embedding vectors that can be compared using measures such as cosine similarity or Manhattan / Euclidean distance. Not only are these distance metrics orders of magnitude cheaper to compute, but the embeddings can also be generated once per sentence (or bio/answer) and subsequently saved and indexed, further decreasing computation time. We achieve this by using SBERT (SentenceBERT) introduced by Reimers et al. [52]. SBERT makes use of a Siamese network structure, initially introduced in the field of computer vision [10]. A representation can be found in figure 3.10. This is an instance of a bi-encoder, which does produce an embedding, and can take individual sentences as input. It should be noted that the BERT component can be substituted by any other LM, as long as it computes semantically meaningful embeddings. Calculating the embeddings for the previously described example with 10,000 sentence pairs takes 5 seconds using this technique, with the running time at inference being reduced to a few milliseconds [52].

What about migratory lovebirds?

Love transcends all borders. Furthermore, Breeze wants to be as inclusive as possible. Therefore, it is important to be able to match profiles in different languages too. Currently, the user profiles on Breeze consist of a mix of Dutch and English, with plans to expand to Belgium and Germany in the near future. Hence, it is paramount that our matchmaking algorithm is language agnostic.

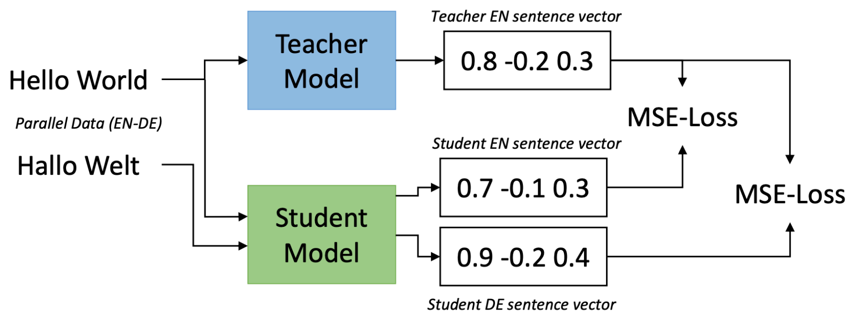


Figure 3.11: Example of knowledge distillation flow. Given parallel data (e.g. English and German), the student model is trained to produce embeddings for both the English and German sentences that are close to the teacher model's embedding. Image taken from [53].

For this, we use a strategy that transforms monolingual embeddings into multilingual embeddings called *Knowledge Distillation*, introduced by Reimers et al. [53]. This approach is based on the idea that a sentence in one language should be mapped to the same position in the vector space when translated to another language. An overview of how it works can be seen in figure 3.11. In essence, a monolingual 'Teacher' model is used to generate sentence embeddings on a specific input. Then, a new 'Student' model is trained using these sentence embeddings as a target, for the (translated) embeddings it generates itself. This way, it uses the same shared vector space for all different languages.

As we do not have an appropriate dataset available to create a model ourselves for Breeze, as well as due to time constraints, we use a pre-trained model which was trained using this technique. Specifically, we use *distiluse-base-multilingual-cased-v1*[4], which supports 15 languages including Dutch, English, French, and German. That is, this model constitutes the 'BERT' block from figure 3.10 in our case.

[4]An overview of the model can be found at `https://huggingface.co/sentence-transformers/distiluse-base-multilingual-cased-v1`.

3.2.2. LoveBERT: the Language Model for Matchmaking

The downside of using the SBERT bi-encoder as described in section 3.2.1, is that it usually boasts lower performance than a cross-encoder [63]. A cross-encoder can simultaneously compare both inputs, while a bi-encoder has to map inputs independently to a meaningful global vector space, which is arguably a more difficult task. Nevertheless, the computational efficiency and ability to save and index the embeddings is simply necessary for larger systems. Ideally, we want to combine the best of both worlds, where we capitalise on the efficiency of a bi-encoder, while preserving the performance of a cross-encoder.

In the context of Breeze, only a few (<15) suggestions per user are made each day. As discussed in section 2.1.1 and table 2.1, RRSs are tailored towards a low number of high-quality recommendations: quality is more important than quantity. This is a typical feature that can be exploited; say there are 50,000 possibilities for suggestions, then realistically, we only really care about the best 1,000. At the moment, this is not a relevant problem from Breeze yet because of the size of the userbase, but it will be in the near future at the current growth trajectory. Regardless, it is important to build a scalable and expandable matchmaking algorithm.

Finding the top 1,000 'best' suggestions would require calculating all 50,000 scores when using a cross-encoder, but can be done for much cheaper using the cosine distance between indexed profile embeddings retrieved from a bi-encoder. Those embeddings could be calculated on profile creation (and refreshed on subsequent profile updates). Ergo, the top-k closest profiles are selected using the embeddings retrieved from a bi-encoder, and the final ranking of this top-k is then decided by a cross-encoder fine-tuned on profile section pairs, yielding superior performance. This final cross-encoder used is LoveBERT, our novel LM designed for matchmaking.

LoveBERT Architecture

LoveBERT is a novel, generalisable LM designed for reciprocal matchmaking systems. It is fully content-based (CB), meaning it does not depend on implicit usage data, and can therefore be applied immediately to cold (i.e. new) accounts. Not all matchmaking systems are the same, they can contain different kinds of textual components on user profiles. Therefore, LoveBERT is built with modularity in mind. The LoveBERT model architecture and training is illustrated in figure 3.12.

For each text section of a user profile which should be considered by the matchmaking algorithm, a separate LM is fine-tuned. In the context of Breeze, this means one for the bios, and one for the user's answers to open questions. The answers are concatenated together to make a single input. These LMs are fine-tuned separately by taking both positive ("real") profile section pairs from two people who liked each

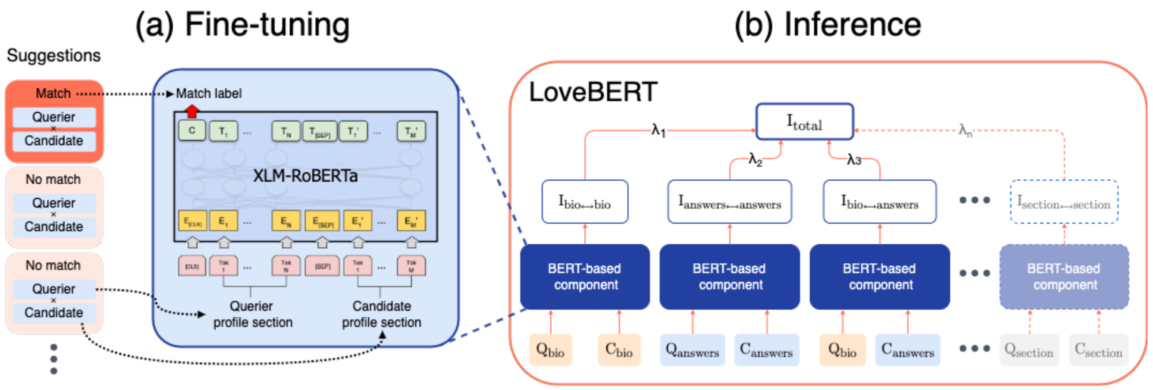


Figure 3.12: LoveBERT receives suggested profiles corresponding to positive (matches) and negative (non-matches) samples, extracted from historical data. During training (a), the relevant profile sections are propagated through the BERT backbone, fine-tuning it to recognise matches. At inference (b), three scores are computed and combined to get the final inferred matching score. LoveBERT can be extended by adding more components to process the relevant free-text profile sections.

other, and negative ("fake") pairs from two users who did not match. These blocks learn which latent factors influence whether or not two users match based on what they put in the corresponding profile sections. The base LM used for each component is *XLM-RoBERTa-base* [5] [13], chosen for its impressive performance in multilingual modelling without sacrificing per-language performance. Note that in practice, a different LM which might make more sense can be substituted as the base for any of the LoveBERT components.

Additionally, based on the link between different profile sections mentioned in our RQ1 conclusion, an additional component is also added which is fine-tuned on intersectional data. In this case, the bio and open answers of two people who have matched are counted as positive samples. This goes both ways, so requires two additional components. These blocks can learn links between the different profile sections which might indicate reciprocity.

The final reciprocal preference score of LoveBERT is subsequently calculated by taking a weighted average of the scores of each component, calculated as follows:

$$\mathcal{I}_{total}(Q,C) = \lambda_1 \mathcal{S}_{Q_{bio} \leftrightarrow C_{bio}} + \lambda_2 \mathcal{S}_{Q_{answers} \leftrightarrow C_{answers}} + \lambda_3 \mathcal{S}_{Q_{bio} \leftrightarrow C_{answers}}$$

where Q is the *querier* profile, C is the *candidate* profile, $\mathcal{S}$ the scores of each individual component, $\mathcal{I}_{total}$ the final reciprocal preference score, and $\lambda_1 \dots \lambda_3$ represent component weights. The component weights could be set manually for greater control, ex-

[5] An overview of the model can be found at `https://huggingface.co/xlm-roberta-base`.

plainability and interpretability, but can also be optimised as hyperparameters as we do in chapter 4.

