Socializing with In-Vivo Intelligence in the Wild

***Abstract*—Social apps help digitally establish and maintain human relationships. As an instance of social-computational integration, social apps highlight the need for seamlessly bringing the human and the computer together in mobile software design. As intelligence is fundamental to the human, a streamlined social-computational interface naturally calls for intelligence from the computer. In this paper, we describe the design and the real-world deployment of MEETCITY, a novel friendship app incarnating our intelligence-by-design approach. Unlike existing apps that rely on geo-proximity and/or static profile information to rank, group, and connect users, MEETCITY in addition models the dynamic user behavior through continuous online learning guided by hierarchical clustering, and applies the result of learning to recommend meetings. Unique to MEETCITY, the features adopted by our online clustering approach are aligned with established personality traits, such as openness, conscientiousness, extraversion, and agreeableness. MEETCITY is a living app that has been in use by thousands of users in the past 3 years. We evaluate the effectiveness of our in-vivo intelligence approach through an in-depth analysis of real-world data collected from MEETCITY users in a large metropolitan area.**

I. INTRODUCTION

In mobile software engineering, social networking applications, or *social apps* for short, represent an important family of mobile apps with distinct features and requirements. Whereas all mobile apps are necessarily designed to help the human *perform tasks*, social apps have the preeminent role of helping the human *be human*. Unlike other families of apps where software quality can often be evaluated in more conventional metrics — *e.g.*, speed for gaming apps, or precision for navigation apps — social apps are faced with a unique and elusive design goal: *a seamless integration of social and computational spheres*.

As intelligence is fundamental to the human, a streamlined social-computational interface naturally calls for intelligence from the computer. It is our belief that a symmetry in intelligence between the human and the computer represents an ideal for seamless social-computational integration.

Bringing machine learning into social app development is an emerging research direction and industrial trend. Make no mistake: social apps such as Facebook, Tinder, and Badoo have long used machine learning algorithms to cluster and classify users based on a wide range of static profile information and geo-location information. The deployment of these algorithms can often lead to limited artificial intelligence (AI) in social apps. For example, an established feature in existing social apps is a recommendation subsystem — be it for recommending friends, dates, or products — guided by profile-driven clustering and classification.

A. In-Vivo Intelligence

In this paper, we advocate the philosophy of *in-vivo intelligence* in the design of social apps, *i.e.*, continuously learning from the *dynamic* behavior of users themselves to understand their *personality* traits, and applying the knowledge gained through learning to socialization.

Take the use scenario of meeting arrangement for example. A traditional app, *e.g.*, MeetUp, generally plays the role of an "information aggregator": participants collaboratively provide the meeting essentials — time, location, invited users — and the app presents such information in user-friendly manners *e.g.*, through reminders and maps. The fundamental drawback of such an "aggregator" design is it goes against the true spirit of human-computational integration: app users are treated as static, mechanical information providers with no fundamental difference from someone sitting in front of a Desktop computer decades ago. Friendship apps, with Tinder and Match as examples, go one step further by taking into account of dynamic geo-location information. This is in sync with typical use scenarios of this family of apps, where new friendships or dates may be established between previously unrelated users. With regards to intelligence, such apps recognize human beings are mobile, but human behavior remains off center in the supposedly *human*-centric computing. Social apps such as eHarmony do consider personality traits to determine similarity between users, but they rely on static data from users, making them in essence no different from an information aggregator.

Our design philosophy of in-vivo intelligence is a systematic rethinking of bringing AI to the social-compuational interface in social apps, consisting 3 ideas.

First, we believe the *dynamic user behavior* should be continuously learned. In friendship apps, a user leaves behind a large amount of behavior data — how she sends meeting requests to others, and how he reacts to requests from others, *etc.* — and such data may provide vivid insight to the user, which in turn may provide guidance to use scenarios of the app, such as determining which users are more likely to be similar.

Second, learning should be designed to reveal the essential *personality traits* of the users. In psychology, an established classification of human personality is the Big 5 Personality [9, 12]: *openness*, *conscientiousness*, *extraversion*, *agreeableness*, and *neuroticism*. Inspired by this classic taxonomy, in-vivo intelligence is a first step toward "digital psychology," designing AI around the well-studied personality traits as features.

Third, more "intelligent" use scenarios should be designed

for social apps. For instance, in addition to finding out which users are likely to be compatible based on AI, social apps should be designed with AI-guided use scenarios. For example, a meeting app may automatically recommend a meeting between compatible users by suggesting a coffee shop that happens to be close to both, and a time based on the participants' past habits.

B. The MEETCITY App

The centerpiece of this paper, MEETCITY, is a friendship app that embodies our philosophy of in-vivo intelligence. MEETCITY is endowed with an online hierarchical clustering algorithm to find similarity among users, and harvest such information to intellgiently arrange meetings. The most innovative aspect of MEETCITY is it continuously tracks the dynamic behavior of each user related to meetings — such as how often a user accepts meetings, and how punctual the user is to attend meetings, and how the user is rated after the meetings — and connect these behavior traits to 4 of the 5 Big 5 personality traits: openness, conscientiousness, extraversion, and agreeableness. This is an exciting frontier toward digital psychology in mobile app design: how artificial intelligence embraces established frameworks in psychology for social app design.

More importantly, MEETCITY is more than a proof of concept for in-vivo intelligence. The app has been available for download on app stores for the last 3 years, with growing momentum for user base expansion. It has been used to arrange 26,000 real-world meetings from around 5,000 active users. This paper draws from the real-world experience we have accumulated through maintaining the app and interacting with real-world users of MEETCITY, leading to an evaluation of the idea of in-vivo intelligence in the wild. As machine learning algorithms are fundamentally data-driven, it is our belief that the MEETCITY project provides a unique vantage point where software engineering meets machine learning: MEETCITY is a live, evolving software lab to draw wisdom on intelligent software engineering.

Our data analysis show that each of the personality features MEETCITY learns over has notable impact on clustering. Together, they help refine the differentiation among users significantly: a 41% increase in the number of clusters compared with clustering over traditional features such as geo-proximity and self-reported profile data. Despite the introduction of more features into clustering, the clustering quality as exhibited by silhouette values remains stable. From the perspective of end users, the post-meeting feedback ratings MEETCITY are significantly higher for meetings recommended by the in-vivo intelligence engine. On a 1-5 scale where 5 is better, the meetings recommended through MEETCITY clustering has an average rating 2 points higher than those of the non-recommended meetings.

