Catalyzing Social Interactions in Mixed Reality

Empirical research study for predicting missed or overlooked in-person connections using ML.

Team Cyberblast - Sparsh Srivastava (ss6381) & Rohan Arora (ra3091)

COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

Elevator pitch

Research questions

RQ1. *What is the impact of mixed reality in catalyzing novel social interactions between co-located people?*

RQ2. *What environmental factors play the largest role in creating or preventing social interactions?*

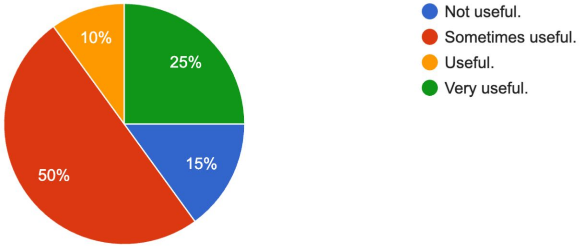
RQ3. *What are the primary user features that make people want to interact with each other?*

Contributions

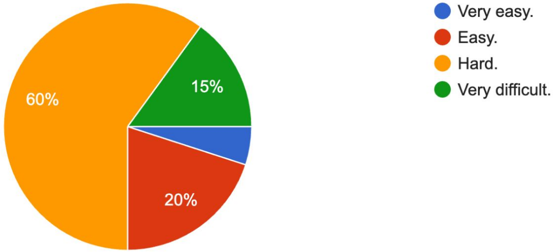
- Focus group analyzing user preferences and present-day perceptions around social interactions with co-located strangers.
- Novel dataset on social preference collected through an empirical study conducted using human participants.
- Four user-to-user recommendation models trained on combinations of MR features, user features, and right-time features for predicting missed in-person interactions.

Focus group

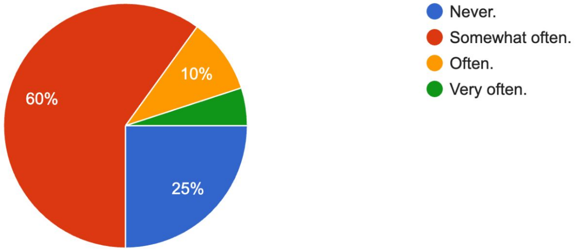
How useful would it be to receive a notification when someone nearby wants to connect with you?
20 responses



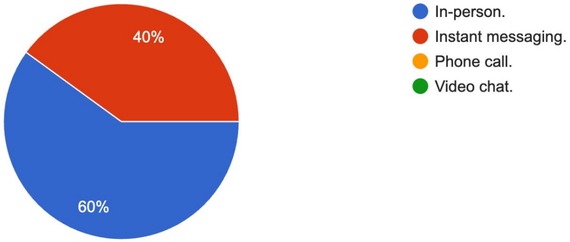
How easy or difficult is it to meet new people in public?
20 responses



How frequently do you want to interact with someone new when you are in public?
20 responses



How would you prefer to interact with new people for your first interaction?
20 responses



Feature classes

User features

- Personal information provided by the self and candidate users.
- e.g. ‘age', 'gender', 'education', 'student', 'workforce', 'industry', 'hobby', 'interest', 'music_genre', 'personality', 'social_media', 'music_listen_time'.

Right-time features

- Environmental data that can be collected by traditional devices, i.e. audio level, location, etc.
- e.g. 'location', 'weather', 'human_noise_level', 'non_human_noise_level', 'day_of_week', 'time_of_day'

Mixed reality features

- Visual data that can only be collected by a mixed reality device.
- e.g. 'height', ‘hair_type’, ‘hair_color’, ‘tattoos’, ‘conversational_intensity’, 'human_congestion_level', 'occlusion', proximity', 'gaze_self_to_candidate', 'gaze_candidate_to_self', ' 'self_clothing’, 'candidate_clothing’

Data collection - User survey

- We sent a survey to 10+ participants to gather feedback on structure and crowdsource answer choices for several of our feature values, such as:
  - “favorite hobbies / interests”
  - “frequently visited locations”
  - “favorite genre of music”.
- Then, we utilized our updated survey to collect user features such as:
  - “age”
  - “gender”
  - “personality type”
  - “clothing preferences by location”
- 40+ participants (< 10% replaced).

```
class User:
    def __init__(self, age, gender, height, hair_type, hair_color, has_tattoos,
                 education, is_student, is_in_workforce, industry,
                 favorite_hobby, favorite_interest, music_genre, personality,
                 listen_or_speak, favorite_social_media, music_listen_time,
                 group_size, clothing_athletic, clothing_casual,
                 clothing_trendy, clothing_formal, clothing_designer,
                 clothing_eyeglasses, clothing_sunglasses,
                 clothing_luxury_watch, clothing_smart_watch, clothing_hat,
                 clothing_necklace, clothing_rings, clothing_earrings):

        self.age = age
        self.gender = gender
        self.height = height
        self.hair_type = hair_type
        self.hair_color = hair_color
        self.has_tattoos = has_tattoos
        self.education = education
        self.is_student = is_student
        self.is_in_workforce = is_in_workforce
        self.industry = industry
        self.favorite_hobby = favorite_hobby
        self.favorite_interest = favorite_interest
        self.favorite_social_media = favorite_social_media
        self.music_genre = music_genre
        self.music_listen_time = music_listen_time
        self.personality = personality
        self.listen_or_speak = listen_or_speak
        self.group_size = group_size
```