While a final fully connected layer could be used instead of weights, this would require retraining of the whole network each time components are added, removed or swapped. Because rapid prototyping and testing is important in the competitive field of dating apps, this is not ideal. Additionally, it would be less transparent, as now there is more direct control of what goes into the decision-making of the algorithm.

Chapter 4

Experiments

The goal of our work is to evaluate the capability of LMs in the context of a matchmaking algorithm in a RRS. We do so through a set of research questions which we attempt to answer by performing experiments. The research questions are as follows:

- **RQ1:** How can textual data from profiles be leveraged to suggest attractive user profiles in a reciprocal recommender system?
  - **RQ1.1**: Is there a difference in the effectiveness of the utilisation of the various kinds of free-text data (e.g. bios, open answers)?
- **RQ2:** How can language models be used in a reciprocal recommender system, and which technique performs best?
  - **RQ2.1**: Can a language model help circumvent the cold-start problem?

4.1. Experimental Setup

In order to answer the research questions, we have fine-tuned LoveBERT as described in section 3.2.2 on historical matching data from Breeze. As the bios are an integral part of our approach, we only used historical suggestions in which both users had a bio for training and evaluation.

RQ1 is answered in section 3.1. To answer RQ1.1, we can look at the cosine distance between the two embeddings of the profile sections of two suggested users generated by the SBERT network, as described in section 3.2.1. Additionally, we perform an ablation study of the LoveBERT components. Each component is trained on a different free-text section of the profile, so comparing their individual results provides insights into the importance of each respective section.

To answer RQ2, we fine-tune a LoveBERT LM with historical Breeze data, and compare the results to several baselines. We use CBOW and the cosine distance between embeddings derived from the SBERT model to gauge the NLP-based performance. Additionally, we use a standard CF approach implemented with xgboost, as well as the current Breeze algorithm, to gauge the overall matchmaking performance.

To answer RQ2.1, we divide our test data in two distinct sets: suggestions with users present in the training data, and suggestions with unseen users not present in the training data. To ensure there is sufficient data for unseen users, we perform the test/train split based on date, so all the new users in the latest portion of the data will not be present in the training set. We then compare and analyse the differences in performance LoveBERT and all baselines between these two sets.

4.1.1. Evaluation

We evaluate the performance of LoveBERT through several evaluation metrics. Xia et al. [72, 73] defined specific Precision and Recall metrics tailored specifically for RRSs. For a given user u, we define three sets of users:

- T: the set of users we have suggested to u.
- L: the set of users who have been liked by u.
- R: the set of users who have been liked by u, and also reciprocated, i.e. liked u back themselves. In other words, a match.

Using these sets, we define the following evaluation metrics as follows:

$$\text{L-Precision } = \frac{|L \cap T|}{|T|}, \text{L-Recall } = \frac{|L \cap T|}{|L|}$$

and

$$\text{R-Precision } = \frac{|R \cap T|}{|T|}, \text{R-Recall } = \frac{|R \cap T|}{|R|}$$

as well as their respective F-scores. By taking the top-k of set T, we can easily evaluate these metrics at different sizes of the suggestion feed.

The R-metrics evaluate the match data, which is quite imbalanced in the context of Breeze. Only a small portion of the suggestions eventually results in a match (for illustration, around ~1% in the training data). Therefore, we also use the Matthews Correlation Coefficient (MCC), which is a score between -1 and 1, and is calculated using the confusion matrix quadrants, and defined as follows:

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)\,(TP + FN)\,(TN + FP)\,(TN + FN)}}$$

The MCC, as Chicco and Jurman explain in [11], is a more informative metric than F-scores in evaluating binary classification problems, particularly imbalanced ones.

Finally, we consider the AUC-ROC curve to measure the performance of the models at different threshold values.

Baselines

All the aforementioned metrics will be used to compare the performance of LoveBERT to several baselines. We include a main example for each of the different approaches to (R)RSs described in chapter 2. One Content-Based (CB) approach, a Collaborative Filtering (CF) approach, and a Hybrid approach. Additionally, we show the results of two embedding-based NLP approaches: Continuous Bag of Words (CBoW) and the Bi-encoder as described in section 3.2.1.

- **Content-based:** The CB approach is implemented using Xgboost. It uses different features of profiles, such as age, height, and the accept rate and reverse accept rate, which are metrics describing how often the user likes other profiles and is liked themselves respectively. Additionally, some features of a suggestion are also used, such as the distance between the two profiles' hometowns or education level. The standard `xgboost` Python library is used for implementation.
- **Collaborative Filtering:** The CF approach is implemented using Latent Factor Models, a proven design in the field of user-item recommendation [26]. We use an implementation, called Latent Factor model for Reciprocal Recommendation (LFRR), specifically designed for RRSs by Neve and Palomares in [45]. In short, it uses latent factor models to compute two unidirectional preference scores, which are subsequently aggregated into a reciprocal score. Latent factor vectors are initialised randomly, and optimised using interaction data to minimise errors. It is implemented using the `surprise` Python library.
- **Hybrid:** The Hybrid baseline is based on a combination of the CB and CF baselines, and inspired by the algorithm powering Breeze at the time of writing. It is simply a weighted combination of the aforementioned CB and CF scored, where the weight of the CB part is $2/3$. While there is no true 'current' Breeze implementation because of the continuous experimentation performed in the never-ending pursuit for better performance, this is also a baseline used for internal testing at Breeze.
- **NLP:** Lastly, two NLP-based approaches are also used. First off, Continuous Bag of Words (CBoW), which simply takes the average of all the individual word embeddings obtained by *GloVe* [48]. The other method to obtain embeddings from the profile text is using SBERT, as described in section 3.2.1. In both cases, the cosine distance is then computed between the two profiles, and the resulting distance is mapped to the range of $[0, 1]$ to represent the matching score.

4.1.2. Dataset & Training

Our dataset consists of historical suggestions made in the Breeze app. A suggestion consists of a *querier* and *candidate* profile, including all their respective profile data. We only used suggestions where both parties had written a bio, resulting in a dataset of 113K samples. Users' open answers were concatenated together as they were overwhelmingly short enough. Additionally, we derive two labels: whether the querier liked the candidate (i.e. like data), and whether the suggestion ended in a match (i.e. match data), which happens if the candidate has already liked the querier at some point in the past. As mentioned in table 2.1 in section 2.1.1, success is determined by both sides of the recommendation in a RSS, meaning only a match is counted as a positive sample. However, matches account for only around 1% of the suggestions, resulting in very imbalanced data which leads to some problems which will be discussed in section 5.1.1. Therefore, we use the like labels for training. Note that the set of liked suggestions L is a strict subset of the set of suggestions part of matches R, i.e. $L \subset R$, as a match requires a like in both directions. The dataset was split 70%/15%/15% into training, evaluation and test sets respectively, resulting in 95K training samples and 17K validation and test samples. The dataset is ordered by date (when the suggestion was made) before making the split to ensure there are previously unseen users in the test set.

All experiments performed for this study were executed on an NVIDIA RTX 3060 Ti GPU. Each LoveBERT component was separately fine-tuned on a pre-trained XLM-RoBERTa$_{base}$ model, using a batch size of 8. The pre-trained models were provided by Huggingface[1]. The output layers of the models consist of a fully connected layer with two output labels. A dropout probability of $p = 0.1$ is used, and a weight decay of $\lambda = 0.01$. The used optimiser is AdamW with an epsilon value of $\epsilon = 1\text{e-}6$. We found training to be very sensitive to the learning rate, and consequently use different learning rates for the fully connected classifier layers and the encoding layers. The optimal learning rate is also different for each LoveBERT component. A warm-up period of the learning rate consists of 10% of the training steps across the board. Max sequence length is $T = 512$ tokens. Each LoveBERT component is trained for a maximum of 8 epochs. A full overview of the hyperparameters for each component can be found in appendix A.

4.2. Results

Now, we present the results of our experiments and discuss the observations made based on them. The threshold for a positive prediction is 0.5 across all experiments. That is, a like or match is predicted if a model's prediction score is ≥ 0.5.

[1] `https://huggingface.co/`

L-metrics

We begin by looking at the L-metrics, where predicting likes is the goal. A like is the first step towards a match, so it is important to analyse to make better suggestions. Furthermore, when a person likes someone as the first party, they always end up in the suggestion feed of the other, aiding the matchmaking system.

Table 4.1 shows the overall performance on the test set of the different models we have tested. Figure 4.1 shows the corresponding ROC curve. There are several interesting observations we can make based on these results.

Model	**Type**	**L-Prec.**	**L-Rec.**	**L-F1**	**MCC**	**AUC**	**TN**	**FP**	**FN**	**TP**
Xgboost	*CB*	0.286	0.185	0.225	0.108	0.587	12753	1333	2359	535
LFFR	*CF*	0.278	0.126	0.173	0.083	0.583	13142	944	2530	364
Breeze	*Hybrid*	0.293	0.174	0.219	0.110	0.589	12870	1216	2389	505
CBoW		0.170	**0.969**	0.289	-0.005	0.511	406	13680	**90**	**2804**
SBERT	*CB-NLP*	0.171	0.905	0.288	0.008	0.519	1426	12660	275	2619
LoveBERT		**0.683**	0.468	**0.555**	**0.496**	**0.862**	**13456**	**630**	1539	1355

Table 4.1: Overall model performances when evaluating predictions using like data as the ground truth. The best score achieved for each metric is displayed in bold.