To place MEETCITY in context, Tinder [16], the industry standard in the category of friendship apps, reports 4 successful dates out of every 53 meeting arrangements, with a successful meeting rate of 7.6%. In contrast, our historical data shows MEETCITY yields a successful meeting rate of 11.8%. In addition, the meeting arrangement on Tinder is a manual and significantly longer process involving many rounds of suggestions from one party and confirmation from the other. Thanks to our innovation on intelligent use scenarios, the same task in MEETCITY is largely automatic: the *when* and *where* of the meeting is automatically suggested by the app. The time needed for arranging a MEETCITY meeting averages at 5-20 minutes.

C. Contributions

In-vivo intelligence reflects a fundamental rethinking in social-computational integration, and MEETCITY represents a concrete instance of unifying AI and software engineering in mobile software design. From a real-world perspective, the MEETCITY experience demonstrates how significant room for technical innovation still exists in an established family of mobile apps. To the best of our knowledge, MEETCITY is the first living social app that systematically embraces AI and psychology by design.

This paper makes the following contributions:

- It highlights the need for in-vivo intelligence to streamline the boundary of the social-computational ecosystem in software engineering.
- It describes the design of a clustering algorithm guided by novel features inspired by psychology.
- It reports a real-world evaluation of the MEETCITY app, with in-depth data analysis on the impact of in-vivo AI in social app design.

II. BACKGROUND

This paper spans several previous largely disconnected areas in research, which we now briefly summarize.

A. Clustering Algorithms

Clustering [5] is a multivariate analysis used to group similar *data items* (or *objects* or *observations*) together in the same group, known as a *cluster*. Unlike supervised learning methods (*e.g.*, classification and regression), clustering-based learning generally does not use any label information. Instead, data are represented as vectors, each of which consists of a set of *variables* (or *features*), and clusters can be formed via the similarity between vectors.

Clustering algorithms may either be hierarchical or non-hierarchical. In non-hierarchical clustering, such as the standard K-means algorithm, all clusters are "peers." Hierarchical clustering on the other hand organizes clusters on multiple levels, while each level can be intuitively viewed as a partitioning of the data items of the previous level. Hierarchical algorithms are generally iterative, attractive for data sets that may experience changes over time, a scenario applicable to MEETCITY.

In hierarchical clustering, one may adopt either an *agglomerative* or *divisive* method to build the cluster hierarchy. Agglomerative hierarchical clustering is a "bottom-up" approach

where each data item starts in its own cluster. "Similar" clusters are subsequently merged. Divisive hierarchical clustering, on the other hand, is a "top-down" approach where all data items start in one cluster, which are subsequently split into different clusters when they are "dissimilar" to each other.

Dissimilarity between clusters is defined through the *distance* between (the feature vectors of) clusters. There are a wide variety of methods to compute the distance, called *linkage methods*. Often used linkage methods include *single linkage* (the shortest distance between data points in two clusters), *complete linkage* (the longest distance between data points in two clusters), *average linkage* (the average distance between data points in two clusters), and *centroid linkage* (the distance between cluster centroids). In this paper, clustering is performed according to another commonly used linkage method called the *Ward linkage* [3]. This linkage choice optimizes over the minimization of the total in-cluster variance.

B. Personality Traits

Big 5 Personality [9] is an influential taxonomy for personality traits. The five factors, Openness, Conscientiousness, Extraversion, Agreeableness, and Neuroticism are also known as forming the OCEAN model. The scientific basis of Big 5 Personality was extensively surveyed [12], and numerous statistical models based on personality questionnaires exist for this taxonomy.

Even though rigorous definitions for the 5 factors have long been debated, there is a general consensus on their characteristics. *Openness* generally refers to the degree of a personal preference for a variety of activities over a strict routine. *Conscientiousness* indicates a high level of dependability, showing self-discipline and acting dutifully. *Extraversion* is a trait for outgoing behavior, aligned with our intuition of being "sociable" or "popular." *Agreeableness* is a tendency to be compassionate and cooperative rather than suspicious and antagonistic towards others. Finally, *Neuroticism* is a tendency for sensitiveness and anxiety. The MEETCITY clustering algorithm focuses on the first 4 personality traits.

Before we proceed, we wish to address the important design concern of privacy. Our design of learning personality traits is based on the behavior of the MEETCITY user while she goes through the intended use scenarios defined by the app. For example, MEETCITY models openness through monitoring how a user reacts to meeting requests from others. At the first glance, this may sound disappointingly simplistic to psychologists, to whom openness may carry much richer semantics. Indeed, this disappointment — if viewed as one — technically results from permission-based app design, where an app such as MEETCITY cannot access, *e.g.*, user emails to analyze personality. In our view however, the *limited* personality characterization precisely stands for what we believe in privacy: in-vivo intelligence has nothing to do with profiling user personality in their private lives; what matters is the personality one exhibits while using the app in intended use scenarios.

C. Social Apps

Social apps are mobile apps with the general goal of enabling, facilitating, and maintaining social networks and promoting social media. In popular app stores, social apps represent a dominating category of apps thanks to the wide adoption of Apps such as Facebook, Instagram, and Wechat. A subcategory of social apps, known as meeting apps (or friendship apps or dating apps) specialize in starting new interactions, as opposed to Facebook-style of social apps optimized for maintaining interactions. Tinder, the leader in this category, has 57 million users as of 2018 [11].

Concretely, MEETCITY belongs to the subcategory of meeting apps. The philosophy of in-vivo intelligence, however, vibrates through the general category of social apps: behind the screen of the app is a living person, and a better understanding of the person's dynamic behavior and personality may improve the user experience of the app.

III. THE MEETCITY DESIGN

A. High-Level App Design

MEETCITY supports a wide variety of common features for social meeting apps, including user account and profile management, profile ranking, mapping and direction support, text-based chatting, privacy enforcement, and multi-media support. The most relevant features to this paper are two folds: how profiles are ranked/recommended, and how meetings are arranged.

Unlike traditional friendship apps, MEETCITY does not recommend meetings solely based on geo-proximity or static profile information, such as self-reported information in profiles (age, preferences, *etc*). The recommendation is in addition guided by a clustering engine where each observation is represented by a six-feature vector:

$$[G, P, O, C, E, A] \tag{1}$$

where G represents the geo-location, P represents aggregated self-reported profile data, and the last 4 features are 4 personality traits that we will discuss shortly. All features are normalized.

Another unique feature of MEETCITY is intelligent meeting arrangement. After users indicate a mutual interest in the meeting, MEETCITY will automatically suggest a location considering the geo-location of the users, and a time based on the patterns of past suggestions from the users. After the meeting is agreed on, MEETCITY will continue tracking the remaining countdown to the meeting, monitor the punctuality of users for the meeting, and request the feedback from users after the meeting.

B. Clustering Algorithm

In MEETCITY, we adopt *online*, *agglomerate*, *hierarchical* clustering based on *Ward linkage*.