Data collection - Scenario survey

You are at a **sit-down restaurant** on a **weekday** in the **evening**.
It is a **rainy** day and it's **crowded**.
There are **many people talking** and **music is playing**.
You see a person **looking at you** who is **nearby**.
The person is with **2 - 3 other people** and they are **speaking to others** in their group.

```
class Scenario:
    def __init__(self, location, weather, human_congestion_level,
                 human_noise_level, non_human_noise_level, candidate_occluded,
                 gaze_self_to_candidate, gaze_candidate_to_self, proximity,
                 day_of_week, time_of_day):
        self.location = location
        self.weather = weather
        self.human_congestion_level = human_congestion_level
        self.human_noise_level = human_noise_level
        self.non_human_noise_level = non_human_noise_level
        self.candidate_occluded = candidate_occluded
        self.gaze_self_to_candidate = gaze_self_to_candidate if not \
                                      candidate_occluded else False
        self.gaze_candidate_to_self = gaze_candidate_to_self
        self.proximity = proximity
        self.day_of_week = day_of_week
        self.time_of_day = time_of_day
```

- 40 participants included from the first surveys.
- We generated scenarios that include all the features that we want to use for our recommendation models
  - Deterministic (corresponding to answers from the user survey)
  - Non-deterministic (chosen randomly from possible values)
- Collect output label “self_decision” with values {“Meet”, “Chat”, “Reject”}.
- 198 data points collected across 73 features + 1 output label.

Model training

- One-hot encoding of categorical variables
- RandomForestClassifier from the Scikit Learn library in Python
- Grid search with 5-Fold Cross Validation:
    - Running time of 10-12 hours, 36,800+ models built.
    - Recorded metrics (accuracy, precision, recall, f1 score).
    - Found best hyper-parameters and stored the best performing models.

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import ParameterGrid

param_grid = {
  'n_estimators': [50, 100, 200, 500],
  'max_depth': [3, 5, 7, None],
  'min_samples_split': [2, 4, 6, 10],
  'min_samples_leaf': [1, 2, 5, 10],
  'max_features': ['sqrt', 'log2', None],
  'min_impurity_decrease': [0, 0.001, 0.01, 0.05, 0.1, 0.2]
}

...

for params in ParameterGrid(param_grid):
  ...

  random_forest = RandomForestClassifier(**params)
  random_forest.fit(X_train, y_train)

  ...
```

Results - Model evaluation

- Predicting label with three values: {Meet, Chat, Reject}.
- Combination model suffered from data sparseness.

Baseline model
includes user features only (24/73)

- Accuracy: 0.58
- Precision: 0.61
- Recall: 0.58
- F1 score: 0.57

Right-time model
includes user & right-time features (30/73)

- Accuracy: 0.55
- Precision: 0.59
- Recall: 0.55
- F1 score: 0.54

Mixed reality model
includes user & MR features (67/73)

- Accuracy: 0.54
- Precision: 0.57
- Recall: 0.54
- F1 score: 0.54

Combination model
includes user, MR, & right-time features (73/73)

- Accuracy: 0.53
- Precision: 0.56
- Recall: 0.53
- F1 score: 0.52

Results - Model improvement

- We combined the two positive classes into a single class, s.t. {Meet, Chat, Reject} => {Accept, Reject}.
- After re-training the model, we improved the performance of the combination model by > 0.15 for each metric.

Baseline model
includes user features only (24/73)

- Acc: 0.58 → 0.72
- Prec: 0.61 → 0.73
- Recall: 0.58 → 0.72
- F1: 0.57 → 0.72

Right-time model
includes user & right-time features (30/73)

- Acc: 0.55 → 0.71
- Prec: 0.59 → 0.74
- Recall: 0.55 → 0.71
- F1: 0.54 → 0.71

Mixed reality model
includes user & MR features (67/73)

- Acc: 0.54 → 0.70
- Prec: 0.57 → 0.72
- Recall: 0.54 → 0.70
- F1: 0.54 → 0.70

Combination model
includes user, MR, & right-time features (73/73)

- Acc: 0.53 → 0.69
- Prec: 0.56 → 0.71
- Recall: 0.53 → 0.69
- F1: 0.52 → 0.69

P(A & B accept) = P(A accepts B) * P(B accepts A) = 0.72 * 0.72
= 0.52

Demo

Q&A

References

- Survey data (.csv): https://drive.google.com/drive/folders/1RGaKiQumhk3FCVpnUQTppiounSXU8Dod?usp=sharing

- Survey data (combined):
  - https://docs.google.com/spreadsheets/d/1TEDkCVeNlVgnwFVOmRGT6hrMG1v0XQW_FpsWLj24IPw/edit?usp=sharing
  - https://docs.google.com/spreadsheets/d/1fZUkagixGKIyNE8xiot_WjMuVHab0Oa9temCVx1f67Y/edit?usp=sharing

- Saved models: https://drive.google.com/drive/folders/1ZXWE6Y5VHFPp89qJi2OdfiwrPzyTnyUh?usp=sharing

- Full dataframe (198 x 74): https://drive.google.com/file/d/1-DPX96ZL1wnBTOS-QU1mLo-_RUM2x-7I/view?usp=sharing