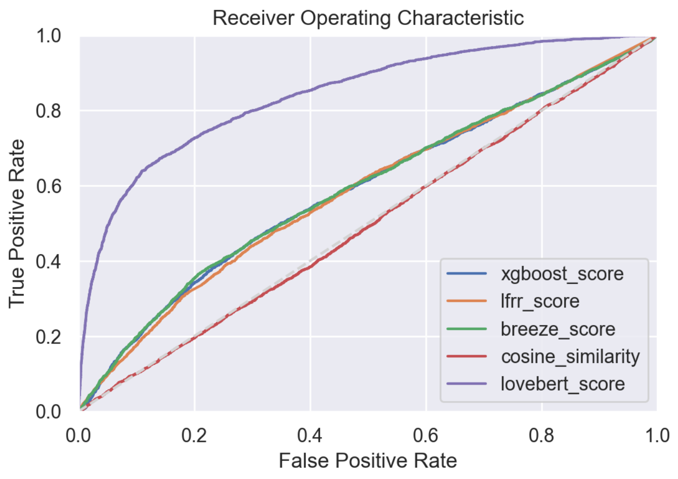


Figure 4.1: ROC curve for like data.

Observation 1: LoveBERT greatly outperforms all baselines in predicting likes. While all the models were trained on like data, LoveBERT significantly outperforms the others in the predictions of likes. It achieves an F1-score of 0.555, and a MCC of 0.496, compared to the 0.225 and 0.110 respectively of the best performing non-NLP approaches. When we look at the ROC curve in 4.1, we can see that LoveBERT is also more robust, performing better than all other approaches for all threshold values.

Observation 2: Embedding-based approaches using the cosine distance make for terrible predictors. Both the embeddings from CBoW and SBERT are barely better than random guessing, and have an incredibly high false positive rate. This can be explained by the fact that the cosine distance is exactly that, a distance metric, and cannot sensibly be used as a prediction probability. The cosine distances are mapped to the [0,1] range. As most of the distances were just over 0, the mapped scores ended up concentrated around 0.55, as shown in figure 4.2. However, this distance can be useful in ranking the closest profiles in terms of their text content.

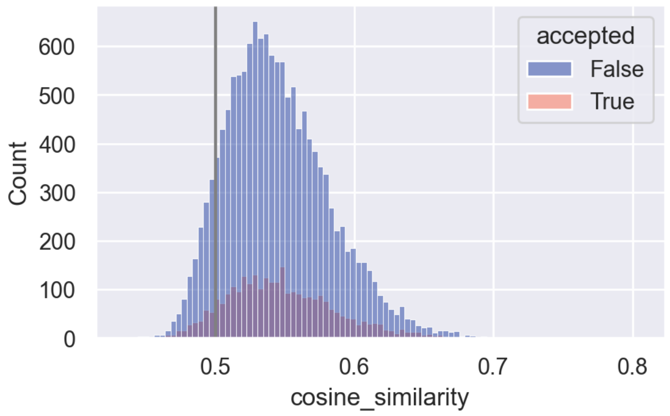
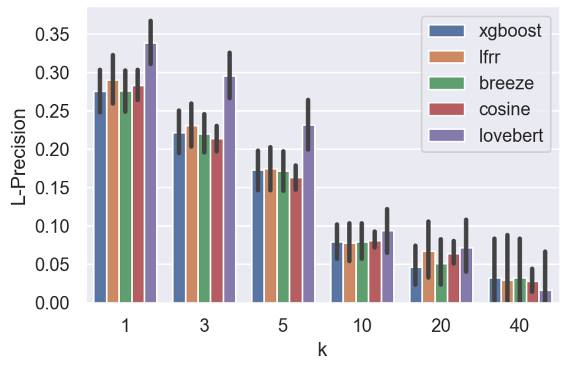


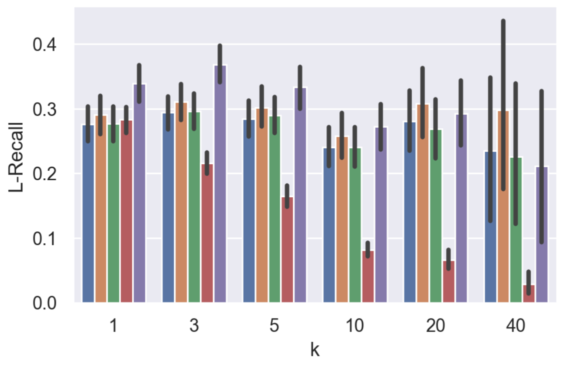
Figure 4.2: Distribution of cosine similarity values of all suggestions.

Additionally, we take a deeper look at the precision and recall at different levels of suggestion feed size. These *Precision@k* and *Recall@k* metrics are shown in figure 4.3, and are calculated by taking the top-k prediction scores of each classifier for each unique querier, and use this as the set T in the metrics as defined in section 4.1.1. For each k, we only include the querier if they had answered at least k suggestions. Different values for k can give an impression of the precision and recall over different time spans, as users only get a couple of suggestions per day.

For example, let us consider user u with three suggestions which are all likes in reality. A model predicts the three scores [0.4, 0.6, 0.1] for the three suggestions. Only one of them is convincing enough (i.e. pass the decision boundary), and predicted as a like, resulting in a recall of 1/3. However, when we look at the top-1 score of u, only the score of 0.6 will be included and therefore classified correctly, with a corresponding recall of 1.0 for the top-1.



(a) L-Precision@k



(b) L-Recall@k

Figure 4.3: L-Precision and L-Recall values for different sizes of suggestion feeds for each different model. The mean value is indicated by the bars, with the lines indicating the confidence interval.

Observation 3: Precision drops off quickly the more suggestions are made. Figure 4.3a shows that as k increases, the L-Precision decreases. The precision of LoveBERT predictions drops off most quickly, but remains the most performant up until a k of 40. A drop in precision as more suggestions are made does make sense. As the predictions are based on the top-k scores of the model, higher k values will include more instances where the model's prediction score is very low, but still in the top-k.

Observation 4: Recall drop is less severe, and the CF approach even increases initially As can be seen in figure 4.3b, the L-Recall also drops as k increases in the LoveBERT predictions. It remains the most performant for the lower values of k, while it loses its edge after k passes 20. It is interesting to note that the L-Recall of the CF method LFRR actually increases as k does as well. This makes sense, as it learns from a user's liking behaviour. Therefore, it should perform better on users with a high amount of rating data, which it does.

R-metrics

It is in the best interest of both Breeze and the users to obtain matches. Breeze because they make money per date, and users because they want to find a suitable partner. Also, as mentioned in table 2.1 in section 2.1.1, not only do we want to make high-quality suggestions because of the limited availability of users in a RRS, many poor recommendations resulting in repeated rejections might also dishearten use of the app. Therefore, the R-metrics are of utmost importance, to a greater degree than the L-metrics.

Model	**Type**	**R-Prec.**	**R-Rec.**	**R-F1**	**MCC**	**AUC**	**TN**	**FP**	**FN**	**TP**
Xgboost	*CB*	0.023	0.371	0.043	0.057	0.671	14124	2686	107	63
LFFR	*CF*	0.023	0.253	0.042	0.045	0.660	**14966**	**1844**	127	43
Breeze	*Hybrid*	**0.026**	0.376	**0.048**	**0.065**	**0.679**	14383	2427	106	64
CBoW	*CB-NLP*	0.010	**0.965**	0.020	-0.004	0.529	490	16320	**6**	**164**
SBERT	*CB-NLP*	0.011	0.947	0.021	0.016	0.583	1692	15118	9	161
LoveBERT	*CB-NLP*	0.017	0.241	0.032	0.028	0.631	14422	2388	129	41

Table 4.2: Model performances when evaluating match-based predictions. The best score achieved for each metric is displayed in bold.