The *online* nature of our algorithm is fundamental to meeting apps: in the mobile and dynamic environment, users

not only change geo-locations, but also exhibit dynamic behavior. The latter is critical in understanding the essence of MEETCITY's notion of in-vivo intelligence: as time goes by, the behavior of a user also *evolves*: she may send out more invites, accept more invites, make it to more meetings on time (or late), cancel meetings, *etc*.

Our design choice of *agglomerate* clustering is also natural to the use cases we wish to model. Recall that agglomerate clustering is a "bottom-up" clustering algorithm, as opposed to the "top-down" divisive clustering. In meeting apps, users frequently change their status, such as from "active" to "inactive" to "offline." If "top-down" clustering were to be applied, the status change of every user would involve global re-clustering, a computation-intensive task. This is especially true for supporting geo-location as a feature in clustering, as users move frequently at very short time intervals, *e.g.*, when they are in moving vehicles. Agglomerate clustering on the other hand exhibits strong stability. The fundamental *locality* exhibited in our features (from geo-locations to other social behavior modeling features discussed below) allows reclustering to follow the bottom-up matter from the leaf node of the changing cluster, and ripple up on the clustering hierarchy. In a nutshell, agglomerate clustering naturally addresses MEETCITY's need for efficient, incremental reclustering.

The "bottom-up" agglomerate clustering is *decentralized* in nature: each user starts as a cluster of her own, and clusters may be merged as needed. This decentralized style in social app design also explains why MEETCITY embraces *hierarchical* clustering at its heart. For example, a popular alternative for clustering would be K-means. That design however requires MEETCITY to have a more centralized view of clustering, such as how many clusters there should be; this fixed *a priori* size goes against the dynamic nature of social networks in meeting apps.

Our concrete choice of the objective function in agglomerate hierarchical clustering follows the Ward's method [3]. This linkage choice optimizes over the minimization of the total in-cluster variance. This is particularly attractive for MEETCITY, as we wish to suggest meetings and dates among *like-minded* users.

The implementation of our algorithm does not veer far from the standard approach, whose specification we omit. The algorithm is initialized with one cluster per user, with distance defined as the squared Euclidean distance. The algorithm iteratively finds a cluster pair that leads to the minimum increase in total in-cluster variance after merging. Our incremental algorithm follows the same, except that reclustering starts from the cluster(s) that contain the user(s) whose representative feature vectors have experienced change.

C. In-Vivo Features

We now take a closer look at how MEETCITY constructs the 4 features reflecting the personality traits.

1) Openness (O): The first important personality trait we define is openness, through *normalized meeting accepts*:

$$O = \frac{Ac}{Rr} \tag{2}$$

where *Ac* is the number of invites accepted by the user, and *Rr* denotes the number of received meeting requests of the user.

One interesting design question is why openness is defined based on meeting *invitee*'s willingness to accept a meeting instead of the *inviter*'s willingness to initiate a meeting. Wouldn't the act of initiating a meeting also indicate an openness (to new experience) too? Here, a key observation to meeting apps is that the role of meeting invitee and that of meeting inviter are *asymmetrical*: the average invites a user *sends* significantly outnumbers the average accepts a user initiates based on our empirical data. In other words, the behavior of the invitee has a stronger influence on whether the key use scenario of the app can happen: a meeting can only be set up when a user accepts a request.

2) Conscientiousness (C): In MEETCITY, we define conscientiousness as the *normalized punctuality*, as:

$$C = \frac{Ac - Mc \cdot \rho - Ml \cdot \upsilon}{Ac} \tag{3}$$

where *Mc* is the number of cancelled meetings, *Ml* is the number of late meetings, and constants ρ and υ are coefficients between 0 and 1. *Mc* and *Ml* together highlight how MeetCity values *punctuality* in meeting arrangement. In other words, a most conscientious user should ideally go to all meetings she accepts, and appear at each meeting on time.

MEETCITY compares the geo-location of the user at the time of the meeting against the location agreed upon (often suggested by the app itself through intelligent meeting arrangement) to track *Mc* and *Ml* values.

3) Extraversion (E): In the human society, only a subset of people are leaders, influencers, or Kevin Bacon. In social apps, users are "not born equal" either: some users are indeed more "popular" than others.

The specific notion of extraversion is indeed app-specific: a "popular" Twitter user may imply more followers, whereas a "popular" Instagram user may post more photos. In our opinion, an app-neutral notion of extraversion should rely on how much the behavior of a specific user may influence the (primary) use scenario of the app. In other words, MEETCITY is less concerned with how extravert a user is in the real life; what matters is how the user has influenced the dynamics of the app. In MEETCITY, we define extraversion as the *adjusted meeting requests*, as:

$$E = \frac{Rr}{Ps} \tag{4}$$

where *Rr* denotes the number of received meeting requests of the user (as before), and *Ps* denotes the number of profile showings of the same user.

Recall from the discussion on openness that the role of the meeting invitee and that of the meeting inviter are asymmetrical. As a result, extraversion is defined from the invitee's perspective as well. This feature as defined shows how popular

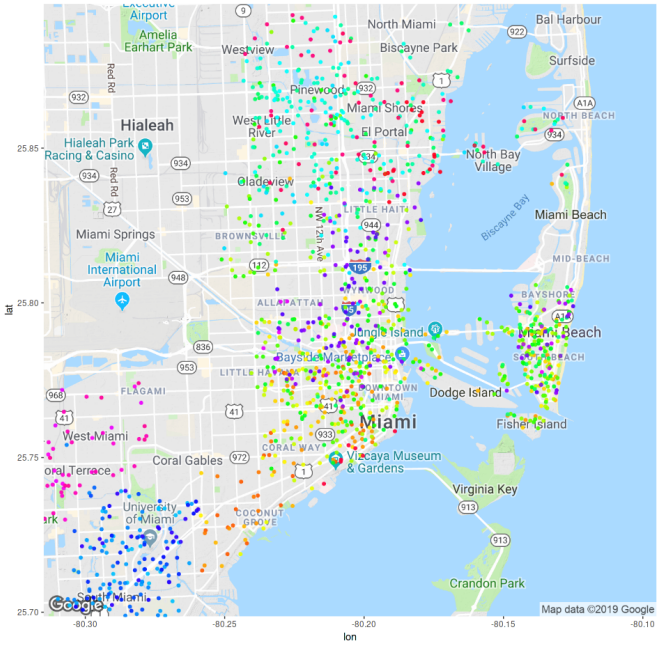


Fig. 1: MEETCITY Clustering with a Map Overlay (Each cluster is represented by a distinct color with similarity threshold of 0.8.)

a user is in the eyes of other users, rather than how active a user reaches out to others. If this invitee persspective may sound counter-intuitive at the first glance, consider the human society: we do not necessarily call a person "an extrovert" because he/she would start the conversation in every social interaction, but often because he/she is well connected in the society.