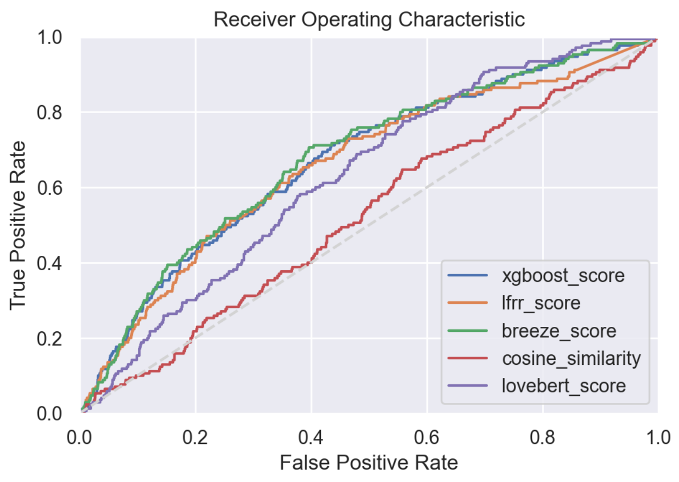


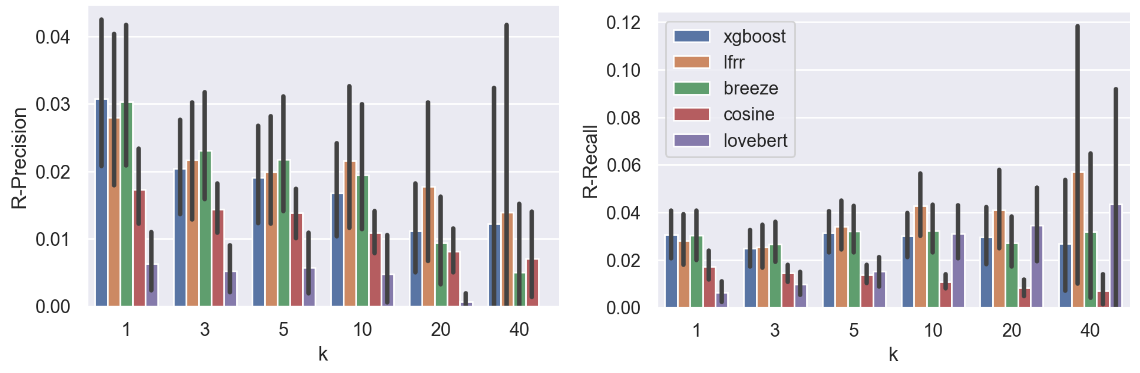
Figure 4.4: ROC curve for match data.

Table 4.2 shows the overall performance on the test set of the different models we have tested when predicting matches is the objective. Figure 4.4 shows the corresponding ROC curve. There are several interesting observations we can make based on this data.

Observation 5: Overall performance is quite low. This data shows that predicting matches is a very difficult problem. The highest R-Precision of any of the classifiers is 0.026, showing it is quite difficult to refrain from suggesting false positives. The imbalanced nature of this data likely plays a significant role in this.

Observation 6: Distance-based classifiers remain woefully ineffective. The match data confirms that it is not useful to use the distance between profile embeddings as a prediction score. SBERT does achieve the highest recall with 0.965, but this is again simply the result of most vectors being close to orthogonal to each other, resulting in a score of just over 0.5 passing the classification threshold. The number of false positives is also extremely high compared to the others again. For the remaining part of this chapter, we will generally disregard the cosine similarity scores.

Observation 7: LoveBERT's superior performance does not transfer to the match domain. Table 4.2 shows that LoveBERT is outperformed in virtually all metrics by the non-NLP based models. The Hybrid model is the clear winner here, boasting an R-F1-score of 0.048 versus the 0.032 of LoveBERT, and an MCC of 0.065, more than double the 0.028 of LoveBERT. The MCC is particularly important here, as that metric is designed for imbalanced data like this. The CB and CF approach both also outperform LoveBERT here. The AUC also shows that more traditional approaches are more robust at different threshold values as well. Although, there do exist some threshold values where LoveBERT outperforms them. This, together with the performance of LoveBERT on like data, shows promise in fine-tuning LMs for matchmaking purposes.



(a) R-Precision@k **(b)** R-Recall@k

Figure 4.5: R-Precision and R-Recall values for different sizes of suggestion feeds for each different model. The mean value is indicated by the bars, with the lines indicating the confidence interval.

Again, we take a deeper look at the precision and recall at different levels of suggestion feed size. The *Precision@k* and *Recall@k* metrics for the reciprocal match data are shown in figure 4.5. Similar to before, they are calculated by taking the top-k prediction scores of each classifier for each unique querier.

Observation 8: Trends from the like data repeat, with overall lower performance. Once again, we see that precision drops as k increases across the board. This indicates it is difficult to preserve the same level of preciseness when you keep trying to make new predictions with a lower confidence level. Of course, in a live environment, there is a steady increase of new users which can potentially have high scores again, somewhat alleviating this problem.

Recall also steadily increases for the CF approach again, confirming that its performance increases as more behavioural data of the users becomes available. Interestingly, LoveBERT's recall also steadily increases this time.

RQ2 Conclusion

RQ2 asks how language models can be used in RRSs, and which technique performs best. We show two main ways to incorporate LMs in the matchmaking algorithm of a RRS. Firstly, using LMs to create profile embeddings and calculating the cosine distance between them. We show that this distance is not suitable to be used as a prediction score, but can effectively rank suggestions based on how similar the free-text profile sections are. Secondly, we fine-tune a pre-trained LM with historical suggestion data in order to predict likes and matches. Our observations show that while a LM-based approach can effectively predict likes, it unfortunately lags behind in performance when evaluating on match data. A more traditional Hybrid approach with a CB and CF component remains the most performant method of accurately prediction matches between users in an online dating context. However, our approach shows promise when considering the results on like data, and the fact that there do exist threshold values where LoveBERT outperforms the Hybrid approach. We also note that all approaches work better for the top-k suggestions, meaning they all fulfil the goal stated in table 2.1 of providing fewer high quality matches over many worse ones.

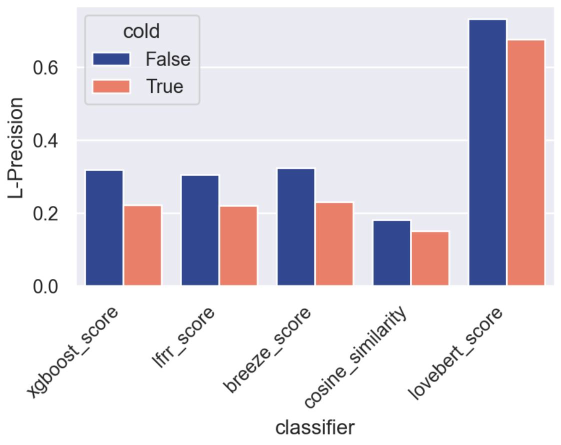
4.2.1. Cold-start Analysis

As mentioned in section 4.1.2, we made the train/validation/test split based on suggestion date to ensure there are previously unseen users present in the test set. Such users are called cold users, where no usage data has been available for training for CF methods. As a result, 25.50% of users in our test set are cold users, or 750 users of the 2941 unique users in total. We compare the performance between the set of cold users and the rest of the test set to assess which method can handle the cold-start problem best. Figure 4.6 shows the differences in precision and recall values for previously unseen (i.e. cold) users, and users present in the training set. We observe several interesting things in these graphs.

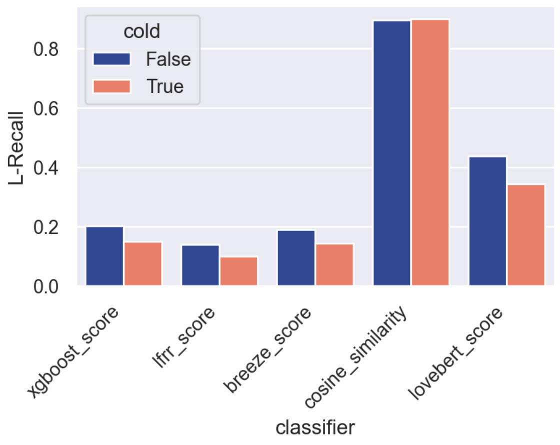
Observation 9: L-Precision and R-Recall both decrease somewhat for cold users. Figure 4.6a and 4.6d show that both metrics slightly drop when evaluating the set of cold users. However, in both cases, the relative drop in the performance of LoveBERT when compared to the other classifier is lower. In fact, for cold users LoveBERT just barely outscores LFRR in R-Recall. Therefore, while LoveBERT is still affected by the cold-start problem in these metrics, it is somewhat more robust to the effects than the other models.

Observation 10: L-Recall drops more significantly for LoveBERT, but it still comfortably retains its performance lead over the other models. As shown in figure 4.6b, all models also show a drop in L-Recall. Only the cosine similarity scores do not drop in, but we have already concluded that is not an effective model to make suggestions regardless. The drop for LoveBERT is comparatively bigger, but it still comes out ahead of the others.

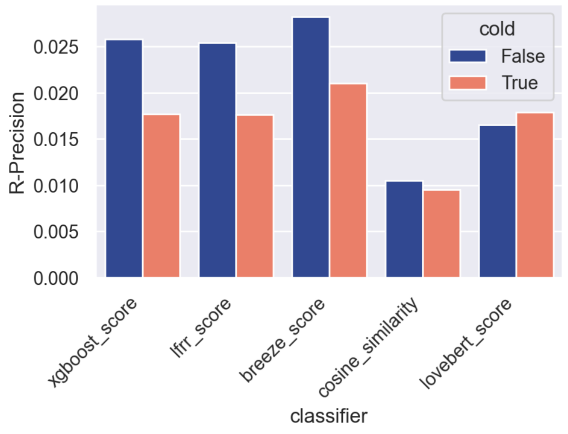
Observation 11: All models apart from LoveBERT lose a lot of R-Precision The three more classical approaches all lose a considerable amount of precision when evaluating on the match data. LoveBERT does not follow this trend, and instead gains a little. However, while the gap tightens, LoveBERT remains unable to achieve a higher absolute score. Note that the scale of figure 4.6c is over an order of magnitude smaller than the other three. So while these changes might appear to be quite drastic, in absolute terms it represents a very small change.