The distinction between Extraversion and Openness reflects the duality between the necessary condition and the sufficient condition for making the key use scenario happen: scheduling a meeting.

4) Agreeablness (A): Agreeableness in MEETCITY is defined as the linear combination of several (normalized) feedback ratings received by the user:

$$A = Fo \cdot \alpha + Fp \cdot \beta + Fa \cdot \gamma \tag{5}$$

where *Fo* is the normalized rating on overall experience for meetings, *Fp* is the normalized rating on personality, and *Fa* is the normalized rating on appearance. Constants α, β, γ are coefficients defined by MEETCITY. This interesting use of user feedback demonstrates an advantage of human-centric computing: the input from users may directly provide assistance to the design of the computational component.

IV. EVALUATION

In this section, we demonstrate the effectiveness of our approach through an in-depth analysis of data collected through deploying MEETCITY in the field. For meeting apps, it is uncommon for users from different geographic regions — such as in different cities — to arrange meetings, let alone

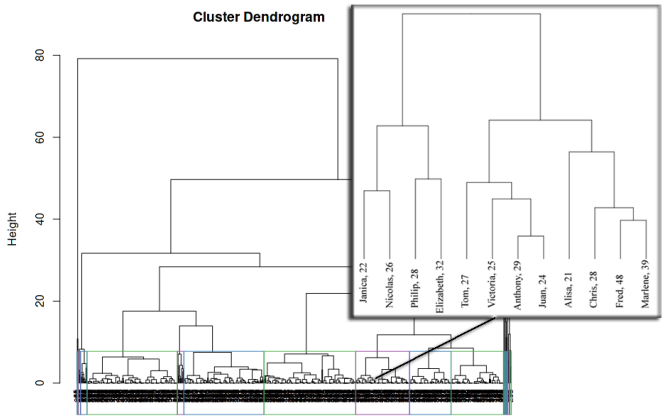


Fig. 2: MEETCITY Clustering: Dendrogram

meetings that rely on intelligent arrangement. As a result, MEETCITY's clustering algorithm is applied separately to each metropolitan area, good news for scalability. The data reported in this section were collected for the metropolitan center of Miami, covering a geographical area of 186.92 square miles, shown in a map in Figure 1. The dataset contains 1342 items, *i.e.*, MEETCITY users.

Figure 2 provides a visual illustration of the algorithm at work. It shows the dendrogram where clusters in the corresponding map are created in the process of hierarchical clustering. As the dendrogram suggests, one may build a hierarchy all the way with each leaf only representing a single data item. In practice, we set the threshold for cluster merging at the cluster distance of 0.2 (a similarity of 0.8). Visually, this means if data items on a subtree of the dendrogram have a similarity of 0.8 or above, we consider them belonging to the same cluster.

A. *Real-World* MEETCITY *Results*

As shown in Figure 1, MEETCITY is able to identify 48 clusters. On one hand, locations remain a factor in clustering, in that data items in proximity frequently belong to the same cluster. This is consistent with the philosophy of meeting apps where location proximity is an important consideration for users. On the other hand, observe that it is frequent to have users close to each other but belong to different clusters. This is particularly true for the downtown area around Bayside Marketplace on the map, where a large number of clusters cover a densely populated area less than 1 square mile. This demonstrates how MEETCITY is able to rely on in-vivo intelligence to build clusters in a diverse population.

The map also demonstrates some distinctive traits of different neighborhoods. In the downtown Bayside Marketplace area, despite the large number of clusters, no cluster dominates. The University of Miami area and the South Beach area however exhibit dominance in 1 or 2 clusters. We think this interesting phenomenon may indicate different levels of diversity in terms of personality traits. The South Beach area for instance, with its renowned beaches, may disproportion-

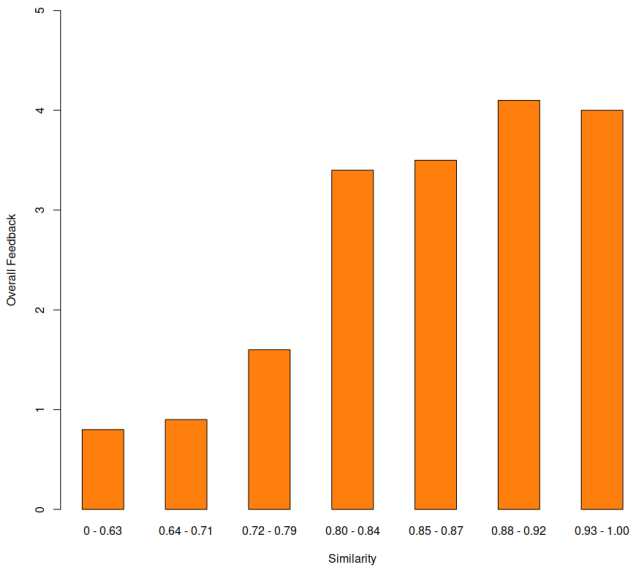


Fig. 3: An *a posteri* Feedback Analysis (The X-axis is demarcated by the ranges of cluster similarity, and Y-axis is the average user rating.)

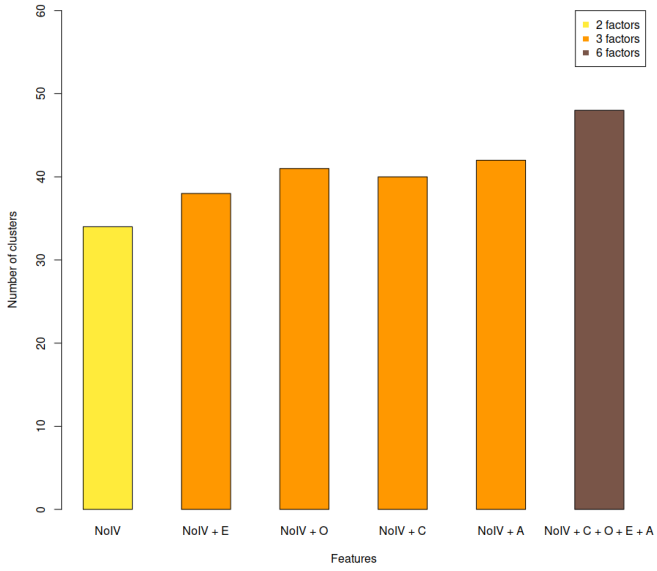


Fig. 4: The Number of Clusters w.r.t. Feature Sets (The X axis is the choice of features for clustering, and the Y axis indicates the number of clusters. All experiments with similarity threshold set at 0.8. Here `NoIV` means the traditional clustering approach based on location/proximity data and profile data. `NoIV+XX` means a variant of our clustering algorithm where feature vector includes the location/proximity, profile, and `XX`, with the latter being one or all of the 4 in-vivo features.)

ately attract users of certain personality traits. In contrast, Bayside Marketplace intuitively has appeal to users from "all walks of life." The clustering results are consistent with our understanding of these neighborhoods, a cross-validation on our approach. Taking this macro view, the MEETCITY may offer a glimpse of a "personality profile" of a city.

B. A Posteri Feedback Analysis

One "end-user" approach to evaluate the effectiveness of our approach is to resort to user feedback. As we described in the previous section, we allow users to rate their experience after the meetings. We conduct an *a posteri* analysis to correlate user ratings with user similarity as computed by our algorithm, with the results shown in Fig. 3. Concretely, for each meeting that has happened, MEETCITY requests the participating users to provide ratings between 1 and 5, the higher the better experience. Fig. 3 averages rating data from all meetings where at least one user has provided a rating, with data "binned" according to the range of similarity computed by the clustering algorithm. Note that even though MEETCITY can use its AI engine to *recommend* meetings between users, it does not preclude users from arranging meetings without recommendation. As a result, some meetings may happen between two users who are not considered "similiar" according to the clustering algorithm of MEETCITY.

Fig. 3 shows that meetings recommended by MEETCITY (with a similarity threshold of 0.8) show significantly higher ratings than those without recommendation. For meetings whose similarity is above 0.8, the ratings are above 3, whereas those meetings below a similarity of 0.8 have average ratings below 2. In a 5-scale rating system, where 3 or above generally refers to a positive experience and 2 or below refers to a negative experience, this *a posteri* analysis demonstrates why our design of clustering-based design matters. More remarkably, observe that location (proximity) data and profile data are readily available to all MEETCITY users, with or without the MEETCITY recommendation. In other words, the users with low ratings (the 3 bars in the Figure) could very well have already taken advantage of location proximity or profile data similarity informally to guide their decision for a meeting. In other words, the gain in user ratings as similarity increases may largely result from our personality-guided clustering design.

C. In-Vivo Features vs. Traditional Features

We now analyze the role of in-vivo features in our clustering-based analysis, especially on what difference they make in addition to traditional features such as location and profile data in clustering algorithm design.

Fig. 4 presents how the inclusion of in-vivo features can help refine the clusters. With traditional features only (which is called the `NoIV` approach in the figure), the hierarchical clustering algorithm is able to produce 34 clusters, whereas when 4 in-vivo features are additionally introduced, the clustering algorithm produces 48 clusters. More clusters imply that users are placed in smaller clusters on average, and in meeting apps, differentiation is good news. To see why, consider the extreme opposite where all users are clustered

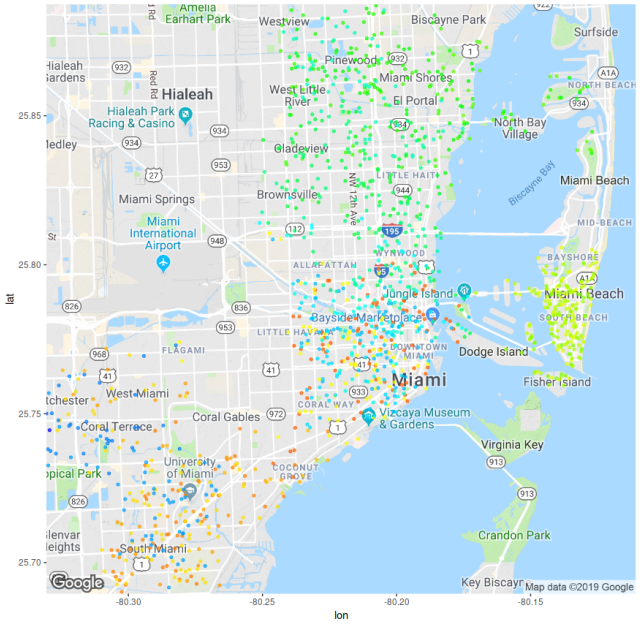


Fig. 5: Traditional Clustering with a Map Overlay (Here, clustering is based on proximity and profile data.)

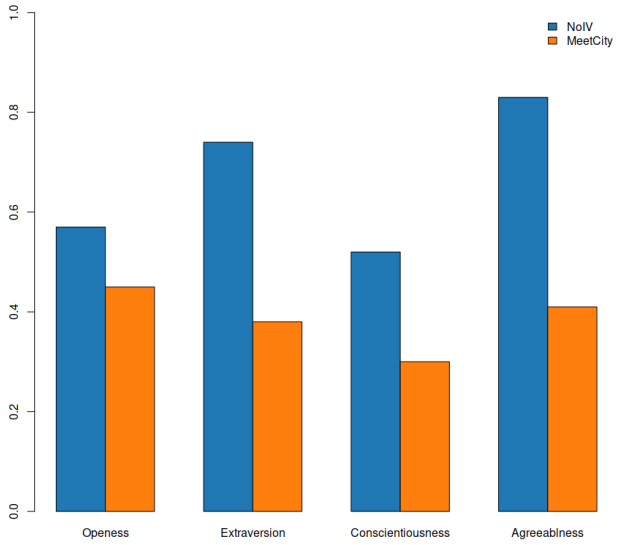


Fig. 6: Intra-Cluster Distance: the Impact on in-vivo Features

in one; meeting arrangement in that scenario would be solely random assignment.

Visually, the impact of refined clusters can be shown by comparing the map we introduced in Fig. 1 (the MEETCITY approach) with the one in Fig. 5 (the NoIV approach). Observe that in key areas such as South Beach, the approach with no in-vivo features ends up clustering all users to one. This demonstrates the limitation of clustering solely based on geo-locations and self-reported profile data. In concentrated areas where the demographics are more homogeneous, neither geo-locations nor self-reported profile data can effectively help differentiate users. Another area that clustering without in-vivo features fares poorly in is north Miami, the area around Little Haiti, a neighborhood with a large population of Caribbean descent. Observe that a large number of users are grouped into the same cluster. The primary cause is that self-reported profile data may exhibit higher similarity. This similarity however is only superficial: when in-vivo features are considered, the result of MEETCITY (see Figure 1) demonstrates the population indeed falls into a variety of clusters, as to be expected.