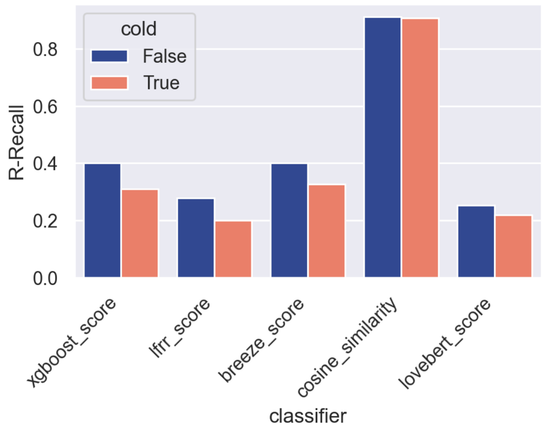
(a) L-Precision of all classifier based on predicted like data.



(b) L-Recall of all classifier based on predicted like data.



(c) R-Precision of all classifier based on predicted match data.



(d) R-Recall of all classifier based on predicted match data.

Figure 4.6: Differences in Precision and Recall scores of each classifier between previously unseen (i.e. cold) users, and users present in the training set, for both like and match data.

RQ2.1 Conclusion

RQ2.1 asks whether or not a LM-based approach can circumvent the cold-start problem. We show that in almost all metrics, every approach takes a hit in performance when asked to make predictions for cold users. This means the cold-start problem is, in fact, a problem. While LoveBERT cannot circumvent the cold-start problem entirely, we do show that it is more resistant to its effects compared to the other approaches. This provides a promising avenue for future research looking to tackle the cold-start problem.

4.2.2. Ablation Study

In order to gauge the performance of each individual component of LoveBERT, we perform an ablation study to find out how effective each part is. This is easily done by setting all weights $\lambda_1 \ldots \lambda_3$ and combinations of them to 0. This way, we only get the scores of those components. Table 4.3 shows the performance of all (combinations of) individual components.

Model	**R-Prec.**	**R-Rec.**	**R-F1**	**MCC**	**AUC**
LoveBERT$_{\lambda_1 \leftarrow 0}$	0.015	0.165	0.027	0.017	**0.654**
LoveBERT$_{\lambda_2 \leftarrow 0}$	0.016	0.118	0.029	0.018	0.606
LoveBERT$_{\lambda_3 \leftarrow 0}$	0.015	0.141	0.027	0.016	0.553
LoveBERT$_{\lambda_1, \lambda_2 \leftarrow 0}$ *(only bio-ans)*	**0.017**	0.218	0.031	0.026	0.652
LoveBERT$_{\lambda_1, \lambda_3 \leftarrow 0}$ *(only ans-ans)*	0.015	0.165	0.027	0.017	0.606
LoveBERT$_{\lambda_2, \lambda_3 \leftarrow 0}$ *(only bio-bio)*	0.011	0.106	0.021	0.004	0.538
LoveBERT$_{best}$	**0.017**	**0.241**	**0.032**	**0.028**	0.631

Table 4.3: Ablation study results. When a single component is deactivated the remaining two weights are set to 0.5. Best results are in bold.

Apart from discovering the influence of each (combination of) components, we also want to know what combination of weights performs best. Therefore, we perform a grid search on the weights in order to find the best performing model. An interval of 0.05 was used for the grid search for all three weights. The resulting combination of weights yielding the highest performance is $\lambda_1 = 0.1, \lambda_2 = 0.25, \lambda_3 = 0.65$.

Observation 12: The bio-bio component of LoveBERT performs significantly worse than all other components or their combinations. Table 4.3 shows the bio-bio component scores significantly lower in all metrics than the other components. Its precision is 0.011, while the next lowest precision is 0.015. Its MCC, which is the most significant metric for the imbalanced match data, is the biggest tell that its performance is lacking when compared to the other (combination of) components. It is only 0.004, 4 times as low as the next lowest score of 0.016. This tells us that bios alone are not too useful to predict matches. Further confirming this is the fact that the bio-bio component's weight is only 0.1. A likely explanation is that many of the bios are extremely short, meaning there is less information available to learn from this component.

Observation 13: The bio-ans component is the most performant individually. While the bios alone might not be enough to reliably predict matches, there is definitely some useful information present in the bio. The bio-ans component is the best-performing individual component. In fact, it performs nearly as well as the best performing model on its own, with an MCC of 0.026 compared to LoveBERT$_{best}$'s 0.028. This suggests that there is a strong link between the information present in the bios of the querier, and the answers of the candidate. We concluded in section 3.1.2 that most people's bios contain information about the user's interests and lifestyle, as well as that people generally look for similarities in other profiles. Combined with the fact that many of the questions direct users to write about their interests or lifestyle, it makes sense that there is valuable information present in the link between bios and open answers that can be learned by our LoveBERT component. The fact that there are often multiple questions answered on a profile also simply provides more information to process, and subsequently to learn from for the model. This is further bolstered by the fact that every combination where at least some component leverages open answers performs fairly well.

RQ1.1 Conclusion

RQ1.1 asks whether there is a discernible difference in the effectiveness of the utilisation of the different free-text profile sections. The difference in performance between all individual components is fairly drastic. The bio-bio component performs significantly worse than the ans-ans component. In turn, the bio-ans component combining the two different sections yields the highest performance on its own. Therefore, we conclude that there is indeed a difference in the effectiveness of the utilisation of the different profile sections. A key takeaway appears to be that longer texts mean more information to learn from, which in turn leads to better results. Additionally, combining different types of free-text data is also highly effective, outperforming both individual profile sections on their own.

Chapter 5

Discussion

We summarise our results, and discuss the different limitations of our approach in this section. Additionally, we briefly describe some explored avenues that failed to produce meaningful results for our research.

5.1. Limitations

There are several limitations to this study which inhibit the performance of LoveBERT. Some are specific to the context of Breeze, while others apply to the more general setting of matchmaking with practical implications that are not solved easily.

5.1.1. Data Imbalance

We have mentioned before that only a very small number of suggestions result in a match. In our case, there are 1232 matches in a dataset of 113K samples, so just over 1%. One problem which presented itself was that it was not practical to get sensible output when training on such imbalanced data. We have attempted some standard ways to deal with imbalanced data, such as undersampling negative samples, oversampling positive samples (including substituting words with synonyms [58]), and using a weighted loss function. In the end, this all yielded lower performance than simply using the unidirectional like data as the label instead of the reciprocal like (i.e. match) data, as described in section 4.1.2. As dealing with data imbalance in NLP tasks is an open task in research which could warrant a research project on its own, we simply use like data and consider the overarching problem out of scope for this work.

5.1.2. Data Sparsity

The problem data sparsity presents is twofold in our case. Firstly, it decreases the size of the usable dataset, because the bio section was introduced fairly recently and is not mandatory. Therefore, the majority of users do not have a bio on their profile. As we are working with suggestions made, that means that if even one side of the equation lacks a bio, we cannot effectively use it. Now, the usable dataset is still substantial enough at 113K samples, but that is just a fraction of the total number of suggestions made (5.3M).

The second side of data sparsity pertains specifically to the open questions. Users have an open choice in which questions they answer on their profile. At the time of conducting my experiments, there were twenty-one options to choose from. We expect the system to learn something from the open answers best when the two users have picked the same question, as in that case their answers are most likely to relate to each other. As seen in section 3.1.2, as well as confirmed by the data, the vast majority of people have 3 or fewer questions on their profile. For illustration, let us see three independently randomly chosen questions of person A as 'good' options, and the remaining 18 as 'bad' options. As the order of the questions does not matter, we can calculate the number of possible combinations of questions using the binomial coefficient C^n_k defined as:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

where n is the size of the set of all elements, and k the number of elements picked. Thus, the probability that two users choose exactly the same 3 questions is:

$$\frac{\binom{3}{3}\binom{18}{0}}{\binom{21}{3}} = \frac{1}{1330} = 7.5\text{e-}4$$

Even an overlap of two questions is unlikely, even in a favourable case where both users have picked 3 questions (which is fairly unlikely in and of itself):

$$\frac{\binom{3}{2}\binom{18}{1}}{\binom{21}{3}} = \frac{3 \cdot 18}{1330} = 0.04$$

The point is, it is very unlikely people pick the same questions, which makes it less likely their answers contain similarities, which in turn makes it more difficult for the model to learn from.

5.1.3. Mutable Training Data and Unreliable Ground Truth

Breeze does not save the history of the contents of profile sections. Therefore, the profile section data corresponds to what it was at the latest point in the dataset, which is early April. This means that like and match data from early January for example, does not necessarily contain the correct corresponding profile data which was shown to the users at that time. Obviously, this is suboptimal, as the training samples might not represent the actuality of the situation at the time. Ideally, the profile information at the time of rating is saved and linked to that suggestion. However, this was not attainable in time for us. This issue would be less pronounced in a production setting where the models are continuously fed new data and deployed.

The mutable training data might also lead to scenarios where both parties did not like each other at the time, but if they had been presented with the updated profile information, they might have. In such cases, negative samples should actually have been positive ones. This can also happen when the same two users are suggested to each other again after a longer time period. Breeze had not done this yet, but it could happen in the future or in other contexts. People are not deterministic beings, and people who do not match at one moment could easily match later in life, meaning the ground truth is not always set in stone.

Finally, another problem with the ground truth in our context is that it is the ground truth for the profile as a whole, and not merely for the free-text portions we focus on. For example, two users can now be a perfect match based on the contents of the bio and open answers, but simply not be physically attracted to each other. Ideally, we would know when this was the case so the suggestion could still be used to improve the text-based predictions. Unfortunately, this is currently not possible.

5.1.4. Bio Content Randomness

One of the less quantifiable problems is the randomness of the contents of people's bios. Unlike specific open questions or more concrete questions, there's no particular sense of direction. The lack of direction being a problem is confirmed by the data we present in section 3.1.2, where we also conclude that people often find it challenging to write a bio, because they do not know what to put in it.

As a result, many people do not write a bio, and those who do often write about a divergent set of things, which makes learning from them more difficult. As suggested before in section 3.1.2, this problem can likely be partially alleviated by providing some sort of prompt or guidance when editing a bio with examples or tips of what to put in it.

5.1.5. Lack of Datasets

Finally, there are no other datasets available for the online dating setting with matching data, which makes it difficult to test our system on different data, as well as confirming or comparing the results of other works. In fact, there are no ready-to-use datasets for RRSs available in general, so we cannot even test LoveBERT on data from a different context.

For the online dating domain, it makes sense as the data is extremely private, and contains privacy-sensitive information. For that reason, we are unfortunately also unable to publish our dataset. Nevertheless, it is a regrettable fact which makes it difficult to gauge the state of the field.

5.2. Practical Implication

Our work reveals the following practical implications. These are matters which should either be considered in future research on matchmaking or RRSs in general, or suggestions which could aid Breeze or other online dating platforms in improving their product.

5.2.1. LoveBERT Provides Possible Cold-start Alleviation

In section 2.1.2 we show that while we cannot circumvent the cold-start problem with our approach, it does appear to be more robust to it. This indicates that CB-NLP approaches might be an effective approach to alleviate the cold-start problem in systems that are particularly vulnerable to it.

5.2.2. Informative Bios are Necessary

Per section 3.1, one of the foremost reasons Breeze users currently do not have a bio is that they are unsure of what to put in it. Additionally, many of the bios that do exist are not necessarily sensible or do not follow a specific format, as we have discussed in section 5.1. The more directed open answers section currently contributes much more to LoveBERT's performance, showing that there is valuable information present in textual data. However, some kind of structure is necessary to extract it. Considering the following two things: (i) both of these factors play a part in the disappointing performance of LoveBERT, especially the bio-bio component, as shown in table 4.3, and (ii) both of these issues could be alleviated by helping users fill out their bios. We believe some kind of guideline or nudge should be present to aid users in writing a bio. The specific form of this nudge is left open, but it should push towards a descriptive, informative bio.

5.3. Unsuccessful Endeavours

In addition to the limitations and implications of our work, there were also some approaches attempted, which unfortunately did not contribute to a meaningful increase in performance. Nevertheless, for the sake of completeness, we still describe the concepts we have explored, what went wrong, and if it could work given more resources.

5.3.1. Humour Detection

A key finding from section 3.1.2 is that people with a humorous bio and/or open answers are looking for humour in others' profiles as well. Inversely, people with more serious profiles tend to look for other serious profiles. Therefore, something like a 'humour index' for all profiles might be a useful metric to use in the matching process.

Humour detection is an active area of research in the field of NLP, and there are several LMs quite effective at detecting humour in (short) pieces of text. We have attempted to apply some of these to our data to attribute a 'humour index' to each profile, to no avail unfortunately. None of the models tried had sufficient performance to be able to use in practice. Overall, there were two main problems. First of all, none of the models were language-agnostic, generally only being able to handle English. This was a big problem for Breeze, where most profiles are in Dutch. Secondly, most humour detection models are trained specifically on jokes, often with a specific **setup** and a **punchline**. For example, an instance from a 200K short text dataset used by one of the models tested [2] is the following:

> **Why do time machines make you happy?**
> **They're an anti de-present.**

Bios on profiles can also be quite humorous, but often do not explicitly follow the format of a joke, i.e. *"My name is Janine but you can call me tonight!"*. This mismatch in format makes it more difficult for the models to effectively detect humour in the profile bios.

In table 5.1, we present a brief overview of specific problems for each of the models we have tried. Due to time constraints and scope limitations, we did not attempt to label part of our own data and train a model ourselves. In the end, we are not discounting the idea of successfully applying humour detection in a matchmaking setting, but conclude that it did not work in our current context.

[1]From `https://huggingface.co/thanawan/bert-base-uncased-finetuned-humordetection`

Model	**Problems encountered**
ColBERT [2]	Trained on a joke dataset with a fairly specific format. Only supports English and Hindi.
Weller and Seppi. [70]	Trained on jokes from a variety of sources, but only supports English.
Thanawan.[1]	No information of dataset used available, unknown which languages are supported (presumably only English).

Table 5.1: Problems encountered for each humour detection LM.

Chapter 6

Conclusion

Attempting to suggest the ideal romantic partner is a daunting task. People are emotional beings, meaning it can be difficult for hard science alone to solve this issue. Nevertheless, there are several online dating platforms who try their best, and spend a lot of resources to develop the most performant matchmaking system. This works contributes to this objective, in the context of Breeze. Breeze is an online dating app where users who match immediately go on a first date, without the ability to chat beforehand. This means that some of the core principles of RRSs, such as favouring quality over quantity, are extra important for them. Therefore, the matchmaking algorithm is perhaps the most integral part of the platform.

This works contributes to the goal of suggesting the best possible matches in several ways. First of all, we performed a user study with active users of the Breeze app, asking about their desires and preferences regarding other user profiles. Second, we have investigated the effectiveness of using NLP methods, specifically leveraging language models, to generate better suggestions for potential romantic interests. To that end, we have introduced LoveBERT, our novel, easily extendable LM-based matchmaking model. We also show the promise of LoveBERT, and provide an analysis of the importance of the different profile sections.

We set out to answer two research questions in this work, each with their own subquestion. Each of the research questions is repeated below, together with the corresponding conclusions.

RQ1: How can textual data from profiles be leveraged to suggest attractive user profiles in a reciprocal recommender system?
To answer this research question, we conducted a survey among active users of the Breeze app to find out the general desires of the userbase. We concluded that people are generally looking for similarities in the suggested user profiles on Breeze. As research also backs up that similarities tend to attract, we conclude that attempting to match two similar profiles is a good approach for a matchmaking algorithm. Additionally, we

concluded that people who describe their own interests and hobbies are responsive to bios and open answers which describe the others interests and hobbies. This initiated the first steps in the design of LoveBERT. Finally, we concluded that users with serious bios are more interested in others who also have a more serious bio, and vice versa for funny bios. We explored this direction, without result, and therefore leave it as an open problem.

RQ1.1: Is there a difference in the effectiveness of the utilisation of the various kinds of free-text data (e.g. bios, open answers)?
To answer this research question, we performed an ablation study of the different components of LoveBERT. We concluded that there is a drastic difference in performance between all individual components. The bio-bio component performed significantly worse than the ans-ans component, while the bio-ans component combining the two different sections yielded the highest performance on its own. Therefore, there is indeed a difference in the effectiveness of the utilisation of the different profile sections. Additionally, we concluded that combining different types of free-text data is most effective, outperforming both individual profile sections on their own.

RQ2: How can language models be used in a reciprocal recommender system, and which technique performs best?
The second research question was answered by comparing LoveBERT, our novel matchmaking algorithm, to existing approaches for RRSs and other NLP approaches. First of all, we concluded that embedding-based distances cannot effectively be used for predicting matches, but may be used to rank the similarity of profiles in suggestions. Secondly, we conclude that LoveBERT is the superior method in predicting likes. Unfortunately, we also conclude its performance does not hold up when predicting likes, being outperformed by the more traditional approaches, especially the Hybrid approach. However, we concluded our approach shows promise, as there exist specific scenarios where LoveBERT outperforms the Hybrid approach.

RQ2.1: Can a language model help circumvent the cold-start problem?
Finally, this research question was answered by splitting our test set in two distinct sets: previously seen and previously unseen users. We concluded that when considering most metrics, every approach takes a hit in performance when asked to make predictions for cold users. We confirm the existence of the cold-start problem. We also conclude that while our LM-based approach cannot circumvent the cold-start problem entirely, it is more resistant to the effects compared to the other approaches.