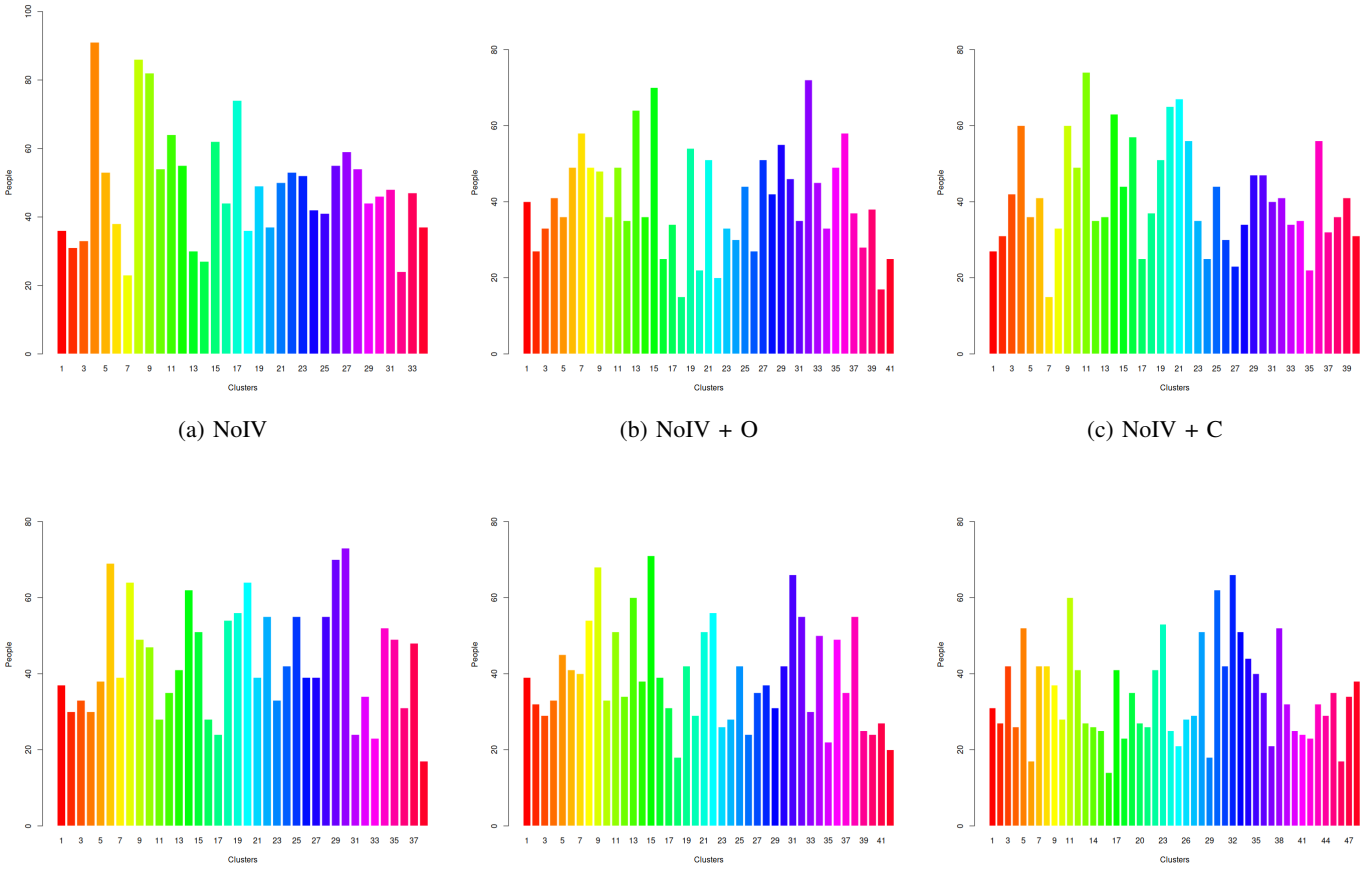
To gain an in-depth understanding on the impact of clustering with in-vivo features, we further analyze the intra-cluster distance for each in-vivo feature, with results shown in Fig. 6. Each group of bars represents (the average distance of) an individual in-vivo feature, such as openness. The first bar in each group represents the NoIV clustering, whereas the second bar represents the MEETCITY clustering. This figure shows that the MEETCITY clustering algorithm can reduce the average distance of openness, extraversion, conscientiousness, and agreeableness. This result naturally derives from our algorithm design — and the trend may not be surprising by itself — but the high-level message is important: MEETCITY *brings together the users who have more similar personality traits*.

D. Feature-Specific Analysis

We next analyze the individual impacts of in-vivo features, focusing on how each personality trait may contribute in clustering.

Fig. 7 shows the cluster size distribution when individual features are used for clustering. Recall that if we only use traditional features, there are 34 clusters (Fig. 7a) and when all in-vivo features are used in addition to traditional features, there are 48 clusters (Fig. 7f). If openness, conscientiousness, extraversion, and agreeableness are added to the traditional features individually, the clustering algorithm will generate 41, 40, 38, 42, clusters respectively. Each feature appears to contribute in differentiating users with comparable impact, and their contribution appears to be composable. Among the 4 features, extraversion is slightly less differentiating. We speculate this may have to do with the nature of social networks. Recall that extraversion is defined through normalized meeting requests, a form of popularity. It is widely known that social networks as a graph are scale-free [4]: popularity is dictated by power law due to preferential attachment. As a result, extraversion is often represented as a large value (for some popular users) or a small value (for other users), and the distribution is far from uniform. As a feature for clustering, its effect is thus more polarizing than differentiating users into numerous clusters of various shades of extraversion.

Fig. 7 also demonstrates that cluster sizes for clustering variants with in-vivo features (all subfigures except the first one) fall into the 20-70 range without overly small or overly large clusters. The relative balance in size is an attractive



(a) NoIV (b) NoIV + O (c) NoIV + C

(d) NoIV + E (e) NoIV + A (f) MeetCity

Fig. 7: Cluster Size Distribution

trait in practice, implying each user has a reasonable pool of potential meeting partners. In contrast, clustering solely based on traditional features (the first subfigure) leads to 3 clusters over the size of 80, with one approaching 100.

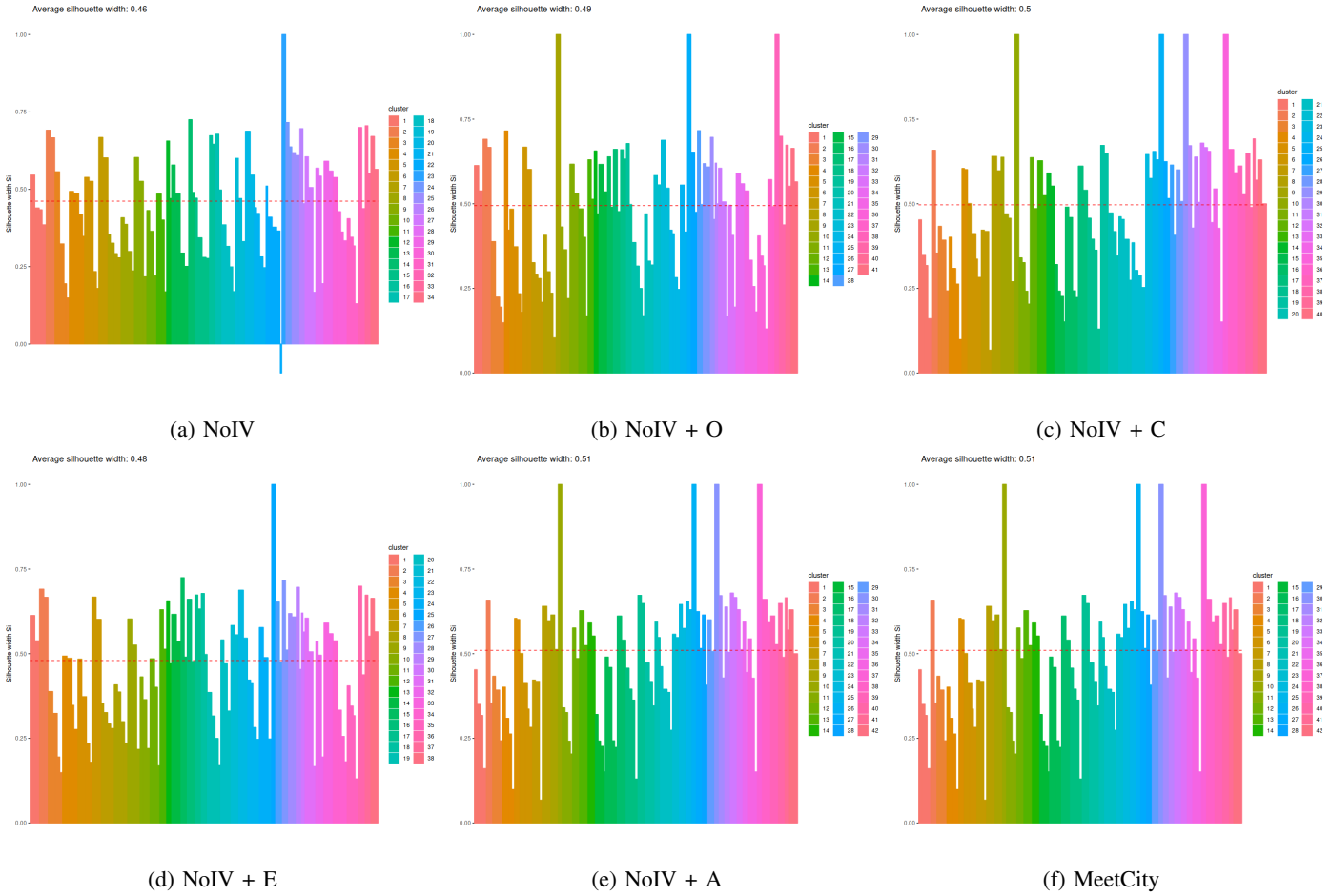
In clustering algorithms, the introduction of additional features intuitively leads to more differentiation in data items. A common concern is whether additional differentiation would lead to the deterioration of the clustering quality. In other words, more clusters can be found, but do they still abide by the principle of clustering: data within the same cluster are similar, but those from different clusters are different? To further study the quality of clustering, we introduce Fig. 8, silhouette diagrams widely used for monitoring clustering quality. A *silhouette value* is a metric to indicate how closely related data items are within the cluster *and* how different clusters are separated from each other. A higher silhouette value (between -1 and 1) indicates better clustering. From Fig. 8a, observe that the average silhouette width for clustering based on traditional features is 0.46. With additional in-vivo features, the silhouette value stays stable, and a mild improvement to 0.51 when all in-vivo features are included (Fig. 8f).

V. Lessons Learned

Three years into running, MeetCity is a rewarding experience of bringing AI and software engineering together in the wild. In this section, we share some lessons learned from this experience, in the hope that they may shed light to the real-world evolution of mobile software systems.

a) Algorithm Plane vs. Data Plane in AI: MeetCity may be among the first meeting apps to systematically embrace AI, but there remains a gap between the algorithm core of MeetCity and the frontier of machine learning. For example, we are often asked about applying deep learning to MeetCity, a potentially fruitful future direction in its own right. Based on our existing experience however, the effectiveness of AI-based design in meeting apps is less driven by the algorithm choice, but more driven by the *types of data* fed to the algorithms. Indeed, earlier versions of MeetCity did not have all 4 in-vivo features, and the progression appears to be aligned with the improvement of overall user experience through the app evolution.

b) Rich Features, Simple Interface: MeetCity is a full-fledged social app with a wide range of functionalities. The question at concern is how to *present* this rich set of features



(a) NoIV (b) NoIV + O (c) NoIV + C

(d) NoIV + E (e) NoIV + A (f) MeetCity

Fig. 8: Silhouette Diagram

to users. In an initial version of MEETCITY, we opted for a tab-based design where main features are presented to the user all at once, with separate tabs representing, *e.g.*, a mapped view of nearby users, a summary of incoming and outgoing date invitations, recent text messages, and a list of upcoming and finished dates. This design did not fare well in the initial launch. As much as users were excited about the AI-guided meeting app, feedback indicated that they felt their experience was highly fragmented, often relying on notifications or familiarity to navigate the complex user interface. Users inexperienced with social apps in general were disproportionately negative with that design.

Learning from this experience, the next version of MEETCITY adopted the philosophy of *guided user experience*. A user could only access and interact with a limited subset of user interface (UI) geared toward completing the next step in their use case — be it browsing nearby user profiles, sending out a request, navigating and attending the dates. This more minimalistic UI design was significantly better received.

c) Software Adoption: Unlike traditional server/desktop software where software development is often commissioned — so much so that (the promised) software adoption is routinely *a priori* to software development — apps are faced with a critical software adoption problem: among all apps within the same umbrella category, such as meeting/friendship/dating apps, only a small fraction will survive the fierce competition. We believe that software adoption should be a critical link in modern software engineering that calls for more study in the era of apps.

In our experience, *feature*-based marketing is most effective. With limited budget, most apps — including MEETCITY — cannot afford to have large campaigns that reach out to end users *en masse* and market their app as a "do-it-all" and "for everyone" artifact. MEETCITY adopted an approach to highlight its focus on AI.