6.1. Future Work

Conducting this study has provided us with several insights into the world of matchmaking in RRSs. In this section, we share some recommendations for future research and avenues to explore.

- **Extending LoveBERT with additional profile section.** LoveBERT was designed with flexibility in mind. Therefore, we encourage further research exploring the possibilities of adding more types of user data in the model. An example could be more research into obtaining an accurate 'humour index', which could be incorporated as a component into LoveBERT. The additional components do not necessarily have to be LM-based. Anything which provides a prediction score or probability could be included.
- **Include LoveBERT in hybrid system to help with cold-start problem.** As LoveBERT appears to be more robust to the cold-start problem, we conjecture it could be included in some hybrid approach which relies more on the CB-NLP components earlier on while there is less data available. As more usage data comes in, the hybrid system could gradually rely more on the more performant classical approaches. Research on such a system could be very valuable.
- **Creating an open dataset for RRSs.** A huge problem in matchmaking research is that there are no public datasets available. This presents a huge opportunity for someone to create such a dataset. A main reason for this is likely the privacy-sensitive nature of the data. However, methods may exist to anonymise such data while still being useful for research purposes. It would also allow for better comparison between different RRS approaches by providing a consistent dataset people could evaluate on.
- **Utilising the cosine distance between user profiles.** We have shown that the distance between two profile embeddings cannot be effectively used as a prediction score. However, these distance metrics are still useful to identify profiles with extremely similar content. Further research on how to effectively use these distances in different approaches could aid in predicting likes and matches more effectively.

Bibliography

[1] J. Akehurst, I. Koprinska, K. Yacef, L. Pizzato, J. Kay, and T. Rej, “Ccra content-collaborative reciprocal recommender for online dating,” in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[2] I. Annamoradnejad and G. Zoghi, “Colbert: Using bert sentence embedding for humor detection,” *arXiv preprint arXiv:2004.12765*, 2020.

[3] J. Beel, S. Langer, A. Nürnberger, and M. Genzmehr, “The impact of demographics (age and gender) and other user-characteristics on evaluating recommender systems,” in *Lecture Notes in Computer Science*. Lecture Notes in Computer Science, 2013, pp. 396–400. DOI: 10.1007/978-3-642-40501-3_45.

[4] P. P. Biemer and L. E. Lyberg, *Introduction to survey quality*. John Wiley & Sons, 2003.

[5] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, “Recommender systems survey,” *Knowledge-Based Systems*, vol. 46, pp. 109–132, 2013, ISSN: 0950-7051. DOI: 10.1016/j.knosys.2013.03.012.

[6] J. Bobadilla, F. Ortega, A. Hernando, and J. Bernal, “A collaborative filtering approach to mitigate the new user cold start problem,” *Knowledge-Based Systems*, vol. 26, pp. 225–238, 2012, ISSN: 0950-7051. DOI: 10.1016/j.knosys.2011.07.021. [Online]. Available: http://oa.upm.es/15302/1/INVE_MEM_2012_123432.pdf.

[7] T. Brown *et al.*, “Language models are few-shot learners,” *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[8] X. Cai *et al.*, “Collaborative filtering for people to people recommendation in social networks,” in *AI 2010: Advances in Artificial Intelligence*. Springer Berlin Heidelberg, 2010, pp. 476–485. DOI: 10.1007/978-3-642-17432-2_48.

[9] H.-H. Chen, “Behavior2vec: Generating distributed representations of users behaviors on products for recommender systems,” *ACM Transactions on Knowledge Discovery from Data*, vol. 12, no. 4, pp. 1–20, 2018, ISSN: 1556-4681. DOI: 10.1145/3184454.

[10] D. Chicco, “Siamese neural networks: An overview,” in *Methods in Molecular Biology*. Methods in Molecular Biology, 2021, pp. 73–94. DOI: 10.1007/978-1-0716-0826-5_3.

[11] D. Chicco and G. Jurman, “The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation,” *BMC genomics*, vol. 21, no. 1, pp. 1–13, 2020.

[12] K. W. Church, “Word2vec,” *Natural Language Engineering*, vol. 23, no. 1, pp. 155–162, 2017.

[13] A. Conneau *et al.*, “Unsupervised cross-lingual representation learning at scale,” *CoRR*, vol. abs/1911.02116, 2019. arXiv: 1911.02116. [Online]. Available: http://arxiv.org/abs/1911.02116.

[14] M. De Gemmis, P. Lops, C. Musto, F. Narducci, and G. Semeraro, “Semantics-aware content-based recommender systems,” in *Recommender Systems Handbook*. Springer US, 2015, pp. 119–159. DOI: 10.1007/978-1-4899-7637-6_4.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv: 1810.04805*, 2018.

[16] G. Farnadi, P. Kouki, S. K. Thompson, S. Srinivasan, and L. Getoor, “A fairness-aware hybrid recommender system,” 2018.

[17] A. T. Fiore, L. S. Taylor, G. Mendelsohn, and M. Hearst, “Assessing attractiveness in online dating profiles,” in *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI ’08*, ACM Press. DOI: 10.1145/1357054.1357181.

[18] A. T. Fiore, L. S. Taylor, X. Zhong, G. A. Mendelsohn, and C. Cheshire, “Who’s right and who writes: People, profiles, contacts, and replies in online dating,” in *2010 43rd Hawaii International Conference on System Sciences*, IEEE, 2010. DOI: 10.1109/hicss.2010.444.

[19] S. M. Al-Ghuribi and S. A. Mohd Noah, “Multi-Criteria Review-Based Recommender System The State of the Art,” *IEEE Access*, vol. 7, pp. 169 446–169 468, 2019, ISSN: 2169-3536. DOI: 10.1109/access.2019.2954861.

[20] J. Gope and S. K. Jain, “A survey on solving cold start problem in recommender systems,” in *2017 International Conference on Computing, Communication and Automation (ICCCA)*, IEEE. DOI: 10.1109/ccaa.2017.8229786.

[21] W. Hong, S. Zheng, H. Wang, and J. Shi, “A job recommender system based on user clustering.,” *J. Comput.*, vol. 8, no. 8, pp. 1960–1967, 2013.

[22] N. W. Hudson and R. C. Fraley, "Partner similarity matters for the insecure: Attachment orientations moderate the association between similarity in partners personality traits and relationship satisfaction," *Journal of Research in Personality*, vol. 53, pp. 112–123, 2014.

[23] A. Kleinerman, A. Rosenfeld, F. Ricci, and S. Kraus, "Supporting users in finding successful matches in reciprocal recommender systems," *User Modeling and User-Adapted Interaction*, vol. 31, no. 3, pp. 541–589, 2021, ISSN: 0924-1868. DOI: 10.1007/s11257-020-09279-z.

[24] ——, "Optimally balancing receiver and recommended users' importance in reciprocal recommender systems," in *Proceedings of the 12th ACM Conference on Recommender Systems*, ACM. DOI: 10.1145/3240323.3240349.

[25] E. C. Klohnen and S. Luo, "Interpersonal attraction and personality: What is attractive – self similarity, ideal similarity, complementarity or attachment security?" *Journal of personality and social psychology*, vol. 85, no. 4, p. 709, 2003.

[26] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[27] J. Launay and R. I. Dunbar, "Playing with strangers: Which shared traits attract us most to new people?" *PloS one*, vol. 10, no. 6, e0129688, 2015.

[28] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International conference on machine learning*, PMLR, 2014, pp. 1188–1196.

[29] M. Lewis *et al.*, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," 2019.

[30] L. Li and T. Li, "Meet: A generalized framework for reciprocal recommender systems," in *Proceedings of the 21st ACM international conference on Information and knowledge management - CIKM '12*, ACM Press, 2012. DOI: 10.1145/2396761.2396770.

[31] X. Li and J. She, "Collaborative variational autoencoder for recommender systems," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM. DOI: 10.1145/3097983.3098077.

[32] X. Li, M. Wang, and T.-P. Liang, "A multi-theoretical kernel-based approach to social network-based recommendation," *Decision Support Systems*, vol. 65, pp. 95–104, 2014.

[33] B. Lika, K. Kolomvatsos, and S. Hadjiefthymiades, "Facing the cold start problem in recommender systems," *Expert Systems with Applications*, vol. 41, no. 4, pp. 2065–2073, 2014.

[34] R. Likert, “A technique for the measurement of attitudes.,” *Archives of psychology*, 1932.

[35] J. Liu, P. Dolan, and E. R. Pedersen, “Personalized news recommendation based on click behavior,” in *Proceedings of the 15th international conference on Intelligent user interfaces - IUI ’10*, ACM Press, 2010. DOI: 10.1145/1719970.1719976.

[36] R. Liu, Y. Ouyang, W. Rong, X. Song, C. Tang, and Z. Xiong, “Rating prediction based job recommendation service for college students,” in *Computational Science and Its Applications ICCSA 2016*. Springer International Publishing, 2016, pp. 453–467. DOI: 10.1007/978-3-319-42092-9_35.

[37] R. Liu, Y. Ouyang, W. Rong, X. Song, W. Xie, and Z. Xiong, “Employer oriented recruitment recommender service for university students,” in *Intelligent Computing Methodologies*. Springer International Publishing, 2016, pp. 811–823. DOI: 10.1007/978-3-319-42297-8_75.

[38] Y. Liu *et al.*, “Roberta: A robustly optimized bert pretraining approach,” *arXiv preprint arXiv:1907.11692*, 2019.

[39] S. Luo and G. Zhang, “What leads to romantic attraction: Similarity, reciprocity, security, or beauty? evidence from a speed-dating study,” *Journal of personality*, vol. 77, no. 4, pp. 933–964, 2009.

[40] I. Malkiel, O. Barkan, A. Caciularu, N. Razin, and O. K. N. Koenigstein, “Recobert: A catalog language model for text-based recommendations,” 2020.

[41] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.

[42] R. Mu, “A survey of recommender systems based on deep learning,” *IEEE Access*, vol. 6, pp. 69 009–69 022, 2018, ISSN: 2169-3536. DOI: 10.1109/access.2018.2880197.

[43] C. Musto, G. Semeraro, M. de Gemmis, and P. Lops, “Learning word embeddings from wikipedia for content-based recommender systems,” ser. Advances in Information Retrieval, Springer International Publishing, pp. 729–734, ISBN: 978-3-319-30671-1. DOI: 10.1007/978-3-319-30671-1_60.

[44] J. Neve and I. Palomares, “Hybrid reciprocal recommender systems: Integrating item-to-user principles in reciprocal recommendation,” in *Companion Proceedings of the Web Conference 2020*, ACM. DOI: 10.1145/3366424.3383295.

[45] J. Neve and I. Palomares, "Latent factor models and aggregation operators for collaborative filtering in reciprocal recommender systems," in *Proceedings of the 13th ACM Conference on Recommender Systems*, ACM, 2019. DOI: `10.1145/3298689.3347026`.

[46] ——, "Aggregation strategies in user-to-user reciprocal recommender systems," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, IEEE. DOI: `10.1109/smc.2019.8914362`.

[47] I. Palomares and et al., "Reciprocal recommender systems: Analysis of state-of-art literature, challenges and opportunities towards social recommendation," *Information Fusion*, vol. 69, p. 103, 2021, ISSN: 1566-2535 1566-2535.

[48] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: `http://www.aclweb.org/anthology/D14-1162`.

[49] L. Pizzato, T. Rej, T. Chung, I. Koprinska, and J. Kay, "Recon: A reciprocal recommender for online dating," in *Proceedings of the fourth ACM conference on Recommender systems - RecSys '10*, ACM Press. DOI: `10.1145/1864708.1864747`.

[50] I. Raeesi Vanani, L. Mahmoudi, S. M. J. Jalali, and K.-H. Pho, "Using text mining algorithms in identifying emerging trends for recommender systems," *Quality & Quantity*, 2021, ISSN: 0033-5177. DOI: `10.1007/s11135-021-01177-9`.

[51] C. Raffel *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," 2020.

[52] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[53] ——, "Making monolingual sentence embeddings multilingual using knowledge distillation," 2020.

[54] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens," in *Proceedings of the 1994 ACM conference on Computer supported cooperative work - CSCW '94*, ACM Press. DOI: `10.1145/192844.192905`.

[55] M. Á. Rodríguez-García, R. Valencia-García, R. Colomo-Palacios, and J. M. Gómez-Berbís, "Blinddate recommender: A context-aware ontology-based dating recommendation platform," *Journal of Information Science*, vol. 45, no. 5, pp. 573–591, 2019, ISSN: 0165-5515. DOI: `10.1177/0165551518806114`.

[56] M. J. Rosenfeld and R. J. Thomas, “Searching for a mate,” *American Sociological Review*, vol. 77, no. 4, pp. 523–547, 2012, ISSN: 0003-1224. DOI: 10 . 1177 / 0003122412448050. [Online]. Available: https : / / dx . doi . org / 10 . 1177 / 0003122412448050.

[57] M. J. Rosenfeld, R. J. Thomas, and S. Hausen, “Disintermediating your friends: How online dating in the united states displaces other ways of meeting,” *Proceedings of the National Academy of Sciences*, vol. 116, no. 36, pp. 17 753–17 758, 2019, ISSN: 0027-8424. DOI: 10 . 1073/pnas . 1908630116. [Online]. Available: https : //dx.doi.org/10.1073/pnas.1908630116.

[58] S. Shaikh, S. M. Daudpota, A. S. Imran, and Z. Kastrati, “Towards improved classification accuracy on highly imbalanced text dataset using deep neural language models,” *Applied Sciences*, vol. 11, no. 2, p. 869, 2021.

[59] M. Y. H. Al-Shamri, “User profiling approaches for demographic recommender systems,” *Knowledge-Based Systems*, vol. 100, pp. 175–187, 2016.

[60] G. Sielis, A. Tzanavari, and G. Papadopoulos, “A review of recommender systems: Types, techniques and applications,” in 2014, pp. 329–339, ISBN: 9781466658882.

[61] Y. Sun *et al.*, “Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation,” *arXiv preprint arXiv:2107.02137*, 2021.

[62] Y. Tay, A. T. Luu, and S. C. Hui, “Couplenet: Paying attention to couples with coupled attention for relationship recommendation,” 2018.

[63] N. Thakur, N. Reimers, J. Daxenberger, and I. Gurevych, “Augmented sbert: Data augmentation method for improving bi-encoders for pairwise sentence scoring tasks,” 2020.

[64] G. Tyson, V. C. Perta, H. Haddadi, and M. C. Seto, “A first look at user activity on tinder,” in *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, IEEE. DOI: 10 . 1109/asonam . 2016 . 7752275.

[65] A. Vaswani *et al.*, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, I. Guyon *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https : //proceedings . neurips . cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[66] V. Vehovar, V. Toepoel, and S. Steinmetz, “Non-probability sampling,” *The Sage handbook of survey methods*, vol. 1, pp. 329–45, 2016.

[67] D. Wang, Y. Liang, D. Xu, X. Feng, and R. Guan, “A content-based recommender system for computer science publications,” *Knowledge-Based Systems*, vol. 157, pp. 1–9, 2018, ISSN: 0950-7051. DOI: 10.1016/j.knosys.2018.05.001. [Online]. Available: https://doi.org/10.1016/j.knosys.2018.05.001.

[68] T. Wang and Y. Fu, “Item-based collaborative filtering with bert,” ser. Proceedings of The 3rd Workshop on e-Commerce and NLP, Association for Computational Linguistics, pp. 54–58. DOI: 10.18653/v1/2020.ecnlp-1.8. [Online]. Available: https://aclanthology.org/2020.ecnlp-1.8%20https://doi.org/10.18653/v1/2020.ecnlp-1.8.

[69] Y. Wang, M. Wang, and W. Xu, “A sentiment-enhanced hybrid recommender system for movie recommendation: A big data analytics framework,” *Wireless Communications and Mobile Computing*, vol. 2018, pp. 1–9, 2018, ISSN: 1530-8669. DOI: 10.1155/2018/8263704. [Online]. Available: https://doi.org/10.1155/2018/8263704.

[70] O. Weller and K. Seppi, “Humor detection: A transformer gets the last laugh,” *arXiv preprint arXiv:1909.00252*, 2019.

[71] *Writing survey questions*, Oct. 2021. [Online]. Available: https://www.pewresearch.org/our-methods/u-s-surveys/writing-survey-questions/.

[72] P. Xia, B. Liu, Y. Sun, and C. Chen, “Reciprocal recommendation system for online dating,” in *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, ACM. DOI: 10.1145/2808797.2809282. [Online]. Available: http://www.cs.uml.edu/%7Ebliu/pub/ordec.pdf.

[73] P. Xia, S. Zhai, B. Liu, Y. Sun, and C. Chen, “Design of reciprocal recommendation systems for online dating,” *Social Network Analysis and Mining*, vol. 6, no. 1, 2016, ISSN: 1869-5450. DOI: 10.1007/s13278-016-0340-2.

Appendix A

Hyperparameters

Table A.1 describes the hyperparameters for the fine-tuning of the different LoveBERT components. We select the best hyperparameter values based on experimentation with different values. Results were particularly sensitive to the Adam ϵ values and the learning rate of the classifier layer.

Hyperparam	**bio-bio**	**ans-ans**	**bio-ans**
Batch size	8	6	6
Max sequence length	512	512	512
Peak learning rate	4e-6	6e-6	4e-6
Learning rate decay	linear	cosine	linear
Learning rate classifier layer	1e-4	2e-4	1e-4
Warmup ratio	0.1	0.1	0.1
Weight decay	0.01	0.01	0.01
Adam β_1	0.9	0.9	0.9
Adam β_2	0.999	0.999	0.999
Adam ϵ	1e-6	1e-6	1e-6
Max epochs	8	8	8

Table A.1: Hyperparameters for fine-tuning the different LoveBERT components.