In the longer time span however, the adoption of apps cannot be driven by one single feature. To sustain the growth, software functionality, reliability, and user interface all matter to end users. Once a stable release can be achieved, software adoption can benefit from marketing. In the case of MEETCITY, we initiated a 6-month campaign with a 3-prong approach: social media marketing, experiential Marketing campaign with promoters distributing flyers, and influencers conducting demonstrations at special events.

d) Priorized Evolution: As a living app, MEETCITY receives a large number of feature requests, often more than

what our development team can complete. In addition to bug fixes, our philosophy of software evolution is consistent with how we orient MEETCITY in software adoption: we prioritize features that may further highlight our uniqueness in in-vivo intelligence. For example, we are currently working on adaptive clustering algorithms where the coefficients (such as α, β, and γ) may evolve online. We are also interested in mining text-based user comments and feedback to enrich our core clustering algorithm. Finally, when a recommended meeting leads to low ratings, it serves as valuable *de facto* "bug report" we wish to analyze, hopefully providing feedback to the adaptive clustering algorithm.

e) Privacy: Users greatly care about privacy. Beyond the excitement of embracing AI to meeting apps, the most common reaction we receive is a concern on privacy. As we discussed in Section II-B, learning personality can cause alarm among users unfamiliar with how machine learning works in general. The process of articulating our design strengthened our belief that user privacy should be maximally respected. For instance, as richer data may lead to better results of learning, it might be tempting to encourage users to share more information. This in our opinion is too high a price to pay to end users.

VI. THREATS TO VALIDITY

First of all, our reported analysis only applies to our experience with one app, MEETCITY. Even though it is our opinion that the philosophy of in-vivo intelligence may be applicable to the design of a large family of social apps, the specific results of data analysis may not generalize to other apps. For example, our personality-based learning may matter for friendship apps, but may not be applicable to LinkedIn-style social apps with a professional purpose. Second, our data set consists of thousands of observations, magnitude smaller than the scale of friendship apps such as Tinder. Our general belief with scalability is that for friendship apps where geo-promixity plays a role, different metropolitan areas should be analyzed and maintained separately. As a result, a natural decomposition exists for the algorithm, making scalability lesser of a concern in algorithm design. This speculation however can only be substantiated with a larger data set. Third, even though our definition of personality traits as clustering features is experimentally effective, there is no fundamental justification on why our definition is superior to alternative definitions. The heuristic nature of this approach is indeed in line with statistical approaches in psychology: unlike the "perfect" computational world where many algorithms come with some notion of "optimality," studying human behavior is often inherently empirical.

VII. RELATED WORK

Our approach calls for synergy between human and artificial intelligence in software engineering, a philosophy well-articulated in previous work [17]. Broadly, AI in software engineering [10, 18] is an active area of research, with prolific and diverse solutions ranging from software maintenance [13], bug prediction [14], specification inference [19], project management [8], and energy optimization [6], to name a few. In this context, this paper focuses on addressing a systematic design to bridge AI and social app design.

The connection with cognitive psychology and software engineering is widely known in requirements engineering [15]. The influence of personality has been exhaustively studied for project management [1, 7] and conceptual design [2].

To the best of our knowledge, MEETCITY is the first mobile software system where in-app user behavior is monitored to help learn personality traits, which is in turn used to guide meeting apps. It brings together several distinct areas of interest in software engineering: social apps design, AI, and digital psychology.

VIII. CONCLUSION

In this paper, we call for in-vivo intelligence for the design of social apps. MEETCITY as a concrete instance of in-vivo intelligence relies on hierarchical clustering of personality-inspired features to learn dynamic behavior of users in meeting apps. As a living app, MEETCITY demonstrates the benefit of building in-vivo features into meeting app design, leading to improved user experience.

REFERENCES

[1] Forty years of research on personality in software engineering. *Comput. Hum. Behav.*, 46(C):94–113, May 2015.

[2] Farshid Anvari, Deborah Richards, Michael Hitchens, and Muhammad Ali Babar. Effectiveness of persona with personality traits on conceptual design. *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, 2:263–272, 2015.

[3] Mariette Awad and Rahul Khanna. *Efficient Learning Machines*. Apress, 2015.

[4] Albert-Laszlo Barabasi and Reka Albert. Albert, r.: Emergence of scaling in random networks. science 286, 509-512. *Science (New York, N.Y.)*, 286:509–12, 11 1999.

[5] G. Bonaccorso. *Machine Learning Algorithms - Second Edition*. Packt Publishing, 2018.

[6] Anthony Canino, Yu David Liu, and Hidehiko Masuhara. Stochastic energy optimization for mobile GPS applications. In *ESEC/SIGSOFT FSE 2018, Lake Buena Vista, FL, USA, November 04-09, 2018*, pages 703–713, 2018.

[7] S. S. J. O. Cruz, F. Q. B. da Silva, C. V. F. Monteiro, P. Santos, and I. Rossilei. Personality in software engineering: Preliminary findings from a systematic literature review. In *15th Annual Conference on Evaluation Assessment in Software Engineering (EASE 2011)*, pages 1–10, April 2011.

[8] Massimiliano Di Penta, Mark Harman, and Giuliano Antoniol. The use of search-based optimization techniques to schedule and staff software projects: An approach and an empirical study. *Softw. Pract. Exper.*, 41(5):495–519, April 2011.

[9] Lewis R Goldberg. Language and individual differences: The search for universals in personality lexicons. *Review of personality and social psychology*, 2(1):141–165, 1981.

[10] M. Harman. The role of artificial intelligence in software engineering. In *2012 First International Workshop on Realizing AI Synergies in Software Engineering (RAISE)*, pages 1–6, June 2012.

[11] Mansoor Iqbal. Tinder revenue and usage statistics, 2018.

[12] Oliver P John, Sanjay Srivastava, et al. The big five trait taxonomy: History, measurement, and theoretical perspectives. *Handbook of personality: Theory and research*, 2(1999):102–138, 1999.

[13] S. Mancoridis, B. S. Mitchell, Y. Chen, and E. R. Gansner. Bunch: a clustering tool for the recovery and maintenance of software system structures. In *Proceedings IEEE International Conference on Software Maintenance - 1999 (ICSM'99). 'Software Maintenance for Business Change' (Cat. No.99CB36360)*, pages 50–59, Aug 1999.

[14] Amir Michail and Tao Xie. Helping users avoid bugs in gui applications. In *Proceedings of the 27th International Conference on Software Engineering*, ICSE '05, pages 107–116, New York, NY, USA, 2005. ACM.

[15] Bashar Nuseibeh and Steve Easterbrook. Requirements engineering: A roadmap. In *Proceedings of the Conference on The Future of Software Engineering*, ICSE '00, pages 35–46, New York, NY, USA, 2000. ACM.

[16] Kari Paul. From 53 matches to 4 dates: What a month on tinder is really like, 2018. Published: Mar 16, 2018.

[17] T. Xie. The synergy of human and artificial intelligence in software engineering. In *2013 2nd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE)*, pages 4–6, May 2013.

[18] Tao Xie. Intelligent software engineering: Synergy between ai and software engineering. pages 1–1, 02 2018.

[19] Hao Zhong, Lu Zhang, Tao Xie, and Hong Mei. Inferring resource specifications from natural language api documentation. In *Proc. 24th IEEE/ACM International Conference on Automated Software Engineering (ASE 2009)*, pages 307–318, November 2009.